

PRIVATE

ENTREPRENEUR

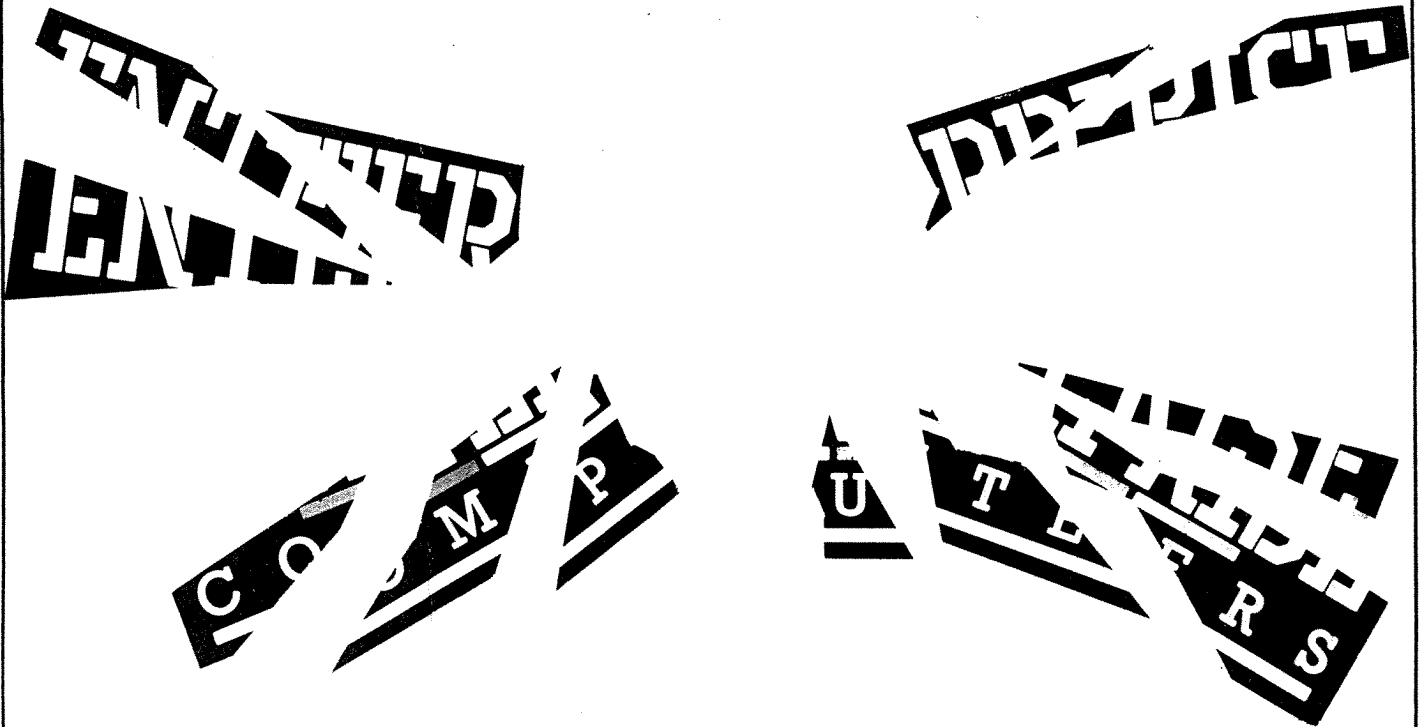
magazine.

LATE SUMMER
1986

ISSUE 6
An I.E.U.G publication

IEUG

INDEPENDENT ENTERPRISE USER GROUP



Picking up the pieces

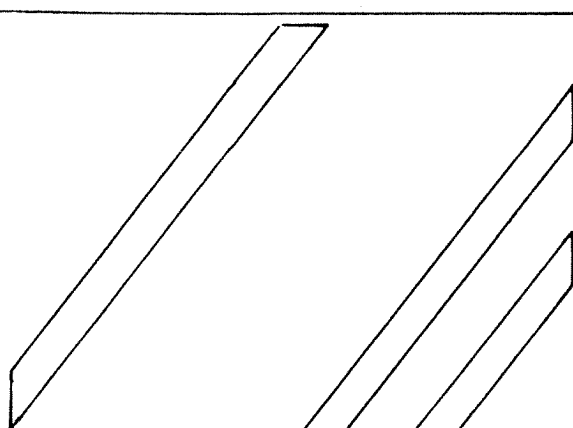
IEUGs
history

Basic
EXTENSIONS

Latest Software
reviews.

Readers
views.

DATABASE
PROGRAM



THIS
SPACE

Available to the trade



**CONTACT THE I.E.U.G
FOR DETAILS AND RATES**

£ 35 128K

Editorial

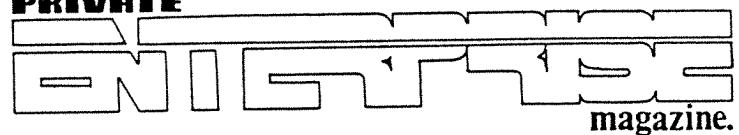
Yes.... it's finally here ! After the longest-ever delay, Issue 6 has finally arrived. Also, it's our Anniversary, although I'd prefer to have celebrated it in happier circumstances.

Anyway, enough of the revelry - down to business. You should all have read my letter by now, and be aware that Enterprise Computers Ltd. no longer exists, and that the User Group is endeavouring to provide you with better support than Enterprise did. The first step towards this goal was taken at the very successful PCW Show (many thanks to all who showed up and talked to us), where we caught the attention of the computing press, and also provoked some very positive reactions from a number of software houses.

However, nothing will be achieved overnight, and you must be prepared to bear with us while things are sorted out. The first move must be one of introspection, for we cannot continue in the chaotic manner in which the User Group has been run up to now - a restructuring is necessary. Towards this end, there will be a meeting in the near future (date and venue to be confirmed in Issue 7), in which, in addition to seeing all the new software, you will be able to take part in the democratic process of reorganising the running of the User Group. Until this moment, there has been no opportunity for anyone to find out how the User Group is run, or contribute any advice, other than by writing letters to Mark or Tim. Now will be your chance, if you wish it, to challenge our positions of ultimate power (and ultimate commitment !) and stand for election against us for a User Group post. A full list of items for discussion will accompany Issue 7. (approx four weeks away) and the minutes of the meeting will be published in Issue 8.

Neil Blaber

PRIVATE



magazine.

CONTENTS...

ISSUE 6

NEWS DESK > Dave Race looks back on a year of Private Enterprise when most things almost certainly were not what they at first seemed. **4**

PRIVATE CORRESPONDENCE > Private Enterprise tries to get to grips with users problems resulting from Enterprises receivership. **6**

PROGRAMMING > Get heavily into detachable hoods, facial diseases and long tangled greecy hair when you break the basic boundries into machine code. **8**

SOFTWARE UPDATE > Two new Utilities emerge from Tim Boxes Boxsoft Programs, including the new Zzip Basic Integer Compiler. **10**

ADVANCED PROGRAMMING > Create your own basic extensions (Strictly for the Machine code mind) **12**

HOME PRODUCE > May the force be with your fingers when you start typing in this mega Database program. **16**

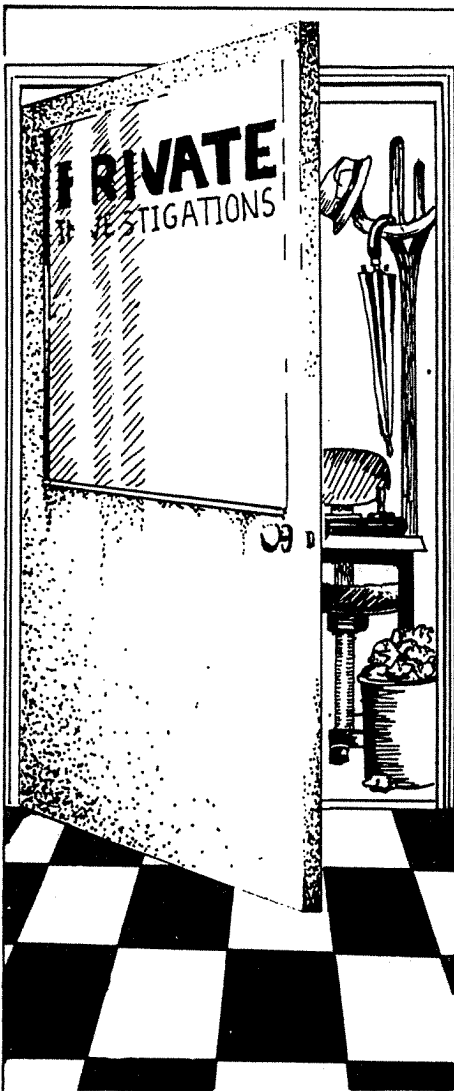
THE INDEPENDENT ENTERPRISE USER GROUP

12 Whitegates,
100 Station Road,
New Barnet,
HERTS,
EN5 1QB,
ENGLAND.

01/440/4110

An Independent Enterprise User Group Publication. Artwork & Layout MARK LISSAK, Private Correspondence DUNCAN TAYLOR. News Editor & Software reviews DAVE RACE. Articles Editor NEIL BLABER. Secretary TIM BOX. Private Enterprise Magazine is a copyright of The Independent Enterprise User Group. No article may be reproduced in whole or in part without written consent from the copyright holders.

256K



News Desk

History of PRIVATE

ENTRISSE

I to V

demise of Enterprise and rumours of the 320, an Amstrad basher which will probably never now see the light of day.

So, I thought I'd spend this issue looking back through the last five issues of the magazine at some of the successes and some of the failures of the last year and a bit.

1

The 128 was launched and proved to be by far the more successful of the two models, offering a larger memory than any other home micro at that time and being considerably quicker than the 64. Enterprise signed a distribution deal with Terry Blood, billed by us as a "major step" - the contract seemed to last all of 10 minutes !

A vast number of hardware peripherals had already been produced for the machines, i.e. a medium resolution monitor (soon to be phased out), a good quality dot matrix printer and a joystick interface, ah heady days ! Also, "in the pipeline", were a disk interface, later to become known as EXDOS (a truly excellent piece of kit), and the much fabled Base Unit, which degraded into what became known as "that horrible piece of plastic that sits between me Enterprise and me Exdos", in polite circles.

Software was coming in thick and fast - titles mentioned included Dambusters, Matchday, Seventh Seal, Super Pipeline II and Frank Bruno's Boxing, all of which have given us a

great deal of entertainment over the year. To be fair, of the 23 titles mentioned in issue 1, 14 did appear including such greats as BEATCHA and JACKS HOUSE OF CARDS. Finally, it was announced that the Enterprise 128 "had been selected for the Design Centre, an accolade only given to three other micros - the ZX81, the SPECTRUM and the BBC". Notice any connection between the four ?

2

Lots happening this issue !

Production of the two machines moved up to GRI Ltd in Scotland. Mr Twine of GRI described the Enterprise as "an excellent saleable product and (with) the soundest possible financial backing" (ho hum).

Enterprise set up business in Germany (and are still going ! - Ed.), but lost their distribution link with Zappo. Zappo were apparently a little peeved at Enterprise signing up with Terry Blood - they shouldn't have worried.

We were all getting over the excitement of running the Enterprise stand at the PCW show (I kid you not). In fact we managed to spend half of the news section extolling the virtues of just about everything we saw there; even the mouse and Speakeasy got a mention - well, one out of two ain't bad ! Surprisingly, just about every piece of software mentioned this issue got out !

As you must all know by now, Enterprise went into Receivership some little time ago; this was on the cards for some time, but was still a shock when it actually happened. It's one of those things that always happens to other people - not someone you know personally.

This sad turn of events will, of course, mean some changes to the magazine and User Group, as mentioned by Neil in his letter (which you should all have read by now !), although everything will keep going of course. One of the items likely to be in very short supply in the future is news - I used to make most of it up as it was ! Indeed, as of this moment, the only real news around is the

News Desk

3

The issue with the cover Jeff Minter would have loved, and the record for the most misspellings of Gary's name - proof readers, who'd 'ave 'em?

The all-powerful McIntyre deal was disclosed. This comprised of a couple of good value systems, one with the 64 and one with the 128, to be sold mail order and backed up by a massive (the likes of which have never been seen before) advertising campaign. This seemed to consist of a series of "subliminal" adverts in Popular Computing Weekly (which included a wonderful misquote of Neil), and a couple of adverts in the national press around Boxing Day. I've never been sure which had the greater impact

Exdos units were available, as were internal RAM expansions. No external RAM packs "until after Christmas".

We started on the long slippery slope to providing you with a dedicated joystick (which Aztec never quite got round to letting us see), and the mouse and Speakeasy were once again very nearly with us!

19 software titles mentioned this time, many of which had been mentioned in issue 1, 8 of which never made it. I'm not counting "View to a Kill" in with that lot!

4

Guess what? The mouse was nearly with us - ooh, the excitement! See issue 4's cover for a more realistic representation of the facts. I was happily singing the praises of Macro-D and Asmon this issue, which just goes to show you what an idiot I am, as neither has seen the light of day.

Wimpy choose an Enterprise for their stand at the Ideal Home Exhibition, obviously not wishing to offend any of the major home computer manufacturers.

Finally it was revealed that IS-DOS was to be pushed by I.S. as "an industry standard 8-bit operating system" and would be producing packages for it. I.S. were unavailable for comment on this (I wonder why?).

5

Oh that front cover, if only we knew!

30% of this issue's news was taken up by a description of our second great meeting, which was admittedly a great success. That just goes to show you how much was happening towards the end.

Aztec bit the dust, shame! I can now reveal that one of the reasons that

they couldn't get the mouse out was that they had a warehouse fire and all their stock was burnt, including the preprods, and the moulds, and the plans - in fact, everything seemed to be in there - strange....

The Technical Manual was launched (hands up everyone who got one) while I was still rabbiting on about Asmon and Macro-D.

Finally, if you want a giggle, read the last paragraph in the news section again in the light of recent happenings.

So there we have it, a year in the life of... It just goes to show you that no-one can get it right all the time, in fact in the face of such adversity I was lucky to get any of it right at all!

**DAVE
RACE**

PRIVATE
ENTERPRISE
PROGRAMS

IEUG's
Greater
Util

VOLUME 1

Available

NOW!

£2.00

Hello and welcome to another Private Correspondence, but this time with a difference; changes have taken place. To be more precise, I have taken on the letters page from Tim. Before you all do nasty things like sending us your Enterprise versions of the "Commodore Rap", I must tell you that the address WILL be the same as before (for the time being... heh heh - Ed). If you have any queries which you would prefer to phone up about, then contact Tim at the number given in Issue 5. Oh yes, before I go any further I must tell you that my name is Duncan Taylor. Now to the serious bit, your letters:

Dear I.E.U.G

Now that Enterprise Computers have gone into Receivership what will happen to all the hardware and software that was about to be released like Eggs of Death, Sprite Handler, Zip Compiler and the mouse?

Will you still be continuing the Private Enterprise? How do you use the Attributes commands properly in Basic? How can I move a character about the screen without disturbing the background? I recently bought a Forth cartridge and I am having problems using it, please could you do an article on Forth? Will you be doing any articles on machine code?

I have just bought a Brother HR5 printer and tried your screen dump program on the Greatest Hits Vol 1 Tape and I keep getting a variable error at the beginning of the program, is this another bug or is it my printer? Can you tell me if there are any Enterprise users in the Humberside area? Is it possible to buy memory upgrades for the 64 in electronic shops?

Nigel Schrimshaw,
South Humberside.

DT: Although Enterprise Computers have gone into Receivership, we will still be supporting the Enterprise as fully as we can. Neil's letter should have

PRIVATE

gossip, outrage, it's your page.

explained this fully. Zip will be available soon, watch this mag for details.

I'm afraid that to do justice to explaining the Attribute command would necessitate writing an article about it - anyone out there interested?

Moving characters about the screen without disturbing the background would have to be done using machine code and would be too long to include in Private Correspondence.

We will do articles on Forth and Machine code if anyone would like to send them in to us. We can only do a limited amount ourselves, and nobody at IEUG HQ has any expertise with Forth, so I can't guarantee anything.

Your printer should work perfectly well with the screen dump program, as it is Epson compatible. Check to see that you are doing everything correctly, and if you are still experiencing problems contact Tim at the IEUG address, as he supplied the offending program!

We can't publish peoples names and addresses or forward them on, unless they specifically ask us to do so. If you would like to become a Local Area Organiser, tell us and we can then tell people in your area to get in touch with you.

The memory upgrade to 128k involves quite a number of components on a circuit board, so to construct one yourself is unfortunately not as simple as putting in a few chips. Keep a look out in further issues for any

news about upgrading.

Dear I.E.U.G

I propose to make all Enterprise users not in the IEUG aware of it by writing to magazines informing them of the IEUG, if that's all right with you guys.

Down to problems now:

a) How does one convert tape to disk when the initial part of the tape contains the load instructions for the next part? I have DEVFAC. How do I go about getting the next bit to load from disk? I already have all the separate parts of a program, but need to get them together.

b) How does one get a modular tape (e.g wriggler, sorcery) to load when the EXDOS is connected? I have to disconnect EXDOS to play them.

c) How does one alter the program on Lands of Havoc to get it to function with the external joystick?

d) I can't understand some of the notation in the Technical Manual. How should IRQ_ENABLE_STATE, FLAG_SOFT_IRQ etc be used? Should they be part of a Basic program or are they just notations for machine code?

e) How does one get to EXOS to put in EXOS variables? How do I input EXOS Kernel functions in basic machine code?

(I'm sorry I can't read your signature)

Private Correspondence

Purton,
Swindon.

DT: a) To get EXDOS to transfer things from tape to disk directly either use:

```
:COPY TAPE: program_name
```

or:

```
10 OPEN INPUT [1:"TAPE:"  
20 OPEN OUTPUT [2:"DISK:file"  
30 COPY [1 TO [2
```

The latter will only copy one file at a time (refer to the EXDOS manual). To link up separate modules of the same program you need to find the name of each of the modules and replace every occurrence of "TAPE:" in a module with the name of the next module in the sequence. To do this you need to use a machine code (or Pascal... Ed.) program.

b) The problem with some tape software is that the code is written in memory which EXDOS uses (unfortunately, Wiggler and Sorcery are in this category); there is no way of getting round this other than completely relocating the program. If you feel in a particularly masochistic mood, and write a program to do this, you will make a lot of people hysterically happy (hint hint).

c) One of the marks of a good program is that it allows users flexibility in its use - unfortunately not all programs are written with this in mind. Lands of Havoc doesn't allow the use of an external joystick, so unless you are willing to reprogram some of it then I am afraid there is nothing simple that can be done. Another example of the above is the number of programs which don't allow the external speaker to be switched off.

d) You are not the first to be mystified at the notation used in either manuals, a little explanation

is necessary.

FLAG_SOFT_IRQ and the like are just labels, they refer to specific areas in memory whose state either changes the operation of the machine, or which give an indication of what is going on. For example, FLAG_SOFT_IRQ refers to address BFF2 (in Hex) in segment 255 of the memory. This byte is set, either by the programmer or the computer during operation, to cause an interrupt. The Technical Manual explains it further.

e) To use EXOS properly you need to write a Pascal or machine code program to do whatever it is you want to do with EXOS. A machine code program can then be accessed from a Basic program, if you want to, using the USR command. See the article on Exos Variables in this very issue for more details!

Thanks for the offer of "spreading the word". Everything to advertise the group to non-IEUG Enterprise users is welcome.

Dear I.E.U.G

I have bought EXDOS, but the Receiver tells me that I cannot now have a free IS-DOS disk. Is there any way I can obtain IS-DOS?

With EXDOS connected most tapes will not load fully even with the command LOAD "tape:". Some tapes just stop loading and others will only continue loading if I press START immediately after the monitor indicates the loading has finished.

I bought a Cumana 3.5" twin disk drive, and having lost the instructions. I am unsure of what disks I can use with it.

J.D.Weir,
Croydon,
Surrey.

DT: There have been a number of problems getting hold of IS-DOS; nothing has been sorted out yet, although news from the Receiver indicates that Enterprise's assets will be sold soon, so hopefully this issue should be resolved in the very near future.

The question about EXDOS and the loading of tapes is a question a number of people have written in about. This problem exists because when EXDOS is connected, the default input device automatically changes to DISK: This is not a problem if you are only dealing with a single file, but most programs consist of a number of files, usually a loader program followed by the main program. The trouble is that a number of these loader programs do not specify TAPE: when looking for the file to be loaded, and of course it will be searched for on disk.

With your Cumana drive, any 3.5" disk will do, although if it's only a single-sided drive don't waste money by buying double-sided disks!

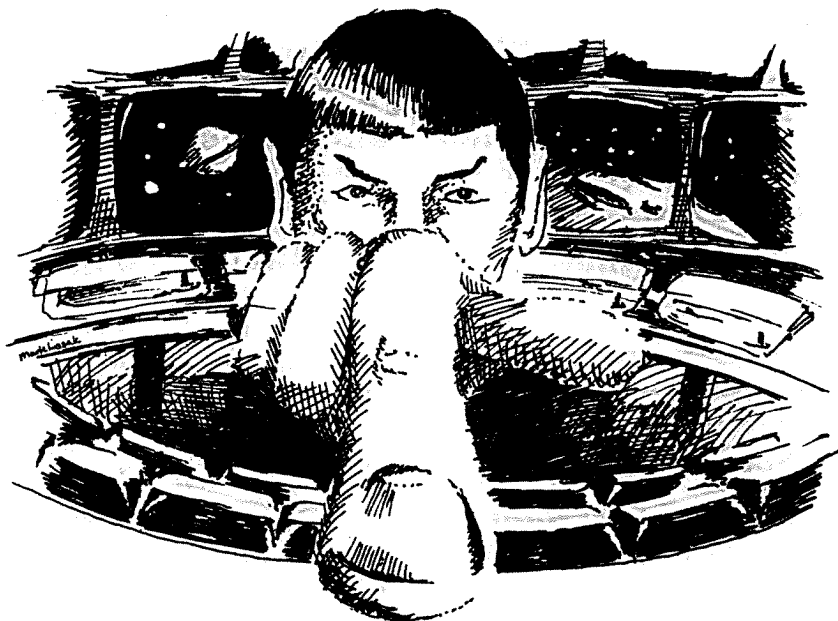
Programming

There are often times when it is necessary to run machine code routines from a main BASIC program, or store large amounts of data without using arrays. To do this you must find a place in the computer's memory where it will not be overwritten by other things. Whilst it is fairly easy to protect it from the needs of the EXOS operating system, protecting it from the BASIC is rather more difficult as it is not properly documented.

The "official" Enterprise way of setting aside memory is to use the ALLOCATE command, as mentioned in the chapter of the BASIC manual entitled "Using Machine Code". You allocate a certain amount of memory for use by your routine, and insert the machine code into the program using CODE statements, typing it in as bytes of hex. This is fairly well explained in the manual. As a method of using machine code it is not bad; you are allowed to use "labels" in the hex, and can CALL these labels by name. However, when you compare it with the likes of "DEVPAC" and "ASMON" (I am lucky enough to have a copy of ASMON, it's brilliant!), the hex method seems a little primitive, and also wastes memory with thousands of CODE lines.

The problem is that there is no way of to include an assembled file from either assembler into an allocated block, as the BASIC lacks a facility to LOAD or SAVE blocks of machine code. It is possible to use relocatable modules via the operating system, but they can be a bit of a mouthful and are not really necessary:

I have written short routines to SAVE and LOAD machine code, and these are printed below. There are two versions of each, BASIC and machine code, although you will of course need a machine code loading routine to load the machine code loading routine (!) unless you convert it into hex and use CODE.



M/C from Basic

LOAD:

```
100 PROGRAM "MLOAD"
110 LET MCLen = <length of code>
120 ALLOCATE MCLen
130 CODE MC = "0"
140 OPEN #106:"name" ACCESS OUTPUT
150 FOR I = MC TO MC + MCLen - 1
160 GET #106:A$
170 POKE I,ORD (A$)
180 NEXT I
190 CLOSE #106
```

```
MLOAD: LD A,106D
LD DE,location of code
LD BC,length of code
EXOS 06D ;Write block
RET
```

SAVE:

```
100 PROGRAM "MSAVE"
110 LET MCLen = <length of code>
120 ALLOCATE MCLen
130 CODE MC = .... ! Your M/C
140 CODE MC = .... ! program
150 CODE MC = .... ! (hex)
160 OPEN #106:"name" ACCESS INPUT
170 FOR I = MC TO MC + MCLen - 1
180 PRINT #106:CHR$(PEEK(I));
```

```
190 NEXT I
200 CLOSE #106
```

```
MSAVE: LD A,106D
LD DE,destination
LD BC,length of code
EXOS 08D ;Read block
RET
```

The BASIC routines are pretty self explanatory, but the machine code routines assume that channel #106 has previously been opened to tape or disk. This channel should be closed afterwards. The machine code was written using ASMON.

As long as the code you load in is assembled to run at the start address of the allocated space (usually 4809), you will be able to call and use it. You can also use it to load data into the allocated space. This method will sometimes do for small routines, but there are unfortunately a few snags.

The allocated space seems to be part of the variables area, and is cleared by a RUN command. It is not possible to extend the allocation over more

Programming

than one segment, so in practice you are limited to about 12K of space. On a 128K machine this is pathetic! Finally, EXOS 2.0 in the 64K machines has a bug in the ALLOCATE command, requiring patching with the program featured in previous issues.

We can thank Intelligent (?) Software for these little features, unfortunately they make the ALLOCATE space quite hard to use.

```
LD A,(FREESEG)      ! Load A with number of previously
LD C,A              ! allocated segment
EXOS 25              ! "Free Segment" EXOS call
EXOS 24              ! "Allocate Segment" EXOS call
RST 18              ! See if an error has occurred
LD A,C              ! C now contains the segment number
LD (FREESEG),A      ! of the allocated segment
LD H,0
LD L,C
RET
```

```
FREESEG DEFB 0      ! Address where user's segment
                   ! number is held. Must not be
                   ! corrupted by any other data.
```

If you want to use a mega-long piece of code, or huge library of data, then there is an alternative - you can set up your own 16K segment or segments for use by your BASIC program. Unfortunately, the set-up code must be put in using the infamous ALLOCATE statement, but that's life!

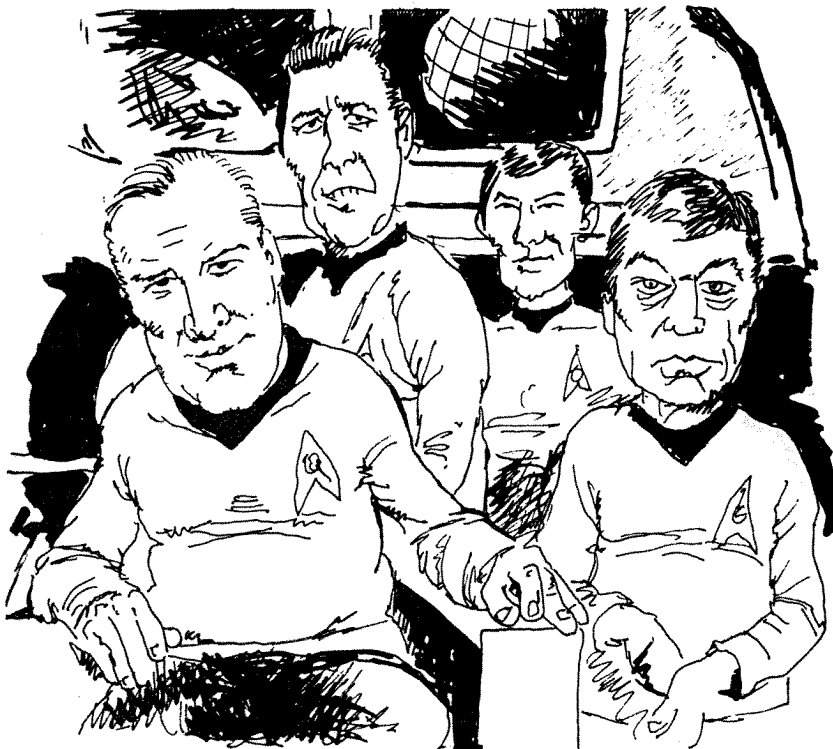
Here's the set-up code. It is very simple because it relies on EXOS calls to do all the work.

The program above will allocate you a 16K segment, and return its segment number to your BASIC program in register HL (see the BASIC manual). Any number of segments may be allocated like this, up to the number free to use, but there is one important point to remember when using the routine. You must make sure that the segment is freed by EXOS, using the free segment call before the segment is allocated, otherwise you will be given a new segment. If this is not done each time, you will soon find the routine happily gobbling up the system's free-memory each time you run your program.

Before you can use a segment, it must be paged into the Z80 address map page 2, using OUT 178, freeseg. This ought to be done each time the segment is used, as the BASIC may sometimes want to use the page. You can then PEEK and POKE to your statement from BASIC - it will be situated from 32768 to 49151 in the address map, but remember to be careful.

You can LOAD or SAVE part or all of your segment(s) using the LOAD and SAVE routines above. If you want to know more about the operating system, allocating memory etc. then the Enterprise Technical Manual is a must to buy.

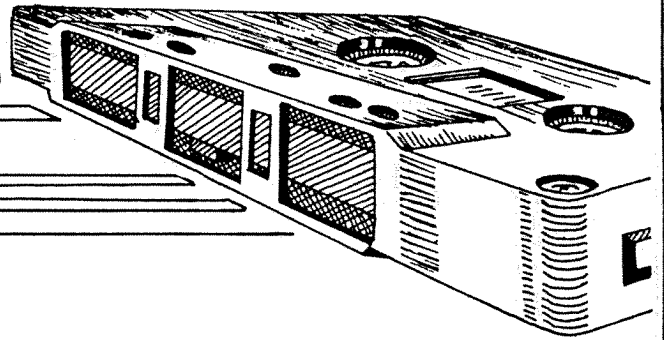
A.S. Burnham



(Editor's note - The manual is no longer available, but all useful technical information contained in it will appear in this magazine in the near future).

Software

Update



KEY TO RATINGS;

ARCADE and ANIMATED ADVENTURES

- GAME CONTENT - Variety of actions / screens
- PLAYABILITY - Ease of use, addictive quality
- GRAPHICS - Quality and use of graphics related to machine
- SOUND - Use of stereo and tune / noise originality.
- VALUE FOR MONEY - Overall impression when compared with price.

ADVENTURES

- GAME CONTENT - Design of plot / background. Puzzle ingenuity.
- PRESENTATION - Atmosphere, graphics (if any), text / screen layout.
- INTERACTION - Parser quality, editing facilities
- VALUE FOR MONEY - Overall impression when compared with price.

PERCENTAGES

- 0 - 25 - Yuk, Bleah !
- 26 - 50 - Bad to Mediocre
- 51 - 75 - Average to Good
- 75 - 100 - Excellent to completely Brilliant

Name : Screen Utilities
 Producer : Boxsoft
 Category : Utility
 Price : £5.95

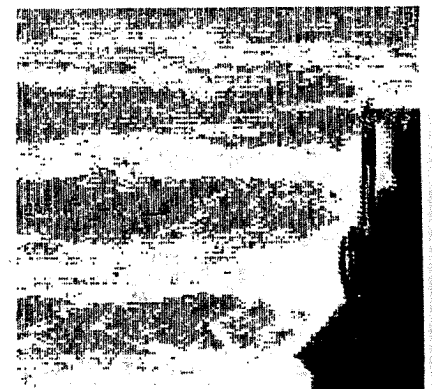
If you went to the P.C.W. Show last year, you probably freaked out over the User Group's artistic talent (read, Mark !), by which I mean those great pictures of deckchairs and horses and things. Well I've a confession to make, we didn't draw them - Tim digitised them. BUT you're probably still pretty amazed at the speed they were loaded from disc; after all the listing in Issue 1 wasn't that fast, and how did we do those nifty printer dumps ?

Well, the software used by us is now available to you, courtesy of Boxsoft (surprise, surprise !) Screen Utilities consists of two programs SCR_SLC, (standing for SCReen Save, Load Copy), and SCR_DUMP, (standing for... wait for it... SCReen DUMP, and called SCR_COPY on the cassette label for some inexplicable reason).

Both programs load as system extensions, so that they sit in the computer out of the way and are called using the EXT command, which is much more convenient than having to include the routines at the beginning of every graphics program that you write. They can also be used in immediate mode in the same way that you would use HELP, for example.

SCR_SLC is very powerful and very quick. Not only does it save the

Boxsoft
 P·R·O·G·R·A·M·S



SCREEN UTILITIES

palette colours along with the picture data, but it will open a video channel of the right type to load the picture and automatically display it if asked. SCR_DUMP dumps sideways on, so that you don't lose the edge of your masterpiece, and the output can be inverted so that it appears the same as the image on the T.V. screen. It can also be set up for just about any printer - except daisywheels !

Both programs have many more features which I don't have space to go into here, most of which were listed in the Boxsoft advert last Issue. All in all if you're the sort of person who does any graphic work these utilities are a useful addition to your software tools, and I can certainly recommend them at £5.95 for the pair (where's that tenner Tim ?).

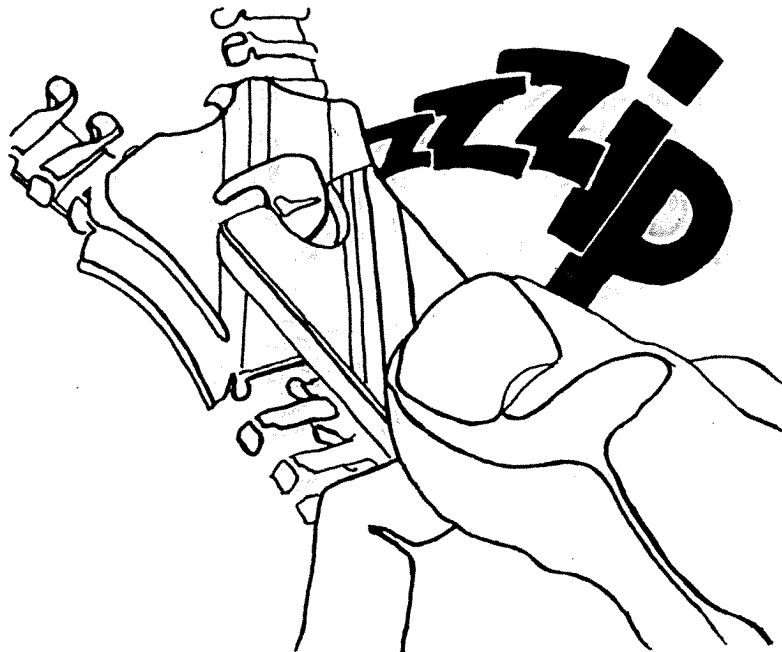
≡ Software Update

Name : Zzip
Producer : Boxsoft
Category : Utility
Price : £17.95

Anyone who does any programming on the Enterprise will know how slow IS-Basic is - a snail with an abacus is probably quicker! Well, help is now at hand in the form of Zzip, a Basic compiler available from Boxsoft, (plug,plug) and written by Peter Miner.

This is the second compiler we have seen on the Enterprise, the first, from Aztec (remember them?) never got past the development stage and only offered a two to five times speed increase. This one gives up to a FIFTY times speed increase on some things and gives, on average, a twelvefold speed increase.

One of the reasons why Zzip is so quick is that it is an integer compiler. This means that any programs which use floating point arithmetic are unlikely to work properly after being compiled without a certain amount of fudging. Normally, this would also mean that trigonometric functions (SIN, COS etc.) wouldn't work, as they return values between -1 and 1; the same applies to LOG. However Zzip has the unusual feature of scaling the results returned by these functions by a factor of 1000, so allowing them to be used in compiled programs with a little work. All of this is explained in detail in the user manual.



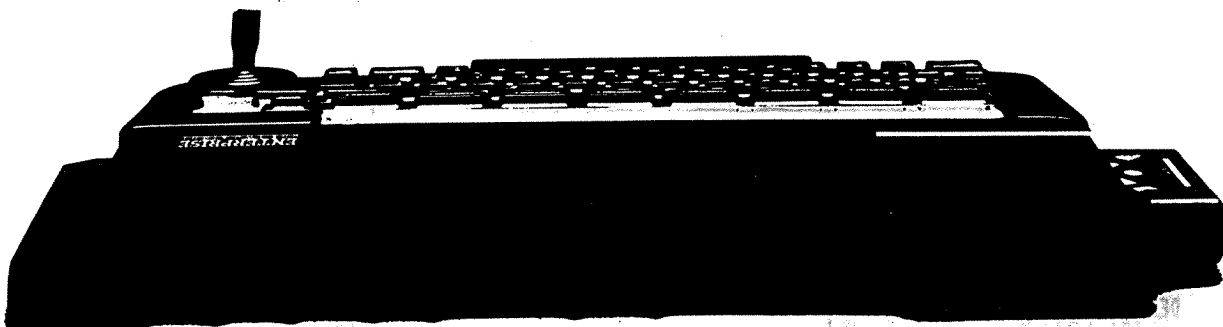
In fact, Zzip seems to handle nearly all Basic functions and commands, although some need to be used with care, and there won't be many programs

Actually using Zzip is simplicity itself. Just load Zzip and tell it the name of the program you want to compile. Zzip will then ask you what to call the compiled program and gets on with the job of compiling your Basic program. It then makes a number of passes through the program and points out any parts that can't be compiled so you can go back to the Basic program and alter them. You will almost certainly find that there is something in your program that will need changing, although it will usually only be something minor, and

the manual gives plenty of advice on more difficult cases. Once the program has been altered to Zzip's satisfaction the compiled program will be saved in two parts - the main program and a loader. All you need to do to run the compiled version of your Basic masterpiece is load and run the loader program and sit back and be amazed by the speed increases.

In conclusion, I would say that Zzip is a good buy, and the hassle of going through a Basic program checking for things it can't handle is more than compensated for by the speed increases obtained, presuming of course that you program in Basic.

Dave Race



Advanced Programming

This article is intended for the machine-code programmer. If you do not understand machine-code, do not expect to understand much of this!

If you wish to write programs using an assembler, they should be assembled as user-relocatable modules. Use ENT \$ to mark the code that initialises and defines all your extensions.

Intelligent Software designed BASIC in such a way that a programmer can add to the language very easily, without having to re-write the whole thing. It is possible to add both commands and functions that can have their own syntax and yet use the normal facilities of BASIC to interpret.

First, I will explain how to add your own function. This is possible in BASIC using the DEF command, but this method will mean that your function will not disappear when you alter a BASIC program or run it. Your function will behave very similarly to an in-built one.

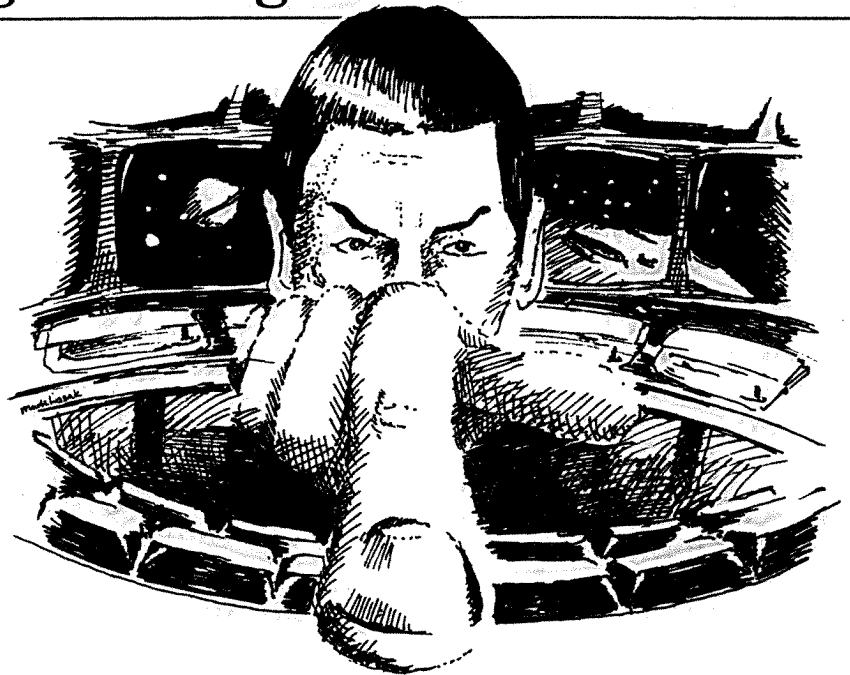
To add a function you should use a CALL to a BASIC subroutine, which will do all the complicated work for you. Here is how to use it :

```
LD HL,function_data
RST 16
DEFB 128,0
```

The 'RST 16' bit is special call like the 'RST 48' of EXOS. It looks ahead to the 'DEFB' and all the numbers after it. Each number represents a specific call. The list of numbers should end with a '0'.

Before calling, HL should point to a block of information about the function, in the following form:

```
fn_data DEFW 0
        DEFB 13
        DEFB name_length
        DEFM "name"
```



Basic Extensions

```
DEFW execute_address
DEFB execute_segment
```

'name_length' is the number of characters in the name of the function

'name' is the name of the function in upper-case letters.

'execute_address' is the address to be called to carry out the work of the function

'execute_segment' is the segment number that the code is in. Just keep it as zero if your code is in page-zero, which it probably will be.

The function will have to get arguments and return them using more 'RST 16' calls. Here is an explanation of the most important ones.

To return a value, first put it in the HL register and then :

```
RST 16
DEFB 2,0
```

e.g.

The built-in WHITE function consists solely of :

```
WHITE LD HL,255 ;HL=white code
      RST 16
      DEFB 2,0 ;Return this value
      RET ;Return
```

To skip over an opening - bracket, use :

```
LD A,8
RST 16
DEFB 34,0
```

To skip over the closing - bracket, use :

```
LD A,9
RST 16
DEFB 34,0
```

To skip over the comma, use :

```
LD A,12
RST 16
DEFB 34,0
```

These three are essential for functions with arguments. Do not forget to include them. If the

Advanced Programming

relevant symbol is missing, then a 'Not understood' error will be caused.

To fetch an argument, use :

```
RST 16
DEFB 35,11,0
```

This returns the argument's value in HL.

Here is an example function that doubles a number.

```
DEFINE ENT $
LD HL, FN_DATA
RST 16
DEFB 128,0
RET
```

```
FN_DATA DEFW 0
DEFB 12
DEFB 6
DEFM "DOUBLE"
DEFW DOUBLE
DEFB 0
```

```
DOUBLE LD A,8 \ Skip
RST 16 > opening
DEFB 34,0 / bracket

RST 16 \ Fetch
DEFB 35,11,0 / parameter
```

```
ADD HL,HL > double it
```

```
RST 16 \ Return
DEFB 2,0 / it
```

```
LD A,9 \ Skip
RST 16 > closing
DEFB 34,0 / bracket
```

```
RET > Return
```

Because 'RST 16' accepts a list of calls after it, it is possible to replace the first five lines of DOUBLE with :

```
LD A,8
RST 16
DEFB 34,35,11,0
```

It is easy to think that it ought to be :

```
LD A,8
RST 16
DEFB 34,0,35,11,0
```

There are two zeros in the original, so why shouldn't there be two zeros when the program is shortened version?

The answer is that the zero is the marker of the end of the list of calls and so should not appear until the end.

Now for string functions.

String functions are defined in exactly the same way as numeric ones, except that the 'DEFB 12' must be replaced with 'DEFB 13', to signify it as a string function. It must also, of course, have a \$ sign on the end of its name. This should be included in the name and length byte.

To evaluate a string expression as an argument, use :

```
RST 16
DEFB 36,0
```

This will put a string on the 'BASIC parameter stack'. This is used for parameters, blocks calculations etc. and is very important.

To find the address of this string, use

```
LD HL,(552)
INC HL
```

HL will now point to its length byte. After that comes the actual string.

When the function ends, this string should have been deleted from the stack, meaning that the value at address 552 must point to the next byte above the end of the string.

Now for how to return a string result.

```
LD HL,(552)
DEC HL
```

Then put the last character in the string at address HL. Decrement HL and put the second-from last character in the byte at HL and so on until the end (or beginning!) of the string.

The length byte should come next and the byte below should hold the length byte+2. Then store the address of this last byte at address 552.

So, to return the character B, use :

```
LD HL,(552)
DEC HL
LD (HL),B
DEC HL
LD (HL),1
DEC HL
LD (HL),3
LD (552),HL
```

Now for a complete function.

```
FN_DATA DEFW 0
DEFB 13
DEFB 4
DEFM "KEY$"
DEFW KEY
DEFB 0
```

```
KEY LD A,105 \ Read character
RST 48 > from keyboard
DEFB 5 / channel
```

```
RST 24 > See to any errors
```

```
LD HL,(552) \ Return
DEC HL \ character
LD (HL),B \ as
DEC HL \ result
LD (HL),1 /from
DEC HL /function
LD (HL),3 /
LD (552),HL /
RET
```

Which returns a key press by reading

Advanced Programming

from the keyboard channel.

Commands are more complicated.

Here is the code for a command that writes a string to the status line.

This bit defines the command

```
DEFINE LD HL,(562)
DEFLOOP LD A,(HL)
        INC HL
        OR (HL)
        DEC HL
        JR Z,EXTEND
        LD A,(HL)
        INC HL
        LD H,(HL)
        LD L,A
        JR DEFLOOP
```

```
EXTEND LD DE,COMTAB
        LD (HL),E
        INC HL
        LD (HL),D
        RET
```

This is the first table. It can have information on as many commands as you like. Here it is only one, though.

```
COMTAB DEFW 0
        DEFB 1 ; number of commands
The following two lines must be
repeated for every command.
        DEFW STATUS$ ;pointer to
                ;data about
                ;the command
        DEFB 4 ; ignored
```

This is the data for each command.

```
STATUS$ DEFW STATUS ;execution address
        DEFW CHECK
        DEFB 83 ;command type
                ;see below
        DEFB 6 ;name length
        DEFM "STATUS" ;name in upper
                ;case
```

This bit is required, but is pointless to explain. just include it with every extension program for commands.

```
CHECK RST 16
        DEFB 32,0
```

```
CHCK1 SCF
        JR C,CHCK2
        RST 16
        DEFB 33,0
CHCK2 LD A,(516)
        CP 16
        RET Z
        CP 2
        RET C
        CP 8
        JR Z,CHCK1
        JR CHECK
```

This is the code that does the actual work.

```
STATUS RST 16 \fetch string
        DEFB 36,0 /parameter
        LD HL,(552)
        INC HL
        LD A,(HL) \ Check
        CP 32 \ its
        JR Z,TBG > length
        OR A / and cause in
        JR Z,TBG / error if too big
        LD C,A \ Put length
        LD B,0 / in BC
        IN A,(178)
        PUSH AF
        LD A,255 \ Find address
        OUT (178),A > of status
        LD DE,(0BFF6h)/ line
        INC DE
        INC DE \ Move over
        INC DE > key lock
        INC DE / section
        INC DE
        INC DE
        LDIR > write into memory
        LD (552),HL > Reset 552
        POP AF \ Restore
        OUT (178),A / segments
        RET
```

```
TBG LD HL,1106 \ Cause string
        RST 32 / too big error
```

As you may have noticed, functions and commands get their arguments in exactly the same way. So commands fetch numbers using the same call as functions.

The 'DEFB 83' bit of the command's



data section is the type byte. It is the normal type byte, meaning that the function can be used in programs, immediate mode, multi-statement lines and that it should be tokenised. All commands should be tokenised. The only exceptions are !, DATA and REM where the information afterwards will not be evaluated and should be kept as text.

The type byte is made up as follows :

bit	meaning if set
0	allowed in immediate mode
1	allowed in program
2	end of block
3	start of block
4	allowed in multi-statement line
5	alters program
6	tokenise
7	not used - keep as zero

Bits 2 and 3 are used by LIST for indentation purposes. If they are both set, then indentation like CASE or ELSE is given.

Call number 34 you have seen before. It compares the character that BASIC's internal pointers are pointing to with the value in the A register. If they differ, then an error is caused. Here are all the values A can have :

0	End of line
1	! -comment marker
3	£ 10 * 17 ; 23 >=
4	\$ 11 + 18 < 25 \
6	& 12 , 19 = 27 ^
7	' 13 - 20 >

Advanced Programming

8 (15 / 21 <)
9) 16 : 22 < =

It is also possible to find out what character is at that position with 'LD A,(516)'. A will have the appropriate value from the table above. To skip over any character or word, use call number 32.

To find out what the pointers are pointing after this symble, use 'LD A,(514)'. A will have a value from the following table :

0 symble
32 ordinary word or numeric variable
64 string variable
96 command
128 string enclosed in quotes
160 line number after GOTO etc.
192 ordinary number

To find out the length of a word, use 'LD A,(515)'. The address of the text of this word can be found out with 'LD HL,(839)'.

Here is a summary of the most important calls :

0 End of list
1 Cause error.Can also use RST 32
2 Put integer HL on BASIC parameter stack
11 Get HL off parameter stack
32 Skip over any word
33 Skip over expression
34 Skip over word as long as it is prefixed with the symble A
35 Fetch numeric parameter
36 Fetch string parameter
128 Add function HL
160 Skip closing bracket
186 List HL to DE

Here is a summary of the more important addresses in page zero.

16 RST 16 call address
32 cause error call
514 word type
515 word length
516 delimiting symble

518 command exit condition
if zero when the command exits with RET, then the program will jump to the address stored at 534.
521 present channel number
522 program number
532 pointer to symble in program
534 pointer to beginning of line if address 518 is two, then this is used as the destination address for jumps
538 start of program memory
540 code pointer
552 BASIC parameter stack
562 pointer to command table
572 error code
574 CONTINUE address
576 address of present HANDLER program
584 length of input
585-838 input
839 pointer to present word

Here is the format of a program line :

length byte (0 for end of program)
2-byte line number (1-9999)
1-byte indentation value (used by LIST) etc.
96 (type byte for command)
command number (0-255)

After that comes the command's parameters. First comes a byte giving the type. The top three bits are put in 514 and are used to find out what the rest of the data means.

symble (514=0)
Bottom five bits (in 515) is the symble's number (see above).

word (32)
Bottom five bits are word's length. Word comes after in upper case. (839) points to this word

string (64)
As ordinary word

command (96)
Number afterwards gives command number.

quotes (128)

Next byte gives length. Bytes after that are the letters between the quotes.

line number (160)

Two bytes afterwards give line number.

number (192)

If bottom five bits equal 2, then the number is an integer, given as a line number. Otherwise, the number is floating-point in the following form :

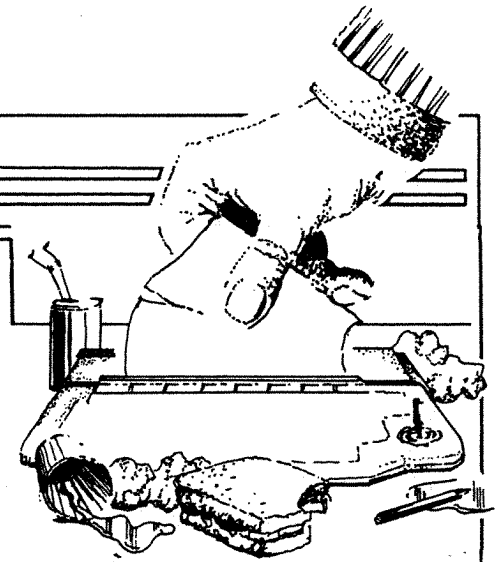
five bytes BCD mantissa
one byte exponent
-bit 7 is the sign bit
-64 is an exponent of zero, below that are negative exponents, and above that, the exponent is positive.

The last byte of the line is zero. If the first command on the line has its tokenisation flag reset, then the above rules do not apply. The whole line is just text ending with a zero. It will, though, have the length byte, line number and command number stored as above.

If you are confused by this, or would like to know more, you can contact me. My address is :

Glebe House
Coalport Road
Madeley
Telford
TF75DS

Home Produce



Once the database is running, a menu is presented to the user giving three options :

1. CREATE A FILE

This (as the name suggests) allows the user to create a file. He will be asked how many fields will be required (up to a maximum of 10). If the following example is considered, the term "field" and others may become clearer.

Name:	Fred Bloggs	= Field 1 ;	
Number + Road:	13, Nowhere Street	= Field 2 ;	
Town:	Villestown	= Field 3 ;	= 1 record
County:	Yorkshire	= Field 4 ;	
Post Code:	AB1 C23	= Field 5 ;	

Each separate piece of information is called a field, so in this example the Name is the first and the Town the third. Altogether, these five fields form one record. For example, it could be thought of as one card in a card index. Once a number of records have been entered, these make up a file (think of a draw full of index cards). The more fields that are used, the smaller the number of records which can be stored before the memory is used up. After the titles have been entered (up to a maximum of 20 characters) the main database routine is started.

2. LOAD A FILE FROM TAPE

This allows data to be loaded into the database from tape after being saved previously. Just enter the filename when asked (or just type ENTER to load the first file found on the tape). The data will then be loaded into the database and all the functions used on it as normal.

3. EXIT PROGRAM

As the title suggests, use this to exit the program.

Once in the main database routine, three separate windows are created; the top one displaying instructions and prompts, the second displays all the commands available to the user and the third (and largest) showing all the entered information. Another function of the top window is to show the current record number, with the total number of records in the file and the total available (depends on the number of fields used - maximum 1400 with 1 field). The main database routine has a number of commands which do need some quite detailed explanation.

COMMAND "A"

Adds new records to the file. This command is used when starting a new file or when extra records are to be

added. Only 20 characters may be used on each field. If more are used, they are automatically added to the beginning of the next field. If the erase key is pressed the whole field is deleted, not just one character. To stop adding records, press ESC.

COMMAND "D"

Delete record currently shown on the screen. You will be asked to confirm this command to prevent accidental erasure. If there are a lot of records, it may take some time to rearrange them.

COMMAND "O" Sort

Order or Sort. The sort in this program is a "shell sort", which is very quick when compared with others such as the "bubble sort". Even though it is a powerful sort, it will take a long time with lots of records to arrange, so be prepared for a long wait. After choosing which field the sort will use, you will be asked how many characters are to be considered. I have found that 2 characters are the most reliable for the data I store in the program, but this can vary and depends on the contents of the fields. If, for example, there are no spaces in the first 4 characters of the field being used then 4 would give a more accurate sort. The reasoning for this is that spaces are also included in the routine and can give unexpected results. Finally, numbers are not sorted numerically, but according to their ASCII codes, so 113 would come before 23 ! Experimentation is really the best solution with this function.

COMMAND "S" Find

Search through the file. You are asked which field the search is to use. If you want it to be over all the fields then type "A". The search string is required next; enter the words and letters which the computer is looking for. Be careful in the use of upper and lower case letters, as they are not the same. All records containing the search string will be displayed (up to a maximum of 50), one at a time - just press any key to go on to the next, or ESC (when prompted) to escape.

Home Produce

COMMAND "L"

List all records from first to last. Press ESC to stop the listing.

COMMAND "F" *Next*

Move forward by one record.

COMMAND "B" *Previous*

Move backwards by one record.

COMMAND "C"

Copy a page to the printer, or alternatively list the whole file onto paper (in a compacted form). Follow the prompts when given.

COMMAND "E" *Quit*

```

100 PROGRAM "DATABASE.Bas"
110 REM
120 REM *****
130 REM ***
140 REM *** main program ***
150 REM ***
160 REM *****
170 REM
180 SET FKEY 1 ""
190 SET FKEY 2 ""
200 SET FKEY 3 ""
210 SET FKEY 4 ""
220 SET FKEY 5 ""
230 SET FKEY 6 ""
240 SET FKEY 7 ""
250 SET FKEY 8 ""
260 SET INTERRUPT STOP OFF
270 SET KEY CLICK ON
280 SET SPEAKER ON
290 LET FIELDS,NUM_REC,CUR_REC=0
300 LET CUR_FLD=1
310 LET ADD,NUM=0
320 DO
330 TEXT
340 PRINT AT 1,14:"Database"
350 PRINT AT 2,14:"-----"
360 PRINT AT 4,7:"By Richard Hudson 1986"
370 PRINT AT 10,1:"Select from the following..."
380 PRINT :PRINT "1}Create a new file."
390 PRINT :PRINT "2}Load a file from tape."
400 PRINT :PRINT "3}Exit program."
410 LOOK F105:B
420 PRINT :PRINT CHR$(B); " ";
430 !
440 !
450 LOOP UNTIL B=49 OR B=50 OR B=51
460 WAIT 1
470 IF B=49 THEN
480 CALL ASK
490 !
500 ! set up data base arrays
510 !
520 LET MAXMEM=INT(1400/FIELDS)

```

Exit from the main database into the initial menu.

COMMAND "U"

Update records. The function operates on the record currently displayed. Each field in turn can be altered, but if any alterations are to be made the ERASE KEY MUST BE PRESSED - otherwise the new version will be added to the old, even though at first it will appear to overwrite. To leave a field unaltered, just press ENTER.

COMMAND "K" *Save*

Save a file to tape. Data can be saved under a given name (letters and underline characters only) of limited length. It is saved in two parts, the first saving the data needed to create the arrays, ie. the number of records/fields and the field titles. The second part actually saves the input data in the file. The two parts are required so that arrays can be formed during loading.

```

530 STRING INFO$(1 TO MAXMEM,1 TO FIELDS)*20
540 NUMERIC Y(FIELDS)
550 STRING TITLE$(FIELDS)*18
560 DIM SER(50)
570 CALL SET_UP2
580 CALL SET_UP1
590 CALL DBASE
600 ELSE IF B=50 THEN
610 !
620 INPUT PROMPT "Enter file name:-":F$
630 OPEN F106:"tape:"&F$ & ".TXT" access input
640 INPUT F106:FIELDS
650 INPUT F106:NUM_REC
660 INPUT F106:CUR_FLD
670 CLOSE F106
680 LET MAXMEM=INT(1400/FIELDS)
690 STRING INFO$(1 TO(MAXMEM),1 TO FIELDS)*20
700 STRING TITLE$(FIELDS)*18
710 DIM SER(50)
720 OPEN F106:"tape:"&F$
730 FOR F=1 TO FIELDS
740 INPUT F106:TITLE$(F)
750 NEXT F
760 FOR R=1 TO NUM_REC
770 FOR F=1 TO FIELDS
780 INPUT F106:INFO$(R,F)
790 NEXT F
800 NEXT R
810 CLOSE F106
820 LET CUR_REC=NUM_REC
830 CALL SET_UP1
840 CALL DBASE
850 ELSE IF B=51 THEN
860 CLEAR FKEYS
870 SET INTERRUPT STOP ON
880 END
890 END IF
900 GOTO 320
910 REM
920 REM *****
930 REM *** Functions ***
940 REM *****
950 REM

```

Home Produce

```

960 DEF ASK
970 CLEAR SCREEN
980 PRINT AT 1,14:"Database"
990 PRINT AT 2,14:"-----"
1000 INPUT AT 5,1,IF MISSING 1000,PROMPT "How many
      fields(Max 10) ?":FIELDS
1010 PRINT:PRINT "Thankyou. Please Wait a moment..."
1020 END DEF
1030 DEF SET_UP1
1040 ! open f1,f2,f3
1050 SET VIDEO MODE 0 ! 40-coloumn text
1060 SET VIDEO X 40
1070 SET VIDEO Y 2
1080 OPEN f1:"video:"
1090 SET f1:PALETTE 8,YELLOW
1100 SET VIDEO Y 3
1110 SET VIDEO MODE 2
1120 OPEN f2:"video:"
1130 SET f2:PALETTE 0,GREEN
1140 SET VIDEO MODE 0 ! 40 coloumn text
1150 SET VIDEO Y 21
1160 OPEN f3:"video:"
1170 SET f3:PALETTE 32,YELLOW,0
1180 DISPLAY f3:AT 6 FROM 1 TO 21
1190 DISPLAY f1:AT 1 FROM 1 TO 2
1200 DISPLAY f2:AT 3 FROM 1 TO 3
1210 PRINT f2:"F=search:", "L=list:", "C=copy page to
      printer:", "A=forwards:", "P=backwards:"
1220 PRINT f2:"O=order(sort):", "U=update record:",
      "A=add new records:", "D=delete record:", "S=save
      file to tape:", "E=Exit to main menu:";
1230 PING
1240 END DEF
1250 REM
1260 REM *****
1270 REM *** Data base routine ***
1280 REM *****
1290 REM
1300 DEF DBASE
1310 CALL DISP_FLDS
1320 DO
1330 CLEAR f1
1340 PRINT f1,AT 2,10:"Enter command:";
1350 IF NUM_REC>0 THEN
1360 CALL DISP_REC(CUR_REC)
1370 PRINT f1,AT 1,1:"Record:";CUR_REC;" of";
      NUM_REC,"Max=";MAXMEM;
1380 END IF
1390 LOOK f105:KEY
1400 IF KEY<31 OR KEY>159 THEN 1390
1410 SELECT CASE KEY
1420 CASE 83,115 / 70, FIN>
1430 CALL SEARCH:LET KEY=0
1440 CASE 76,108
1450 CALL LIST:LET KEY=0
1460 CASE 73,105
1470 CALL INFO:LET KEY=0
1480 CASE 67,99
1490 CALL COPY:LET KEY=0
1500 CASE 70,102 / 78, NEXT
1510 CALL FORWARD:LET KEY=0
1520 CASE 66,98 / 80, PREVIOUS
1530 CALL BACKWARD:LET KEY=0
1540 CASE 79,111
1550 CALL SORT:LET KEY=0
1560 CASE 85,117
1570 CALL UPDATE:LET KEY=0
1580 CASE 65,97
1590 CALL ADD RECORDS:LET KEY=0:LET ADD=ADD+1
1600 CASE 68,100

```

```

1610 CALL DELETE_REC:LET KEY=0
1620 CASE 75,107 / 83
1630 CALL SAVE:LET KEY=0
1640 CASE 69,101
1650 CLEAR f1
1660 PRINT f1:"Do you really want to exit ?(Y/N)
      all the variables will be destroyed !";
1670 LOOK f105:KEY2
1680 IF KEY2=89 OR KEY2=121 THEN EXIT DO
1690 CLEAR f1
1700 CASE ELSE
1710 GOTO 1390 ! look for key again
1720 END SELECT
1730 LOOP
1740 CLEAR f1
1750 RUN
1760 END DEF
1770 REM *****
1780 DEF ADD RECORDS
1790 LET CUR_REC=NUM_REC
1800 CLEAR f1
1810 CLEAR f3
1820 CALL DISP_FLDS
1830 IF NUM_REC=MAXMEM THEN
1840 CLEAR f1
1850 PRINT f1:"File full....."
1860 WAIT 5
1870 GOTO 2200
1880 END IF
1890 IF NUM_REC>0 THEN LET CUR_REC=CUR_REC+1:LET NUM_
      REC=NUM_REC+1
1900 STRING BUFFER$
1910 LET BUFFER$=""
1920 PRINT f1,AT 2,1:"Type Esc to stop entering";
1930 DO
1940 FOR F=1 TO FIELDS
1950 LET X=(F*2)-1
1960 FOR L=21 TO 40
1970 !
1980 PRINT f3,AT X,L:""
1990 LOOK f105:KEY1
2000 IF KEY1=13 THEN EXIT FOR
2010 IF KEY1=27 THEN 2200
2020 IF KEY1=164 THEN
2030 !
2040 PRINT f3,AT X,21:""
2050 LET L=21
2060 LET BUFFER$=""
2070 GOTO 1980
2080 END IF
2090 PRINT f3,AT X,L:CHR$(KEY1)
2100 LET BUFFER$=BUFFER$&CHR$(KEY1)
2110 NEXT L
2120 LET INFO$(CUR_REC,F)=BUFFER$
2130 LET BUFFER$=""
2140 NEXT F
2150 LET NUM_REC=NUM_REC+1
2160 LET CUR_REC=CUR_REC+1
2170 CLEAR f3
2180 CALL DISP_FLDS
2190 LOOP
2200 LET NUM_REC=NUM_REC-1:LET CUR_REC=CUR_REC-1
2210 CLEAR f1
2220 END DEF
2230 DEF LIST
2240 CLEAR f1
2250 PRINT f1,AT 1,5:"Press Esc to stop listing"
2260 PRINT f1,AT 2,7:"use HOLD if required";
2270 FOR CUR_REC=1 TO NUM_REC-1
2280 LET Z=0
2290 CALL DISP_REC(CUR_REC)

```

if len(Buffer) = then ping.

Home Produce

```

2300 IF Z=1 THEN 2350
2310 FOR DELAY=1 TO 50
2320 IF INKEY#=CHR$(27) THEN 2350
2330 NEXT DELAY
2340 NEXT CUR_REC
2350 CLEAR F1
2360 END DEF
2370 DEF DISP_REC(REF CUR_REC)
2380 FOR F1=1 TO FIELDS
2390 PRINT F3,AT(F1*2)-1,21:" " ! 20 spaces
2400 PRINT F3,AT(F1*2)-1,21:INFO$(CUR_REC,F1)
2410 IF INKEY#=CHR$(27) THEN
2420 LET Z=1
2430 GOTO 2460
2440 END IF
2450 NEXT F1
2460 END DEF
2470 DEF FORWARD — NEXT
2480 IF CUR_REC=NUM_REC THEN 2500
2490 LET CUR_REC=CUR_REC+1
2500 END DEF
2510 DEF BACKWARD — PREVIOUS
2520 IF CUR_REC=1 THEN 2540
2530 LET CUR_REC=CUR_REC-1
2540 END DEF
2550 DEF SAVE
2560 LET FILE$=""
2570 CLEAR F1
2580 PRINT F1,AT 1,1:"Input file name (max 12 chars.)"
2590 FOR F=1 TO 12
2600 LOOK F105:A
2610 IF A=13 THEN EXIT FOR
2620 IF A<65 OR A>122 THEN 2600
2630 LET FILE$=FILE$&CHR$(A)
2640 PRINT F1,AT 2,1:FILE$;
2650 NEXT F
2660 CLEAR F1
2670 PRINT F1,AT 1,1:"Is ";FILE$;" okay ?"
2680 PRINT F1,AT 2,1:"type Y or N.";
2690 LOOK F105:B
2700 IF B=89 OR B=121 THEN 2720
2710 GOTO 2560
2720 CLEAR F1
2730 LET FILE$=FILE$&".TXT"
2740 PRINT F1:"Start recorder, then press any key."
2750 SET REM ON
2760 LOOK F105:A
2770 OPEN F106:"Tape"&FILE$ ACCESS OUTPUT
2780 PRINT F106:FIELDS
2790 PRINT F106:NUM_REC
2800 PRINT F106:CUR_FLD
2810 CLOSE F106
2820 OPEN F106:"tape"&FILE$ ACCESS OUTPUT
2830 FOR F=1 TO FIELDS
2840 PRINT F106:TITLE$(F)
2850 NEXT F
2860 FOR R=1 TO NUM_REC
2870 FOR F=1 TO FIELDS
2880 PRINT F106:INFO$(R,F)
2890 NEXT F
2900 NEXT R
2910 CLOSE F106
2920 END DEF
2930 DEF SET_UP2
2940 PRINT AT 10,1:"Input the titles for each field:"
2950 FOR FLD=1 TO FIELDS
2960 PRINT AT 10+FLD,1:FLD;
2970 INPUT AT 10+FLD,4:TI$
2980 IF LEN(TI$)>=19 THEN 2940
2990 LET TITLE$(FLD)=TI$
3000 NEXT FLD

```

```

3010 INPUT AT 21,1,PROMPT "Are These Correct ?(y\n) ":R$
3020 IF UCASE$(R$(1))="Y" THEN 3080
3030 IF UCASE$(R$(1))="N" THEN
3040 CLEAR SCREEN
3050 GOTO 2940
3060 ELSE
3070 GOTO 3010
3080 END IF
3090 END DEF
3100 DEF DISP FLDS
3110 FOR F=F TO FIELDS
3120 PRINT F3,AT(F*2)-1,1:TITLE$(F);
3130 PRINT F3,AT(F*2)-1,20:""
3140 NEXT F
3150 END DEF
3160 DEF SEARCH — F1 ~>
3170 LET NUM=0
3180 CLEAR F1
3190 PRINT F1,AT 1,1:"Enter field to be used(0=10 &
A=all)"
3200 LOOK F105:S
3210 IF S>=49 AND S<=57 THEN
3220 LET C=VAL(CHR$(S)) ! for No. entered
3230 IF C=0 THEN LET C=10:PING
3240 ELSE IF S=97 OR S=65 THEN
3250 LET C=-1 ! to show blanket search
3260 ELSE
3270 GOTO 3180
3280 END IF
3290 IF C>FIELDS THEN 3180
3300 CLEAR F1
3310 PRINT F1,AT 1,1:"Enter search string...(max
19 chars.)"
3320 LET SEARCH$=""
3330 FOR F=1 TO 19
3340 LOOK F105:L
3350 IF L=164 THEN
3360 LET SEARCH$=""
3370 LET F=1
3380 GOTO 3300
3390 ELSE IF L=13 THEN
3400 GOTO 3470
3410 ELSE IF SEARCH#=CHR$(27) THEN
3420 GOTO 3920 ! exit routine
3430 END IF
3440 LET SEARCH$=SEARCH$&CHR$(L)
3450 PRINT F1,AT 2,F:CHR$(L);
3460 NEXT F
3470 CLEAR F1
3480 PRINT F1:"Is ";SEARCH$;" okay ? (y/n)"
3490 LOOK F105:L
3500 IF L=121 OR L=89 THEN 3520
3510 GOTO 3300
3520 CLEAR F1
3530 PRINT F1,AT 1,10:"Searching"
3540 IF C=-1 THEN
3550 FOR S=1 TO NUM_REC
3560 FOR S1=1 TO FIELDS
3570 IF POS(INFO$(S,S1),SEARCH$)>0 THEN
3580 LET SER(NUM)=S
3590 LET NUM=NUM+1
3600 GOTO 3630
3610 END IF
3620 NEXT S1
3630 NEXT S
3640 ELSE IF C>0 THEN
3650 FOR S=1 TO NUM_REC
3660 IF POS(INFO$(S,C),SEARCH$)>0 THEN
3670 LET SER(NUM)=S
3680 LET NUM=NUM+1

```

Home Produce

```

3690     END IF
3700     NEXT S
3710     END IF
3720     IF NUM=0 THEN
3730         CLEAR F1
3740         PRINT F1,AT 1,1:"No record found containing
           string."
3750         WAIT 3
3760         GOTO 3920
3770     ELSE
3780         CLEAR F1
3790         PRINT F1,AT 1,1:"There are";NUM;" records found."
3800         PRINT F1,AT 2,1:"Type any key to list them.";
3810         LOOK F105:A
3820         FOR F=0 TO NUM-1
3830             LET CUR_REC=SER(F)
3840             CALL DISP_REC(CUR_REC)
3850             CLEAR F1
3860             PRINT F1,AT 1,1:"Press any key to continue.Esc
           to exit."
3870             PRINT F1:"Record";SER(F);" of";NUM_REC;
3880             LOOK F105:A
3890             IF A=27 THEN EXIT FOR
3900         NEXT F
3910     END IF
3920 END DEF
3930 DEF UPDATE
3940     CLEAR F1
3950     PRINT F1,AT 1,1:"Press ENTER to leave field
           unaltered"
3960     PRINT F1:"Press Esc to exit:Erase to change field";
3970     FOR F=1 TO FIELDS
3980         LET BUFFER$=INFO$(CUR_REC,F)
3990         LET X=(F*2)-1
4000         FOR L=21 TO 40
4010             PRINT F3,AT X,L:""
4020             LOOK F105:A
4030             IF A=27 THEN 4160
4040             IF A=13 THEN EXIT FOR
4050             IF A=164 THEN
4060                 LET L=21
4070                 LET BUFFER$=""
4080                 PRINT F3,AT X,21:"";
4090                 GOTO 4010
4100             END IF
4110             LET BUFFER$=BUFFER$&CHR$(A)
4120             PRINT F3,AT X,L:CHR$(A)
4130         NEXT L
4140         LET INFO$(CUR_REC,F)=BUFFER$
4150     NEXT F
4160 END DEF
4170 DEF SORT
4240     STRING T$(1 TO FIELDS)*21
4250     CLEAR F1
4260     PRINT F1,AT 1,1:"Enter field to be used:"
4270     PRINT F1:"1 to 9 and 0=10";
4280     LOOK F105:S
4290     IF S>=48 AND S<=57 THEN
4300         LET SF=VAL(CHR$(S))
4310         IF SF=0 THEN LET SF=10
4320     ELSE
4330         GOTO 4250
4340     END IF
4350     IF SF>FIELDS THEN 4250
4360     CLEAR F1
4370     PRINT F1,AT 1,1:"How many characters to be
           considered?"
4380     PRINT F1,AT 2,1:"1 to 9 (2 is the most reliable)";
4390     LOOK F105:S
4400     IF S>=49 AND S<=57 THEN
4410         LET LE=VAL(CHR$(S))
4420         IF LE=0 THEN LET LE=10
4430     ELSE
4440         GOTO 4360
4450     END IF
4460     CLEAR F1
4470     IF NUM_REC<50 THEN PRINT F1,AT 1,17:"SORTING"
4480     IF NUM_REC>=50 THEN
4490         PRINT F1,AT 1,1:"You'd better have a cuppa while
           these"
4500         PRINT F1:"records are being sorted !";
4510     END IF
4520     LET N=NUM_REC
4530     LET L=(2*INT(LOG(N)/.693))-1
4540     LET L=INT(L/2)
4550     IF L<1 THEN 4740
4560     FOR J=1 TO L
4570         FOR K=J+L TO N STEP L
4580             LET I=K
4590             FOR R=1 TO FIELDS
4600                 LET T$(R)=INFO$(I,R)
4610             NEXT R
4620             IF UCASE$(INFO$(I-L,SF))(1:LE)<=UCASE$(T$(SF)
           (1:LE)) THEN 4680
4630             FOR R=1 TO FIELDS
4640                 LET INFO$(I,R)=INFO$(I-L,R)
4650             NEXT R
4660             LET I=I-L
4670             IF I>L THEN 4620
4680             FOR R=1 TO FIELDS
4690                 LET INFO$(I,R)=T$(R)
4700             NEXT R
4710         NEXT K
4720     NEXT J
4730     GOTO 4540
4740     LET CUR_REC=1
4750     IF NUM_REC>10 THEN
4760         FOR F=1 TO 3
4830     PRINT F1,AT 1,1:"Are you sure you want to delete
           this file ? (Y/N) ";
4840     LOOK F105:REPLY
4850     IF UCASE$(CHR$(REPLY))<>"Y" THEN 5060
4860     PRINT F1:"ok";
4870     IF CUR_REC=NUM_REC THEN
4880         FOR F=1 TO FIELDS
4890             LET INFO$(CUR_REC,F)=""
4900         NEXT F
4910         LET CUR_REC=CUR_REC-1
4920     ELSE
4930         CLEAR F1
4940         PRINT F1,AT 1,1:"Please wait while files are moved
           down one place";
4950         ! Move other files down one place
4960         FOR M=CUR_REC TO NUM_REC-1
4970             FOR F=1 TO FIELDS
4980                 LET INFO$(M,F)=INFO$(M+1,F)
4990             NEXT F
5000         NEXT M
5010         FOR F=1 TO FIELDS ! scrub last record
5020             LET INFO$(NUM_REC,F)=""
5030         NEXT F
5040     END IF
5050     LET NUM_REC=NUM_REC-1
5060 END DEF
5070 DEF COPY
5080     CLEAR F1
5090     PRINT F1,AT 1,1:"Type 1 to copy page or 2 to list
           all data Esc to exit.";
5100     LOOK F105:REPLY
5110     IF REPLY=27 THEN 5220
5120     IF REPLY=49 THEN COPY FROM F3 ! to F104
5130     IF REPLY=50 THEN
5140         FOR R=1 TO NUM_REC
5150             LPRINT "Record ";R
5160             FOR F=1 TO FIELDS
5170                 LPRINT
5180                 LPRINT TITLE$(F);";";TAB(20);INFO$(R,F)
5190             NEXT F
5200         NEXT R
5210     END IF
5220 END DEF

```

