NATGUG NEWS

Volume 8 Issue 1 July 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

I've just been reading May 80-Micro and they have some quite interesting figures on Tandy's financial and market positions in the US of A. In the last quarter of 1985 Tandy claim to be the top selling PC compatible and taking 1985 overall claim to be second only to Compaq selling 185,000 units during the year - this includes Model 1000's and 1200's in the various guises. In financial terms Tandy seem to be pulling out of the doldrums with sales for the six month period ending Dec. 85 totalling $1.8 billion, a 19% increase over the same period of the previous year. One should always take these sort of figures with a liberal pinch of salt since the accountants will have performed various 'massages' to present the best picture but none the less this would seem to be a reasonably healthy state of affairs. I still wonder how Tandy are going to fare in competition with the Far East clones, since a no-frills 2 drive, 256K, mono screen machine can be had for under #500. However I have seen the 1000 HD with 128K and a 10MB Hard disk for #1195. This machine is reputed to have a high degree of compatability and of course is backed by Tandy and not Ctl-Alt-Deli and other evanescent companies. The big compatibility problem is with the short slots, although cards for the XT and Portable will fit.

An interesting little tit-bit (no this isn't page three) for PCW 8256 users is that a firm called Citadel Products is selling an upgrade kit for the PCW. The advert claims that it is very easy you only need a screw driver and hacksaw blade (!). The kit adds a 1MB 3-inch disk drive and the extra memory. Cost is #189.95 or nearly 40% of a new 8512 - I wonder how much the individual items can be got for.

On the software front there are a couple of interesting front-ends (definitely not page three !) available to run on PC compatibles, namely 1DIR and PCBOSS. These are somewhat similar to Southern Software's File Commander in that they initiate by doing a directory, display the said directory plus various items of system information and then give you a number of options that you can use on the displayed files such as rename, copy, delete, execute etc. Both programs offer multiple file tagging and file selection is by scrolling with the arrow keys or wild-card. Both allow movement up and down directory trees, making this somewhat tedious process rather less error prone than from the normal command line. 1DIR (do they actually pay people to think up these names) also allows you to build command menus for running applications and batch files etc. Both programs are memory resident and attach themselves do the DOS so that when an application is terminated you automatically return to the front-end rather than DOS. In fact, if you execute one or other from the AUTOEXEC.BAT you need never see A> again. Both programs are to be recomended although the version I have of 1DIR does not work with DOS 3.X only 2.X. 1DIR is from Bourbaki Inc., PCBOSS is from Windmill Software.

For Model 4 owners who are suffering from withdrawl symptoms because they cannot PEEK and POKE to the screen I found this small BASIC routine that you can add to your own programs to give you back this capability.

```
1 ASSEMBLY$=STRING$(26,32) : A=VARPTR(ASSEMBLY$) +1 :
  A=PEEK(A) + PEEK(A+1) * 256
2 FOR I = 0 TO 25 : READ B% : POKE A+I,B% : NEXT : RESTORE
3 CHAR%=1 : A2=VARPTR(CHAR%) : IF A2<0 THEN A2=A2+65536
4 B2%=INT(A2/256) : B1%=INT((A2/256-B2%)*256)
5 POKE A+10,B1% : POKE A+11,B2% : PEAK=A : POAK=A+14
6 A=0 : B%=0 : A2=0 : B1% =0 : I=0
7 DATA 126,103,26,111,62,15,6,1,239,33,32,32,119,201
8 DATA 126,103,26,111,10,79,62,15,6,2,239,201
```

To read a screen character --- CALL PEAK(X%,Y%)
To write to the screen --- CALL POAK(X%,Y%,C%)
Where X% is an integer defining the row (0-23)
Y% is an integer defining the col (0-79)
and C% is an integer defining the character (0 -255) to poke.
The parameters must be integer variables not numeric constants
The variable CHAR% returns the value of the character peaked from the screen. I have not personally tried these routines since I don't run a Model 4.

My apologies to Don Harrison whose advert appeared in last month's issue with the incorrect telephone number, the ad is repeated at the end of this issue with the correct number - I hope.

Finally thank you to those who have sent their contributions on disk, it certainly makes my life easier (if there are typos in telephone numbers I can blame them on you !). For those who include tables or use word processors which put a CR/LF at the end of each screen line, we actually only use a line width of 65 chars since we are using a 10 cpi daisy to allow for reduction to A5 on printing.

LETTERS

Dear Ed.,

Best wishes in your post as Editor. I have every issue of the newsletter since issue 1 in January 1979 and they are a very useful reference.

Being a NEWDOS80 person I am always intrigued by the effort required to field the buffer and file data to discs by other systems. I have recently had to use the IBM system and this is still as antiquated.

Can anybody tell me where I can find a CMD "O" type sort for these new PCs?

The one area in which our newsletter falls short is the answers to questions which are asked by our members. I would like to see the answers to the questions, even if they are a couple of months later. (Agreed, if you manage to sort a problem on the phone, write in with the solution, it could help others, Ed.)

I learnt my NEWDOS80 file handling at a group meeting in a school in North London some years ago and feel that alone justifies my lifelong membership.

I shall be attending the Blandford show on Sunday 7th September, could we have a session on compiling basic programs?

Over the last seven years I have learnt a few tricks and the following one I think may be of use to other members.

Most times your computer will prove very reliable and errors are generally of your making. However, when your basic program crashes with a parity error try typing CONT and press ENTER, you can do this several times and you will often find you can get past the problem.

Finally, can I just say a public 'Thank you' to OS of Blandford Computers, who has provided more information and help for Tandy machines and their users than Tandy themselves.

Yours sincerely,

John Bennett, 11 Haywards, Pound Hill, Crawley, Sussex RH10 3TR.

Members letters cont.

Dear Ed.,

Having spoken to Mr Brian Pain on the telephone this evening, he advised me to drop you a line regarding my requirements.

Firstly a little about my interests. I studied for the City and Guilds Amateur Radio certificate, and was very pleased to pass the set examination, my call sign being G1JQH.

I have been the proud owner of the EG 3003 Genie I computer for almost a year now, adding a secondhand EG 3014 expander unit, two half height 40t SS disc drives and finally a secondhand printer.

I am looking into the possibility of adding Packet Radio and/or Amtor to my existing RTTY system, and I am given to understand that to do this, I would need the RS-232 interface board and interconnections etc.

Having been in touch with LOWES they tell me that the RS-232 Interface is no longer available, and even if were it would cost about #160, which is rather more than I am prepared to pay.

Firstly, are there any other members who share my interest in both computing and amateur radio and secondly, is there anyone with a second-hand RS232 board suitable for use in my Genie.

Thanking you in anticipation,

Dennis Spalding, 171 Minster Road, Minster-on-Sea, Sheerness, Kent ME12 3LH.

Model 4 extra memory - an update

Having installed a 512k memory upgrade in my model 4p and had it working for a month I am able to give a more considered opinion on the value of the extra memory. Like all things it has good and bad points. A good point is that it seems to work without snags and a bad point is that you miss the Ramdisk if it is not available.

When is it not available? I have a model 4p and a 4 and only the 4p has the 512k memory. The model 4 has only a 128k memory. Not only that but I cannot run my model III programs under TRS-DOS 6 and therefore Ramdisk 6 software cannot be used. I can use the older memdisk program TMDD to give me a 64K memdisk on the 128k machine but that will not work with some of the programs which do not respect Highmem.

In other respects I was pleased with the conversion and this left two alternatives. I could get round the software problems by purchasing another driver. There are two available and the original driver was designed to operate with Newdos 80 v 2. As most, if not all of my model 111 programs will run under Newdos, I sent off for the Ramdisk 1.14 driver. This will work on the 128k machine and give it the advantages of Memdisk which comes with TRS-DOS. For that reason the program has some advantage for the 128k machine when working with Newdos. TRS-DOS 6 users get very little for their money unless they upgrade the memory because they already have a 64K memdisk.

One of the reasons for having two systems is to provide a backup and it is highly desirable to have the model 4 upgraded. I had already found the that there is no such thing as too much memory and I decided to opt for the 1 meg expansion. It might be worth noting that the upgrades can be done in stages although it would not make too much sense to upgrade to 128k if you were likely to want 256k or above in the near future. The reason being that the 128k expansion uses 64k chips which are removed for large expansions. On the other hand the 256k and above, expansions are achieved by adding more chips of the same type.

Having waited for six weeks for the model 4p conversion to arrive, I made the point of asking for delivery advice. I got back a reply that said that the model 4p kit had been delayed because of the complexity ot the fitting instructions. This was understandable from the technical point of view as they were both complete and accurate when they arrived. On the other hand they must have known about the delay when they advertised in December and I appear to have got the first delivery in June. Had they mentioned the delay when they quoted prices in April, I would have ordered the Model 4 version.

The delivery note came back in well under two weeks and the kit arrived in the second post of the same day. It also brought some news that needs to be considered against my earlier report. The 4p conversion was complicated because it was done to the motherboard and made the minimum of alterations so that the

expansion could be removed. This is highly desirable as Tandy do not like servicing non standard machines. The model 4 being much larger inside, had room for a board which was grafted onto the motherboard and could be removed if required. Having told me last time that the board cannot be fitted to the 4p and that 512k was the largest upgrade that could be fitted, the new manual now stated the opposite. A board is available for the 4p and a full meg can be fitted.

The new instructions say that a board is now available for the 4p but it requires the Z80 microprocessor and socket to be removed. The Z80 goes back on the board and this is obviously one difference between the two boards. The board supplied for the model 4 has one or two PALs and some odd chips and components. One PAL is marked 512k and the other 1 meg. The 512k PAL is mounted directly to the board but the 1 meg one is in a socket. This would suggest that it is designed for ugrading in stages. While the socket would make it easier to fit the PAL, the upgrade would still require the chips to be piggybacked. This is not a technique for the novice and especially on expensive chips.

The Newdos and TRS-DOS ramdisk are claimed to be different but work alike. Having gained some experience on the TRS-DOS version, I quickly got the Newdos ramdisk operational. I had a slight problem because the Newdos version would not accept the seventh Bank and I had to change the Last=7 to Last=6. This may be due to the slight differences in the way they format but at the moment I have not had time to look at the calculations. With memory to spare it is not too significant. Having had a brief trial I shut down and resumed later.

On my second attempt I could not get the Formated ramdisk to work. It seemed to crash with a Wrong Data Record Type error (or something similar) and eventually I gave up. Later I got my son to have a go and it worked perfectly. Since that time I have not had any trouble except once when I tried to back up my booting disk to the Ramdisk. This may not make much sense to any one that has not used a 4p with Newdos or a Ramdkisk. Firstly the 4p will not boot off a standard Newdos disk unless MODELA/III has been installed by another DOS. I have a zapped disk which came from John Kilpatric and I believe was breathed on by Leo Knaggs. This disk is a hybrid of TRSDOS 6 and Newdos 80. So its not surpising if something odd happens if you move it into memory.

One of the advantages of the Ramdisk is that it can replace the system disk and any other disks. To do this you have a program that copies all the system files to the Ramdisk and makes it drive 0. From then on the system looks straight to the Ramdisk and saves the time taken in disk I/O. In fact you do not need a system disk and can have data disks on all drives.

To further speed up the working of the computer, it is possible to copy across all the programs and data files. All this can be done under Newdos by adding ILF=$ to RAMDISK/PAR.

This copies all the files (including SYS) from drive 0 and saves typing in a long list. It was this part that seemed to lock up with the zapped DOS disk. Using an ILF file it is possible to specify all the files and this is required for the TRS-DOS version. Although TRS-DOS does not support ILF files the Ramdisk program adds the facility. Both drivers can be set up by a JCL if more complex loading is required but I have not tried it.

Another facility on the Ramdisk is the ablity to change the number of the Ramdisk. This means that you can have it as drive 0 and make the former drive 0 into drive n, or you can set it up as drive n and have it as an extra drive. The decision is made on the bases of the software and your specific requirements. There is an obvious problem with having the Ramdisk as drive 0 as files are usually saved to drive 0 as default and the contents will be lost if you shut down without saving the data to disk.

A Ramdisk allows faster and sometimes more convenient working but rarely does it do anything that cannot be done another way. Perhaps the nearest may be the facilty to have a large data base that is largely protected from disaster. If a data base and the sysfiles are in the Ramdisk as drive 0. It is possible to write protect the Ramdisk and remove all the disks. That way the computer is fairly safe and can be left for access by a lot of relatively inexperienced people. They can pull the plug but cannot easily corrupt the data in memory and the data on the disks will remain safe.

The other feature of large Ramdisks is the way they can be used to suppliment the other drives. A large Ramdisk is more convenient than another drive for backing up and copying etc. The standard machine with its 40 track single sided drives can be given a large tempory drive but some care has got to be taken in saving data when the ramdisk can easily overfill a data disk.. It is also more convenient if you have a 40 and an 80 track drive. It is very easy to backup the 40 track drive to a switched 40/80 and then find that the disk is unreadable on a standard 40 track drive. On the otherhand a 40 track disk can be backed up via the ramdisk without resorting to disk swapping. Single 80 track drives can be backed up in the same way provided that there is sufficient memory available. (main reason for choosing 1 meg instead of 512k)

No system can ever be perfect and I have been disappointed in some respects. Firstly the speedup is nowhere near the 20X claimed. It is true that they only said 'up to 20X' and we all know that 'from #xxxx' means a second hand clapped out Mini and not the Rolls in the picture. The position is very complex because disk I/O is not the only factor in operating speed. I timed one program with a sequentional GET and PRINT and was surprised how long the PRINT took. Speeding up the GET will not affect the PRINT and I found that several programs were only accessing three times faster.

If you load a file into Scripsit from ramdisk it is speeded up dramatically and the same goes for Saves. If you load and save once, there is no overall gain as the files have to be copied to and from the disk in the beginning and at the end. If you have a virtual memory program like SuperScripsit the time saving is dramatic.

Loading data into Visicalc is only slightly faster than from disk. This is due to the processing time taken when the data is put into the cells. Multiplan loads its overlays from the ramdisk and is noticably faster in a notoriously slow program. The improvement is appreciable. Wordstar would also benefit but no CPM driver is available. This is particularly galling when you see how the small memdisk on Montezumas CPM can improve the program but lacks the capacity to have the program, overlays and data.

Sometimes programs do not benefit from the Ramdisk because of the way that they are written. SIR has overlays and makes frequent accesses to the data disk. It would benefit greatly from the faster loading. However the program is written to look for the modules on drive 0 and the data on drive 1. This means that the Ramdisk can only be used for one or other but as mentioned earlier it would be ideal to have it entirely in memory. This is not the fault of the Ramdisk and can only be resolved by modifiying the program.

The earlier report did not mention the price as this was not to hand. In fact the Access charge has only just arrived and appears as £167 charged in Dutch Guilders but there were no extras to pay. As the price is quoted in US dollars there is a three way currency calcualation. The price includes the memory, PALS as required and the Ramdisk driver. In my case I needed both model III software drivers (Newdos 80) and a model 4 driver (TRS-DOS). As I was purchasing two upgrade kits I got both drivers included but they and the chips are available separately.

Some people might like to buy a small upgrade and add the extra when the prices of memory fall. The Ramdisk drivers are the same for any upgrade from 64k to 1 meg. A 64K machine can have a 256K kit installed and the additional 256k added later. A 128k machine would become 320k. The 256K chips are available from Seatronics but other suitable chips could be used. To expand above 512k it is necessary to fit another PAL and one or two sets of 256k RAMS. Seatronics claim that they supply 125 nano second refresh chips to allow for the microprocessor to be speeded up. Slower and cheaper chips might be adequate but I have no experience in this field.

Derek Traylor Hornchurch, Essex

FINDWORD

(Ed. Like all good structured programs, this follows top-down principles and tre explanation procedure is near the end !. You will see that Geoffrey has used the dreaded GOTO in the later part of the program, I hope that he has included it just to show that it is possible to use it in PASCAL. I can't see at the moment why A CASE construct would not have done the job rather more elegantly, but programming isn't always about elegance, practicality is often more important)

```
PROGRAM Findword; (* To find a key word in TEXT files*)
(*G.H. &  A. Taylor*)

LABEL     Loop1, Loop2;

TYPE
  longline = STRING [255];

VAR
  dfile                                          : text;
  dfilename,Qfind                                : STRING [14];
  dline                                          : longline;
  n,linecount,pagecount                          : INTEGER;
    (*n is a general counter.
     linecount counts the lines in the file being searched.
     pagecount counts  the lines  used on the screen for printing
     the file.*)
  c                                          : CHAR; (*for inkey*)
  lprint                                     : BOOLEAN;

PROCEDURE Printline;

VAR len_dline      : INTEGER;

BEGIN

  IF lprint THEN (*if printer output*)
     WRITELN (lst,linecount:3,' ',dline)  (*print line*)
   ELSE  (*Not Printer output*)
     BEGIN
        len_dline := LENGTH(dline);
        IF (len_dline > 235) THEN pagecount := pagecount + 4
        ELSE
        IF len_dline > 155 THEN pagecount := pagecount + 3
        ELSE
        IF len_dline > 75 THEN pagecount := pagecount + 2
        Else
        pagecount := pagecount + 1;   (*paging routine*)

     END; (*If not printer output*)

  WRITELN(linecount:3,'  ',dline); (*On screen in any event*)

   IF pagecount >= 16 THEN   (*only if NOT lprint*)
```

```
        BEGIN
          GOTOXY(1,25);
          WRITE('Press any key to continue');
          REPEAT UNTIL KEYPRESSED;
          CLRSCR;
          WRITELN(dfilename);
          WRITELN(' Line    Text');
          WRITELN;
          WRITELN(linecount:3,'  ',dline);
          pagecount := 0;
        END; (*inkey$ routine*)

END; (*PROCEDURE Printline*)


PROCEDURE Start_chk_file;

BEGIN
  linecount := 0; (*init vars*)
  pagecount:=0;
  CLRSCR;
  WRITELN(dfilename);
  WRITELN('Line    Text');
  WRITELN;
  IF lprint THEN
     BEGIN
        WRITELN(LST);
        WRITELN(LST,'The     file      being     searched      is
.......',dfilename);
        WRITELN(LST,'Looking for ..................... ',Qfind);
        WRITELN(LST);
        WRITELN(LST,'Line    Text');
        WRITELN(LST);
     END; (*If lprint*)

END; (*PROCEDURE Start_chk_file*)

PROCEDURE End_chk_file;
BEGIN
  WRITELN;
  WRITELN('End of File');

WRITELN('---------------------------------------------------------
--------');
  IF lprint THEN
     BEGIN
        WRITELN(LST);
        WRITELN(LST,'End of File');

WRITELN(LST,'---------------------------------------------------
------');
     END; (*If lprint*)

  CLOSE(dfile);
```

```
  IF NOT lprint THEN
    BEGIN
      WRITELN;WRITELN('Press any key to continue');
      REPEAT UNTIL KEYPRESSED;
    END; (*IF NOT lprint*)

END; (*PROCEDURE End_chk_file*)


PROCEDURE Query_print;
BEGIN
 REPEAT
    CLRSCR;
    GOTOXY(1,8);
    WRITELN('Do you  want a printed list  of the lines  in  which
the word is found?');
    WRITE('  Enter Y or N   :');
    READ(KBD,c);
    c := UPCASE(c);
 UNTIL (c = 'Y') OR (c = 'N');
 WRITE(c);

 IF c = 'Y' THEN
    BEGIN
      lprint := TRUE;
      GOTOXY(1,18);
      WRITELN('When the Printer is ready press..... "C"');
      WRITELN;WRITELN;
      WRITELN('Or to abort the printout press........"A"');
      REPEAT
         READ(KBD,c);
         c := UPCASE(c);
      UNTIL (c='C') OR (c='A');
      IF c = 'A' THEN lprint := FALSE;
      CLRSCR;
    END  (*IF c=Y*)
  ELSE lprint := FALSE; (*IF c <> 'Y'*)

END; (*PROCEDURE Query_print*)


PROCEDURE Getfile;  (*get filename & open file*)

 var OK : Boolean;
BEGIN
  CLRSCR;
  REPEAT   (*UNTIL FILE OPENED OK or dfilename = ''*)
     dfilename := '';
     gotoxy(1,8);
     WRITELN('Please Enter the name of  the TEXT file you wish to
search');
     WRITELN('(   with   the   Drive   and   file   extension   as
necessary)');
     WRITELN;WRITELN;
     WRITELN(' or just press <ENTER> for Menu');
     WRITE(':');
```

```
     READLN (dfilename);
     IF dfilename <> '' then
      BEGIN
         ASSIGN(dfile,dfilename);
         (*$I-*)   RESET(dfile)  (*$I+*) ;    (*turn  off e/check
while                                              opening    for
reading*)
         OK := (IORESULT = 0);

         IF NOT OK THEN
           BEGIN
             CLRSCR;
             GOTOXY(5,5);
             WRITELN(' Cannot find File ',dfilename);
           END;

      END (*IF dfilename <> ''*)

    UNTIL OK OR (dfilename = '');  (*file opened OK or dfilename
                                    is blank*)

END; (*PROCEDURE Getfile*)


PROCEDURE Findword;
VAR  xline     : Longline;
     II        : INTEGER;
     len_x     : INTEGER;

BEGIN

   xline:= dline;
   len_x:= LENGTH(xline);
   II:=1;
   WHILE II < len_x + 1 DO        (*Convert line to upper case*)
     BEGIN
        xline[II]:=UPCASE(xline[II]);
        II:= II+1
      END; (*WHILE II < ETC*)

   IF POS(Qfind,xline) <> 0 THEN Printline;

   IF (POS(Qfind,xline) = 0) AND (Qfind = '') THEN Printline;

 END; (*PROCEDURE Findword*)


PROCEDURE Heading;  (*Not necessary for running programme*)
BEGIN
CLRSCR;
WRITELN;
WRITELN('                      * * *  F I N D W O R D  * * *');
WRITELN;
WRITELN('Written   by  G.H.  &  A.  Taylor  42  Davenham   Avenue
Northwood Mddx HA6 3HQ');
WRITELN;WRITELN;WRITELN;
```

```
WRITELN('This programme searches through a TEXT file for the
word you specify');
WRITELN('and displays each line in which the word is found.');
WRITELN;WRITELN;
WRITELN('The search is not case dependent.');
WRITELN;
WRITELN('A TEXT File is an ASCII file');
WRITELN('with the lines terminated with a carriage return/line
feed.');
GOTOXY(1,25);
WRITE('Press any key to continue');
REPEAT UNTIL KEYPRESSED;
END; (*PROCEDURE Heading*)


PROCEDURE Enter_word;
BEGIN
   CLRSCR;
   GOTOXY(1,8);
   WRITELN(' Please ENTER the word you wish to find ');
   WRITELN;
   WRITE('or just press <ENTER> to print whole file  :');
   READ(Qfind);
   For n := 1 to LENGTH(Qfind) DO Qfind[n]:=UPCASE(Qfind[n]);
END; (*PROCEDURE Enter_word*)



BEGIN     (*MAIN*)

Heading;

Loop1:
Getfile;
IF dfilename <> '' THEN Query_print;

Loop2:
IF dfilename <> '' THEN
  BEGIN
    Enter_word;
    Start_chk_file;

    WHILE NOT EOF(dfile) DO
      BEGIN
        READLN(dfile,dline);
        linecount := linecount + 1;
        Findword;
      END; (*DO*)

    End_chk_file;
  END; (*IF dfilename <> ''*)

CLRSCR;
GOTOXY(1,8);
WRITELN('Press -');
WRITELN;
```

```
WRITELN('          1  To find a word in a new file');
IF dfilename <> '' THEN
 BEGIN
   WRITELN;
   WRITELN('          2   To find  a new keyword  but in the same
file');
 END;
WRITELN;
WRITELN('          3  To exit programme');
REPEAT
   GOTOXY(1,19);
   READ(KBD,c);
   c := UPCASE(c);
UNTIL (ORD(c)=49)  OR  (ORD(c)=51) OR ((ORD(c)=50) AND (dfilename
<> ''));
(*i.e until c = 1 or 2 or 3*)

IF c = '1' THEN GOTO Loop1;

IF c = '2' THEN
   BEGIN
      RESET(dfile);
      GOTO Loop2;
   END;

IF c = '3' THEN HALT;
END.
```

Geoffrey & Ariela Taylor.

Using Binary Coded Decimals (The DAA command).

I have recently been working on a program that needs a counter value. The solution that I came up with involved the use of binary coded decimals (BCD). BCD arithmetic is often used where the round-off problem associated with floating point math as normally implemented on micros could cause embarrassing problems in some software such as accountancy programs. The principles behind BCD are not well covered in the usual books and therefore you may interested in the routines that I used.

My program contained a simple M/C routine that uses the Byte Counter (BC) to tell it how much to add to a decimal value. The actual counting was done with the aid of DAA (Decimal Adjust Accumulator). This little used instruction applies the necessary correction to the contents of the Acc. if you are using BCD format. Before we can look at the routine it is necessary to understand BCD's.

The BCD represents a decimal digit (0,1,...,9) by the corresponding 4 bit binary number. Here are the ten BCD codes:

Decimal	0	1	2	3	4	5	6	7	8	9
BCD	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

Here is an Example: 2039 => 0010 0000 0011 1001
7847 => 0111 1000 0100 0111

Note that 2 decimal numbers can be contained in 1 byte and that numbers of 1010 or greater are invalid. If a mathematical operation results in 1010 or greater then a BCD correction has to performed, ie do a carry as we would in normal arithmetic. One could write a routine to do this but the Z80 provides this automatically with the DAA instruction.
Lets look at an example

```
decimal         binary
   6719         0110 0111 0001 1001
 + 1126         0001 0001 0010 0110
   ----         -------------------
   7845         0111 1000 0011 1111
        HEX=>    7    8    3    F   Wrong !
```

If we perform a DAA on the low order byte, the following steps will be carried out:

```
   00111111 (3F)
 + 00000110 (+6 as 1111 is not a BCD value)
 = 00110101 (Answer: 35 and Half carry set)
 + 00010000 (+Half carry)
 = 01000101 (45)
```

If adding the half-carry resulted in a number greater than 1001

then the full carry would be set and is therefore used as a flag to transfer 1 to the next byte. This may all be a little confusing for you, so let's see the command in action.

```
 1                ; DAA Counter Demo Program - DAA/SOF
 2                ;  Copyright (C) 1983 BY P.Knaggs
 3                ;
 4                          ORG  0A000H
 5                ;
 6 A000 00        NUMBERH:  DEFB 0    ; Reserve space for counter
 7 A001 00        NUMBERL:  DEFB 0
 8                ;
 9 A002 010100    START:    LD   BC,1           ; Set add value
10 A005 CD1AA0             CALL AddToNumber    ; Add it
11  A008 DA50A0             JP   C,PROGEND      ; Is Counter over
the limit?
12 A00B CD34A0             CALL DisplayNumber  ; No - Print
13                ;
14                ; A simple delay so as you can see the value
15 A00E 0605               LD   B,5
16 A010 48       LOOP:     LD   C,B
17 A011 06FF               LD   B,255
18 A013 10FE               DJNZ $
19 A015 41                 LD   B,C
20 A016 10F8               DJNZ LOOP
21 A018 18E8               JR   START
22                ;
23                ;
24                ;
25                ; ****************************
26                ; ***   THE SUBROUTINES.   ***
27                ; ****************************
28                ;
29                ;
30           AddToNumber: EQU  $
31 A01A 78                 LD   A,B       ; Is it all complete
32 A01B B1                 OR   C
33 A01C C8                 RET  Z              ; Yes - Return
34 A01D 0B                 DEC  BC             ; No - minus 1
35 A01E 3A01A0             LD   A,(NUMBERL)    ; Get the value
36 A021 3C                 INC  A              ; Add one to it
37 A022 27                 DAA                ; Decimal adjust it
38 A023 3201A0             LD   (NUMBERL),A    ; Put it back
39 A026 30F2               JR   NC,AddToNumber ; Is it over 99 ?
40                            ; No - Repeat for all of add value
41 A028 3F                 CCF      ; Yes - Stop DAA taking the
41                                     ;the carry into account
42 A029 3A00A0             LD   A,(NUMBERH) ; Get the high value
43 A02C 3C                 INC  A              ; Add one to it
44 A02D 27                 DAA              ; Decimal adjust it
45 A02E 3200A0             LD   (NUMBERH),A    ; Put it back
46 A031 D8                 RET  C            ; Return if over 99
47 A032 18E6               JR   AddToNumber ; Else repeat for
48                ;
49                ;
```

```
50                DisplayNumber: EQU  $
51 A034 21103C                   LD   HL,3C10H   ; Point to the screen
52 A037 3A00A0                   LD   A,(NUMBERH); Get the high part
53 A03A CD40A0                   CALL DISPLAY      ; Print it
54 A03D 3A01A0                   LD   A,(NUMBERL) ; Get the low part
55                ;
56 A040 F5        DISPLAY:       PUSH AF         ; Save the BCD value
57 A041 0F                       RRCA
58 A042 0F                       RRCA
59 A043 0F                        RRCA            ; swop high  nyb  for
low nyb
60 A044 0F                       RRCA
61  A045  CD49A0                   CALL  NYB              ;  Print  the
Nybble
62 A048 F1                       POP  AF       ; Get the original value
63                ;
64 A049 E60F      NYB:           AND  0FH       ; Strip High Nybble
65 A04B C630                     ADD  A,'0'     ; Add the ASCII offset
66 A04D 77                       LD   (HL),A    ; Put it on the screen
67 A04E 23                       INC  HL        ; Move the pointer on
68 A04F C9                       RET            ; Return from routine
69                ;
70                ;
71                ;
72 A050 C350A0    PROGEND:       JP   $              ; End of routine
73                ;
74                               EXEC START
75                               END
```

Cross Reference Table

NUMBERH	A000	6	42	45	52
NUMBERL	A001	7	35	38	54
START	A002	9	21	74	
LOOP	A010	16	20		
AddToNumber	A01A	30	10	39	47
DisplayNumber	A034	50	12		
DISPLAY	A040	56	53		
NYB	A049	64	61		
PROGEND	A050	72	11		

This program simply fills up the counter to 99 as a BCD digit. The routine AddToNumber adds one to the counter value. The routine DisplayNumber will move the high Nybble (1/2 a byte) to the low nybble. It then makes the value an ASCII character and puts it onto the screen. It then repeats the process for the low nybble. Play with it and see what you think.

P.Knaggs
12, Seymour Road, CHIPPENHAM, Wilts. SN15 3NH
Tel: (0249) 654940

OGGY OGGY OGGY

Sorry to have missed last month - pressures of work can be but a poor excuse !

Now, although I'm writing this in July, you should be reading it in September, just in time to be reminded about THE BLANDFORD WORKSHOP - SUNDAY SEPTEMBER 7th - see you there ! The Swindon agenda is coming together nicely. The emphasis this year seems likely to be on "applications" software, with talks on CP/M software common commands, DOS window routines, and, hopefully, a demonstration of Deskmate running on a 2000. Brian is offering a comprehensive session on 'Sage', the very powerful Model 4 accounting package, and Ariela will be giving us a further dBase II tutorial. I can only repeat my previous comment - if there is ANYTHING that YOU would like to see discussed or demonstrated at Swindon, then ring either myself or Bob Sparling - WITHOUT DELAY ! We're here to oblige, you know!

After what seems like years, I've suddenly had hundreds (well, actually three) requests for info from the 80-Micro library and so I've been galvanised into action to update the listings for same. (For the benefit of the new readers, 80-Micro is an American magazine dedicated to the Tandy computer, and we have a copy of each issue from which photocopies are available to NATGUG members). Since 80-Micro now publishes an annual index, I am now only recording all subsequent references and updates so that these can be supplied with any requested article.

That's all for this month folks, get your Swindon booking forms in as soon as possible (it helps the Hotel to know how much extra beer to get in !).

PS. If you do want to ring me at home (0373 72739) please remember my 8.30pm curfew - Thanks !!

KERMIT - Part IV

Rules and Heuristics

During a file transfer, one KERMIT sends information packets - file headers, data, and so forth, and the other KERMIT sends only ACKs or NAKs in response. The most important rule in the KERMIT protocol is

1. "Wait for a response before sending the next packet."

This prevents buffer overruns, and allows participation by half duplex systems. Of course, KERMIT should not wait forever; a timeout should occur after a few seconds if the expected packet has not arrived. Upon timeout, a sending KERMIT retransmits the current packet; a receiving KERMIT re-ACKs the current packet or NAKs the expected one. Some interesting heuristics are used in the KERMIT protocol to boost efficiency and improve error recovery. A number of important rules take care of the cases when packets are lost in transmission. The first can be stated as

2. "A NAK for the next packet implies an ACK for the current packet."

A NAK with packet number n+1 means the receiving KERMIT got packet n, sent an ACK that was never received, and not knowing that the ACK didn't get through (since we don't ACK an ACK), is now waiting for packet n+1, which was never sent. In this case, we simply send packet n+1. An important exception is when the missing ACK is for a Send-Init packet; do you see why? The next rule,

3. "ACK and discard redundant packets."

handles the situation where the same packet arrives again. The sending KERMIT timed out waiting for an ACK which was sent, but lost, and retransmitted the packet. The receiver must discard the redundant data and ACK the packet again; to do otherwise could have undesirable effects, like adding the same data to a file twice or opening the same file multiple times. Note that the situation resulting from a lost ACK depends upon which side times out first. KERMIT must handle another situation arising from possible lost packets:

4. "NAK the expected command."

The potential problem occurs in either the Receive Init state or when a KERMIT server is waiting for a command. In either case KERMIT won't know whether communication has begun if the other side's initial packet was lost. KERMIT can't assume the other side will time out and retransmit, so it must check periodically by sending a NAK for packet zero. If KERMIT gets no response it assumes nothing has happened yet and goes back to sleep for a while. But sending periodic NAKs opens the door to the buffering problem. Some systems buffer input for a device; when a program isn't looking for input some or all the input is saved by the computer for future re quests. This can cause problems

when talking to a KERMIT server or sending a file to a KERMIT in receive wait. If some time has elapsed since activating the remote KERMIT and escaping back and starting up the local KERMIT, a number of NAKs may have accumulated in the local KERMIT's input buffer, so:

5. "Clear the input buffer at the beginning of a transfer."

If the input buffer is not cleared, the local KERMIT will think the remote side is having trouble receiving the first packet. In an effort to get the packet through, it will be sent again; this repeats for every NAK waiting in the input buffer. By the time the first ACK is finally encountered in the buffer, a number of duplicates of the first packet will have been sent out. If this number exceeds the NAK threshhold, the connection will be broken. If not, however, the second packet will be retransmitted once for each of the extra ACKs the remote KERMIT correctly sent for the duplicate first packets. This can continue throughout the file transfer, causing each packet to be sent many times. So, in addition,

6. "Clear the input buffer after reading each packet."

Any computer that buffers its input should clear its input buffer before the transfer and after each packet that arrives successfully. But since not all systems provide a clear-buffer function, we may add another rule:

7. "Discard redundant ACKs."

In the situation just described, the first packet would be sent out multiple times, once for each buffered NAK. Upon receipt of the first ACK, the second packet would go out, but the next response would be another ACK for the first packet; by rule 7, KERMIT would simply take in the redundant ACK, ignore it, and look for the next ACK, until it got the desired one, in violation of the spirit of Rule 1. If we allowed ourselves to violate Rule 1, we could add a final rule, "An ACK for packet n also ACKs all previous packets," as is done in network protocols like DDCMP, allowing data packets to be sent in a continuous stream. KERMIT cannot use this rule for many reasons: sequence number wraparound, buffer overflows, locking out half duplex systems (how can they NAK a bad packet if they can't get control of the line?). Thus if we violate rule 1, it must be only in a very minor way. Here is a sequence of packets from a real file transfer. Each packet starts with Control-A, shown as ©A.

```
©A) SH( @-#©                                          !Send Init
©A) YH( @-#%                                    !ACK for Send Init
©A+!FMOON.DOC2                                        !File Header
©A#!Y?                                         !ACK for File Header
©AE"D  No celestial body has required J  !First Data packet
©A#"Y@                                  !ACK for first Data packet
©AE#Das m%%%uch labor for the study of its#      !Second Data
                                                  !packet, Bad
```

```
©A##N8                                  !NAK for second Data packet
©AE#Das much labor for the study of its#    !Second Data packet
                                            !again
©A##YA                                      !ACK for second
                                            !Data packet
©AE$D#M#Jmotion as the moon.  Since ClaA    !etc...
©A#$YB      (many packets omitted here)
©AD"Dout 300 terms are sufficient.#M#JU     !Last Data packet
©A#"Y@                                      !ACK for last Data
©A##ZB                                      !EOF
©A##YA                                      !ACK for EOF
©A#$B+                                      !EOT
©A#$YB                                      !ACK for EOT
```

In the first packet, we see following the control-A the packet length ")" (41 , less 32, or 9), followed by the packet type, S (Send-Init), followed by the appropriate parameter declarations: maximum packet length is H (72-32=40), timeout is "(" (40-32=8), number of padding characters is 0 (space=32-32=0), the padding character is 0, end-of-line is "-" (45-32=13, the ASCII value of carriage return), the control-quote character is "#", and the remaining fields are omitted, defaulting to appropriate values. The final character "©" is the single-character checksum, computed as follows (all numbers and computations in decimal, and "sp" represents a space):

```
    )    sp   S    H    (    sp   @       #
    41 + 32 + 83 + 72 + 40 + 32 + 64  + 45 + 35 = 444
```

Remember the various bits are combined in the single character checksum in the following way

CHK = S + ((S AND 192) / 64) AND 63

where S is the sum of the ASCII values

therefore

CHK = 444 + ((444 AND 192) / 64) AND 63 = 62

convert to a printable character with the char function

char(62) = 62+32 = 94 = "©"

The receiver ACKs with its own parameters, which are the same. Then comes the file header, the file, EOF, and EOT. One data packet was corrupted by a burst of "%" characters, NAK'd, and retransmitted.

* Performance

For text files (documents or program source), assuming an average line length of 40 with lines separated by a carriage-return/linefeed pair, the only control characters normally found in the text file, we see about 5% overhead for prefixing of control characters. Assuming no line terminators for packets (although one or both sides may require them), no retransmissions or timeouts, and no time wasted for the line to turn around between packet and response, then for average packet length p, using a single-character checksum, the KERMIT protocol overhead consists of:

5 control field characters in the data packet
5 characters in the acknowledgement packet
+ 0.05p for control character quoting

This gives 10/p + 0.05 overhead. E.g. if the packet length is 40, there is 30% overhead. If p is 96 (the maximum), there is about 15%. These figures will vary with the average line length and the frequency of other control characters (like tabs and formfeeds) in the file, and will go up with immediate retransmissions, and way up with delayed retransmissions. For binary files, the quoting overhead will be higher. But transmission overhead can also go down dramatically if prefix encoding is used for repeated characters, depending on the nature of the data (binary data containing many zeroes, highly indented or columnar data or program text will tend to benefit). Each file transfer also gets a fixed overhead for the preliminary (Send Init, File Header) and terminating (EOF, EOT) packets. If the mainframe end of a connection is heavily loaded, it may take considerable time to digest and process incoming characters before replying. On half duplex mainframes, there may be a pause between sending and receiving, even if the load is light; this might be used to advantage by preparing the next packet in advance while waiting for the current ACK. Another problem may occur on heavily loaded mainframes, that of undesirable timeouts. Timeouts are intended to detect lost packets. A heavily loaded system may take longer than the timeout interval to send a packet. For this reason, mainframe KERMITs should take the requested timeout interval only as a minimum, and should adjust it for each packet based on the current system load, up to a reasonable maximum. On a noisy line, there is a greater likelihood of corrupted packets and therefore of retransmission overhead. Performance on noisy lines can be improved by reducing the packet length, and thus the probability that any particular packet will be corrupted, and the amount of time required to retransmit a corrupted packet. A KERMIT program can unilaterally adjust the packet length according to the number of retransmissions that are occurring. Short packets cut down on retransmission overhead, long packets cut down on character overhead.

* "User Interface"

KERMIT was designed from a mainframe perspective. Like many mainframe programs, KERMIT issues a prompt, the user types a command, KERMIT executes the command and issues another prompt, and so on until the user exits from the program. Much care is devoted to the command parser, even on microcomputer versions. The goal is to provide English-like commands composed of sequences of keywords or operands, with abbreviations possible for any keyword in any field down to the minimum unique length, and with "?" help available at any point in a command. Not all implementations need follow this model, but most do. The basic commands are SEND and RECEIVE. These allow most KERMITs to exchange files. Operands can be the name of a single file, or a file group designator (e.g. with "wildcards") to transmit multiple files in a single operation. Although some systems may not provide wildcard file processing, the KERMIT

protocol allows it. The CONNECT command provides the mechanism for logging in and typing commands at the remote host, which is necessary in order to start the KERMIT on that side. The CONNECT facility provides character-at-a-time transmission, parity selection, remote or local echoing, and the ability to send any character, including the "escape character" that must be used to get back to the local KERMIT. However, there is no error detection or correction during CONNECT, just as there normally is none between an ordinary terminal and a host. When two systems are dissimilar, a SET command is provided to allow them to accommodate each other's peculiarities, for instance SET PARITY ODD to add odd parity to all outbound characters, or SET LOCAL-ECHO to do local echoing when connected as a terminal to a half duplex system. The SET command must sometimes be used to supply information to the target system on how to store an incoming file with respect to block size, byte size, record format, record length. Most KERMIT implementations take special care to reassure the user during file transfer. The names of the files being transferred are shown, and a dynamic display is made of the packet traffic, showing successful transmission of packets as well as timeouts and retransmissions. Messages are issued when the user connects to the remote system or escapes back from it, and KERMIT prompts identify the implementation. Helpful error messages are displayed when necessary; these may emanate from either the local or the remote system. The final disposition of the transfer is clearly stated, complete or failed. The actions required of the KERMIT user depend upon the degree to which the KERMIT programs involved have implemented the specification. Minimal implementations require that the user connect to the remote host, start KERMIT there, issue a SEND (or RECEIVE) command, escape back to the local machine, and issue the complementary RECEIVE (or SEND) command. All this must be done for each transfer. More advanced implementations allow the remote side to run as a "server" and to take all its instructions in special command packets from the local KERMIT; all that is required of the user on the remote end is to connect initially in order to start the server. The server will even log itself out upon command from the local KERMIT. A minimal server can process commands to send files, receive files, and shut itself down. Given overpage is an example of a session in which the user of an IBM PC gets files from a DECSYSTEM-20. The actions shown are required for minimal KERMIT implementations. The parts the user types are underlined, comments begin with "!". Everything else is system typeout.

```
A>kermit                          ! Run Kermit on the PC.
Kermit V1.20
Kermit-86>          ! This is the Kermit prompt for the PC.
Kermit-86>connect      ! Make connection with mainframe.
  [Connecting to host. Type CTRL-]C to return to PC.]
@terminal vt52     ! Set your terminal type (optional).
@login my-id password  ! Login using normal login method.

    (The DEC-20 prints various messages.)

@kermit                           ! Run Kermit on the DEC-20.
 Kermit-20>                       ! This is Kermit-20's prompt.
 Kermit-20>send *.for             ! Send all FORTRAN files.
 ©]c                              ! Type the escape sequence to
                                  ! return to the PC.
          [Back at PC.]
 Kermit-86>receive                ! Tell the PC files are coming.

  (The progress of the transfer is shown continuously on the
                                                       screen.)
          Transfer Complete.
 Kermit-86>connect                ! Get back to the DEC-20.
 [Connecting to host. Type CTRL-]C to return to PC.]
 Kermit-20>exit                   ! Get out of Kermit-20.
 @logout                          ! Logout from the DEC-20.
   Logged out Job 55, User MY-ID, Account MY-ACCOUNT, TTY 146,
     at 24-Apr-83 15:18:56,  Used 0:00:17 in 0:21:55
  ©]c                             ! Now "escape" back to the PC,
 [Back at PC.]
  Kermit-86>exit                  ! and exit from the PC's Kermit.
```

The session is somewhat simpler when the remote KERMIT is being run as a server. The user must still CONNECT, log in, and start KERMIT on the remote end, but need never again CONNECT to issue subsequent SEND, RECEIVE, EXIT,LOGOUT commands, even though many transactions may take place. All actions can be initiated from the PC.

Next month really will be the final session on Kermit where I will detail the commands available in the version that run on Model 1/3 (under TRSDOS etc), Model 4/Genie CPM, Model 4 TRSDOS 6.x and PC versions.

READERS ADVERTISEMENTS

FOR SALE

Tandy DMP 110 printer and Tandy Model I computer.
H. Sayer 79, Miller Drive, Fareham.
Telephone: Fareham 236582.

FOR SALE

Tandy Model I Level 2 64K with expansion interface, RS232 and cable, three disk drives, LPVIII printer and varied software library. All enquiries to -
A.M. Morwood-Leyland, 26 Parklands, Billericay, Essex CM11 1AS.
Telephone: (02774) 56253

FOR SALE

1 Tape-based VIDEO GENIE with many tapes (mainly games)	£30
1 Disc-based GENIE I (twin disc drive, amny discs with RS232-Tandy compatible with software & desk)	£150
1 Modem 300 Baud originate only with Telephone	£35
1 Acustic Modem 300 Baud originate and answer	£30

ORIGINAL SOFTWARE WITH DOCUMENTATION

Multidos 1.7 for Model 1 Double Density	£25
Dotwriter with many fonts	£20
Copycat - copying utility	£10
Duplidisk V. 2.3	£ 5
Modem 80	£10

The above is offered O.N.O. - buyer collects.
Please call Don Harrison on 01-575 2678
24 Gifford Gardens, Hanwell, London W7 3AN.

NEW ADDRESS
As from 7th July, 1986 Leighton Davies will be contactable at Glanmor, Brynna Road, Pencoed, Bridgend, Mid Glamorgan.
Telephone - (0656)-860398.

RB080486 A survey of NATGUG computers.

Original Micro computer system.

1) Micro computer model:
2) Date when purchased :
3) Price :
4) List ALL later additions :

About yourself.

1) Are you ;
i) a student . . . . . :
ii) a professional. (please specify):
iii) other (please specify) . . . . :

About the system

1) Why did you buy a computer ?

2) Why did you buy a model from the Tandy range?

3) What uses did you envisage for your computer ?

4) What was it actually used for ?

5) Where you content with system purchased ?

6) If no to 5 then please state your upgrade path ?

7) What high level languages did you use on your system ? If more than one, please put in order of use:

8) Did you use an assembler on your system ?

9) Please list the top 5 application programs that you used ?

10) How many HOURS were spent on your system per week ?

11) What (if any) features of your system were most irritating ?

12) What (if any) features of your system were most pleasing ?

13) What is your impression of the reliability of the system

14) What problems (if any) have you had with your system ?

15) Please comment on :-
i) Support given by Tandy or supplier

ii) Support given by NATGUG or other local user groups:

Documentation.

1) Was the documentation for setting up :-
/ EXCELLENT / GOOD / POOR / USELESS /

2) Was the documentation for using BASIC :-
/ EXCELLENT / GOOD / POOR / USELESS /

3) Was the documentation for using Machine Code :-
/ EXCELLENT / GOOD / POOR / USELESS /

3) List the best features of the documentation.

4) List the worst features of the documentation.

5) Make any additional comments about documentation here:

Changing system. (only applicable if you have changed system)

1) Name new micro :
2) Date . . . . . :
3) Price. . . . . :

4) What was your reason for changing ?

5) Where did the old machine go ?

6) What features do you like about new system ?

vi) What features don't you like about new system ?

vii) What features did you miss when comparing new with old ?

Please return the completed form with any additional comments to :
R. Bains, 452 Leagrave Rd., Luton, Beds., LU3 1RH.
Thank you for taking the time to complete this questionaire.