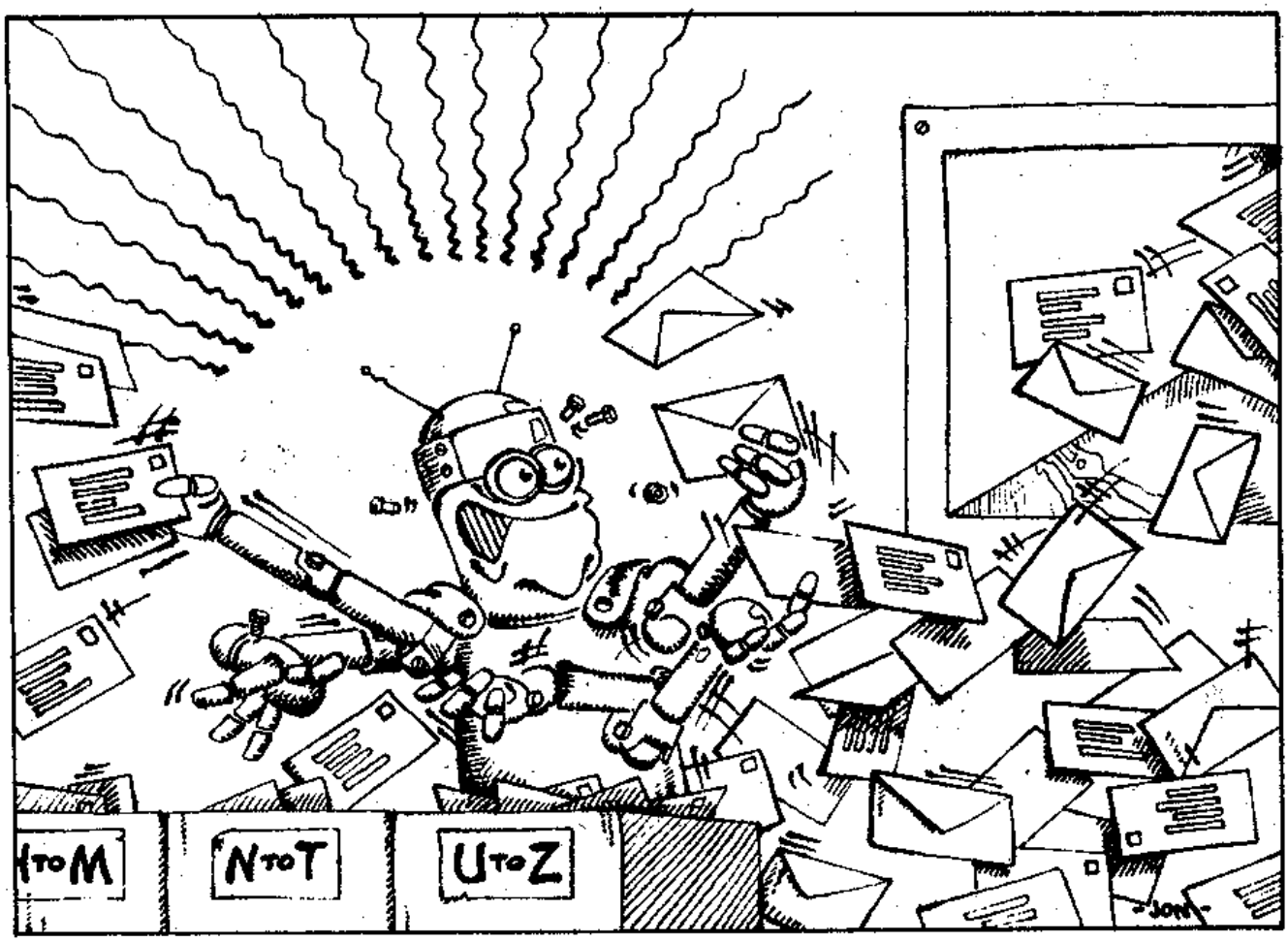
Vol 2 - No 10. June 1989.

FORMAT

THE MONTHLY MAGAZINE FOR SPECTRUM, DISCiPLE & PLUS D USERS



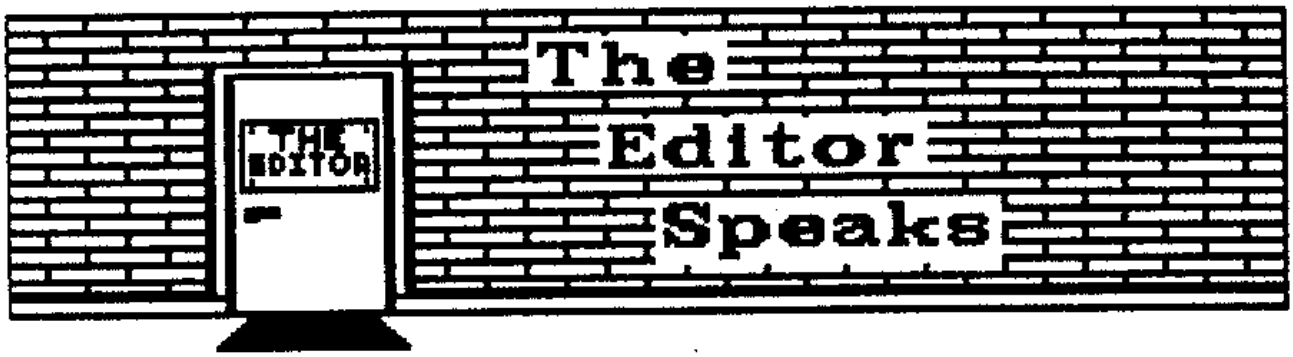
SORT Things Out With FORMAT

Vol 2 No 10. CONTENTS June 1989.

THIS MONTHS ADVERTISERS:-

FORMAT is published by **INDUG.** 34 Bourton Road, Gloucester, GL4 0LE, England. Telephone 0452-412572. **DISCiPLE** and **PLUS D** are trade marks of MILES GORDON TECHNOLOGY LTD., Lakeside Technology Park, Phoenix Way, Swansea, South Wales, SA7 9EH. Telephone 0792-791100.

Printed by O.S.LITHO. Gloucester. Tel:- (0452) 23198.



Another month and lots to interest you in this issue, look out for S.O.S. the first game to be printed in FORMAT and well worth typing in even if you dont normally play games.

Also some good news for recent subscribers. The early issues of FORMAT, some of which have been out of print for many months, are now available again. I had planned to do a compilation - Best Of FORMAT Vol 1 - when all the first years issues were out of print. But the work involved (as some of the text files no longer exist) could not have been entertained for some time to come. So "by popular demand" as they say, I have arranged for reprints of early issues which will be available from the end of June (see BACK ISSUE SERVICE).

Since FORMAT was launched in August 1987 its readership has grown steadily to the point where we now have nearly 1500 subscribers. We are the largest Spectrum group in the world, but we could be even bigger... There are, I believe, around 1.5 million Spectrum users in the UK alone. Of these, I would estimate that 10% are 'SERIOUS' users - people who use their machines for programming, wordprocessing and other non-game playing activities. Doing the sums that means around 150 thousand people are using their Spectrums for the same sort of things as most FORMAT readers. So why dont we have a circulation of 150,000 ?

Until recently the only way we publicised FORMAT was through MGT and the introductory issue they sent out to all new PLUS D users on our behalf. In the Dec'88 and Feb'89 issues of the Sinclair User magazine we ran small adverts to attract new readers. These adverts produced only a handful of replies and were not worth the effort. In the Nov'88 issue of FORMAT we tried to encourage you to introduce new readers in return for a two month extention to your membership. This brought a much better response with over thirty introductions. But even this is a drop in the ocean when you consider 150,000 as a possible target.

It is obvious that the good news of FORMAT is spread better, and faster, by our own readers. So now its up to you. How do we get more readers? Do you care? Well you should do, most of the increase in the size and scope of FORMAT has been financed by the growth in membership. Remember that we started at just 16 pages a month, if you want to see 40, 48 or even more pages per issue we need to grow. Your ideas will be considered and I will award a years subscription to the best one received by the end of June. Get your thinking cap on - your ideas are really needed.

Many people have asked why we have not published anything on the new SAM computer in recent months. Well SAM development is still going ahead, MGT are in the process of mailing people who have registered an interest. But there is little that we havent already published that is not better left to a time when the machine is in peoples hands. There is little point in printing long articles now, only to have them repeated when the machine is available. But keep reading FORMAT, several articles are in a very advanced state and will appear as soon as SAM is launched.

Bye for now.

Bob Brenchley. Editor.

NEWS ON 4

PIRATE CASE DROPPED.

The Scottish Procurator Fiscal's office have dropped the case against members of a Glasgow based software piracy ring. Their excuse for dropping a case that the computer industry considered to be very important was that they thought the case was to complicated for a jury to understand. Can it really be true that after 10 years of the micro-computer boom the Procurator Fiscal's office still believes people are so ignorant that they cant understand that breach of software copyright is simple theft?

CHINA TOPS LIST.

Several countries have been targeted by the International Intellectual Property Alliance, an organization set up to protect the rights of owners of copyright material such as books, films, music and of cource software.

China tops the list of major offenders but the list also includes India, Brazil, Egypt, Taiwan and Nigeria. It is estimated that it cost American industry alone over $760 million in lost sales each year.

GOT ANY OLD GAMES MATE?

Oxfam are after your old and unwanted games. The games will be sold through their main London stores in an effort to raise over £20,000 for their projects. Software will be accepted at any of their shops nationwide during June and July. Please make sure the software still loads OK and that instructions are complete.

PSION BOOST.

After just one year on the Unlisted Securities Market Psion have announced pre-tax profits up by 48% over last year at £2.75 million.

Psion, well known in Spectrum circles for their early software association with Sinclair, are now best known for their ORGANISER II range of hand held computers which has recently been expanded with two new top of the range models.

I bet Alan Suger is green with envy after his much publicised drop in profits.

B.M.C. INTO PRODUCTION.

The B.M.C. - Bruce's Mega Chip - is now in production in Japan. This chip forms the real power-house of the new MGT SAM COUPE computer due for launch very soon.

The chip, which correctly called an ASCIC - Application Specific Custom Integrated Circuit - if being made by Figitsue. Most of the work on SAM has been in the design of this chip which reduces the number of chips required on the board from over 130 to just 8.

ON-LINE TO FRANCE.

Ever fancied logging on to MINITEL? (thats the French version of Prestel for the uninitiated. Well Nicholas McKenna of Runcorn has produced a program that allows Spectrum users with the VTX 5000 modem to dial up this massive database. Like Prestel and Micronet some pages/sections are free while others are only available to subscribers. So, provided you read French, it should be an interesting change. Contact:- N.O.McKenna, 277c Falcons View, Southgate, Runcorn, WA7 2XW, for further details.

If you have any news items you want to pass on then send them in. Please mark the envelope NEWS in the top corner.

SHORT • SPOT

By: John Wase.

The first offering this month is from Nick Fleming of Cleethorpes, who mentions that essentially the screen is mapped in two main sections (the first from address 16384 is in three further pieces and indicates if a pixel is selected or not: from 22528 to 23295 contains the attributes file which controls colour, bright and flash) and he sends two little subroutines. His first converts a screen address to an attribute address:-

Routine 1.	Routine 2.
LD A,H	LD A,H
RRA	RLA
RRA	RLA
RRA	RLA
AND 3	AND 24
OR 88	OR 64
LD H,A	LD H,A

And his second routine converts an attribute map address to a screen file address. These are, of course, merely for use as segments in a larger program.

Now a routine for DISCiPLE/PLUS D owners. As you know, entering CLS# with either of these interfaces attached clears the screen to black ink, white paper, white border, no bright, no flash: great if you like it but dead boring. If you don't; well, you can alter it (it's in the interface RAM) using the familiar POKE @ routines. The actual addresses for poking the numbers are different for the DISCiPLE and the PLUS D but the principles are the same. In Sinclair Basic, INK and PAPER are both represented by a number from 0-7. We can therefore conveniently use one decimal number to represent both together, let's call it SCREEN. So SCREEN = ((8*PAPER)+INK). Now enter one of the following pokes.

PLUS D:- **POKE @3780,SCREEN**
DISCiPLE:- **POKE @4570,SCREEN**

And then do CLS#. Up comes your screen in the chosen colours, the BORDER colour is automatically set to the PAPER colour. You can always save your customised system file for future use.

Harold Burton of Edinburgh is one of those people who abound in ideas, and has written again. Although longer than I would like, his program is clearly so useful to those of you who keep a mixture of programs on a disc that I am going to include it. Harold writes that one of the most useful features of the system is the abbreviated load - LOAD Pn - where n is a number. Most programs consist of a number of files, but always starting with a Basic Loader. Harold's routine has two alternatives: either a full concise CAT, (complete with program numbers) is output, or the list contains names and numbers only of Basic programs, Auto-run code files and snapshot files - enough for one to select a number and load the loader and all the associated files. Harold numbers all his discs by saving the system file for his DISCiPLE with an extension - something like "Sys 3d 4", and, provided this is file number one, the program prints it out as well; otherwise one is asked for a number as well as a name. Here's the listing.

```
 1 REM No & Name
 2 REM By: Harold Burton. 1.5.89
10 POKE @5,80: POKE @6,1: LPRINT CHR
   $ 27;CHR$ 64;CHR$ 27;CHR$ 48;: PO
   KE @6,0
20 INPUT "Enter date of printout:  "
   , LINE a$
30 PRINT AT 6,0;"Choose Printout:"''
```

```
'"1. ALL FILE NAMES"'"2. BASIC,
AUTORUN CODE FILES &     SNAPSHOT
 FILES ONLY"
 40 PRINT AT 20,0;"Key No. of choice:
    ": PAUSE 0
 50 GOTO (60 AND INKEY$="1")+(130 AND
    INKEY$="2")+(40 AND INKEY$<>"1"
    AND INKEY$<>"2")
 60 GOSUB 280: LPRINT AT 0,4;"COMPLET
    E LIST OF ALL FILES - ";a$
 70 GOSUB 200
 80 FOR p=1 TO 80: IF PEEK ml<>0 THEN
     LPRINT TAB q;(" " AND p<10);p;"
    ";: FOR n=1 TO 10: LPRINT CHR$ PE
    EK (ml+n);: NEXT n: LET q=q+15: I
    F q>72 THEN LET q=3
 90 LET ml=ml+256: NEXT p: LPRINT CHR
    $ 13;: LPRINT
100 PRINT #0;"Any more discs to CAT?
     Y/N": PAUSE 0
110 IF INKEY$="Y" OR INKEY$="y" THEN
    GOTO 70
120 GOSUB 280: GOTO 1e4
130 GOSUB 280: LPRINT AT 0,4;"LIST OF
     BASIC FILES, AUTORUN CODE FILES
    AND SNAPSHOT FILES - ";a$
140 GOSUB 200
150 FOR p=1 TO 80: IF PEEK ml=1 OR PE
    EK ml=5 OR (PEEK ml=4 AND PEEK (m
    l+218)+256*PEEK (ml+219)<>65535 A
    ND PEEK (ml+218)+256*PEEK (ml+219
    )<>0) AND PEEK ml<>0 THEN LPRINT
    TAB q;(" " AND p<10);p;" ";: FOR
    n=1 TO 10: LPRINT CHR$ PEEK (ml+n
    );: NEXT n: LET q=q+15: IF q>72 T
    HEN LET q=3
160 LET ml=ml+256: NEXT p: LPRINT CHR
    $ 13;: LPRINT
170 PRINT #0;"Any more discs to CAT?
     Y/N": PAUSE 0
180 IF INKEY$="y" OR INKEY$="Y" THEN
    GOTO 140
190 GOSUB 280: GOTO 1e4
200 CLS: PRINT AT 6,1;"INSERT DISC th
    en press any key": PAUSE 0
210 LET m=30000: FOR t=0 TO 3: FOR s=
    1 TO 10: LOAD @1,t,s,m: LET m=m+5
    12: NEXT s: NEXT t
220 GOSUB 280
230 IF PEEK 30009>32 THEN LET c$=CHR$
    PEEK 30008+CHR$ PEEK 30009
240 IF PEEK 30009=32 THEN INPUT "ENTE
    R DISC NUMBER  "; LINE c$
250 CLS: INPUT "DESCRIPTION  "; LINE
    d$
260 POKE @6,1: LPRINT CHR$ 27+CHR$ 14
    ;: POKE @6,0: LPRINT TAB 2;"DISC
    NUMBER  ";(" " AND VAL c$<10);VAL
     c$;"   ";d$: LPRINT
270 CLS: LET q=3: LET ml=30000: RETUR
    N
280 LPRINT "-------------------------
    ---------------------------------
    ----------------------": RETURN
```

Finally, from the sublime to the trivial. Malcolm Perry of Kidderminster, one of Bob's more prolific correspondents, has sent in a huge pile of material. Here is a silly holiday game - yes, it's that old favorite "hangman". One player looks away while the other types in a word and presses "enter": the letters in the word initially appear as stars (*) at the top of the screen. Guaranteed to keep young vistors happy for hours, so Malcolm says.

```
 10 CLS: LET H$="HANGMAN": PRINT AT 1
    2,0;h$
 20 INPUT "THINK OF A WORD";A$
 30 LET C=0: LET L=LEN A$
 40 LET H=0: PRINT AT 0,0
 50 FOR A=1 TO L: PRINT "*";: NEXT A
 55 PRINT AT 12,0;" "
 60 INPUT "THINK OF A LETTER";B$: LET
     D=C
 70 FOR A=1 TO L
 80 IF B$=A$(A) THEN PRINT AT 0,A-1;B
    $: LET C=C+1: IF C=1 THEN GOTO 10
    00
 90 NEXT A
100 IF C=D THEN LET H=H+1: GOTO 2000
110 GOTO 60
1000 PRINT AT 20,0;"WELL DONE"
1010 INPUT "ANY KEY TO PLAY AGAIN";A$
1020 GOTO 10
2010 PRINT AT 12,H;H$(H)
2030 IF H=7 THEN GOTO 5000
2040 GOTO 60
5000 PLOT 138,134: DRAW 4,0: PLOT 127,
     115: DRAW 12,-12: DRAW 12,12: PL
     OT 144,140: PLOT 136,140: PLOT 14
     0,136: CIRCLE 140,138,10: PLOT 14
     0,129: DRAW 0,-20: PLOT 125,65: D
     RAW 15,15: DRAW 15,-15: PLOT 140,
     100: DRAW 0,-20
5010 PRINT AT 20,0;"AAAAAGGGGgggg....."
5020 PLOT 155,115: DRAW 4,0: DRAW 0, -
     10: CIRCLE 160,95,8
5030 GOTO 1010
```

Keep sending those small routine to me at: Green Leys Cottage, Bishampton, Pershore, Worcs, WR10 2LX.

ADVENTURE CORNER

By: Paul Rigby.

Last month I talked about the latest developments of the PAW. One feature amongst the discussion was the promotion, which Gilsoft are staging, of User overlays. The aim being that, via the menu option "Z", the user will have access to a variety of User Overlay programs which will enhance and improve the PAW in the same way that the Patch and the Press did for the Quill. Once the overlay has been selected, via it's own identity letter, it loads into the main program. Within the later versions, A16 and above, of the PAW Gilsoft have thoughtfully provided the first User Overlay, a data management program which anticipates the possibility of the adventure author being swamped with a mess of User Overlays. When it was produced it was thought to be a very handy program but not of much use without other overlays to manage. Well, Gilsoft have decided to start the ball rolling by producing, with G.T.Kellet of Kelsoft, the first three additional User Overlays. They are PAW-Phosis, PAW-Tel and PAW-Mega.

PAW-Phosis, accessed by typing "P" in the User-Overlay "Z" option, is a powerful editor for use, mainly, with Process tables. Phosis will allow you to take one or more entries within the Process table and examine it, delete it, copy or transfer it. It will also allow the manipulation of whole Process tables around your database. So you are now able to copy or transfer them in bulk. Copying can now be done exactly, that is an exact copy can be made, or it can be copied into a different Verb Noun pair. You may have further control over the replacement of copies by designating a specific area in which it will go to.

The Transfer option of Phosis is very similar to the copy option but with several important differences. If, for example, you had designated the word "carefully" as a verb when you should have designated it as an adjective you would, previously, have had to lose all the entries you had associated with "carefully" and then start again. However, now you can simply change the designation by transferring all of the original entries you had assigned to "carefully" to the new word value, without losing any data.

The Delete option is controllable too. You can either delete all of the entries in a Process table or select a range of entries for deletion. Similarly, the Print option will show, on screen, either all the entries from a specified Process table or just a range of values. The Save/Load/Verify option will save and load Process tables to tape or disk with the inclusion of a file name for each Process table. While the Verify option will verify the Process table which has just been saved, before any changes have been made to it.

The final two options are Recall, which displays the last line which was entered and the Info option which displays the amount of locations in the adventure so far, the messages, the total objects and Process tables and so on. PAW-Phosis is very handy, therefore, for building a library of routines which can always be used for future adventures using the copy/transfer routines. An immense amount of time can be saved using this method.

PAW-Tel's main function is to debug a database, using it's own set of prescribed rules. This is another

time-saver because it means that you can now throw away that listing you have been pouring over all night and let Tel, accessed by keying "M" in the "Z" option, do it for you. The main menu options for the debugger are Hunt, Search, List and Info. Info simply displays a variety of information on screen in a similar way that Info does in PAW-Phosis.

You can, through the Hunt option, ask it to hunt through a specified Process table(s) for a list of parameters. After it has done this it will list them on the screen. The List option can become quite complicated depending on what you want it to do. For example you can list all of the Noun, Verb and entry numbers of all the entries in a range of Process tables which contain conditions that refer to a specific Flag. The List option can also list the flags, messages and objects, for example, that are used in one particular Process table or tables. You can also list all of the Flags used within a Process table, or all the messages. In fact the permutations are endless!

The Search option will search a Process table, or tables, for Conditional Actions (CondActs) that match the CondAct specification on the command line. For example, you can go to the Response Table and list all of the Verb, Noun and entry numbers of the entries that contain ISAT CondActs which test for object seven at any location number.

Another interesting option for Tel is the Map feature. This gives a pictorial representation of all of the exits from a given location number. At the end of each direction the connecting location number is given. For example, if you are in location one a number one will be displayed in the centre of a compass rosette. If you wished to go North you would see a line drawn northwards and, at the end of the line, would be the location number you would arrive at if you actually **did** go north, location five, for example. The full rosette is displayed on screen with all of the destination locations displayed too. UP and DOWN as well as ENTER and LEAVE can also be displayed. You can print this display out thus giving a valuable and more recognisable picture of how things are progressing. I am sure that some enterprising adventure author could adapt this feature to be displayed on-screen while you play the finished adventure thus keeping up with the times. It seems that the majority of the newer commercial adventures have this feature. Incidently, there is part of the Map option within PAW-Phosis. This was not intentional but it had to be done because of memory limitations within Tel.

One of the major criticisms of the PAW is that it is very difficult to use for the owners of 48K Spectrums. Memory limitations involve prospective authors to constantly load and reload overlays for editing and so on. Well Gilsoft have responded to the cry with PAW-Mega! What Mega does is supply all of the functions which are supplied in overlays four and five into one large overlay. Overlay four normally gives you the Process/Response tables, Vocabulary, Connections and Words tables while overlay five gives the Messages, Locations, Objects, Initially At, Object Weight and Background Colours. You can see why, having crunched that lot into one overlay, Gilsoft call it Mega! Actually, I tell a lie. They have had to exclude the background colours option but the lack of this option in Mega is hardly catastrophic.

The functions of Mega are divided up into two principal sections - the internal and external options. The internal options are those options listed above, with the obvious exclusion of the background colours. Those "internal" options can be accessed instantly with no problems to the user. The external options return you to the main menu of the PAW. From this menu options such as the Test Adventure, Graphics and so on can be accessed. However, the Main Menu options are, in the main, overlays themselves so when they are loaded

they overwrite the Mega overlay. It has to born in mind, therefore, that when using Mega you do not forget and think you are in the main menu area. Of course, while Mega will be of immense use to 48K owners that is not to say that it will be of no use to 128K owners. It will be useful for all of those 128K authors who have created a giant sized adventure, or who are working in 48K mode.

The three new User-Overlays are extremely useful. Congratulations must be given to Gilsoft and G.T. Kellet for giving their continued support to the PAW while providing worthwhile additions to the system. I can't wait for the next batch of User-Overlays. Don't forget though, if you are reading this and you think you may have a small program which you think may enhance the PAW then contact Gilsoft. They will give genuinely interested people a list of the function calls and the database structure. Just include an A5-sized SAE. The overlays have to be written in assembly language and be around 5K in length. Tell them Format sent you!

The whole question of utilities is a touchy one. Especially among the high exalted thrones of the noble Adventure Reviewer. Many times have utilities come under criticism, mainly from the press, with accusations ranging from Utility driven adventures looking all the same to the finished adventure being totally boring. One has to ask the question "Why?". Is it the fault of the Utility or the adventure author? Should Utility driven adventures be compared with commercial products at all? After all, is it fair to compare a home produced adventure, which arrives in a jiffy bag, complete with a batch of type-written material in a standard cassette case (sometimes none at all - to save postage), with the latest release from Infocom. You receive the latter in a high quality box with a variety of glossy instructions and background material which has been professionally typeset, complete with high quality artwork and illustrations, and then a key-fob, or a box of matches or some other piece of trivia pertaining to the adventure. Imagine the adventure reviewer who works for a glossy magazine receiving the latest Infocom adventure and the latest mystery adventure from BackbedroomSoft. Even before the respective adventures are loaded up the reviewer has already formed an opinion about them. It is not the fault of the enthusiast that he or she cannot provide the quality that the commercial adventure houses can. Even so, I have seen some very creditable efforts by individuals who have tried to present their creation in the best possible light. It is all a question of money, time and manpower.

Walk into a commercial software house. Fred and Bill over there are concentrating on the code using fast PC compatibles. Jim, who is provided with an Amiga, is dabbling with some graphic routines on Deluxe Paint II (the upgrade to "III" is in the post). If he gets stuck for ideas he might ask Bertha, the up-and-coming art graduate, for some help. In the meantime, another department is designing the artwork for the box cover while a lady sitting at her PC is contemplating the instructions using Wordperfect V.5. Further work is also being done on a novella while Mr.Big struts around the office with a fat cheque book in case the money runs out. Meanwhile Anita Amstrad and the gang who are producing the actual adventure have decided that a day's work has been done, so they retire to the pub to contemplate life and their next holiday in the Bahamas. Meanwhile, in the office, Laser Printers silently whirr because the "rough" copies of the novella are being produced, after being set by Ventura DTP, to be sent down to the typesetters. On a lonely desk at the back sits the tape with the cover music by Jean Michel-Jarre. Meanwhile, at number 6 Downthe Road, Mr.Smith sits in front of his Spectrum with a copy of the PAW, a typewriter to one side and a smile on his face as he discovers that he has found a way to rescue 36 bytes of code which can be used to enhance the graphic in location two of his new adventure -

"It came from Mars, via Milton-Keynes"..

A slight exaggeration, perhaps, but do you see what I mean when I question whether the two types of adventure should be reviewed side-by-side? Surely there has to be a new set of rules when reviewing small, mail-order adventures. The packaging should largely be ignored, unless it stands out as being especially good, so should any graphics within the game, with the same proviso. What should be reviewed are how the puzzles and the storyline hang together, are there any puzzles and storyline to begin with?, the quality of the descriptions, atmosphere?, spelling mistakes?, and so on. I remember that excellent adventure, Jekyl and Hide, by Essential Myth, being reviewed (a game which recently won the "Best mail-order adventure" by the Adventurers Club Limited) in a "glossy" magazine. The reviewer heaped praise upon it and then gave it a mediocre set of percentage marks! He was surely, subconsciously at least, comparing it to the latest from Magnetic Scrolls, or somesuch. Fair enough, if a home-produced adventure **is** boring or severely lacking in imagination then the reviewer should say so. But, for goodness sake, credit where credit's due!

See you next month - temper intact.

* - * - * - *

WRITING FOR FORMAT

We are always on the look-out for articles and programs to publish in FORMAT. Articles can be on any subject related to the Spectrum or computing in general. From half a page to a long series.

Don't worry too much about spelling and things like that (the Editor can't spell either) we will sort things out. Just put it down as clearly as you can. It is best if you send your article as a word processor file, on disc or tape, but please include a printed copy so we can look at it straight away. Pack any pictures flat or better still include SCREEN$ files so we can print them out here.

We are urgently looking for writers to produce articles on the following subjects:-

COMMUNICATIONS - THE 128K ROM

COMPUTER ART - EDUCATION

We also require REVIEWS of software and hardware, both new and old. If you want to do a review please ring first to confirm no-one else is reviewing the same product.

Send your work to our address on page 2 or give us a ring to talk about it.

* - * - * - *

BACK ISSUES

For members who have missed past issues of FORMAT (or perhaps worn theirs out through constant use) we run a back-issue service. The cost is £1 per issue (£1.25 overseas) incl p&p. Your copies will be sent out with your next monthly issue of FORMAT (provided we receive your order at least a week before). Make cheques (drawn on UK bank or Euro-Cheques, P.O., cash) payable to **FORMAT.**

AVAILABLE ISSUES

Volume 1

Issues #1 (Aug'87) - #12 (Jul'88).

Volume 2

Issues #1 (Aug'88) - #9 (May'89).

Note:- Early issues are high-quality reprints.

Please **WRITE YOUR ORDER ON A SEPARATE PIECE OF PAPER.** DO NOT include letters with your order. Remember to quote your membership number or we wont be able to send you your order.

HACK-ZONE

By: Hugh J. McLenaghan.

To start off this month I would like to make another plea for ideas, conversions, tips, and anything else you can think of. Send then direct to me at the address given at the end of this article.

Next, following a request from a reader, I have rewritten the Cat-Sort routine (which was printed in Vol.2 - No. 5) to work with the PLUS D. At the time I only had a DISCiPLE but FORMAT has now lent me a PLUS D so I have been able to make the changes. If you use the Hexloader printed in issue Vol. 2 - No. 2 to enter the data, and save it with **SAVE d*"CatSorCd" CODE 65000,524.** You can then use the same BASIC program that was used for the DISCiPLE version. Here is the code:-

```
65000: CDF8FDCD7EFFCDDF7
65008: FECDEFFECDAAFFC99
65016: CD02FE3AA9FFB720A
65024: F7C9110100AF32A95
65032: FFD5CD84FE28183A9
65040: D63CB72815D3E7D1C
65048: 1C7BFE0B20EB1E01E
65056: 147AFE0420E3C9AF5
65064: 18023E01D3E73200A
65072: 5BED53015BFE01288
65080: 0ADBE73AD63CD3E7D
65088: B7202FD11C7BFE0BB
65096: 20071E01147AFE04C
65104: C8D5CD84FE20183AC
65112: D63CB72015D3E7D1C
65120: 1C7BFE0B20EB1E01E
65128: 147AFE0420E3C9AF5
65136: 18023E01D3E732034
65144: 5BED53045BCD8DFEA
65152: D1C309FECF3FDBE70
65160: 3AD63BB7C921D63BC
65168: 84671100D0CDD5FEE
65176: ED5B015BD5CF3F3A9
65184: 005B21D63B8467E59
65192: 1100D1CDD5FED121D
65200: 00D0CDD5FED1CF3EF
65208: ED5B045BD5CF3F3A5
65216: 035B11D63B8257213
65224: 00D1CDD5FED1CF3E1
65232: 21A9FF34C9010001E
65240: DBE7EDB0D3E7C906F
65248: FF21007F04247EB72
65256: 20FA7832065BC93AF
65264: 065BFE02D821017FC
65272: 3EFF22085B32075B1
65280: 3A075B3C32075B479
65288: 3A065BB8C82A085B0
65296: 2422085BCD19FF184
65304: E7E5D1220B5B3A079
65312: 5B320A5B2A0B5B24E
65320: 220B5B3A0A5B3C329
65328: 0A5B473A065B3DB85
65336: D82A085BED5B0B5B8
65344: 1ACDD7FF46CDEBFFA
65352: B82813380218D52BD
65360: 1B06001A4F7E12717
65368: 231310F718C606092
65376: 23131ACDD7FF46CD0
65384: EBFFB838082004235
65392: 1310EF18AF2A085BA
65400: ED5B0B5B18D11101F
65408: 00210080D5E5CF3F0
65416: D121D63BD50100020
65424: DBE7EDB0D3E7E1243
65432: 24D11C7BFE0B20E40
65440: 1E01147AFE0420DCB
65448: C90011010021008O5
65456: D5E511D63B010002A
65464: DBE7EDB0D3E7E1D18
65472: D5E5CF3EE1D12424F
65480: 1C7BFE0B20E21E014
65488: 147AFE0420DAC9474
65496: 3A815CFE5B280A785
65504: FE61D8FE7BD0CBAF9
65512: C978C9F578CDD7FF3
65520: 47F1C900101010000
```

At last that's that all finished for now. I know a lot of you do not like HEX listings, but its the fastest way of putting data into your computer. With my hexloader program, it even has an automatic checksum which is more accurate than a lot of others.

Next is a short conversion of the POKE -FIND (Vol.1 No 11 to Vol 2 No 1) programs to work with the PLUS D. This follows for both the 48k and 128k versions. In the DATA statements replace all:-

```
219,187 with 207,71
 214,27 with 214,59
 214,28 with 214,60
211,187 with 211,231
```

Both versions should then work without any other changes.

If you wish to hide files from being seen, but can be loaded, then you can use the following program to do this. It will also UnHide any hidden files. All that you need to do is give the start and end file numbers.

Here is the program:-

```
10 REM Written by
20 REM Hugh J. McLenaghan
30 REM On 1st April 1989.
40 REM
50 CLS: CAT *
60 INPUT "First position ";ST
70 INPUT "Last position ";FIN
80 IF FIN<ST THEN GOTO 60
90 FOR A=ST TO FIN
100 LET S=INT ((A+1)/2)-1: LET T=INT
    (S/10): LET S=S-(T*10)+1
110 LET D=256*NOT (A-(INT (A/2)*2))
120 LOAD @*,T,S,4e4
130 IF PEEK (4e4+D)<128 THEN POKE 4e4
    +D,(PEEK (4e4+D))+128: GOTO 150
140 POKE 4e4+D,(PEEK (4e4+D))-128
150 SAVE @*,T,S,4e4
160 NEXT A
170 CLS: PRINT "All Done!"
```

Try it out as it can be useful if you have a lot of files on disc which are only accessed by a main program. This allows you to only have the main programs listed in the catalogue, but the other files are still there, but are just not displayed.

Well that's it again for another month. Please remember that I need all the letters that I can get. I'm sure that out of the 1600 or so of you out there that a lot of you can program or have ideas that you would like to see in a program. If so then send your ideas and comments to:-

Hugh J. McLenaghan (Hack Zone),
36 Floorsburn Cresc.,
Johnstone,
Renfrewshire,
PA5 8PF.

Back next month.

INSIDE G+DOS

Part 3.

By: Stephen Warr.

This month I want to look at the Disc Directory, this is the same for both the PLUS D and its older brother the DISCiPLE so there is much in this article for DISCiPLE users..

Each directory entry takes up 256 bytes and contains all the data needed to load the file that it is referring to. The layout is as follows:-

PLUS D/DISCiPLE DIRECTORY LAYOUT

BYTE	CONTENTS	
0	File Descriptor 0-11 (see last month)	
1-10	File Name	
11	File Sectors Used - Low	
12	File Sectors Used - High	
13	Start of File - Track	
14	Start of File - Sector	
15-209	Sectors Used Map (195x8 bits)	
210	File Size High	
211	File Type	(HD00)
212	File Size Low	(HD0B)
213	" " Mid	
214	Start Address Low	(HD0D)
215	" " High	
216	Size - Vars Low	(HD0F)
217	" " High	
218	Auto-Run Line Low	(HD11)
219	" " " High	
220-241	Snapshot Register Area	

The bytes from 242 to 255 are not used by the current DOS but may be in future versions.

In double density mode the first 40 sectors of the disc (tracks 0-3) each hold 2 directory entries giving a total of 80 files. In single density (only available on the DISCiPLE) each sector holds 1 entry, hence only 40 files per disc.

Byte 210 is only used by extremely long OPENTYPE files (over 64K). It is the most significant of 3 length bytes with 212 & 213.

Bytes 211-219 are a copy of the UFIA (see Dos Command Codes - FORMAT Vol 2 No 4.) they are also stored as the first nine bytes of most files.

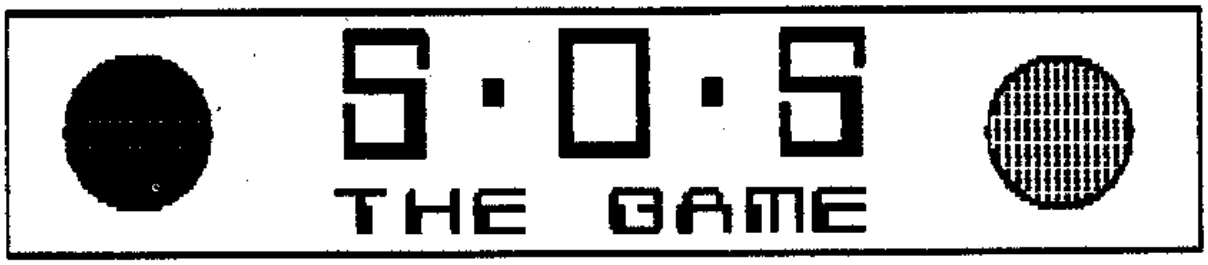
Once you have found the correct directory entry using the directory access routine at 2469 (#9A5), you can point to the data in the entry by loading IX+13 with the displacement, IX will already be set to 15043 (#3AC3), and then calling the routine at 3479 (#D97). HL will then hold the address of the data - i.e. if the directory access routine found a BASIC program:-

```
LD (IX+13),218
CALL 3479
LD E,(HL)
INC HL
LD D,(HL)
```

DE now holds the auto-run line number (or #FFFF if there isn't one). NB. If you want to get the file type, CALL 3475 (#D93) is equivalent to loading IX+13 with zero before a CALL 3479.

For those who are interested, the sector containing the directory entry will in fact have been loaded to address 15318 (#3BD6), ie. in the PLUS D RAM. Each sector in the directory holds two entries and IX+14 will either hold 0 or 1. This value, together with the value in IX+13, gives a two byte displacement from 15318 so the value in IX+13 alone actually gives a displacement from either 15318 or 15574 depending on which entry is being examined.

Back next month with a super routine for the PLUS D.



By: Mark Hill.

S.O.S. (Save Our Souls, Sanity, Spectrums - You choose)

Many years ago I obtained a small board game, it was a puzzle, but one designed in such a way that it could be played time and time again, even after it was solved. It was a sort-of board with holes and many round pegs and I enjoyed many frustrating hours playing it (I'm easily pleased). Needless to say that on rediscovering it a couple of years later, most of the pegs had gone missing. Soon after that I decided to write a spectrum version (no more missing pegs!).

S.O.S. INSTRUCTIONS

When you run the program you will be greeted by a colourful (if rather small) title screen. Press a key and a cyan coloured grid of 25 squares will appear, this is the playing board. The score on the upper right of the screen tells you how many squares you have filled. The first four moves are made for you and are always on the four outer squares. The game itself is played with 25 coloured discs which comprise of 5 BLUE, 5 RED, 5 GREEN, 5 YELLOW and 5 WHITE.

A disc at a time is chosen at random (the first four are picked and put around the board for you). It is then up to you to finish the game by filling the rest of the board. You do this by moving a curser, which is a thick, hollow, cyan, square, around the board using these keys:-

Q = up A = down O = left P = right
SPACE = fire 5 = abort game

When you have chosen the square you wish to fill you press the 'SPACE' button to register the move, you can also abort at any time by pressing the '5' key.

After each move you make, the computer will check to see if you can move again, if not the game will automatically end. You will not be allowed to make 'invalid' moves. Each counter picked randomly from the 25 will be displayed in the 'NEXT DISC TO PLACE' area on the top right of the screen. You then have to decide where to put this disc on the board, But these rules must be followed for a valid move:-

1: Only one disc may ocupy a square, and it may not be moved once it has been left there.
2: After the four initial moves, each disc has to be left in a square that is next to another disc either vertically or horizontally (but not just diagonally).
3: One colour disc may not be left next to another of the same colour, in any direction, vertically, horizontally or diagonally.

You complete the puzzle if you fill all the squares, or score points for the number filled when the game ends (between 4 and 24). Unless you abort, or complete the game, it will end because you cannot make the next move legally (by following the above rules).

This is a one player game but two can play by taking it in turns and the winner is the last one able, or better still, unable to move.

I hope you understood all that, I realise that it may seem a bit heavy going at first, but is simplicity itself, when you get used to the game (simple to understand that is - not to

complete).

IMPORTANT NOTE: This game uses five colours from the Spectrum, which may be hard to tell apart if you use a black & white T.V. set. If this is the case, change the value of the variable 'col' in line 10 to zero (10 LET col=0), then RUN. This will then display any disc with its colour number in the centre for easy distinction. i.e. BLUE=1, RED=2, GREEN=4, YELLOW=6 & WHITE=7.

S.O.S. PROGRAM LISTING

O.K. Here is the listing. Please note - In places the listing contains U.D.G. characters. When you come across this box: {} its contents show you what to type as follows: {G A} would mean enter Graphics mode then type A, {GS 8} would mean enter Graphics mode then while holding CAPS SHIFT press '8' etc... In line 6120 there is an awful lot of graphics characters, please enter carefully (thanks to Mr Nixon for his article on control characters within listings in FORMAT Vol 1 No 10).

```
  1 REM ** PROGRAM: S.O.S.! **
 10 LET col=1
100 PAPER 0:BORDER 0:INK 0:CLS :GOSUB
     9500
110 INK 5:FOR a=1TO 19STEP 3:PRINT AT
     a,0;TAB 20:NEXT a
120 LET pe=7:RANDOMIZE
399 REM MAIN PROG LOOP
400 DIM p(5):LET move=0:LET x=9:LET y
    =9:LET x1=9:LET y1=9
500 GOSUB 6000:GOSUB 7000:GOSUB 2400
510 INPUT ;:PRINT #1;"CHECKING MOVE-P
    LEASE WAIT.":GOSUB 9000:IF pos=0T
    HEN GOTO 700
520 IF move=24THEN PRINT AT 16,20;FLA
    SH 1;PAPER RND*7;INK 9;"  WELL DO
    NE  ";AT 17,19;"you can do it"
530 GOSUB 9100: GOSUB 1000
540 IF pos=0 THEN GOTO 600
550 LET a=x: LET b=y: LET pos=9: GOSU
    B 9020: IF pos=0 THEN INPUT ;: PR
    INT #1;"INVALID MOVE...": BEEP .1
    ,-15: PAUSE 25: GOTO 530
560 BEEP .05,30: IF move=24 THEN GOTO
     800
570 GOSUB 2200
580 GOTO 510
599 REM ABORT GAME
600 PRINT AT 20,18;TAB 0;AT 21,20;TAB
     0
610 BEEP .2,50: INPUT ;: PRINT #1;"GA
    ME ABORTED...": LET pos=8: GOSUB
    9120: GOTO 400
699 REM CAN'T MOVE GAME OVER
700 PRINT #1;AT 1,0; BRIGHT 1; INK 7;
     PAPER 2;" YOU CAN'T MOVE-BAD LUC
    K! "
710 PRINT AT 16,20; INK 4; FLASH 1; B
    RIGHT 1;"  GAME OVER "
720 BEEP .5,-30: GOTO 400
799 REM A WINNER
800 LET pos=9: GOSUB 9125
810 PRINT AT 1,17;TAB 0;AT 2,17;TAB 0
    ;AT 3,28;TAB 0;AT 6,29; BRIGHT 1;
     FLASH 1;"25"
820 PRINT #1; PAPER 6; INK 1; FLASH 1
    ;"PUZZLE COMPLETED...WELL DONE!"
830 GOTO 400
999 REM KEY MOVEMENT
1000 LET a$=INKEY$
1010 IF a$="q" THEN LET x1=x1-3: GOSUB
     9100
1020 IF a$="a" THEN LET x1=x1+3: GOSUB
     9100
1030 IF a$="o" THEN LET y1=y1-3: GOSUB
     9100
1040 IF a$="p" THEN LET y1=y1+3: GOSUB
     9100
1050 IF a$=" " THEN LET pos=1: BEEP .0
    2,50: RETURN
1060 IF a$="5" THEN LET pos=0: RETURN
1070 INPUT ;: GOTO 1000
2199 REM PRINT UPDATE COUNTER
2200 LET move=move+1: PRINT AT 1,28;TA
    B 0;AT 2,28;TAB 0;AT 3,28;TAB 0;A
    T 6,29;move: LET pos=9: GOSUB 912
    5
2210 LET pe=INT (RND*5)+1
2220 IF p(pe)>4 THEN GOTO 2210
2230 LET p(pe)=p(pe)+1
2240 IF pe=3 THEN LET pe=4: GOTO 2260
2250 IF pe>3 THEN LET pe=pe+2
2260 PRINT INK pe;AT 1,28;"{G A}{GS 8}
    {G B}";AT 2,28;"{GS 8}{GS 8}{GS 8
    }";AT 3,28;"{G C}{GS 8}{G D}": IF
     col=0 THEN PRINT PAPER pe; INK 9
    ;AT 2,29;pe
2270 RETURN
2399 REM FIRST FOUR MOVES
2400 GOSUB 2210: RESTORE 2410: FOR a=1
     TO 4: READ x,y: GOSUB 2200: BEEP
     .02,40: NEXT a: LET x=9: LET y=9
    : RETURN
2410 DATA 0,9,9,18,18,9,9,0
```

```
5999 REM TITLE SCREEN
6000 INK 5: LET c=1: LET b=9: PRINT AT
     17,19;TAB 0
6020 PRINT AT 20,18; INK 7; BRIGHT 1;"
     PRESS ANY KEY";AT 21,20;"TO STAR
     T... "
6030 RESTORE 6120: FOR a=8 TO 14: READ
     a$: PRINT INK b;AT a,22;a$: LET
     b=b+1: IF b>8 THEN LET b=1
6040 IF c=1 AND INKEY$<>"" THEN NEXT a
     : GOTO 6030
6050 LET c=0: IF INKEY$="" THEN NEXT a
     : GOTO 6030
6100 PRINT INK 6;AT 1,17;"NEXT DISC{G
     I}";TAB 0;AT 2,17;"TO PLACE {G I}
     ";TAB 0;AT 3,28;TAB 0; PAPER 1;AT
     16,20;"BY MARK HILL"
6110 PRINT AT 5,22;"DISCS";AT 6,22;"PL
     ACED:0 ";AT 20,19;TAB 0;AT 21,20;
     TAB 0: BEEP .03,30: RETURN
6120 DATA "{G A}{GS 8}{G D}{G A}{GS 8}
     {G B}{G A}{GS 8}{G D}","{GS 8}  {
     GS 8} {GS 8}{GS 8}","{GS 8}{TRUE
     VID}  {GS 8} {GS 8}{GS 8}","{G C}
     {GS 8}{G B}{GS 8} {GS 8}{G C}{GS
     8}{G B}","  {GS 8}{GS 8} {GS 8}
     {GS 8}","  {GS 8}{GS 8} {GS 8}  {
     GS 8}","{G A}{GS 8}{G D}{G C}{GS
     8}{G D}{G A}{GS 8}{G D}"
6999 REM SET UP PLAYING GRID
7000 INPUT ;: PRINT #1;"PLEASE WAIT...
     "
7010 FOR a=8 TO 14: PRINT AT a,22; OVE
     R 1; INK pe;TAB 0: NEXT a
7040 RESTORE 9080: FOR a=0 TO 18 STEP
     3: READ c: FOR b=9-c TO 9+c STEP
     3
7050 PRINT AT a,b;"{G E}{G F}{G G}";AT
     a+1,b;"{G H}{G I}{G J}";AT a+2,b;
     "{G K}{G L}{G M}": NEXT b: NEXT a
7060 PRINT AT 20,18;"(PRESS ""5""";AT
     21,23;"TO ABORT)": RETURN
8999 REM CAN I MOVE AGAIN
9000 RESTORE 9080: FOR a=0 TO 18 STEP
     3
9010 READ c: FOR b=9-c TO 9+c STEP 3
9020 IF ATTR (a,b)<>5 THEN GOTO 9060
9030 IF ATTR (a-2,b)=5 AND ATTR (a+1,b
     -3)=5 AND ATTR (a+1,b+3)=5 AND AT
     TR (a+4,b)=5 THEN GOTO 9060
9040 IF ATTR (a-3,b-3)=pe OR ATTR (a-3
     ,b)=pe OR ATTR (a-3,b+3)=pe OR AT
     TR (a,b-3)=pe OR ATTR (a,b+3)=pe
     OR ATTR (a+3,b-3)=pe OR ATTR (a+3
     ,b)=pe OR ATTR (a+3,b+3)=pe THEN
     GOTO 9060
9050 LET pos=1: RETURN
9060 IF pos=9 THEN LET pos=0: RETURN
9070 NEXT b: NEXT a: LET pos=0: RETURN

9080 DATA 0,3,6,9,6,3,0
9099 REM MOVE CURSER
9100 IF x1<0 OR y1<0 OR ATTR (x1,y1)=0
      THEN LET x1=x: LET y1=y: RETURN
9120 INK 8: IF ATTR (x,y)=5 THEN PRINT
      AT x,y;"{G E}{G F}{G G}";AT x+1,
     y;"{G H}{G I}{G J}";AT x+2,y;"{G
     K}{G L}{G M}": GOTO 9135
9125 IF pos=9 THEN INK pe
9130 PRINT AT x,y;"{G A}{GS 8}{G B}";A
     T x+1,y;"{GS 8}{GS 8}{GS 8}";AT x
     +2,y;"{G C}{GS 8}{G D}"
9131 IF col=0 THEN PRINT AT x+1,y+1; I
     NVERSE 1; PAPER 9;ATTR (x,y)
9135 IF pos>1 THEN INK 8: LET pos=1: R
     ETURN
9140 LET x=x1: LET y=y1: PRINT AT x,y;
     "{GS4}{G 3}{G 7}";AT x+1,y;"{GS 5
     } {G 5}";AT x+2,y;"{GS 1}{GS 3}{G
     S 2}"
9150 IF ATTR (x,y)=5 THEN PRINT AT x+1
     ,y+1;"{G I}": GOTO 9170
9160 IF col=0 THEN PRINT PAPER 9;AT x+
     1,y+1;ATTR (x,y)
9170 BEEP .01,10: RETURN
9499 REM GRAPHIC SET-UP
9500 RESTORE 9510: FOR a=USR "a" TO US
     R "m"+7: READ b: POKE a,b: NEXT a
     : RETURN
9510 DATA 0,3,15,31,63,63,127,127,0,19
     2,240,248,252,252,254,254,127,127
     ,63,63,31,15,3,0
9520 DATA 254,254,252,252,248,240,192,
     0,0,127,64,64,64,64,64,64,0,255,0
     ,0,0,0,0,0,0,254,2,2,2,2,2,2,64,6
     4,64,64,64,64,64,64
9550 DATA 0,0,24,36,36,24,0,0,2,2,2,2,
     2,2,2,2,64,64,64,64,64,64,127,0,0
     ,0,0,0,0,0,0,255,0,2,2,2,2,2,2,254,
     0
9999 CLEAR : SAVE "S.O.S." LINE 1
```

THE SECRETS OF WORD MANAGER

SPECTRUM MACHINE CODE MADE EASY

By: Francis Miles.

This month I want to finish looking at **WORD MANAGER**'s methods of printing screen messages.

System 4. There is a very slightly more economical system, which doesn't use any of the ROM routines except RST 16; it is only worth using in a fairly large program with a lot of messages which are only used once each (like **WORD MANAGER**). It saves three bytes per message, but these are partly cancelled out by the need for an extra set of subroutines.

To use these subroutines you do not do any preliminary loading of registers; but the call to the subroutine is _immediately followed by the message string, which has its last character incremented by 128 as in the last system. Eg:-

```
0120            CALL P.BYC
0125  ;PRINT AT 0,8;INVERSE 1
0130  ;see Note last month
0135            DEFB 22,0,8,20,1
0140            DEFM "DISPLAY CONTROLS"
0150
0155  ;PRINT AT 2,0;INVERSE 0
0160            DEFB 22,2,0,20,0
0170            DEFM "Controls are now"
0180            DEFB " set as follows:"
0190            DEFB 22,4,0
0200            DEFM "1. Automatic "
0210            DEFM "justification"
0220            DEFB " "+128
```

The additional subroutines look like this:-

```
5730  P.BCOL  EQU $
5740  ;CLS with colours as in A
5750  ;and print bytes on screen.
5760          CALL SCOL
5770          JR P.BY2
5780
5790  P.BY1   EQU $
5800  ;Print bytes in input.
5810          CALL A1C
5820          JR P.BY
5830
5840  P.BYC   EQU $
5850  ;Clear screen and print bytes.
5860          CALL CLS
5870
5880  P.BY2   EQU $
5890  ;Print bytes on screen.
5900          CALL A2C
5910  P.BY    POP HL;return address
5920  P.BYLP  LD A,(HL)
5930          RES 7,A
5940          PUSH HL
5950          RST 16
5960          POP HL
5970          BIT 7,(HL)
5980          INC HL
5990          JR Z,P.BYLP
6000          PUSH HL
6010          RET
```

You will find lines 5910 and the following hard to understand unless you have grasped how the CALL and RET instructions work. We think of them as meaning "do the subroutine indicated!" and "that's the end of the subroutine!"; but you have to think of them at the moron level, because that's how the Z80 chip reads them.

To the chip, CALL xx means "PUSH the address of the next byte after this command on to the stack and JP xx", and RET means "POP an address off the stack and JP to it". And that's <u>all</u> they mean.

So if you POP the top of the stack when you are inside a subroutine (line 5910), you get the address of the next byte following the CALL xx instruction: normally this would be the return address, but in this case it is the first byte of the message.

So you print the byte at that address with RST 16 (line 5950) - having first made sure its first bit is reset, in case it's the last of the message (line 5930). Now check the address again (line 5970); if its first bit is set, it's the last byte of the message, so you push the next address on the stack and RET.

If the address at HL wasn't the last of the message (line 5990) you loop round to print the byte at the next address. This trick of popping the return address is, of course, copied from the Spectrum ROM, where it's often tangled up with all kinds of other clever tricks. Perhaps the clearest example is in the ERROR.2 routine that starts at 83(53h).

This system 4. is used for all the messages in **WORD MANAGER** except

- some single-byte messages (system 1.)
- variable messages (system 2.)
- messages called for more than once in the program (system 3.)
- messages calling for choice depending on a flag setting (system 3.).

System 5. is not used in any commercial version of **WORD MANAGER**. It requires the construction of a "vocabulary list", containing the 127 most frequently used words (or character strings) in the messages of the program, coded from 128 to 254; often the first letter is left off, so it can be used with either a capital or lower-case initial, and some of the strings have a trailing space, others do not.

Compiling this list (it can't be done till the program is substantially finished) and coding the message strings using the list is very hard work, and I was doubtful when I first embarked on it if it was going to be worth while. To my amazement, when I assembled **WORD MANAGER** with this new system replacing systems 3 and 4, I found it saved about 10% on the whole size of the program, which I was able to use for extra text buffer: increasing the length of the text buffer from 357 lines in the commercial versions to no less than 410 lines in the one I use privately.

The operating subroutine is a new version of P.BY (see system 4.): P.BCOL etc are unchanged. As with system 4., the message is a string of bytes following the subroutine call; but some of the bytes are ordinary characters, to be printed as they stand, some have their hi bit set, signifying they represent strings to be looked up in the vocabulary list - in the list, the last code of each entry again has its hi bit set - and the last byte of the string is 255, the terminator. See the example below.

```
5930  P.BY    EQU $
5940  ;Print bytes from string. If
5950  ;byte has hi bit set, print
5960  ;that number from vocabulary
5970  ;list. String ends with 255.
5980          POP HL
5990          LD A,(HL)
6000          INC HL
6010          PUSH HL
```

[Now the code is checked twice: if it is 255, return to the address on the stack.]

```
6020  ;Terminate now?
6030          CP 255
6040          RET Z
```

[If its hi bit is zero, print it as a normal character and loop.]

```
6050  ;Is hi bit set?
6060  PB.1    BIT 7,A
6070          JR NZ,PB.2
6080  ;No. Print byte and move on.
6090          RST 16
6000          JR P.BY
```

[Otherwise print the "message" from the M.TAB table, much as in system 3., except that the table is now a vocabulary list.]

```
6010  ;Yes. Print message number A.
6020  PB.2    LD DE,M.TAB
6030          CALL 03082;PO.MSG
6040  ;Loop back for next byte.
6050          JR P.BY
```

Here is how this subroutine is used to print out the main menu (if the string is followed by slash, "/", it has a built-intrailing space, if by back slash "\" there is none):

```
0800            LD A,7 ;INK 7 PAPER 0
0810  ;Print main menu
0820            CALL P.BCOL ;as before,
leads into P.BY
0830            DEFB 22,0,9 ;AT 0,9
0840            DEFB 117+128 ;INVERSE 1
0850            DEFB 16,5 ;INK 5
0860            DEFB 126+128 ;"WORD MAN
AGER 5\"
0870            DEFB 105+128 ;"13,13,20
,0"
0880  ;two newlines and INVERSE 0
0890            DEFB 16,7 ;INK 7
0900            DEFB "P",80+128 ;"ress/
"
0910            DEFB 117+128 ;"20,1" IN
VERSE 1
0920            DEFB "s"
0930            DEFB 116+128 ;"20,0" IN
VERSE 0
0940            DEFB " "
0950            DEFB 100+128 ;"to/"
0960            DEFB 94+128 ;"show/"
0970            DEFB "s","89+128 ;"cript
\"
0980            DEFB " ",71+128 ;"on/"
0990            DEFB "s",88+128 ;"creen
\"
```

... etc (several pages of it!) ending with

```
1920            DEFB 255
```

Although this looks so cumbersome, and is very laborious to do, the saving from reducing "WORD MANAGER 5" (even though it only comes in the machine code twice) or "to " (which comes dozens of times) to a single byte each is enormous. The messages print out a little slower than with the other systems, but not enough to be bothersome.

Next month I will start to look at machine code arithmetic.

PLUS D PATCH

By: Bob Brenchley & Steve Nutting.

The PLUS D has proved to be very bug free since the upgrades to G+DOS 2a were published last year. However some readers may have encountered a small problem that has been traced to a bug in the Z80 chip (thats the CPU on the Spectrum) rather than to PLUS D. It manifests itself following a disc operation when you find the keyboard wont respond when you press a key. What has happened is that a return from the DOS has been made with interrupts disabled. As the spectrum uses an interrupt routine to read the keyboard 50 times a second if there are no interrupts then there is no keyboard scan.

The problem lies in a bug in the Z80 chip as I have already said. When the PLUD D tests for the state of interrupts (by doing an LD A,I) it expects the Parity Flag to contain the current state. Most of the time this is true, but on the Z80 an interrupt can occur while the LD A,I instruction in being processed. What then happens is that IFF1 (the internal flip-flip used by interrupts is copied to IFF2 (its back-up) and IFF1 is set to zero. This means the PLUS D is fooled into thinking that interrupts were disabled even though they might not have been. Bruce Gordon (famed designer of the PLUS D) ran tests when this problem was discovered and found that in normal operation the test of interrupt state was wrong around .05% of the time. Not much I know, but it only needs to be wrong once to lock you out of your program.

Now, alas, the problem had no instant solution. The DOS could be changed to do the test twice, and then only to take the answer if it was the same both times, but the routine is in the ROM on the PLUS D and it was considered too small a problem to warrant issuing a replacement EPROM. So I dropped the problem into Steve Nuttings lap and asked him to come up with a fix that did not involve changing the ROM. His answer turned out to be a very simple one although it must have taken lots of work to find it. His routine (printed below) adds a new function to the Snapshot button. By pressing the button and then the 0 key (zero) he forces interrupts to be enabled. So if you ever find yourself with a keyboard lock-out you can now escape from the jaws of disaster.

BOOT your disc system, then type out this program and RUN it. It should work with all versions of G+DOS.

```
 5 CLEAR 24999: LET C=0
10 FOR I=25000 TO 25057
20 READ N: LET C=C+N: POKE I,N: NEXT a
30 IF C<>6505 THEN PRINT "ERROR": STOP
40 RANDOMIZE USR 25017
50 DATA 203,99,202,49,33,6,239,237
55 DATA 88,203,67,192,251,225,195,178
60 DATA 0,207,71,58,31,41,254,177
65 DATA 40,5,62,192,50,183,97,33
70 DATA 168,97,17,237,49,1,17,0
75 DATA 237,176,62,195,50,46,33,33
80 DATA 237,49,34,47,33,195,80,0
85 DATA 243,201
```

When finished save the System by:-
SAVE d1"+sys"CODE 8192,6656

Now, as a test, type RANDOMIZE USR 25056, this will DIsable the interrupts and you will find the keyboard locked out. To return to normal press the Snapshot button and then key zero and you should be back in business.

My thanks to Steve Nutting for his help.

SORTING in BASIC

By: Nev Young.

The first thing to do is try and explain what arrays are. There can be no doubt that arrays can be used in very complex ways but fortunately the principle of arrays is very simple.

Imagine that you have a program that, for whatever reason, needs to hold 20 numbers. You could store each number in a separate variable. e.g.

```
LET NUM1 = 42
LET NUM2 = 23
LET NUM3 = 56
etc. etc.
```

This is fine but the only way of getting at the number held in the variable is by naming that variable. So to print all the numbers you would write:-

```
PRINT NUM1
PRINT NUM2
PRINT NUM3
etc. etc

or PRINT NUM1,NUM2,NUM3, .....
```

An array would be a better way of holding all these numbers, but they do require a little bit of work to set up. First you have to ask the computer to create the array for you. On the Spectrum you use the **DIM** command. So to hold 20 numbers you would use **DIM A(20)**. This creates, in the system variables area, space for 20 numbers but only one variable name. In this case the array is called **A**.

The problem, you have now is, how do you get to each of these numbers if they are all called **A**? This is done by using a subscript or index. So to print the 9th number from our array we would use **PRINT A(9)** where (9) is the subscript and 9 the index. Likewise to store a value in the 12th number we would use **LET A(12)=42**. From now on arrays are easier to use than separate variables. This is because we can replace the index with another variable. So now to print all the 20 numbers out becomes a simple one line program:-

```
FOR N=1 TO 20: PRINT A(N): NEXT N
```

Now lets create an array full of data and save it for later use. Program 1 does just this. (NOTE non DISCiPLE and PLUS D users should change the save and load commands by omitting the **d1** in the command. This will save to and load from tape. Microdrive users should replace the **d1** with ***"m";1;**).

```
10 REM PROGRAM 1.
20 DIM A(20)
30 FOR N=1 TO 20
40 LET A(N)=INT (RND*100)
50 NEXT N
60 SAVE d1"Array" DATA A()
```

We can use this array to show how sorts work. If we print out the contents of the array you will see that the numbers are not in any type of order. Use program 2 to print then out.

```
10 REM PROGRAM 2.
20 LOAD d1"array" DATA A()
30 FOR N=1 TO 20
40 PRINT A(N)'
50 NEXT N
```

Now to get the numbers sorted into order. If we compare every number with the number after it, and swap them if the first is greater than the next. Then by the time we get to the last number we will have moved the largest number to the last position. Program 3

does just this. Line 20 jumps to the start of the program where the data created by program P1 is loaded. Lines 30 to 70 print out the numbers so we can see what is going on. lines 100 to 120 swap a number with the one after it. Lines 200 to 240 compare every number with the one next to it. Notice that the loop in line 200 stops at 19 rather than going to 20 as you might expect. This is because we do not need to compare the last number as it is already at the end of the array and besides there is not a 21st number to compare with.

```
 10 REM PROGRAM 3.
 20 GOTO 1000
 30 REM printout
 40 PRINT AT 0,0;
 50 FOR Z=1 TO 20
 60 PRINT A(Z),'
 70 NEXT Z
 80 RETURN
100 REM swap two items
110 LET T=A(N)
120 LET A(N)=A(N+1)
130 LET A(N+1)=T
140 RETURN
200 FOR N=1 TO 19
210 IF A(N)>A(N+1) THEN GOSUB 100
220 NEXT N
230 RETURN
1000 LOAD d1"Array" DATA A()
1010 CLS
1100 GOSUB 30: GOSUB 200
1200 GOSUB 30
```

As you will see when you run the program all the smaller numbers seem to ripple upwards as the larger numbers are moved towards to end. Running this program once moves the largest number to the end. So if we repeat it 19 times we will have sorted the array. Merge the lines of program 3A to do just this and run it.

```
 10 REM PROGRAM 3A.
300 REM do the sort
310 FOR M=1 TO 19
320 GOSUB 30
330 GOSUB 200
340 NEXT M
350 RETURN
1100 GOSUB 300
```

This is a simple example of a **RIPPLE** or **BUBBLE SORT.** It may seem strange that the loop on line 300 counts downwards but there is a good reason for this. You should see that it is a waste of time to do the comparisons for the last numbers every time as we have already got them sorted. So by changing the program not to do this by merging program 3B the sort is made much faster.

```
 10 REM PROGRAM 3B.
200 FOR N=1 TO M
310 FOR M=19 TO 1 STEP -1
```

There is still a problem. Run the program and then type **GOTO 1010.** This will rerun the program without reloading the data. The program will try and do the entire sort again even though the array is sorted. The next improvement would be to stop the program once the data is sorted. If you now merge program 3C and the extra lines will do this. Run the new program. Then once the data is sorted type **GOTO 1010.** The reason it is so much faster the second time is that if no numbers are swapped then the program knows that the data is sorted. The **NEXT M** on line 320 will not be executed and the program will stop.

```
 10 REM PROGRAM 3C.
135 LET DONE=0
315 LET DONE=1
340 IF NOT DONE THEN NEXT M
```

This is about the best you can do with a bubble sort. There is, however, another type of sort called a swap sort that can be much faster. There is an example of this in program 4. (Do not merge this program with the other type it all in).

```
 10 REM PROGRAM 4.
 20 GOTO 1000
 30 REM printout
 40 PRINT AT 0,0;
 50 FOR Z=1 TO 20
 60 PRINT A(Z),'
 70 NEXT Z
 80 RETURN
100 REM swap two items
110 LET T= A(N)
120 LET A(N)=A(M)
130 LET A(M)=T
```

```
 140 RETURN
 300 FOR M=1 TO 19
 305 GOSUB 30
 310 FOR N=M+1 TO 20
 320 IF A(N)<A(M) THEN GOSUB 100
 330 NEXT N
 340 NEXT M
 350 RETURN
1000 LOAD d1"Array" DATA A()
1010 CLS
1100 GOSUB 30: GOSUB 200
1200 GOSUB 30
```

This time instead of comparing every number with the one next to it we compare every number with the first. If it is smaller then we swap them. After the first time through the loop 310 to 330 the smallest number will be at the top of the array. The next time we do the loop (lines 310 to 330) we compare every number with the second in the array. This continues comparing with the third forth fifth and so on. Of course this still has the same problem as the simple bubble sort in that it does not stop when the sort is done. However, we can still make it faster by each time through the loop, if we do not swap the number with the smallest number, then we test it against the last number. Then reduce the loop from both ends. If you merge program 4A you will see how this works.

```
  10 REM PROGRAM 4A.
 150 LET T=A(N)
 160 LET A(N)=A(SZ-M)
 170 LET A(SZ-M)=T
 180 RETURN
 300 FOR M=1 TO INT (SZ/2)
 310 FOR N=M TO SZ-M
 320 IF A(N)<A(M) THEN GOSUB 100: G
OTO 330
 325 IF A(N)>A(SZ-M) THEN GOSUB 150
1020 LET SZ=21
```

For a complete sort this is almost twice as fast as the bubble sort but the bubble sort can be faster if the data is not badly out of order.

Both the sorts discussed so far are designed primarily to sort an array full of data, but what if we already have the data sorted and just want to add another number. Both the sorts shown are able to do this as can be seen if you merge program 3D with the bubble sort and program 4B with the swap sort. As you see one is much better than the other. You should also note that if you were sorting in the opposite order by changing the > and < in the loop then it works better if you add the new number at the end.

```
  10 REM PROGRAM 3D.
1000 DIM A(20)
1030 FOR L=1 TO 20
1040 LET A(1)=RND*100
1210 NEXT L
```

```
  10 REM PROGRAM 4B.
1000 DIM A(20)
1030 FOR L=1 TO 20
1040 LET A(1)=RND*100
1210 NEXT L
```

It should come as no surprise to find that there is a third type of sort that is designed just for adding new data into an already sorted array. This is called an insert sort because that is what it does. There is an example of one in program 5.

```
  10 REM PROGRAM 5.
  20 GOTO 1000
  30 REM printout
  40 PRINT AT 0,0;
  50 FOR Z=SZ-20 TO SZ
  60 PRINT A(Z),'
  70 NEXT Z
  80 RETURN
 100 REM make space
 110 FOR R=2 TO POS
 120 LET A(R-1)=A(R)
 130 NEXT R
 150 LET A(POS)=NUM
 160 RETURN
 200 REM do insert
 210 FOR N=SZ TO 1 STEP -1
 220 IF NUM<A(N) THEN NEXT N
 230 LET POS=N
 260 GOSUB 100
 270 RETURN
1000 LET SZ=32: DIM A(SZ)
1010 CLS
1020 FOR L=1 TO 20
1030 LET NUM=RND*100
1100 GOSUB 30: GOSUB 200
1200 GOSUB 30
1210 NEXT L
```

You will see that this time the new number is not placed into the array but is held in a separate variable. This is then compared with each number in the array until its place is found. Then all the numbers are shuffled along to make room. This shuffling can be made much faster if we keep track of where the first number is. Then we do not need to shuffle all the unused items in the array. If you merge program 5A you will see how this speeds things up.

```
  10 REM PROGRAM 5A.
 110 FOR R=NXT TO POS
 140 LET NXT=NXT-1
1010 CLS: LET NXT=SZ
```

Just think of what a difference it would make if there were 1,000 items in the array. Finding the correct place for the new number can also be made much faster provided that you are prepared to have an array that is a power of 2 long. (Powers of 2 are 2, 4, 8, 16, 32 etc). So if you want to hold 1000 numbers use an array of 1024. The reason for this is that if you have 1024 numbers you can find the correct position with only 10 compares. This search method is called a binary chop.

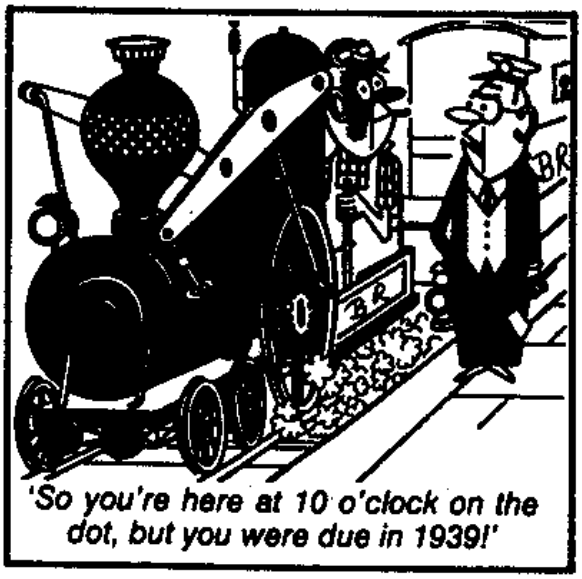
Imagine we have 1024 elements in the array that are already sorted in to the correct order. First we compare the new number with the 512th number, that is half way along the array. Now as all the numbers are in order if the new number is greater than the 512th it must also be greater than all those before the 512th. In the same way if it is smaller than the 512th then it must also be smaller than all those after the 512th. So now we only need to consider the last 512 elements of the array. We now compare with the (512+256)th element, again half way along and once more we can discount half of what is left and each time we do a compare we half the number of elements left to test. After only 10 tests we will have found the correct place for the new number.

There is an example of this in program 5B. This uses an array of only 32 elements but if you replace the 32 on line 1000 with 2048 you will probably not notice any difference in the speed even though the array is now 64 times larger.

```
  10 REM PROGRAM 5B.
 200 REM do insert
 210 LET N=SZ/2
 220 LET POS=1+N
 225 LET N=INT (N/2)
 230 LET POS=POS+N*(NUM>A(POS))-N*(
NUM<A(POS))
 240 IF N<>0 THEN GOTO 225
 250 LET POS=POS-( NUM < A(POS))
 260 GOSUB 100
 270 RETURN
```

So to summarise. You now know of three types of sort BUBBLE, SWAP and INSERT. Which you use will depend on what you intend to do with the data and how you get it. If you have arrays that already contain much unsorted data then the swap sort will usually be best but if much of the data is in order then a bubble may be better especially if you intend to add or change small amounts of numbers. The binary chop is easily the fastest way of building an array by adding numbers one at a time and definitely so if the array is large. Of course the final choice is up to you as the programmer, and you can use more than one type of sort on the same array.

I hope this has been of interest to you and if there are any area of programming that you would like me to write about then please write to me via FORMAT and I'll see what I can do.



'So you're here at 10 o'clock on the dot, but you were due in 1939!'

INTRODUCTION TO

* FORTH *

Part 2.

By: Alan Cocks.

In my previous article on Forth I explained something about it and some of the obstacles I had come across when trying to learn it. One I did not mention was the apparent difficulty in obtaining tuition, although I did manage to find evening classes which were just within travelling distance.

The previous article ended with the "stack" being put through an example like:- 7 3 42 <ENTER>. + . <ENTER> You may remember that spaces are used to separate numbers or characters. The full stop ("dot") is the symbol which takes a number from the top of the stack and displays it on the screen.

The stack can be emptied by systematically entering . <ENTER> until the "stack empty" error message appears. The stack will also be emptied automatically when any error message appears; this gives a "fail safe" approach. The mention of error messages brings to mind that Forth does not go in for error checking much at all, the idea being that to give software that "lean, hungry, fast" look, the soft springing is dispensed with. Things are left to you as the programmer to insert checking.

Back at the stack; you may have tried using the '/' character (divide) in the example sequence, in place of the '+' shown, and found that the answer is given as a whole number, rounded towards zero. In this case the remainder left over from the division is lost, although it would have been retained if a slightly different approach had been used ('/MOD' also uses the second position on the stack; for the adventurous).

The stack is like a railway station; lots of arrivals and departures, and spending too long there can be tedious.

Let's move on. Forth brings memory addresses "up front", frequently putting them on the stack or expecting to find them there and using them. The sequence **PAD <ENTER>** will place on top of the stack a memory address of a "scratch pad" area which is safe to use. That is, the rest of Forth is not expecting to use it. To read this address, use **U. <ENTER>** This puts the address on the screen and removes it from the stack as usual. "U." (pronounced "U dot") treats numbers as positive unsigned numbers, and is used for addresses. Now put the number 327 into the memory location of PAD as follows:-

327 <ENTER> PAD <ENTER> ! <ENTER>

This introduces the character '!', used in English as the exclamation mark, and used in Forth to store a number at a given address (called "store").

What has happened in this second example is a sequence of three things. First 327 is entered on the stack. Then PAD leaves the pad address on top of the stack, pushing 327 onto second place. Then ! ("store") goes to the required address and puts 327 in that location. Both 327 and the pad address are removed from the stack during this process. This is more elegantly described by this form of notation:-

327 (...327)
PAD (327...327 address)
! (327 address...)

The contents of the brackets indicate the stack contents before and after each operation, with the operation

shown as '...' DON'T PANIC. It's all perfectly simple.

For example, the line showing PAD reminds us that before PAD was used, 327 was top of stack. After PAD, the address was top of stack with 327 pushed down to second place. After '!' was used the stack was empty again. This form of notation is very useful indeed and is worth cultivating.

What has happened in this second example is that actions were taken in Forth, as the entries at the keyboard were made. In order to compile a routine, all that it is needed is to begin the sequence with ':' (colon) and also the name of the routine, and end the sequence with ';' (semicolon).

As a third example, let us compile the previous sequence, naming it "STORETEST" as follows. Remember to follow the colon with a space, and also to follow the word STORETEST with a space etc.

: STORETEST 327 PAD ! ; <ENTER>

This routine will now have been compiled. When it is activated by keying **STORETEST <ENTER>** it will have the same action as before. You may like to try this.

"Ah!" I hear you say. "How do I know that 327 really is stored at the correct address?". OK, you don't have to just believe it. There is a routine called '@' (pronounced "fetch") which goes to the memory address of your choice and brings to the stack a copy of what number it finds.

The fourth example uses '@'

: FETCHTEST PAD @ . ; <ENTER>

The routine compiles as you enter it. At compilation, the total amount of compiled code increases slightly, and the memory location allocated to PAD is shifted, so before doing anything else, type STORETEST <ENTER> to store the number at the new position, then FETCHTEST <ENTER> and the response on the screen should be '327'.

A more elegant way of setting out the sourcetext for FETCHTEST would be as follows

```
: FETCHTEST   ( ... )
  PAD         ( ... Pad address)
  @           (Pad address ... 327)
  .           ( ... )
;
```

This indicates how the routine - FETCHTEST - expects to begin with the stack empty and will end by leaving the stack also empty. PAD causes an address to be left on the stack. The routine '@' ("fetch") expects to find an address on the stack, copies the contents of that location to the stack. Then they are displayed on the screen, leaving the stack empty.

You may not be surprised to learn that ':' (colon) turns the compiler on, and ';' (semicolon) turns it off.

I made mention of an Editor previously. A good Editor will allow the entry of source text which looks just like the elegant example of FETCHTEST above. This can be of great benefit in developing programs. The Forth standards do not include such an editor. The various standards are well respected starting points for Forth. The Forth Standards Team have specified these from time to time; also well used is FIG Forth (Forth Interest Group).

As mentioned in the first article, a version which is limited tightly to a standard is not really suitable for beginners. This is because the standards assume that you will yourself write some routines for handling strings and numbers. As a beginner this can be hard work.

The standards you are most likely to come across are Forth 79, Forth 83, and FIG Forth. They have a lot in common, most of their routines are identical. In one or two areas there are differences, and when these become important to you it is a clear sign that you are moving away from being a beginner in Forth.

More next month.

BEYOND SIMPLE UDGs

Part 3.

By: Clyde Bish.

In this (and the next) issue we'll be moving into the Big Time. No more minipics. We'll be talking Full Screen Extravaganzars. And I'll be showing you how to produce your own "Mugsy"-type game.

But let's start with smaller fry. How about a 16 x 8 character illustration top centre screen with room on each side for scores, strength etc. and scrolling text underneath? Firstly the pictures. You obviously get to design these yourself. (Try two simple ones first. Leave your upgrade of "Runestone" until later!). Once you have these drawn out on a suitable pixel/character square grid help is at hand to store and colour them. Load in the "character generator" program and convert each character square along the top row to a udg, then save these to tape. (remember you only want A to P). Repeat the process for the second and subsequent rows, and repeat for the second picture, saving each of the codes in order, one after the other on the tape (in these articles I will refer to tape but you could use Microdrive or disc just by altering the relevent SAVE/LOAD instructions). Now you need a utility to store them as complete picture in high memory, and also add colour information. Program 1 is what you need.

Type it in, SAVE for future use then RUN. Answer the first prompt with the number of pictures to store. (In this case 2). The program clears the required space and tells you the address of the start of the first picture. Make a note of this (and subsequent addresses). You'll need them later. You'll then be invited to play the data tape you made with those blocks of 16 udgs. The program will load these in to the correct addresses. When the eighth block of a set is loaded stop the tape, and add colour. You'll see your picture displayed with the first characer square flashing. Input the attribute for this and subsequent squares. You can work them out using the table in the last issue (or by INK + PAPER * 8 + (128 for FLASH) + (64 for BRIGHT) if you don't have it). If the attribute is the same as the previous square just press ENTER. When all 128 have been filled in you'll move to the next picture to repeat the process, then save the combined code, ready to load back to the correct address.

To use this data you'll need a short machine code routine to move it instantly to the Display and Attribute Files. Use "codeloader" (from my last pair of articles) to enter the data from Table A, changing the FOR/TO values to 65324 and 65367, then save as **SAVE "downcode"CODE 65324,44.**

I'd like to explain how the routine works but space is fast running out so I'll press on and give you the routine you'll need to have in your program to make use of it. One important point, though. You must set the variable n to the start of the required picture data (noted from Program 1) before calling it. Assuming that you did just save two pictures, program 2 will allow you to switch instantly between them. Type it in and try it out (making sure you still have your picture data and "downcode" on board). Fast wasn't it?

Now for the promised scroll. This is called from the ROM using **RANDOMIZE USR 3583.** It scrolls only the bottom 11 lines, so anything on the top half of the screen, for example, your illustration is quite safe. Add the following lines to program 2:-

```
15 RANDOMIZE USR 3583: PRINT AT 21,0;"
   This is picture 1"
25 RANDOMIZE USR 3583: PRINT AT 21,0;"
   This is picture 2"
```

then change between them a few times and you'll see the effect.

Now a move to the Big Time. But first the bad news. Big pics mean large memory blocks in which to store them. 6912 bytes (memory spaces) to be precise if you want them in glorious Sinclaircolor. If you're envisaging a comic strip, "Mugsy"-style, driven by a relatively short program this may not matter. Let's assume for a moment that it doesn't and see how we can store and recall the illustrations.

First of all, how many can you store in memory? Answer - 5, with about 6k left for the driver program. ("128" owners will obviously do much better than this). Next, how to store them. In principle what you have to do is to transfer the contents of each byte of the display file (D FILE) and attributes area (ATTR) where the on-screen picture is stored into high memory, knowing where you're putting it. You could do this by PEEKing each byte, then POKEing the contents up, but this would be quite slow. The solution, a short machine code routine which uses the instruction LDIR. This stands for 'Load / Decrement / Increment / Repeat' which probably doesn't make you much the wiser. In essence, you set one counter to the number of bytes to be transferred, another to the start of the D FILE, and a third to the destination address. LDIR does the rest, continually transferring bytes until the first counter is reduced to zero.

The necessary code is included in Program 2, which will make the transfers for you, modify the code to make it work in reverse, then save it along with the picture bytes. Type it in, and let's try it out. If you have some pics ready made with a drawing utility you could use those. Otherwise the "rainbow" and "logo" screens at the beginning of your Horizons tape if you have it) would do nicely. Generate then SAVE the first with **FOR F=1 TO 704: PRINT "A";: NEXT F: SAVE "A"SCREEN$.** then change the "A" to "B" and repeat the process.

RUN the program, and you'll be asked how many pics you want to save. Let's try two. Answer the prompt for a title with the first pic, then play the tape. The picture will load in and more or less instantly be transferred to high memory, the address of which will be given. Make a note of this, and the POKE number as well. You'll need this information later. Repeat for the second picture, and there you are. You can then save the data for the two pics plus the machine code to drop them back down.

To make use of this routine you will need to include a subroutine in your own program such as **9999 POKE 65358,h: RANDOMIZE USR 65356: RETURN** having set variable h to the POKE number you noted for that picture before you **GOSUB 9999.** Try it, but don't blink at the wrong moment. You'll miss it!

PROGRAM 1.

```
 10 INPUT "No. of pictures to store? "
    ;P: POKE 23681,P
 20 LET D=65367-(P*1152+44): RANDOMIZE
     D: POKE 23728,PEEK 23670: POKE 23
    729,PEEK 23671: CLEAR D-1: LET D=2
    56*PEEK 23671+PEEK 23670: LET S=D:
     LET P=PEEK 23681: FOR I=1 TO P: C
    LS
 25 PRINT "Picture ";I;" starts at ";D
 30 PRINT "PLAY TAPE"
 40 FOR F=1 TO 8: PRINT AT 4,0;"Loadin
    g row ";F: LOAD ""CODE D+(F-1)*128
 60 NEXT F: PRINT "Stop the tape": PAU
    SE 200: CLS
 70 LET C=0: FOR F=D TO D+1020 STEP 12
    8: RANDOMIZE F: POKE 23675,PEEK 23
    670: POKE 23676,PEEK 23671: PRINT
    AT C;8;"ABCDEFGHIJKLMNOP": LET C=C
    +1: NEXT F: REM Caps in quotes are
     udgs
 80 LET D=D+1024: FOR F=22536 TO 22760
     STEP 32: FOR N=F TO F+15: POKE N,
    135
 90 INPUT "attribute? ";A$: IF A$="" T
    HEN POKE N,A: GOTO 110
100 LET A=VAL A$: POKE N,A
110 POKE D,A: LET D=D+1: NEXT N:NEXT F
```

```
120 NEXT I
130 INPUT "title for code? ";A$
140 SAVE A$CODE S,P*1152
```

TABLE A.

33	0	125	17	8	64	62	0	1
16	0	237	176	6	16	19	16	253
60	254	64	56	241	17	8	88	62
0	1	16	0	237	176	6	16	19
16	253	60	254	8	56	241	201	

PROGRAM 2.

```
10 INPUT "No. of pictures to store? ";N
20 POKE 23728,N: LET A=65367-N*6912-12: REM replace 6912 with 2048 for one third of screen, 4096 for two thirds, 6144 for D_FILE only
25 CLEAR A: LET N=PEEK 23728: LET A=65367-N*6912-11: REM 6912 replaced as above
28 LET S=A: PRINT "CLEAR ";S-1;" before LOADing": FOR F=65356 TO 65367: READ I: POKE F,I: NEXT F
30 FOR F=1 TO N: INPUT "title of pic ";(F);"? ";A$: LOAD A$SCREEN$
40 RANDOMIZE A: LET L=PEEK 23670: LET H=PEEK 23671: POKE 65357,L: POKE 65358,H: RANDOMIZE USR 65356
50 PRINT "Data start for pic ";(F);" = ";A'"(POKEs : ";L,H;")": LET A=A+6912: REM replace 6912 as before
60 NEXT F: POKE 65356,33: POKE 65359,17
70 STOP : SAVE "pics"CODE s,n*6912-1
1: REM replace as before
100 DATA 17,0,0,33,0,64,1,0,27,237,176,210: REM replace 27 with 24 for D_FILE only, 16 for two thirds, 8 for one third only. Replace 64 with 72 for middle third, 80 for bottom third
9999 REM Call with POKE 65358, high byte of data start address: RANDOMIZE USR 65356
```

Watch out for next months instalment.

YOUR LETTERS

STAR*LETTER* *STAR*LETTER

Dear Editor,

Why do MGT neglect the DISCiPLE? First they produce the TWO-FACE which they say is not DISCiPLE compatible, then the release PICK-POKE-IT which will only work on the PLUS D. Have they totally forgotten the DISCiPLE?

Yours Sincerely, Hans Sleeman.

I dont think MGT are neglecting DISCiPLE users but it must be remembered that it was Rockfort Products that marketed the DISCiPLE so MGT are bound to concentrate their efforts on their own interface - the PLUS D. There is also a logical explanation for not producing, in DISCiPLE form, the two items you list.

First the DISCiPLE has its own 'Built In' inhibit button and 'Through Port', so most hardware will work without the need for an extra two way switching box (which is what the TWO-FACE really is).

Second, PICK-POKE-IT, the reason this can never be converted to DISCiPLE is that there just is not the room available in the DISCiPLE's RAM area for the program to work. When I did the updates to GDOS 3d last year it took ages to find a few bytes here and a few bytes there to hold the extra code needed.

So dont be too hard on MGT, after all they will still repair DISCiPLEs and look after DISCiPLE owners in many other ways. ***Ed.***

Dear Editor,

Thank you for a marvelous magazine, I cant wait for each months issue to drop through my door. However there is one thing missing from your coverage of the Spectrum. Could someone write an article on using modems? I have a VTX5000 but, apart from Micronet, I dont know what to do with it.

Yours Sincerely, John Freemore.

Dear Editor,

I've seen mention in your magazine of the Timex-Sinclair 2068 computer. Is this available in the UK? Do you have any more details?

Yours Sincerely, K.S.Hardy.

The TS2068 was the American version on the Spectrum. It had extra memory, three screen modes, built in ROM socket and a better keyboard than the early Spectrums. Alas, due to the dithering of Timex it was released nearly two years after the Spectrum appeared in the UK. Its sales, although quite large, were disappointing to Timex and ultimately Timex pulled out of the home computer market in the USA.

However a European version was produced by Timex Portugal. This had a PAL TV modulator and 220/240 volt transformer but was only available (as far as I know) in Portugal.

Perhaps one of our American readers can supply more information on what the computer is capable of. ***Ed.***

Dear Editor,

Can I appeal to readers for help in finding a program to help with Genealogy. Thats research into family trees if you dont know already. I'm also interested in Astronomy but have only been able to find Star Seeker from Mirrorsoft so far. Are there any others?

Yours Sincerely, Colin Mac Kenzie.

Dear Editor,

Thanks for last months TOTAL RECALL program, it was worth the whole years subscription just on its own.

Yours Sincerely, Brian Candy.

Letters printed may be edited for length or clarity. The writer of each months STAR LETTER wins an EXTRA 6 months subscription to FORMAT.

SMALL ADS. SMALL ADS.

PLUS D issue 2 + tape + manual £30. Dual 40/80 DS/DD Cumana disc drives £125. Unused Dysan 5.25" discs box of 10 £7.50 Used discs 10 some with software £5. Original software with manuals - Kemsoft PCB Designer v2 £15. Number One's Electronic Circuit Analyser £20. Call Stuart Williams on Luton (0582) 458799.

Games, Books etc for Spectrum send SAE for list. Philips Videopak games computer and games pack £25. S.Sunderland, 54 North Road, Three Bridges, West Sussex, RH10 1RK.

SERIOUS SPECTRUM OWNERS wanting UTILITIES PROGRAMMING HELP, PRACTICAL software and USEFUL articles, GRAPHICS, INFO, IDEAS and MORE from LIKE-MINDED THINKERS, TRY...

OUTLET monthly on MDRIVE, OPUS, DISCIPLE, PLUS D, TAPE

£2 gets YOUR FIRST issue! A blank disc or cartridge (not cassette) gets a FREE demo!

CHEZRON SOFTWARE, 605 LOUGHBOROUGH ROAD, BIRSTALL, LEICESTER LE4 4NJ

PLUS D HACKER Advanced On-Line hacking tool for the PLUS D only. Works with 48k or 128k Spectrums. New advanced version. As reviewed in Your Sinclair April'89. Send cheque for £4 (£5 overseas) to Steve Nutting, 7,Narrow Close, Histon, Cambridge, CB4 4XX.

DISC CONVERSION SERVICE. Got a game that wont snapshot? Send stamped addressed envelope for list of converted games. By Hugh (Hack Zone) McLenaghan, 36 Floorsburn Crescent, Johnstone, Renfrewshire, PA5 8PF.

WANTED Volex TTX 2000S Teletext Adaptor with software and manual. H.D.Rothwell, 49 Belgrave Rd, Great Boughton, Chester, CH3 5SA. Tel 0244 311327.

TAPESNAP will transfer snapshot files to tape and allow you to reload even without your disc system. Available in two versions TAPESNAP 48 for 48k snaps and TAPESNAP 128 for 128k snaps.
The programs are supplied on tape for auto-save to disc. Price - £4 each or £6 for both incl UK recorded delivery (Overseas +£1). Please send postal orders or cheques (payable to S.Young) to Shimon Young, 21 Colchester Road, Southend-on-Sea, Essex, SS2 6HW.
NOTE: Until late July orders may be delayed due to imminent exams.

Can you

Write programs in Z80 Machine Code ?

Have you

Written any good Spectrum Programs/Utilities in Code or Basic ?

THEN CONTACT DAVE HOOD AT

BETTERBYTES

YOUR ADVERT

Buying, Selling, Pen Friends, etc.

Any PRIVATE advert, up to 30 words (subject to acceptance), will be printed FREE in the next available issue. Any software sold must be original copies, in working order and with full instructions. The publishers will not be held, in any way, responsible for adverts in this column. Trade advertisers should contact the publisher for rates.

· NAMES & NUMBERS ALL YOUR DISCS ·

EASY TO USE

THE ORIGINAL

Disciple DISC MANAGER

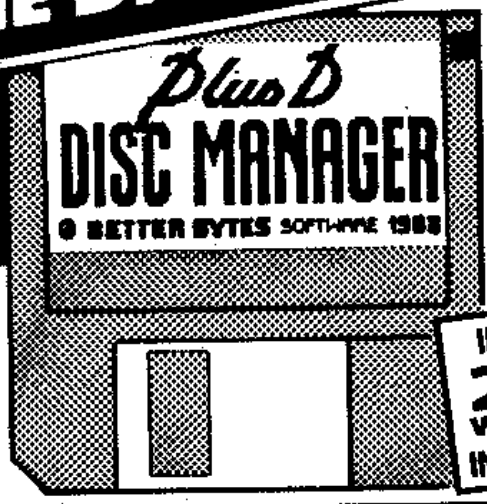
DISC MANAGER

FOR THE DISCIPLE & PLUS D

PERSONALISE YOUR DISCS · AUTOMATIC CATALOGUER · FAST SEARCH & LOAD · PULL DOWN MENUS

· SPECIAL PRINT-OUT PROGRAM · 16 PAGE MANUAL · FLEXIBLE · GREAT GRAPHICS

A MUST FOR ALL DISCIPLE & PLUS D USERS



NEW UPGRADE VERSION

INCLUDES THE BEST EVER **2 WAY SEARCH** IN M/CODE FOR SPECTRUMS

GET TOTAL CONTROL OF YOUR DISC COLLECTION

CUSTOMERS THROUGHOUT THE WORLD – HERE'S WHAT SOME OF THEM SAY...

'IMPRESSIVE' A.S., NEW ZEALAND

'MARVELOUS GRAPHICS' I.A.R., RENFREWSHIRE

'A DELIGHT TO USE' F.W.J., CHIPPENHAM

'VERY USEFUL' R.C.M., NETHERLANDS

'EXCELLENT' R.H. MIDDLESEX

'ESSENTIAL' R.W. CALIFORNIA

The NEW upgrade version of the original DISC MANAGER is the most powerful programme ever written for the DISCIPLE/PLUS D.

Designed to take advantage of Disc Drive ownership, the Manager keeps track of all the programs on all your discs, offering unrivalled benefits and features.

- Storage of up to 27,000 records on one Disc, or 79,920 total. Random File Access.
- Name & Number Discs with fast Autonumber and user pre-defined titles features.
- No typing in of Data. Press a key and Discs are automatically added to appropriate catalogue.
- Fastest ever M/Code Search. 2 modes – Search and Load or Search and List all occurances, then select program to load.
- Special program to print the contents of all your discs with fast sort option. Can selectively print by disc type or number.
- List contents of any disc with ease.
- Multiple erase or rename options.
- Plus many other unique features.
- Comes with 16 page manual and demonstration catalogues.
- Operates with 48K or 128K Spectrums.

Send for the DISC MANAGER today... and you'll soon wonder how you ever Managed without it!

NORMAL PRICE £14.95

SPECIAL INDUG PRICE ONLY

£12.95 INC. P&P

OVERSEAS ORDERS PLEASE ADD £1.50

UPGRADES Existing users can upgrade to the NEW Version.
Send old Manager Disc plus £5.00

STOP PRESS NEW NOW AVAILABLE **DISC ORGANISER**

Recover Erased Programs · Re-organise Directory · Repair Sectors · Wild Card Copy, Erase or Re-name ·
Supplied on Cassette ONLY **£5**

PROBABLY THE LARGEST PROGRAM EVER WRITTEN FOR THE SPECTRUM! COMPRISES

12 PROGRAMS

6 MAIN – 6 SUPPLEMENTARY
A total of 36 files supplied on disc – over 250K of Program Magic!

BETTERBYTES

10 SPITTAL TERRACE · GOSFORTH
NEWCASTLE UPON TYNE NE3 1UT · TEL: (091) 285 6185

· SUPPLIED ON 3·5 OR 5·25 DISC · WRITTEN SPECIALLY FOR DISCIPLE/PLUS D ·

PLEASE STATE YOUR DRIVE TYPE, SIZE, ETC, INDUG MEMBERSHIP No. AND IF REQUIRED FOR DISCIPLE OR PLUS D