NATGUG NEWS

Volume 7 Issue 11 MAY 1986

OFFICIAL JOURNAL OF THE

National TRS-80

& Genie Users

Group.

INFORMATION ON THE GROUP

Membership of the group is by subscription to the Newsletter, which is published monthly. Membership details are obtainable from the Group Secretary. Membership of the group is open to anyone with an interest in computers but special emphasis is placed on equipment in the TANDY range.

Details of the Group accounts, and the constitution of the Group, are available from the Secretary.

Members requiring assistance with problems related to the TRS-80 / Video Genie may call the Secretary. An attempt will be made to put them in touch with a member who can help with the problem.

Workshops are arranged from time to time in various parts of the country.

Sub-groups exist in many areas. A list is provided in the Newsletter from time to time.

The Group maintains two software libraries (Models I and II) which are free to members. Library lists are available from the Secretary.

For confidentiality reasons, the membership list is not generally available, but members may ask the secretary for a list of members in their area, and mailshots to all members may be arranged.

Back numbers of the Newsletter are available from the Secretary.

Please send all contributions for the Newsletter to the Editor, on disk if at all possible (5.25", NEWDOS-80 v2 or Montezuma Micro CP/M prefered, any combination of density, sides or tracks, but please say what it is). Your disk will be returned.

Newsletter Editor
Geof Smith
17 Homefield Rd.,
Bushey,
Herts WD2 3AP
01-950-6345

Secretary & Newsletter Publisher
Brian Pain
24 Oxford Street,
Stony Stratford,
Bucks. MK11 1JU
0908-564271

CONTENTS

EDITORIAL

Apologies for the slight delay in producing this issue of the newsletter, but I've been soaking up the sunshine in a somewhat more pleasant climate than that which prevails here at present. There seemed little point in putting the newsletter together prior to my holiday since only ONE contribution has arrived. On my return this situation had improved to the grand total of three articles but even as I write this there have been no letters/comments at all this month. It is not very easy too judge if you are happy with content etc. if there is no feedback - Enough chastisement for the present.

When I took over the role of Editor of the Newsletter it seemed to me that the one of things I could do was to try and improve the relationship (what relationship ?) between TANDY UK and the NATGUG. On the advice of Os House I wrote to John Sayers, the managing director of TANDY UK. A copy of my of my letter and his reply are reproduced below.

Dear Mr Sayers,

I have recently taken on the post of Editor of the Newsletter of the National TRS-80 User Group. The group has done much in the past to aid TRS-80 users in both the hobbyist and business fields but the number of members has dwindled dramatically in the last few years and at present is just hovering on the limits of viability. Those who are left are in the main what I would term semi-professionals or skilled amateurs, most of the members either use Tandy equipment to run small business concerns or are computer professionals who use Tandy equipment as an adjunct to their work involvement. This represents a considerable core of knowledge of Tandy equipment, operating systems and software that will be dissipated in the next year or two unless interest in the group can be revived. In the past there seems to have been an unwillingness on the part of Tandy UK to acknowledge the existence of the group and doubtless there were commercial reasons for this. However, in the current climate I would have thought that the ability to refer clients to a source of free (to Tandy) expertise would offer distinct advantages. To this end I would ask if it is possible as a general policy for branch managers to mention the existence of a Tandy User Group when making a sale of computer equipment and perhaps give a telephone number where the group secretary can be contacted (Mr. B. Pain 0908 564271).

Additionally, as editor of the newsletter I would like to write informed editorials on new Tandy equipment and software. This would be made very much easier if I could receive any press releases and other product information for incorporation into the newsletter,

yours sincerely,

Dear Dr. Smith,

Thank you for your letter of the 4th may regarding the National TRS-80 User Group and the work done by yourself.

I certainly would be prepared to instruct our stores of the work carried out by the User Group. I presume the address on your letter is the one to which all communication should be sent. I will let you have a copy of the recommendation to our store managers notifying them of your Group as soon as this has been circulated.

As regards to your final paragraph, I will ensure that our Merchandising Department keeps you informed of the new Tandy equipment and software that is coming to the market.

Thank you for taking the time to write. Can I perhaps suggest that you contact Mr. Ted Russel, Director in charge of Tandy Computers, and maybe he can arrange regular meetings between our technical support and merchandising personnel and your group,

Yours sincerely,

J.R.C. Sayers,

I found this reply to be highly encouraging and think we should all to our utmost to foster this relationship. With respect to Tandy equipment, two of the latest to appear are the TANDY 3000 and the TANDY 600. The 3000 is the TANDY rival to the IBM PC/AT and Ariela Taylor who is buying one of these for her hospital department has promised a review in the near future. Initial impressions are that it is a nice piece of equipment and very, very fast. The 600 is TANDY's new MS-DOS lap-held portable, with a built-in 3.5" drive. I'm not sure that there are any in the country as yet but would be interested if any one has seen any initial reports. Will it be as successful as the Model 100 I wonder, since there is a lot more competition in the lap-held market now.

In this newsletter we have one of the first reviews that I have seen on an extended memory board for the Model 4. The degree to which you can extend 8-bit technology does not cease to amaze me. Six or seven years ago a 48K upgrade for the Model 1 seemed out of this world. Now we have 1 megabyte for the Model-4. In this context Trevor Hutchinson (08206 23996) has just finished a prototype 64K/256K board for the Genie III. I have been inveigled into attempting to get some CPM-software together to implement a RAM disk for the board, and I hear tell that a certain Belfast guru may be doing the same for NEWDOS-80. Trevor intends to expand the board's capacity to 1 Megabyte and also envisages that the board will be adaptable to the Model-4. We certainly hope to have some demonstration kit running by next Swindon.

SORTING DATA WITH DSM4

Some time ago I was asked to print a list, in Alpha and Numerical order. As such it was an easy job for SIR but I ran into two problems. The data was handwritten and had many foreign names which looked odd, even when spelt correctly. The lists had to be printed on sheet paper with each letter of the Alphabet starting on a new page and 1.5 line spacing for easy reading.

The problem of poorly written data is that it makes for errors which have to be corrected. SIR requires that a field is re-entered if one letter is wrong and this leads to more errors. I also found that some of the numbers had a prefix letter and this added to the complications.

All this made a simple job much more difficult and when the next one arrived, I looked a little harder before I started. With over 2,000 records and each one almost filling a line at 12 chr per inch, I had nearly 200K of data. That alone is more than a standard disk will hold and splitting the file between disks makes life very difficult when the whole file needs to be sorted.

I opted to use a home-made program because that way I could control the results but on the other hand I lost some of the facilities that the commercial programs offer. The solution is to make a hybrid program with a mixture of Basic and standard packages.

One very good way of entering data is to use a word processor but few offer a Sort routine built in. Where the records are to be printed out in columns, it is easier to use a Calc program. Visicalc and Multiplan are especially good. Visicalc on the model 4 will hold more records than Multiplan but cannot have variable column widths. On balance it would appear that Muliplan was preferable but as neither program will hold 200k, I decided to use neither. My choice was eventually decided for me. Wordstar will hold the full data and retain Tabs but above all it is easier to get a Wordstar experienced typist to enter the data.

The advantage of the model 4p is that it suits the home typist and the on-screen prompts of Wordstar save looking through manuals. On the negative side and possibly the reason for the commercial failure of the 4p was the small drives. A file in Wordstar should not exceed one third of the disk capacity because at any one time it is possible to have three versions of the same file on the disk. This can be reduced by putting the backup file on another disk but this is of limited use on a two drive machine.

An advantage of Wordstar is that it is safe and can easily hold sufficient for one sessions work and the blocks can be merged at a later date. This suits most typists because it takes some 22 hours to put in 200K. By using several data disks I could keep backup copies and make sure the disks did not get too full.

To assist the typist, the data was compressed so that each line could be viewed on screen. This meant inserting spaces before printout but on a large job it is the best way because a little bit of processing at the end can make the entry much easier. All this meant the use of Basic and this is where my experience in CPM gets rather ragged. I chose to change back to TRS-DOS 6.2 for the later stages.

The simplist way to use CPM data under TRS-DOS is to use Hypercross which can read and write under both systems. This allows the data to be transfered without problems and from then on I could use all the facility of TRD-DOS.

At this stage I had all the data on TRS-DOS but in the order that the records were entered. The 200K file was too large for CMD"O" sorts and even an index sort (Newdos 80 or Dosplus 4), could only sort the first three letters. Thus Hong Kong, Honolulu and Honiton were all equal and this was not adequate. The book tells us to sort in blocks and merge the data back. I managed it once but could not face the effort and the time required for a really large job. So I opted for DSM4 which is the Rolls Royce of sorts.

DSM4 is not only a sort routine but half way to being a data base. It will select, delete and sort data in almost any form and very rapidly. Before you can begin, you need to describe your data but Text is always the default. You state the position of the field on which the sort is to start and all other fields in order. Thus you can sort a name like FRED BLOGGS the 6th character followed by the first character. The only restraint being that the data must be in the same position in each string.

Having made up this sort parameter you are invited to save it as a Map file and this can be used over and over again. You can edit the Map file and those that have done sorting before will appreciate the value of being able to change a parameter after seeing the result of the first attempt.

Once the Map file is written, you can get on with the sort and this requires the data to be read from disk. This is where you see the difference between a professional and a novice. My routine took in the records at about one a second and with 2000 records there was time for a cup of tea before finding that I had a sector not found. DSM4 grabbed the records and ticked up the 100's. I checked it with a watch and found that it averaged about 100 X 98bytes in 1.5 seconds. Thus the whole file was accessed in about 30 seconds.

The screen gives an active display as it sorts the blocks and puts them into a work file. A short delay before it starts merging them back and in less than 90 seconds its all over. If the file/sort fields are short enough to handle in one pass then it is even quicker. Because it does not use Basic arrays it can sort much larger files in memory than CMD"O" but the time penalty of larger files is not really worth worrying about.

Unlike CMD"O" and other sorts, the data is not moved and the disk file is still in the same order. The index file gives the order on which the files are accessed. This worried me a little although I could see the advantage because I could have one index in Alpha order and another in Numerical order. I was also pleased to see that there was a simple Basic program which showed the principle of accessing by index.

The index is really a long list of numbers which correspond to the Record number in the data file. Thus the first record in sorted order may be 123 and the next 005. Each number is stored in a string as a two byte integer. Although I roughly knew what that meant I was please to see the syntax given.

```
10 OPEN "R",1,"MYINDX/IND",2:'       Open index file (lrl=2)
20 FIELD 1,2 AS S$:'                    Field the index file
30 OPEN "R",2,"MYDATA/DAT",lrl:'        Open data file
40 FIELD 2,??????????????:'             Field data file
50 FOR L%=1TO LOF(1):'                 For as many records as
                                       are indexed
60 GET 1,L%:'                    Get record number from index
70 GET 2,CVI(S$):'             Get that record from data file
80 'do what you want with data.:' Print or write to new file
90 NEXT L%:'                       continue with next record
100 CLOSE:'                             close when done
110 END
```

The file can be printed via line 80 or it can be written to a new file in sorted order. If the index file in 10 is changed the file can be read in a new order and there is no limit to the number of indexes that can be used. Although DSM4 can sort in either direction it is also possble to reverse the sort in line 50 FOR 1%=LOF(1) TO 1 STEP -1:

A couple of points about DSM4. The sorted index only contains those records that meet with the select criteria. The default is to select all records but you can specify records using logical operators AND,OR,= etc. Another feature allows records to be 'marked delete'. You could mark a record as deleted by having a "*" or some other unique character in a specific place. DSM4 then ignores the record for the sort and can make a note of the deleted record numbers. This allows a record to be deleted and the number be re-used without rewriting the whole file. I found it useful for duplicates where I could add a single marker and from then on the record would not print. If the deletion turned out to be an error it can be undeleted by removing the marker.

I had a few problems on the way but these were nearly all down to me. The first one was when the sort seemed to take a long time. The problem being that I did not know how long to wait. It turned out that there was a small fault on the data disk and the program aborted when it could not read a sector. I thought it had sorted the index and could not understand why it did not work as described. Later I found that the screen is always active and tells you what is going on. When it is sorted, it tells you x records sorted..

The second problem was absolutely my fault in that I left off the lrl=2 in line 10. This did not show up as a Syntax error but it meant that the record number was garbage. All I saw was 'Input past end' or some such error code. I also found one strange fact that has bothered me before. If you use LOF(n) it is possible to find that LOF(n) is greater than the number of reords in the file. I believe it is something to do with the number of sectors/lrl leaving a remainder. DSM4 knows exactly how many records there are and thus I found that LOF(1) in the index file was always correct but not on the data file.

Using DSM4 to do the Sorting, I was well on my way to achieving my printout but I was still left with the need for 1.5 line spacing and forcing a new page for each new letter. The simplist way of getting 1.5 line spacing is to use the variable line spacing in the printer. The Epson and the Quen 1120 will both accept print codes to set the line spacing although both are different. This seems to upset the Forms

Filter so I used a counter to keep the pages to 40 lines and print a Page No as a footer. I also use a second routine to look at the first chaaracter to detect when the B's started and force a new page.

My first attempt started with chr$(65) and incremented one after each letter. Thus after finding the first "A" it looked for "B" but this failed to separate all the letters. It turned out that there were no records starting with "U". I therefore changed it to look for the first letter of the first record and then pick up the change. The new letter then becomes the standard and the program looks for the next change.

To force the new page and still get the footer at the bottom, I found that it was easiest to send CHR$(13). Thus I sent STRING$(40-C,13). As C=the number of lines printed and 40 the lines on the full page, 40-C gave me the blank lines to the end of the page. When I was using normal spacing I just sent TOF and the forms/flt did the rest, but I could not get this to work with non-standard line feeds.

The last problem occured when I found that I could not get all the data on an A4 page and leave sufficient for ring binding. I then resorted to the print codes again. The standard spacing for a 12chr wheel is 10/120th inch. By reducing this to 9/120ths I was able to gain 3/4 of an inch without reducing clarity significantly.

Derek Traylor, Hornchurch, Essex.

OGGY OGGY OGGY

You can tell that Summer is coming because we are having to work hard at producing the programmes for Blandford and Swindon!

The Blandford Workshop, with its legendary House Hospitality, will be held on SUNDAY SEPTEMBER 7th - don't miss it. (Blandford is only a mere 30 minutes from Poole & Bournemouth, so you can drop the rest of the family there and thus make everybody happy !!). Scheduled so far is The Model 4 in a Multi-User Environment, an Introduction to MS-DOS, Model 4 Networking, and demonstrations which could

well include an Amstrad and a Tandy conversing with one another. For those who have not visited Os recently, the Blandford by-pass has made it an absolute doddle - just a few yards up the road signposted for Melbury Abbas. The Blandford Bulletin Board (0258 54494) will be updated with the programme, so get the modems out. Remember, thanks to Os, the Blandford Workshop is F R E E !!!

After that, the only bad news is that Swindon (October 17th-19th) is not also free - but the GOOD NEWS is that prices should be held to the same level as applied in March. Quite apart from that phenomenally popular Systems Room (which appears in itself to justify the whole weekend for many members), there will be hands-on sessions on dBase II, demonstration talks on windowing and using Monte's Window, MBASIC vs CBASIC, and a full session on the Wordstar commands which have become standard for so much CP/M software. Laurie Shields will be there, with a variation on his recent themes, and hopefully Carl will present his D-I-Y "memdisk look-a-like" program.

At the time of writing this article neither schedule is finalised. If YOU have any ideas, thoughts, requests or desires relating to these weekends, then please get in touch. For the Blandford meeting, please ring either Os House or Leo Knaggs, on 0258 53737. Regarding the Swindon weekend, which will of course be held in The Wiltshire Hotel, please ring me on either 0225 61636 (8.00am to 5.30pm) or 0373 72739 (6.30pm to 8.30pm - no later please, when I'm not at a workshop then I go to bed very early!). It would be nice to hear from some Amstrad users about what they would like to see and hear.

Many of us nowadays use our machines for business, and so I leave you with this report from a recent 'Times' :

Computers are now so commonplace in the States that fraud by a keyboard operator or user is almost immediately detected; the art of auditing transactions has been well developed. However, grave concern is now being expressed at the virtual impossibility of auditing the program itself, especially since so few users even possess a listing of their software - worries about that old chestnut of siphoning-off the rounded-up or rounded-down cents have now become extremely serious. The report concluded by stating that so far as the UK was concerned, the potential problem was either just not recognised, or if it was then it was immediately 'swept under the carpet' !!

I hope that Brian has included a renewal notice with this issue - but if not, then he won't mind if you send your cheque in with your name and address on the back !

David Washford, 6 Houston Way, FROME, Somerset BA11 3EU

A USE FOR BOOLEAN VARIABLES

The small Turbo Pascal Pascal program (getrec) overpage is not particularly exciting and was merely written to take individual records from a file produced by Cardbox Intrisic Format and put each into a separate file. Each field in this format is separated by a NULL (ASCII 0) and each record by two NULLS. The interesting piece of the program is the section outlined by (***...***).

This is a REPEAT...UNTIL loop which will exit when the Boolean variable fin becomes TRUE. Line 1 reads a character from the file into the variable letter. The next line (Line 2) illustrates the difference between ':=' and '='. The expression (letter = chr(0)) returns a value of FALSE for every character assigned to letter except ASCII 0. This is then AND'd with the other boolean variable zero that is initially set to FALSE. The result of the AND can be obtained from the truth table:

x	y	x AND y
FALSE	FALSE	FALSE
FALSE	TRUE	FALSE
TRUE	FALSE	FALSE
TRUE	TRUE	TRUE

This value is passed to the loop variable fin. The next line (line 3) then sets zero to true if the character read was a NULL. The end effect of this is that the loop will only exit when two NULL's are read in succession, ie. at the end of each record. Pretend that you are a computer and step around the loop if you don't believe me. Boolean logic might look a little mysterious but can be very useful on occasion.

I have recently done a comparison of Turbo PASCAL versions 2 and 3 on both CPM-80 and MSDOS. Under MSDOS there is no doubt that although version 3 produces larger code it compiles and executes about twice as fast as version 2. However, under CPM-80 the code is not only about 2k larger with version 3 presumably because of a larger run time library, but there is NO advantage gained in either compilation or execution speed. Therefore unless there are any version 3 extensions that you need for your programs, you are best advised to stick with version 2 if running under CPM-80.

Ed

```
program getrec;

var infile,outfile:text;
    inname,outname:string[20];
    letter,ttl:char;
    zero,fin:boolean;
    icount:integer;

begin
write(con,'Enter name of inputfile');
readln(con,inname);
write(con,'Enter initial name for 1st output file ');
readln(con,outname);
assign(infile,inname);
reset(infile);
icount:=0;

REPEAT
   OUTNAME[6]:=CHR(ORD(OUTNAME[6]) + 1);
   ASSIGN(outfile,outname);
   rewrite(outfile);
   writeLN(con,outname);
   zero:=FALSE;

(*******************************************************);
   REPEAT
      read(infile,letter);                    (* Line 1 *)
      fin:= zero and (letter = chr(0));       (* Line 2 *)
      zero:= letter = chr(0);                 (* Line 3 *)
      write(outfile,letter);
   UNTIL fin;
(*******************************************************);

   icount:= icount + 1;
   write(outfile,chr(0));
   close (outfile);
   write(con,icount:5,'records read');
   write(con,'Type any key to continue, space bar to top');
   read(con,ttl);
UNTIL ttl <> ' ';
close(infile);
end.
```

SOME MORE ON BASIC FILING ROUTINES

Another small contribution in connection with filing routines. In common I suspect with most people, my string input routines control the permitted length of the input. This is done with a variable LE(ngth), which is set before calling the subroutine. As most string input in my programs is destined for a random file, it occurred to me that I could kill two birds with one stone.

At the start of the input routine a string of spaces is set up of the required length. Immediately before return from the routine the input is LSET into this string, so we return with a standard length string.

The routine may be of interest to some people. It runs as follows:-

```
10 Z$="":Z1$=STRING$(LE,32):PRINT CHR$(14);:REM GET CURSOR
20 R$=INKEY$:IF R$="" THEN 20 ELSE R=ASC(R$)
30 IF R=13 THEN LSET Z1$=Z$:PRINT CHR$(15);:RETURN
40 IF R=8 AND Z$="" THEN 20:REM CANT BACK-SPACE IF NO INPUT
50 IF R<>8 AND LEN(Z$)=LE THEN 20:REM NO MORE ROOM
60 PRINT R$;:IF R=8 THEN Z$=LEFT$(Z$,LEN(Z$)-1):GOTO 20
70 Z$=Z$+R$:GOTO 20
```

You will see that this permits the construction of Field strings by simple concatenation.

This process has a further advantage in that it permits neatly formatted printing without tabs. Suppose we have A$ 20 characters and B$ 15, we can PRINT A$+" "+B$, and a series will be neatly aligned one under the other with three spaces between each string.

The only minor complication I have so far found is with dates. In my accounting programs these are entered as a 6 character string, 12 Jan. This is alright, but if the date is 1 Jan we need RSET instead of LSET. I deal with this on return from the subroutine by inspecting the string to see if the last character is a space.. If it is then it is switched to the front.

```
IF RIGHT$(Z1$,1)=" "THEN Z1$=" "+LEFT$(Z1$,5)
```

I hope this is helpful to someone

M.C.Matthews, Dorset.

MEMORY EXPANSION FOR THE MODEL 4

One of the nicest features of the model 4 is the extra memory. It allows a larger sheet in Visicalc and bigger files in Le Script and Scripaid and some other wordprocessors can have Banks available. The memdisk allows rapid loading and saving and seems to get round 'those sector not found errors' as long as you remember to save before turning off and nobody pulls the plug. However, a disadvantage of the model 4 is that you cannnot use the banks and memdisk at the same time and the capacity is rather limited. When 35 track SD was the standard, a 56K memdisk would have seemed adequate but today it is rather miserly.

The Amstrad CPC128 starts off equal with the expanded TRS-80 but a 256K ramdisk is available to bolt on the back. The thought that the rotten little Arnold with its Hi-res and flash colour graphics should go one up on sheer bulk capacity is too much. It was time for the TRS-80 to fight back.

Do you remember when the when the model 1 had 4K and the new chips took it up to 16K? Then the same keyboards started to sprought 64K when the even newer chips became available. That was the limit, we were assured because the Z80 will only handle 64K of memory. Suddenly 128K was possible and the Joyce came out with 256K. So if Amstrad can do it for the Joyce and and the Arnold why not the model 4.

Lured by promises of 1/2Meg, 1Meg and perhaps even 1.5Meg, I started to look at upgrades. The problem being that they are not cheap and I have burnt my fingers before. It is one thing to upgrade the hardware but will it do what I want. Remember the 80 col display for the model 1 that would not work with the Wordprocessors that most needed it. The CPM that would not run normal CPM programs and the Bigmem board that lacked usable software. All sounded great but were generally disappointing in some way.

Having waited for others to take the plunge, I was forced to go on alone and sent of to Holland for a Kit. Nothing happened at first but after a couple of weeks my order was returned because I had not included the expiry date. After the delay I had some doubts about the company, but through lack of alternative, I complied and re-ordered.

Having sent of the order at the end of March, I heard nothing until the middle of May when out of the blue, the package arrived. It consisted of a 30cm tube of chips and a small bag of miscellaneous chips and componants. I found a letter and a manual. The letter was partly in English but one paragraph stated "Please let it anybody else with knowless about

electronics while it is a lot of work". That was just the letter and I turned to manual with real fear. Now there are probably a few people that 'knowless' about electronic than me but they do not have to read 'Double Dutch'. However my fears were unjustified and whoever wrote the manual, it was not the writer of the letter. The manual is clear, as well explained as a complex matter can be, and seemed to make sense. It also helped translate the letter. It more or less meant that 'if you think this is going to be easy then you don't know enough to attempt it'.

Joking aside, it is not for the faint heart or the inexperienced or you are likely to lose the use of your model 4p for some time. Model 4 owners may well find the proposition much easier. The 4p does not have room for the extra board and so the modification has to be done on the existing board. As I am no expert I will not try and explain in detail but in essence you need to remove the 64K chips and replace them with the 256K chips supplied.

Even a novice hacker would not find that hard but the conversion does not end there as the circuitry needs some modification. The problem being that cutting tracks is easy but it makes it difficult to restore the board if it need servicing. The kit chose the safer method of removing pins so that at worst you needed new chips. My resident experts all approved of that approach.

The letter also stated that 512K is the most that can be put in a model 4p because there was not room for more but the model 4 can have 1Meg by piggy-backing chips. They had supplied the kit on approval and asked me to decide whether I could manage it or get somebody else to do it and return it for refund if not. The instructions warned that it could take five hours to complete the conversion and obviously you cannot use the computer in the mean time. The only proviso was that the seal on the ramdisk must not be broken otherwise no refund. As you only need the disk after conversion that seemed fair enough.

Having looked at the instructions and had some advice, I decided that I could have done it if I had really wanted to. Of course I could have run the London Marathon or swum the English Channel if I had really wanted to, it was just the lack of time. So off went the kit to our local expert Phil Haswell. Phil is a professional and owns a TRS-80. Phil confirms that the instructions are clear, adequate but even he felt the pressure with the sheer volume of alterations and confirms that the computer could be returned to it orginal state as only one track has been cut.

Once converted the model 4 looks just the same, not even a go faster stripe or a TURBO badge to show off but inside it has hidden depths. In fact so ordinary does it look that if you use ordinary software it will behave quite 'ordinarily'. Which after all, is quite reassuring.

To use the extra memory you have got to set up the mendisk as you do with TRS-DOS but in this case it goes further. You can leave Bank 0 for Lescript, Pronto, Double Duty etc and start on bank 1. You can down load your SYS files and even switch drives. You can load in programs as with JCL and no doubt even more once you ge the hang of it.

The Memdisk is based round the Newdos 80 ILF files and at first it was only available for Newdos 80 v2 or 2.5 and later v3 (v3! where did that one come from. As one of the few legitimate owners I had never heard of an 80 col version). I ordered the Newdos driver but was sent the TRS-DOS version. TRS-DOS 6.n does not have ILF files so a special program is included.

Essentially you generate some JCL type files with the parameters that you want or else you use the default ones provided. This goes on the disk as RAMDISKn/PAR (note the n). You then have a list of the files you want put in the Memdisk. SYS0-13, Basic, and so on. These are in RAMDISKn/ILF and you can probably see that this is similar to having the TRS-DOS Memdisk parameters and Copy command in a JCL. To initialise, you type RD n (n being the PAR and ILF files that you want ie RAMDISK1/PAR). For most purposes you might only need one type of memdisk but you can have one with BASIC and one without. In my case I need a print driver for Scripaid but it could be Multiplan overlays. The ILF files can be on any drive as long as it is specified.

This sound a bit complicated but the default settings do most of it. The only snag being the time taken. From switch on to 'ready' can take nearly two minutes (OK so I am impatient and I do know how long tapes take but if I had wanted to wait two minutes, I would have got up later). But the nice thing is that given the correct ILF your memdisk is now drive 0 with the speed up that that can give and drive 0 is now called drive 2 and the system disk can be replaced by a data disk. If you need to Reset, the program looks through and checks the directory of the memdisk. If it is intact, it will use it without format and retaining the data.

In operation, the program appears invisible (can something appear invisible) and the speed take you by surprise. Type (BREAK) S in Scripsit and the cursor blinks and carries on . There is none of the drive activity while it loads the overlay from 0 and then drive 1 to Save and the silence is uncanny.

Nevertheless, saved it has been. Just watch!. I have just saved the whole file and I bet most of you did not even notice.

Bottom Line. If you like Banks and find the Memdisk too small then this is for you. While it could expand the use of a single drive machine, it is more expensive than a second drive. If you have a 1 Meg memdisk you are going to have trouble saving it on a 40 SS drive. If you use Newdos 80 then it will allow you to use the second bank but there are other programs, although none of these allow more than 128K. On top of the Ramdisk, you get the facility to change the system drive but again this can be achieved without. It would be a great thing for virtual word processors like Wordstar or others that use overlays but there is no CPM driver. If you use Dosplus 4 or another DOS then you will have to go back to TRS-DOS but if the upgrade takes off, the Software companies will have to think again.

If you have a 4p then 1/2 Meg is the limit and think twice about work involved. You may find a friendly hacker but you are prevailing on his kindness if you expect it done for nothing. I think the model 4 version should be quicker and easier but it is not a plug 'in and go' board. The 4 can take the full Meg but you have to piggy-back chips. I have heard the model 3 can take 1.5 Meg but it was not mentioned by this supplier.

The kit was supplied by Seatronics inc.
Kasteel str. 4-abc
6223 BJ Maastricht.
The Nederlands

It was advertised in the 80 micro that has also printed a review of the model 4 version. The company seemed OK but the delay in supplying and answering letters was too long for my liking and I do not seem to be alone. I have since noticed that the above address is slightly different and maybe they have moved recently (any truth in the story that successfull micro-electronics firms move into smaller factories.

The price ($200 - 512K) reflects the cost of the 256k chips which are not in plentiful supply but in time, and under the pressure from the newer 512K chips, the price will probably fall. The software is not protected but is is claimed that the disks are registered and pirate copies can be traced back. On the plus side, it does everything claimed and without hidden snags or costs. The 1/2 Meg limit may not be too much of a problem for 4p users but you are better off than without. Maybe the 512K chips will be the next upgrade but one thing is certain, the Amstrad, Spectrums and what have you, are going to be sproughting 256K before long and the old 64K machines are going to be yesterday systems in the shops.

Derek Traylor.

KERMIT PART II

Last month I talked about some of the potential compatability problems that had to be taken into account when the KERMIT protocol was designed. This month we will take a look first at the characteristics of KERMIT that resulted from those design considerations.

Characteristics of KERMIT

- Communication takes place over ordinary terminal connections.

- Communication is half duplex. This allows both full and half duplex systems to participate, and it eliminates the echoing that would otherwise occur for characters arriving at a host job's controlling terminal.

- The packet length is variable, but the maximum is 96 characters so that most hosts can take packets in without buffering problems.

- Packets are sent in alternate directions; a reply is required for each packet. This is to allow half duplex systems to participate, and to prevent buffer overruns that would occur on some systems if packets were sent back to back.

- A timeout facility, when available, allows transmission to resume after lost packets.

- All transmission is in ASCII. Any non-ASCII hosts are responsible for conversion. ASCII control characters are prefixed and then converted to printable characters during transmission to ensure that they arrive as sent. A single ASCII control character (normally SOH) is used to mark the beginning of a packet.

- Binary files can be transmitted by a similar prefix scheme, or by use of the parity bit when both sides have control of it.

- Logical records (lines) in textual files are terminated during transmission with quoted carriage-return/linefeed sequences, which are transparent to the protocol and may appear anywhere in a packet. Systems that delimit records in other ways are responsible for conversion, if they desire the distinction between records to be preserved across unlike systems.

- Only a file's name and contents are transmitted - no attributes. It is the user's responsibility to see that the file is stored correctly on the target system. Within this framework, invertible transfer of text files can be assured, but invertible transfer of non-text files depends on the capabilities of the particular implementations of KERMIT and the host operating systems.

- KERMIT has no special knowledge of the host on the other side. No attempt is made to integrate the two sides. Rather, KERMIT is designed to work more or less uniformly on all systems.

- KERMIT need not be written in any particular language. It is not a portable program, but a portable protocol.

Thus KERMIT accommodates itself to many systems by conforming to a common subset of their features. The resulting simplicity and generality allow KERMIT on any machine to communicate with KERMIT on any other machine, micro-to-mainframe, micro-to-micro, mainframe-to mainframe. The back-and-forth exchange of packets keeps the two sides synchronized; the protocol can be called "asynchronous" only because the communication hardware itself operates asynchronously.

As far as the user is concerned, KERMIT is a do-it-yourself operation. For instance, to transfer files between your micro and a mainframe, you would run KERMIT on your micro, put KERMIT into terminal emulation mode, which "connects" you to the mainframe, log in and run KERMIT on the mainframe, then "escape" back to the micro and issue commands to the micro's KERMIT to send or fetch the desired files. Any inconvenience implicit in this procedure is a consequence of the power it gives the ordinary user to establish reliable connections between computers that could not otherwise be connected.

As described last month, the process of information transfer centres around the sending and receiving of discrete pieces of data called packets. We will now consider the design of the packet in a little more detail

Packets

KERMIT packets need to contain the data that is being transferred, plus minimum information to assure (with high probability) that the expected data arrives completely and correctly. Several issues come up when designing the packet layout: how to represent data, how to delimit fields within the packet, how to delimit the packet itself, how to arrange the fields within the packet. Since the transmission medium it self is character-oriented, it is not feasible to transmit bit strings of arbitrary length, as do some of the bit-oriented protocols. Therefore the smallest unit of information in a packet must be the ASCII character. This precludes some techniques that are used with other communication media.

Control Fields

Since KERMIT packets are short, it is important to minimize the amount of control information per packet. Therefore the decision was made to limit the length of each control field to 1 character with the exception of the checksum. There are 95 printable characters to work with (128 ASCII characters, less

DEL and the 32 control characters), therefore values from 0 to 94 can be represented with a single character. (From last month you may remember that the ASCII control characters are not used since they mean different things on different systems). This lead to the following implementation of control fields

- The packet SEQUENCE NUMBER is used to detect missing or duplicate packets. It is unlikely that a large number of packets could be lost, especially since packet n is acknowledged before packet n+1 is sent. So the sequence number can be a small quantity, which "wraps around" to its minimum value when it exceeds a specified maximum value.

- To prevent long packets, a small maximum length can be enforced by specifying the PACKET LENGTH with a single character; since there are 95 printable ASCII characters, this would be the maximum length, depending on how the control fields are counted.

- The CHECKSUM is also of fixed length. The actual length depends on the desired balance between efficiency and error detection.

The packet length and checksum act together to detect corrupted, missing, or extra characters. These are the essential fields for promoting error-free transmission. But so far, we've only considered packets that carry actual file data; special packets are also required composed only of control information, for instance to tell the remote host the name of the file that is about to come, or to tell it that the transmission is complete. This can be accomplished with a packet TYPE field. The number of functions that needs to be specified in this field is small, so a single character can suffice here too.

Packet Framing

The beginning of a packet is marked by a unique start character, SOH (Start Of Header, ASCII 1, Control-A). This character cannot appear anywhere else within the packet. SOH was chosen because, unlike most other control characters, it is generally accepted upon input at a job's controlling terminal on most mainframes as a data character, rather than an interrupt or break character. This is probably no accident, since it was originally intended for this use by the designers of the ASCII alphabet. Should a system be incapable of sending or receiving SOH, it is possible to redefine the start-of-packet character to be any other control character; the two sides need not use the same one.

There are three principal options for recognizing the end of a packet: a fixed length, a unique packet-end character, and a length field. There are arguments for and against each involving what happens when characters, particularly a length or terminator control field character, is lost or garbled, which

will be mentioned later. KERMIT uses a LENGTH field.

To take in a packet, a KERMIT program gets characters from the line until it encounters the SOH. The next character is the length; KERMIT reads and decodes the length and then reads that many subsequent characters to complete the packet. If another SOH is encountered before the count is exhausted, the current packet is forgotten and a new one is started. This stratagy allows arbitrary amounts of noise to be generated spontaneously between packets without interfering with the protocol.

Encoding

In order to make each character in the packet printable, KERMIT "quotes" any unprintable character by transforming it to a printable one and precedes it with a special prefix character. The prefix is normally "#"; the transformation is done by complementing bit 6 (adding or subtracting 64 , modulo 64). Thus control-A becomes "#A", control-Z becomes "#Z", US (control-underscore on most terminals) be comes "# ". The prefix character is also used to quote itself: "##". Upon input, the reverse transformation is performed. Printable characters are not tranformed. The assumption is that most files to be transferred are printable, and printable files contain relatively few control characters; when this is true, the character stream is not significantly lengthened by quoting. For binary files, the average quoting overhead will be 26.6% if all bit patterns are equally likely, since the characters that must be quoted (the control characters, plus DEL, and "#" itself) comprise 26.6% of the ASCII alphabet.

KERMIT also provides a scheme for indicating the status of the 8th bit when transferring binary files between systems that must use the 8th bit for parity. A byte whose 8th bit is set is preceded by another special quoting character, "&". If the low-order 7 bits coincide with an ASCII control character, then control-character quoting is also done. For instance, the byte 10000001 would be transmitted as "&#A". The "&" character itself can be included as data by quoting it (#&), and the control-quote character may have its 8th bit set (&##). 8th-bit quoting is only done when necessary; if both sides can control the parity bit, then its value is preserved during transmission. If the 8th bit is set randomly on binary files, then 8th-bit quoting will add 50% character overhead. For some kinds of binary data, it could be less; for instance, positive binary numbers in 2's complement notation do not to have their high-order bits set, in which case at least one byte per word will not be quoted.

A third kind of "quoting" implements rudimentary data compression. At low speeds, the bottleneck in file transmission is likely to be the line itself, so any measure that can cut down on use of the line would be welcome. The special prefix character "ç" indicates that the next character is a repeat count (a single character, encoded printably) and that the character after that (which may also have control or 8th-bit

prefixes) is repeated so many times. For instance "¢†A" indicates a series of 93 letter A's; "¢H&#B" indicates a series of 40 control-B's with the parity bit set. The repeat count prefix itself can be included as text by quoting it with "#". To keep the protocol simple, no other transformations are done.

Error Detection

Character parity and Hamming codes are forms of "vertical redundancy checks" (VRCs), formed by combining all the bits of a character in one way or another. The other kind of check that can be used is the "longitudinal redundancy check" (LRC), which produces a "block check character" formed by some combination of each character within a sequence. The sending side computes the LRC and sends it with the packet; the receiving side recomputes it for comparison. There are various forms of LRCs. One form produces a "column parity" character, or "logical sum", whose bits are the exclusive-ORs of the corresponding bits of the data characters. Another is the "checksum" which is the arithmetic sum of all the characters in the sequence, interpreted numerically. Another is the "Cyclic Redundancy Check" (CRC) which passes the characters through what amounts to a shift register with imbedded feedback loops, producing a block check in which each bit is effected in many ways by the preceding characters.

All of these techniques will catch single-bit errors. They vary in their ability to detect other kinds of errors. For instance, a double bit column error will always go undetected with column parity, since the result of XORing any two bits together is the same as XORing their complements, whereas half the possible double bit errors can be caught by addition because of the carry into the next bit position. CRC does even better by rippling the effect of a data bit multiply through the block check character, but the method is complex, and a software im plementation of CRC can be inscrutable.

Standard, base-level KERMIT employs a single-character arithmetic checksum, which is simple to program, is low in overhead, and has proven quite adequate in practice. The sum is formed by adding together the ASCII values of each character in the packet except the SOH and the checksum itself, and including any quoting characters. Even non-ASCII hosts must do this calculation in ASCII. The result can approach 12,000 in the worst case. The binary representation of this number is 10111011100000, which is 14 bits long. This is much more than one characters worth of bits, so we can discard some high order bits and still have a viable validity check.

The KERMIT protocol also allows other block check options, including a two-character checksum and a three-character 16-bit CRC. The two- character checksum is simply the low order 12 bits of the arithmetic sum, broken into two printable characters. The CRC sequence is formed from the 16-bit quantity generated by the CCITT-recommended polynomial

$$x^{16} + x^{12} + x^5 + 1$$

which is also used in some form with other popular transmission techniques. The high order 4 bits of the CRC go into the first character, the middle 6 into the second, and the low order 6 into the third.

Some care must be taken in the formation of the single-character block check. Since it must be expressed as a single printable character, values of the high order data bits may be lost, which could result in undetected errors, especially when transferring binary files. Therefore, the 7th and 8th bits of the sum are extracted and added back to the low order bits; if the arithmetic sum of all the characters is S, then the value of the single-character KERMIT checksum is given by

(S + ((S AND 300)/100)) AND 77

(numbers are in octal notation). This ensures that the checksum, terse though it is, reflects every bit from every character in the packet.

A final note on parity -- a parity bit on each character combined with a logical sum of all the characters (VRC and LRC) would allow detection and correction of single-bit errors without retransmission by pinpointing the "row" and "column" of the bad bit. But control of the parity bit cannot be achieved on every system, so the parity bit is used for binary data whenever possible can, but if needed is surrendered to the communication hardware. If we have use of the 8th bit for data, then it is figured into the block check; if we do not, then it must be omitted from the block check in case it has been changed by agents beyond the knowledge or control of the KERMIT program.

Next month we will look at the actual structure of a packet and see how the whole system works. Meanwhile on the subject of how KERMIT got it's name, an eagle-eyed reader spotted a letter in Practical Computing which stated that KERMIT is the acronym for KL-10 Error-free Reciprocal Micro Interchange over Tty lines, KL-10 being the CPU for DEC-10 & DEC-20 mainframes. It sounds boring enough to be true!.

NB. On the print wheel I'm using
¢ = tilde = decimal 126
† = right curly bracket = decimal 125
Anyone know a good 10 cpi ASCII sequence Qume wheel

Ed