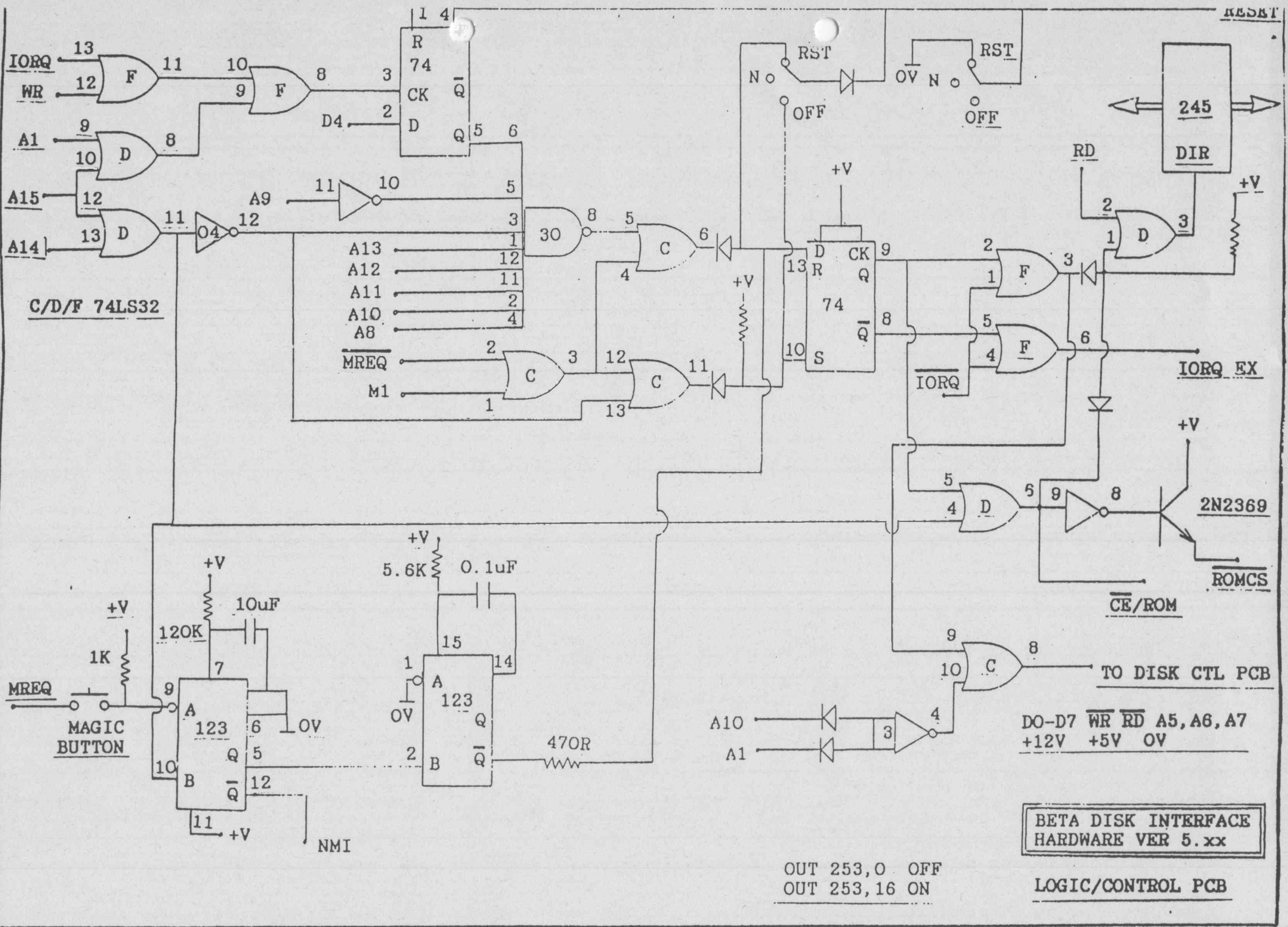


DO-D7 $\overline{WR}$ $\overline{RD}$ A5,A6,A7
+12V +5V OV

BETA DISK INTERFACE
HARDWARE VER 5.xx

OUT 253,0 OFF
OUT 253,16 ON

LOGIC/CONTROL PCB

B.D.U.C.

BETA DISK USERS CLUB

BETA DISK NEWSLETTER NO. 1

Welcome to this the first issue of the BDUC newsletter. This newsletter is intended to provide hints, examples and advice to users of the Technology Research Beta 128 and Beta Plus disk interfaces. The club is non-commercial and non-profit making and run on a voluntary basis. Your subscription covers the cost of postage, stationery and photo-copying etc.

Since this is the first issue, I had better introduce myself. I am Martyn Smith, a computer engineer. I have owned a 48K Spectrum since 1982. I started BDUC recently when I recognised there was a need for users to exchange information and thought a users club might fill the gap or "close the loop". I set out initially with the informal association of users that had already been established by Sam Estall. This group used the pages of Prestel as means of communicating, I did'nt want to restrict membership to Prestel, so I set out to contact as many known users as possible. I used ~~Prestel~~ contacts and Technology Research's user lists to form the club. I might add that this is the "official" Beta Disk Users Club and I must acknowledge the help of Technology Research who have given BDUC support.

I hope in the next issues to include more readers contributions so I need your input. This may take the form of written articles of general interest, specific programming hints or anything connected with Beta Disk use.

You should send your contribution to the address given in this issue. You may send a tape, microdrive, floppy disk (5.25" or 3.5") using Tasword 2 or Tasword 3 text file format. (Tasword Word processors will be used extensively in the preparation of this newsletter). BDUC will return all tapes, cartridges and disks. You may also write, but make sure the letter is either typed or dot matrix printed.

If you have an interesting application of a Beta Disk such as club accounts, stock control or bulletin board, the club would like to hear from you. Also if you know any other users you think would like to join, tell them about BDUC.

BY M.J.SMITH, BDUC.

This program gives "extended" information about the disk and allows manipulation of data available in the LIST output.
Unfortunately due to a bug the Beta cannot LIST into a data file which has been opened for writing on the same disk.
The following example highlights the problem.

```
  10 RANDOMIZE US R DOS: REM : OPEN #9,"CAT",W
  15 RANDOMIZE US R DOS: REM : LIST #9
  20 RANDOMIZE US R DOS: REM : CLOSE #9
```

All apparently looks well but if an attempt is made to read the file and print on the screen thus,

```
  10 RANDOMIZE US R DOS: REM : OPEN #9,"CAT",R
  15 RANDOMIZE US R DOS: REM : MOVE #9 TO #2
  20 RANDOMIZE US R DOS: REM : CLOSE #9
```

The file appears to contain random data. So the microdrive is used to buffer the data.
The first listing creates the data file from the disk LIST and gives the following information: data file type,sectors used,basic length,code length,auto start line,date of LIST, sectors used,sectors free,k-bytes used, k-bytes free,and number of files. You should input the date into line 20,the mdrive number in line 30, disk title in line 40 and alter the variable DOS if your TRDOS calls 15363.

```
  10 LET DOS=15619: REM CALL TRDOS
  20 LET Y$="28:01:87": REM TODAY'S DATE
  30 LET D=1: REM MDRIVE SELECT
  40 LET D$="1": REM DISK TITLE
  50 PRINT AT 0,0;"PUT A SPARE CART IN DRIVE ";D;AT 2,0;"PRESS A
     KEY": PAUSE 0
  60 ERASE "M";D;D$
  70 OPEN #6;"M";D;D$
  80 PRINT AT 6,0;"INSERT DISK:";D$;" IN DRIVE A";AT  8,0;"PRESS
     A KEY": PAUSE 0
  90 RANDOMIZE USR DOS: REM : LIST #6
 100 REM CALL TO TRDOS LIST DISK CONTENTS INTO MDRIVE STREAM
 110 PRINT #6;" "
 120 PRINT #6;" "
 130 PRINT #6;" "
 140 PRINT #6;Y$ : REM DATE LABEL
 150 PRINT #6;"0": REM END OF FILE MARKER
 160 PRINT AT 10,0;"DISK:";D$;" OK"
 170 CLOSE #6
```

Once you have created the data file in mdrive 1 you may test all is well by entering the following:

```
  10 MOVE "M";1;"1" TO #2
```

The disk name in this example is "1" as in the listing above. This line displays the data file in a similar way to the normal LIST but note the date stamp and the end of file marker "0". The second listing calculates the information from the data file and outputs to the screen, also creating a file "*LIST" on drive A, this file now contains the extended information. Remember to set DOS to suit.

```
  10 CLEAR #
  11 LET DOS=15619:RANDOMIZE USR DOS: REM : OPEN#7,"*LIST",W
  20 LET S=2544: REM SECTORS SET FOR YOUR CONFIG
     80T DS 2544,40T DS 1264,40T SS 624
  30 DIM Z$(62): DIM D$(62)
  40 LET X$="1": REM DISK TITLE
  50 LET D=1: REM MDRIVE SELECT
  60 OPEN #9;"M";D;X$
  70 INPUT #9;D$
  80 INPUT #9;D$
  90 INPUT #9;Z$
 100 IF Z$(3)=":" AND Z$(6)=":" THEN LET P$=Z$(1 TO )
 110 IF Z$(1 TO 8)="0       " THEN GO TO 210
 120 IF Z$(3 TO 7)="File " THEN GO TO 140
 130 GO TO 90
 140 INPUT #9;Z$
 150 IF Z$(3)=":" AND Z$(6)=":" THEN LET P$=Z$(1 TO )
 160 IF Z$(1 TO 8)="0       " THEN GO TO 210
 170 IF Z$(1 TO 12)="            " THEN GO TO 90
 180 IF Z$(10)="B" THEN LET Z$(18 TO 22)="0"
 190 PRINT Z$(10);":";Z$(1 TO 8);":";Z$(12 TO 14);":";Z$(18 TO 2
     2);":";Z$(24 TO 28);(Z$(29 TO 32) AND Z$(10)="B")
 191 PRINT #7;Z$(10);":";Z$(1 TO 8);":";Z$(12 TO 14);":"Z$(18 TO
     22);":";Z$(24 TO 28);(Z$(29 TO 32) AND Z$(10)="B")
 200 GO TO 140
 210 LET L=LEN D$: LET N=VAL (D$(1 TO 2)): LET FS=VAL (D$(L-4 TO
     L)): LET US=S-FS: LET FK=FS/4: LET UK=(S/4)-FK
 220 PRINT '''DATE:";P$(1 TO 9);TAB 16;N;" FILE"+("S" AND N>1)
 221 PRINT #7;'''DATE:";P$(1 TO 9);TAB 16;"N;" FILE"+("S" AND N>
     1)
 230 PRINT "USED:";INT US;TAB 10;INT UK;"K"
 231 PRINT #7;"USED:";INT US;TAB 20;INT UK;"K"
 240 PRINT "FREE:";INT FS;TAB 10;INT FK;"K"
 241 PRINT #7;"FREE:";INT FS;TAB 20;INT FK;"K"
 250 CLOSE #9
 260 CLOSE #7
```

The third part transfers the extended LIST back to disk.

```
 10 RANDOMIZE USR DOS: OPEN #8,"LIST",W
 20 MOVE "M";1;"*LIST" TO #8
 25 RANDOMIZE USR DOS: CLOSE #8
```

Once this is transferred back to disk the extended information is now available for display or can be used in the auto load program. The auto load program selects the BASIC files from the extended catalogue filename *LIST. It prints the files on the screen which may then be selected by number and loaded. The extended information file *LIST must however be kept up to date since titles actually on the disk may not be loaded if they are not in the *LIST file.
This is an easy way to load BASIC files since only one keypress is required. Further modification to the listing is required if CODE files are also to be loaded in the same way.

```
 10 CLEAR #
 20 DIM Z$(60)
 30 DIM T$(50,10): LET T=1
 40 LET E=USR DOS: REM: OPEN #9,"*LIST",R
 50 INPUT #9;Z$
 60 IF Z$(1)="0" THEN GO TO 140
 70 IF Z$(3 TO 7)="File " THEN GO TO 90
 80 GO TO 50
 90 INPUT #9;Z$
100 IF Z$(1)="0" THEN GO TO 140
110 IF Z$(1 TO 12)="            " THEN GO TO 50
120 IF Z$(10)="B" THEN LET T$(T)=Z$(1 TO 8): LET T=T+1
130 GO TO 90
140 LET E=USR DOS: REM: CLOSE #9
150 CLS
160 FOR X=1 TO T-1 STEP 2
170 PRINT X;TAB 4;T$(X);TAB 16;X+1;TAB 20;T$(X+1)
180 NEXT X
190 INPUT "SELECT TO LOAD ";X
200 LET E=USR DOS: REM: LOAD T$(X)
```

Suggestions for improvements or enhancements to any programs published in the newsletter are most welcome. Please send them for evaluation. I will publish any improvements in future issues.

)

BY M.J.SMITH, BDUC.

This program is for Beta Disk users who also use Beta Basic by Betasoft. Beta Basic is an extension to Sinclair Basic with many additional commands and functions.
The first part transfers Beta Basic to disk and the second part defines a procedure which eliminates the call to TRDOS by calling RANDOMIZE USR 15616, instead the word DOS is typed. This saves time and is most convenient.

To transfer Beta Basic to disk. Load the normal tape version of Beta Basic or your own customised version. Whilst Beta Basic is running type in the following:
In direct mode POKE 58457,195:POKE 58458,100:POKE 58459,228 this disables the normal Beta Basic initialisation.

```
 1 LET RT=DPEEK(23730)
 2 RANDOMIZE USR 59904
 3 RANDOMIZE USR 15619: REM: SAVE "boot" LINE 6
 4 RANDOMIZE USR 15619: REM: SAVE "BB"CODE RT+1,65367-RT
 5 STOP
 6 CLEAR RT: BORDER 0: PAPER 0: INK 7: CLS
 7 RANDOMIZE USR 15619: REM: LOAD "BB"CODE
 8 RANDOMIZE USR 58419: PAUSE 0: CLS: DELETE 1 TO 8
```

This saves Beta Basic to disk, the second part defines PROC DOS.

```
 1 DEF PROC DOS
 2 RANDOMIZ E USR 63243
 3 LET E=USR 15616
 4 END PROC
```

In PROC DOS the RANDOMIZE USR 63243 statement disables Beta Basic's IM2. This means that Beta Basic lives more happily with TRDOS and there is no need to switch Beta Basic off before using any TRDOS commands. However with IM2 disabled the interrupt driven clock does not update. You can restore IM2 with RANDOMIZE USR 61369. The PROC DOS can be hidden in line 0 with the following PROC, which may be merged with the program above.

```
101 DEF PROC HIDE
102 LET X=DPEEK(23635)
103 DO
104 LET X=X+DPEEK(X+2)+4
105 EXIT IF PEEK (X+1)=100
106 POKE X+1,0
107 LOOP
108 END PROC
```

Here's an article of interest to all disk users, it's from Keith Burton (Spector) who runs the PHANTOM BB.

DISKS-ARE THEY ALL THE SAME?

About five years ago you'd be lucky to buy a pack of ten floppy disks for less than £30 (the 'new' 80 track disks would be even more!!) Considering inflation over the last five years, it's heartening to see that we are not now paying around £50 for a pack of ten. Disks are now as little as £10 for a box of 10.
Of course there are pitfalls in going for the cheaper disks and this article should help you select the right disk for the right job. First of all, what do all those abbreviations mean?

DD = Double Density	SD = Single Density
DS = Double Sided	SS = Single Sided

TPI = Tracks Per Inch

Translating this into something understandable is fairly straightforward. All disks are manufactured in batches with each batch undergoing certification testing. A batch of disks passing all the tests becomes DD/DS 96TPI - Some companies even go as far as claiming Quad Density certification (QD/DS !!!)
Batches of disks only passing the minimum test levels will be certified SD/SS 48TPI. There are various intermediate levels too, but all discs from a certain manufacturer usually come off the same production line, regardless of their certification rating. When choosing disks you should be able to match the specifications of your DOS and disk drives to those of the required disks. 96 and 48TPI ratings equate roughly to 80 and 40 track disk drives however TPI has no relation to density as thats to do with the way the data is packed on the disk. Double density DOS's make greater demands on the performance of the disk. Obviously DD DOS's should be used with DD disks. Single density 96TPI disks may not be adequate for the demands of a DD DOS. However more often than not you will be able to format 40 track SD disks on 80 track drives in DD - successfully!!
Beware - cheap disks are usually just that CHEAP! However there are plenty of sources of perfectly good cheap disks if you shop around. If you are buying a new brand of budget disks for the first time, ensure that you can get a refund or an exchange if they prove to be unreliable. Its a good idea to format a few or even all the box of disks as disk formatting and verifying is a good test of the disk. Other things to look out for when shopping for new disks are hub reinforcement rings, strong or stiff disk jackets, which have been neatly glued on the underside and a good set of labels and stickers. Finally some dealers are selling disks in handy plastic boxes - they will help protect your disks and are well worth looking out for.

DISK CARE- A QUICK QUIDE.

We have all at some time come across a disk that won't read a sector or file, the usual cause is the disk itself. It's a flexible device with a lot of data stored in a small area and so that data is prone to failure.
You will have seen the warnings on disk sleeves and boxes, temperature, dust, food and drink can all cause harm to your disks. One of the most important things is to never bend your disks (that is not why we call them floppy!!) A slight kink in the surface of the disk will make it unreadable.
Disks shouldn't be left near sources of strong magnetic fields like speakers or monitors. Some users of larger micros have experienced problems due to the drive being directly below the monitor and the manufacturer has not provided adequate protection for the disk in the form of screening of magnetic feilds.
Don't smoke near disks - the particles that fly off a cigarette are potentially devastating to the disk surface. Another problem that can occur is the misalignment of the drive heads themselves. The heads inside the drive which read or write data to the disk are mounted on a moving platform, which is moved across the disk surface by a stepper motor. If the heads become misaligned the disk will become unreliable, disks become suddenly unreadable. This misalignment may occur if the drive is dropped or recieved a knock, wear and tear can also cause it.
If misaligmment is your problem take it to a repair shop. DIY alignment can do more harm than good!

Well that's all for this issue!

I hope you have found something of interest in this issue. The next issue should be ready towards the end of May. Please don't forget to send your contributions!

You may contact BDUC by phone during evenings and weekends on 0706 218354 or Prestel 204885283.

B.D.U.C.

BETA DISK USERS CLUB

BETA DISK NEWSLETTER NO. 2

STOP PRESS...STOP PRESS...STOP PRESS...STOP PRESS...STOP PRESS

March.20th. Reports reaching BDUC confirm that TR ceased trading as of today. BDUC advised members not to return disk interfaces or drives to TR for repair, or send cash for goods. Enquiries should be directed to the Official Receiver in Reading.

Welcome to this the second issue of the BDUC newsletter. After the depressing news of TR's collapse I can assure members that this does not signify the end of BDUC!

Thank You to BDUC members for comments and contributions. The BDUC survey data is currently being compiled and should be available next issue, in the mean time if you have not returned your survey questionaire, please do!

Some users with older versions of TRDOS (4.xx or earlier) have indicated that these early revisions did not support the LIST function, it should be relatively easy to use CAT to provide information for an extended catalogue similar to the program in issue 1.

WHOOPS!

In the conversion from programs developed using Beta Basic for inclusion in the newsletter one or two bugs crept in. Here are the corrections.

Page 3 line 260
```
 260 RANDOMIZE USR DOS: REM: CLOSE #7
```

Page 4 line 10,20,25
```
  10 RANDOMIZE USR DOS: REM: OPEN #8,"LIST",W
  20 MOVE "M";1;"1" TO #8
  25 RANDOMIZE USR DOS: REM: CLOSE #8
```

Page 4 line 40
```
  40 LET E=USR DOS: REM: OPEN #9,"LIST",R
```

In future editions BDUC will wherever possible include programs for Beta+ and Beta 128 interfaces.

TRDOS and RANDOMIZE USR 15616. BY M.J.SMITH, BDUC.

The 48K Spectrum operating system in ROM occupies 16K of memory from location 0 to 16383. It can therefore be seen that 15616 is within this range, so why does TRDOS make a call to what is normally a ROM address?

First of all we need to investigate the contents of address 15616 without the Beta disk connected. So armed with your favorite disassembler look at the address 15616 (3D00H). You should find something like this:

```
          15616  3D00H  00  NOP   TO   15624  3D08H  00  NOP
```

This location is the start of the Sinclair character set bit-map and is character code 32 which is 'SPACE'. The next step is to refit your Beta and save TRDOS to disk. This is easily accomplished by entering TRDOS and typing directly:

```
                    SAVE "TRDOS" CODE 0,16384
```

This saves 16K of code whilst TRDOS is paged in. The actual hardware that pages in TRDOS is complex but here is the basic idea explained.

This is my own personal interpretation, if anyone has disected a Beta interface please correct me if I'm wrong!

A call to 15616 causes a particular logic state on the address bus of the interface, in this case:

```
                               A15                 A0
      15616 = 3D00H            3    D    0    0
                               0011 1101 0000 0000
      15619 = 3D03H            3    D    0    0
                               0011 1101 0000 0011
```

The address lines are decoded by logic gates and lines A15 to A8 of the address bus are used. Once the condition is met, then TRDOS is paged in a similar way to the Interface 1 ROM is paged (Interface 1 uses address 8 which is easily decoded) the Spectrum operating system is temporarily 'replaced' by TRDOS. The code at location 15616 is then executed. Here's a quick disassembly of the code at 15616, notice now that TRDOS has replaced the eight NOP's with the code:

```
      15616  3D00H  00     NOP        ;wait one m cycle
      15617  3D01H  182E   JR 15665   ;jump to routine
      15619  3D03H  00     NOP        ;wait one m cycle
      15620  3D04H  1814   JR 15642   ;jump to routine
```

The reason for a time delay is because hardware paging can take longer than 1 m cycle and the time delay is included to ensure paging has occured correctly before execution of code.

The best way to look at this is to load your TRDOS code at 32000 (7D00H) and disassemble, remember to subtract 32000 from all the absolute addresses. So 15616 = 47616 (BA00H) and 15619 = 47619 (BA03H).

TASWORD 3 TEXT TO BETA DISK. BY M.J.SMITH, BDUC.

This program enables security back-up of Tasword 3 text files to disk. It's quite frustrating to have disk storage on line and be forced to use microdrive with an excellent program such as Tasword 3. BDUC has approached Tasman of Leeds about producing a Beta compatible version but at this present time they have no plans to do so.

Several BDUC members have asked about Beta compatible versions of Tasword 3, I do hope to have further news in the next issue.

Listing 1 produces text files on disk from cartridge and listing 2 transfers files back to microdrive. I use this method to archive my text files.

This version was written for 5.01 or 5.03 TRDOS.

Listing 1.

```
 10 PRINT TAB 9;"TASWORD3-BETA"''"TASWORD3 TEXT IN MDRIVE 1 AND
    BACKUP"'"DISK IN DRIVE A"'" PRESS ANY KEY "
 20 LET F=0
 30 PAUSE 0
 40 CLEAR #
 50 LET E=USR 15619: REM : OPEN #9,"TEMP",W
 60 CAT #9,1
 70 LET E=USR 15619: REM : CLOSE #9
 80 PRINT '"CATALOGUE COMPLETE"
 90 LET E=USR 15619: REM : OPEN #9,"TEMP",R
100 INPUT #9;D$
110 INPUT #9;D$
120 INPUT #9;Z$
130 IF LEN Z$=0 THEN GO TO 210
140 LET F=F+1
150 PRINT AT 12,0;"MOVING:";F;" ";Z$;AT 12,18;"  "
160 LET E=USR 15619: REM : OPEN #10,Z$,W
170 MOVE "M";1;Z$ TO #10
180 LET E=USR 15619: REM : CLOSE #10
190 PRINT AT 12,18;"OK"
200 GO TO 120
210 LET E=USR 15619: REM CLOSE #9
220 PRINT F;" FILES MOVED"
```

The second part transfers archive files back to the current text file cartridge in drive 1.

Listing 2.

```
  10 PRINT "TASWORD3 TEXT DATA IN DISK A"''"MICRODRIVE IN DRIVE
     1 "''; FLASH 1;" PRESS A KEY "
  20 LET F=0
  30 PAUSE 0
  40 LET E=USR 15619: REM : OPEN #9,"TEMP",R
  50 INPUT #9;D$
  60 INPUT #9;D$
  70 INPUT #9;Z$
  80 IF LEN Z$=0 THEN GO TO 160
  90 LET F=F+1
 100 PRINT AT 12,0;"MOVING:";Z$;"                "
 110 RANDOMIZE USR 15619: REM : OPEN #10,Z$,R
 120 MOVE #10 TO "M";1;Z$
 130 LET E=USR 15619: REM : CLOSE #10
 140 PRINT AT 12,18;"OK"
 150 GO TO 70
 160 LET E=USR 15619: REM : CLOSE #9
 170 PRINT F;" FILES MOVED"
```

Both listings could be joined together and modified to give backup of individual files. One point to bear in mind is that the cartridge to be transferred to disk must contain DATA files only since the CAT in line 60 will give all file types but the transfer lines are for DATA files only and a BASIC halt will be produced if a transfer is attempted.

****

BDUC SPECIAL OFFER

BDUC has negotiated a special price on BETA BASIC by BETASOFT.The normal price is £14.95 the club offer price is £12.95. This program is highly recommended by BDUC.
IMPORTANT:To qualify for discount members must send their order to BDUC on the order form enclosed. Overseas members should enquire about discounts.

****

)

Here is a very useful utility that you can include on all your disks it's form Nick Cooper of Leicestershire.

AUTOBOOT BY NICK COOPER.

There are two parts of Basic to type in, but I feel sure you'll find the effort worth it.. The first part is a small program to poke some machine code into the first line of BASIC.
First type in listing one and run it. Delete all the lines except line 10 and save this to disk. Type in the listing two, and save this also.
Now, with the second listing in memory, MERGE in the first listing and save the merged program as 'boot' with auto start line 15. And that's it! Save a copy onto every disk that you want a bootfile on.
As far as I'm aware this program will only work with the Beta 128 interface (Version 5.03) and should work with both the Spectrum 48+ and the 128k machine.
Due to another bug in the Beta i/f, the program will return to Basic with the drive running, if a disk is'nt present although this should'nt occur as a disk has to be present to load it in the first place.

INSTRUCTIONS

Autoboot supports BASIC, CODE and SNAPSHOT files but not data, random and serial access files.
Once loaded, you are offered the following options:-
Keys A-P
Will load the file corresponding to that letter on the screen.

If that file is BASIC, then it will load and control will return back to basic. Before a CODE file is loaded, however, you're first asked where you want it to load to. This can be in the range 30000 upto the top of memory. Should you attempt to load it too high, then you will be told to try again. If you just press ENTER at the prompt the file will load to the address it was saved from. Pressing STOP will abort.
You are then asked if you want to move ramtop. This will move ramtop without destroying any variables. If you enter an address from 30000 to top of memory, then ramtop will move to that address. If you just press ENTER at the prompt then the following will happen. If you are attempting to load below ramtop (or indeed over it) then ramtop will be moved to the load address minus one. On the other hand, if you're attempting to load anywhere above ramtop, then ramtop will be left alone. Pressing STOP will abort.

Once the file is loaded, you are then asked for the run address. Again, any address from 30000 to the top of memory will run the file from that address. If you just press ENTER at the prompt the file will run from the load address. STOP aborts.
If you do not wish the file to run, but return to basic then just press cursor down.
Snapshot files will ask for confirmation before loading as loading one accidently could be disasterous!

Cursor left-right

Will page through each screen full of files. (There are 16 files to a screen making 8 screensfull for a full disk.) Going past the last screen will return to the first.

Options 1,2,3

1. Extended catalogue.
2. Change Disc. If you want to reboot a different disc.
3. Call TRDOS. Will return to the bootfile.

Type in the following listing and follow the instructions given above. Line 10 contains 69 x's, this space should be reserved for machine code poked in by the rest of the prog.

The facility has also been added so that multiple drives are catered for and disc title, number of files, and free space is given in Kbytes.

Listing 1.

```
  10 REM xxxx 69 x's xxxxxxxxxxxxxxxxxxxxxxxxx
  50 RESTORE
  60 LET x=(PEEK 23635+256*PEEK (23636))+5
  70 FOR a=0 TO 68: READ z: POKE x+a,z: NEXT a
  80 STOP
 100 DATA 1,5,1,17,1,0,33,25,105,205
 101 DATA 19,61,201,17,0,0,24,11,17,96
 102 DATA 234,42,178,92,175,237,82,56,39,175
 103 DATA 27,237,83,178,92,103,111,57,68,77
 104 DATA 42,61,92,35,229,183,237,66,68,77
 105 DATA 3,225,62,62,18,27,27,27,237,83
 106 DATA 61,92,19,237,184,19,235,249,201
```

Listing 2.

```
 15 CLEAR : DIM a$(16,16)
 20 LET y=(PEEK 23627+256*PEEK (23628))+8
 30 LET x=(PEEK 23635+256*PEEK (23636))+5
 40 POKE x+7,y-256*INT (y/256): POKE x+8,INT (y/256)
 45 POKE x+4,8: RANDOMIZE USR x: LET t$=a$(16,6 TO 12): LET d$=
    CHR$ ((PEEK 23833)+65): LET fl=CODE (a$(15,5)): LET bf=INT
    ((CODE a$(15,6)+CODE a$(15,7)*256)/4)
 50 LET f=0: LET n=1: POKE x+4,0: RANDOMIZE USR x
 60 CLS : PRINT : PRINT " *Autoboot N.Cooper 1987*"''"  Title
    :";t$," Disc Drive: ";d$,"  ";fl;" file(s)"," Free Kbyte:";
    bf''TAB 22;"Screen{";n;"}: PRINT
100 FOR a=1 TO 16
110 IF a$(a,1)=CHR$ 1 THEN NEXT a
120 IF a$(a,1)=CHR$ 0 THEN LET f=1: GO TO 150
130 PRINT "  ";CHR$ (a+64);":";a$(a, TO 8);"<";a$(a,9);">",
140 NEXT a
150 PRINT : PRINT : PRINT TAB 22;("   more.." AND f=0)+("no mor
    e.." AND f=1)
160 PRINT AT 18,8;"[1] List   Disc ";AT 19,8;"[2] Change Disc";
    AT 20,8;"[3] Enter TRDOS"
170 PLOT 8,0: DRAW 0,175: DRAW 247,0: DRAW 0,-175: DRAW -247,0
200 GO SUB 510: LET z$=INKEY$
210 IF z$=CHR$ 8 AND f=1 THEN GO TO 50
220 IF z$=CHR$ 9 THEN LET n=n+1: POKE x+4,n-1: RANDOMIZE USR x:
    GO TO 60-
225 IF z$=CHR$ 8 AND n=1 THEN LET n=INT (fl/16)+1: POKE x+4,n-1
    LET f=1: GO TO 60
230 IF z$=CHR$ 8 AND n>1 THEN LET n=n-1: POKE x+4,n-1: LET f=0:
    RANDOMIZE USR X: GO TO 60
240 IF z$>="a" AND CODE z$<=a+95 THEN GO TO 300
250 IF z$>="A" AND CODE z$<=a+63 THEN GO TO 300
260 IF z$="1" THEN GO SUB 540: GO TO 580
270 IF z$="2" THEN GO SUB 620: RUN
280 IF z$="3" THEN RANDOMIZE USR 15616
285 IF z$="3" THEN RUN
290 GO TO 200
300 IF z$>="a" THEN LET p=CODE z$-96
310 IF z$<="P" THEN LET p=CODE z$-64
320 LET p$=a$(p, TO 8)
325 IF a$(p,1)=CHR$ 0 OR a$(p,1)=CHR$ 1 THEN GO TO 350
330 IF a$(p,9)="B" THEN INPUT ;: PRINT #0;" Loading ";p$;"<B>":
    RANDOMIZE USR 15619: REM : LOAD p$
335 IF a$(p,9)="C" AND a$(p,14)="USR " THEN GO TO 600
340 IF a$(p,9)="C" THEN LET n$="Load addr? ": GO SUB 400: GO TO
    360
```

Listing 2. continued.

```
350 INPUT ;: PRINT #0;" <B> and <C> Files only...": FOR z=1 TO
    100: NEXT z : GO TO 200
360 IF q$=" STOP " THEN GO TO 200
365 IF q$<>"" THEN IF VAL q$+(CODE a$(p,12)+256*CODE a$(p,13))>
    65536 THEN INPUT ;: PRINT #0;" Loading to high:Press a key.
    ..": PAUSE 0: GO TO 340
375 IF y$=CHR$ 226 THEN GO TO 60
380 IF y$="y" OR y$="Y" THEN GO SUB 550: GO TO 400
390 GO TO 370
400 POKE x+14+(5 AND y$=""),rt-256*INT (rt/256): POKE x+15+(5 A
    ND y$=""),INT (rt/256): RANDOMIZE USR (x+13+(5 AND y$=""))
410 INPUT ;: PRINT #0;" Loading ";p$;"<C>": IF q$="" THEN GO SU
    B 500: RANDOMIZE USR 15619: REM : LOAD p$CODE
420 IF q$<>"" THEN LET r=VAL q$: RANDOMIZE USR 15619: REM : LOA
    D p$CODE VAL q$
430 LET n$="Run addr? ": GO SUB 460: CLS : IF q$=" STOP "  THEN
    GO TO 60
440 IF q$="" THEN RANDOMIZE USR r: GO TO 1e4
450 IF q$<>"" THEN RANDOMIZE USR VAL q$: GO TO 1e4
460 INPUT " ";(p$);"<C>  ";(n$); LINE q$
470 IF CODE q$>=50 AND CODE q$<=54 AND LEN q$=5 THEN RETURN
480 IF q$="" OR q$=" STOP " THEN RETURN
490 GO TO 460
500 LET r=CODE (a$(p,10))+256*CODE (a$(p,11)): RETURN
510 IF INKEY$<>"" THEN GO TO 510
520 IF INKEY$="" THEN GO TO 520
530 RETURN
540 INPUT ;: PRINT #0;" Insert Disc and Press any key..": GO SU
    B 510: RETURN
550 INPUT " New Ramtop? "; LINE y$: IF y$="" THEN GO SUB 500: L
    ET rt=r: RETURN
560 IF CODE y$>=51 AND CODE y$<=54 AND LEN y$=5 THEN LET rt=VAL
     y$: RETURN
570 GO TO 550
580 RANDOMIZE USR 15619: REM : LIST
590 PRINT #0;" Press a key": PAUSE 0: GO TO 60
600 INPUT ;: PRINT #0;" Snapshot:";p$;"    Sure?(y/n)": GO SUB
    510: LET y$=INKEY$: IF y$<>"y" AND y$<>"Y" THEN GO TO 60
610 INPUT ;: PRINT #0;" Loading Snapshot File  ";p$: RANDOMIZE
    USR 15619: REM : GO TO P$CODE
620 INPUT ;: PRINT #0;" Insert Disc :  Drive A,B,C,D ?": GO SUB
    510: LET z$=INKEY$: IF z$>"d" OR z$<"A" THEN GO TO 620
630 IF z$>"D" AND z$<"a" THEN GO TO 620
640 RANDOMIZE USR 15619: REM :*z$;":"
650 RETURN
```

BDUC, 2, DOWNHAM AVENUE, RAWTENSTALL, ROSSENDALE, LANCASHIRE. BB4 8JY

B.D.U.C.

BETA DISK USERS CLUB

BETA DISK NEWSLETTER NO. 3

Welcome to this the third BDUC newsletter. We also welcome new members in Europe. BDUC now has members in France, Germany, Belgium and Denmark. Thanks to members who have sent contributions, I should be able to feature most of them in forthcoming issues. Apologies for the late return of your contribution disks and tapes.

All members are invited to contribute, I can accept text files in the following formats, Tasword 2 and 3, The Writer and The Last Word. Both 3.5" and 5.25" disk formats may be submitted. Any Beta interface related material will be reviewed for inclusion.

BDUC has no information about TR's business since the last newsletter, but Cumana Ltd. (who it seems manufactured the floppy controller PCB for Technology Research) are repairing Beta+ and Beta 128 interfaces. Service may be obtained by sending your interface with a description of the fault to Cumana at the following address: Cumana Ltd.,
Service Department,
Pines Trading Estate,
Guildford,
Surrey.
GU3 3BH.

There is a minimum charge of £15+VAT. The minimum charge is then plus parts plus return carriage. You should send your interface by recorded post or secured delivery. Cumana Ltd. may be contacted on 0483-503121.
European users should contact Cumana Ltd. for carriage and terms of payment.

You are advised that before sending your unit for repair the cost of repair may well exceed the cost of purchasing a second hand unit. The FDC chip costs about £15 or £20 and the TRDOS EPROM costs £12, Beta interfaces are usually available for £35 approx.

There is a small correction to Nick Cooper's Autoboot program from the second issue, line 225 should be changed to:

```
 225 IF z$=CHR$ 8 AND n=1 THEN LET n=INT (f1/16)+1: POKE x+4,n-
     1: RANDOMIZE USR x: LET F=1: GO TO 60
```

)

THE SECRETS OF TRACK Ø. BY HENDRICK BROOTHAERS.

What follows relates to a "dual density" interface.

First some basic things:

- The use of track zero is the same for each type of format, 4Ø or 8Ø tracks and single or double sided. This is necessary since first we have to read what kind of format is used before we can use this information for other purposes.
- Each track is divided in 16 sectors.
- Each sector contains 256 bytes.

Use of track zero:

- Only nine of the sixteen sectors are used.
- Sector eight contains all information about the disk such such as name, password (earlier versions only), number of files etc.
- Sectors zero to seven contain information about the files on the disk.
- Each file uses sixteen bytes, thus we can store information for sixteen files in each sector.
- Because we use eight sectors (Ø to 7) we can store information for a maximum of 128 files on track zero.

Sector eight contains general information.

- Only the last 31 bytes have a meaning. The first 225 bytes are all zero.

BYTE	MEANING
225	sector } First free sector and
226	track } First free track on the disk.
227	format type (see following table)
228	number of files (deleted files included)
229	} number of free sectors
23Ø	} on the disk.
231	16
232	Ø
233	Ø
234	to 242 password (9 chars long)
243	Ø
244	number of deleted files
245	to 253 name of the disk (9 chars long)
254	Ø
255	Ø

Table for byte 227:

22	=	80 track/double side	=	2560 sectors
23	=	40 track/double side	=	1280 sectors
24	=	80 track/single side	=	1280 sectors
25	=	40 track/single side	=	640 sectors

Sectors zero to seven contain file information.

- As mentioned before, each file uses 16 bytes.
- Some of these bytes have a different meaning, depending on the type of file. (Basic, Code or Data)

```
                          BYTE MEANING
   0  to  7 = file name

   8        = file type B(asic) C(ode) D(ata)

   9  to 10 = load address for a Code file
            = total length for a Basic file
            = address for a Data file

  11  to 12 = run address for a Code file
            = program length for a Basic file
            = array length for a Data file

  13        = length in number of sectors

  14   sector }
              } where the file is located on the diskette
  15   track  }
```

REMARKS:

- For a DELETED file byte 1 is set to value 1.
- The RUN line number for a BASIC program is directly behind the actual basic program on the disk, and NOT on track zero.
- The end of the directory is recognized when byte 1 of a program name has the value zero.

The next part describes using machine code calls and Beta DOS it was written as a reference for earlier versions of DOS.
Some features of these early versions are included in 5.03 but the call address may differ.

MACHINE CODE AND BETA DOS.

FUNCTION:	3.0	4.03 4.04	4.06	4.07 4.09	4.10	4.1	4.11 4.12	TRDOS VERSION
DOS ROM ON	3CAAH	15467	15467	15467	15467	15467	15467	
READ DISK	3C09H	11958	11992	11958	11991	11934	11990	
WRITE DISK	3C0CH	11974	12008	11974	12007	11950	12027	
DOS ROM OFF	3CBEH	15484	15484	15484	15484	15484	15484	

To use the read and write calls set the registers as follows:

DE =	D =track, E =sector
HL =	buffer address
BC =	B number of sectors to do, C=0

Don't forget to turn TRDOS on/off before/after using.

The following group may be used to address the FDC (Floppy Disk Controller Chip) directly.

PORT OUT	PORT IN
1F COMMAND to command reg.	1F STATUS from status reg.
3F TRACK to track reg.	3F TRACK from track reg.
5F SECTOR to sector reg.	5F SECTOR from sector reg.
7F DATA to data reg.	7F DATA from data reg.

The following controls the BETA interface port, FC OUT generates a clock to enable some circuits in the BETA interface.

```
F7 }
     }-These two during OUT generate a clock to the interface to
FF }  store info such as drive number/side/master reset to FDC.
```

-during IN they check if a DRQ (data request) or an INTRQ (interrupt request) is pending from the interface. FF is also used to switch the BETA ROM on and off.

The following BETA DOS system variables are for the earlier versions mentioned above.
BETA DOS VARIABLES.

5CC8	Format for drive A
5CC9	Format for drive B
5CCA	Format for drive C
5CCB	Format for drive D

BETA DOS VARIABLES continued.

5CCC	Track to read from
5CCD	Sector to read from
5CCE	ØØ=read / FF=write
5CD9/A	Effective load address
5CDB/C	Effective length
5CCD	File name (8 chars long)
5CE5	File type B/C/D/# (B:Basic C:Code D:Array #:Print file)
5CE6/7	Load address for code file/total length for basic
5CE8/9	Program length for basic or length for code file
5CEA	File length in sectors
5CEB/C	Sector and track start of file
5CF6	Drive number
5CF8	Drive to read from
5CF9	Drive to write to
5DØ2	Temporary storage for address for text to be printed
5D19	Drive number Drive A=Ø,B=1,C=2,D=3

BETA DISK INTERFACE HARDWARE - PART 1 LOGIC AND CONTROL.

The circuit diagram in this issue relates to BETA 128 with 5.xx hardware. It is known that many revisions to the hardware were carried out and this diagram may not be accurate, but it does provide a useful illustration of BETA hardware principles.

Component List.
1 27128 EPROM 16384 WORDS X 8 BITS NEC D27128D

3 74LS32 QUAD 2-INPUT OR

1 74LS74 FLIP-FLOP D-TYPE

1 74LS30 8-INPUT NAND

1 74LS04 BUFFER HEX INVERTING

1 74LS123 DUAL MONOSTABLE MULTIVIBRATOR

THE LAST WORD BETA EXTENSION. BY M.J.SMITH BDUC.
BETA 128 5.xx DOS.

The Last Word Beta extension as supplied is coded for version 4.xx DOS. To enable correct operation of the CAT command from TLW the overlay code must be loaded as described and the following patch included. Also the Beta lines at 4000 must be inserted EXACTLY as shown below. This is required since TLW code inserts the text file title to be loaded or saved directly into the BASIC lines for execution it also preserves the return address and executes a RST 8 with the PRINT USR 8 command in line 4020. The sequence to modify your TLW is as follows:

With your configured TLW resident return to BASIC. Load the Beta extension code from tape. LOAD "BETA2CODE" CODE this loads 488 bytes at address 50000, now in direct mode POKE 50112,61. Add or edit the existing lines as below.

```
  40 CLEAR 26500: GO SUB 100: RANDOMIZE USR 15619: REM: LOAD
     "TLW2"CODE
  45 GO TO 1000
  68 RANDOMIZE USR 15619: REM: ERASE "boot"
  69 RANDOMIZE USR 15619: REM: ERASE "TLW2"CODE
  70 RANDOMIZE USR 15619: REM: SAVE "boot" LINE 40
  80 RANDOMIZE USR 15619: REM: SAVE "TLW2"CODE 50000,15535
4000 REM BETA LINES
4010 RANDOMIZE USR 15619: REM: SAVE "A:12345678"CODE 12345,
     12345
4020 PRINT USR 8
```

The values in quotes and the code start and length in line 4010 are dummy, TLW overwrites these with the current filename to load or save.

The configured code and the boot loader can be saved or resaved by GO TO 68 (resave) or GO TO 70 (save).
NOTE: Resave deletes the original versions on the disk!

The overlay code contains pointers to the Beta DOS call routines starting at 50440 is LOAD, 50442 is ERASE, 50444 is SAVE, 50446 is CAT.

The Beta prompt text strings start at 50450.

BDUC,2,DOWNHAM AVENUE,RAWTENSTALL,ROSSENDALE,LANCASHIRE.BB4 8JY

B.D.U.C.

BETA DISK USERS CLUB

BETA DISK NEWSLETTER NO. 4

Hello again! Thanks and a quick mention for Dr. Andy Wright of BETASOFT in Birmingham for including a line about BDUC in the BETA BASIC newsletter. This resulted in enquiries from Spain, Sweden, Austria, Netherlands, Canada, India and Botswana.

C O N T R I B U T I O N S . C O N T R I B U T I O N S
Please send your contribution now! More material is required for forthcoming issues. Please help make this newsletter interesting and worthwhile. Anything related to Beta hardware or software.

Bernhard Lutz of Germany writes to say that he is in the army and was a member of tHe Beta user group set up by Per Henneberg Kristensen in Denmark. He was disapointed to receive only one newsletter and no reply to his letters after sending a large subscription fee, it now seems the club no longer exists. He sent this RAM Image splitter program which was developed for TRDOS version 4.11.

RAM IMAGE SPLITTER. BY BERNHARD LUTZ. TRDOS 4.11.

When using the "magic button" feature of the Beta disk interface the RAM image created on disk provides a useful method of saving programs quickly. However the block of code created is not really useful for any other purpose because of the way it has been saved. This program "strips" the screen from the magic file and enables the code to be saved as two blocks with or without the screen and enables use of tape and microdrive as well as disk.

The two code blocks created may then be loaded as required and may be transferred to microdrive or tape. Future enhancements to this program will be included soon. To use the RAM splitter program, type in listing 1 and save a security copy to tape. A formatted disk must be used to save the magic file on since the code expects to find the magic file on track 1 sector 0.

STOP PRESS...STOP PRESS...STOP PRESS...STOP PRESS...STOP PRESS..
A version of this program suitable for TRDOS 5.xx is now ready and will be included in the next issue.

The splitter program should be saved onto and loaded from another disk for the same reason as above using the name "boot".

How to use the RAM IMAGE splitter software.

1. With the software you wish to snapshot loaded and running, press the magic button at a suitable point as normal.
2. When the magic file has been saved. Load the splitter program "boot" from the second disk.
3. Option 7 may be used to test whether the copy will run correctly when it is loaded as a split version. If the test fails points A to E should be read and the splitter program run again.
4. Select option 8 (save & copy 16k/48k or screen$). A seven character file name should be entered, and the program will will split the previously saved RAM image as described. Files are saved in the following format. Screen$ are saved with "name"+"0". Programs are saved with "name"+"1" and "name"+"2".
5. A loader for the split image could be written as

```
10 CLEAR 27295
20 LET DOS=15363
30 RANDOMIZE USR DOS:REM:LOAD "name0"CODE     Optional line.
40 RANDOMIZE USR DOS:REM:LOAD "name1"CODE
50 CLS
60 RANDOMIZE USR DOS:REM:LOAD "name2"CODE
70 RANDOMIZE USR 16384
```

WHEN THE SPLIT RAM IMAGE FAILS TO WORK CORRECTLY.

A. Sometimes programs use Interrupt Mode 2, so it is necessary to use option 5 to alter the Interrupt Mode in the machine code part of the splitter program which is self loading from lines 9100 to 9130. This operates in a similar way to the "$" in a magic file name.
B. Some 16k programs test whether there is 48k available and self relocate so that a 16k program must sometimes saved as a 48k.
C. The disk may be full since the magic files are 192 sectors long.
D. The magic button may have been pressed at the wrong moment, if the program is testing what is on the screen, try using the magic button elswhere in the program.
E. Some programs should have any interrupts disabled using option 6, which modifies the self loading machine code.

The locations 27296 to 65535 may be poked as normal but locations 23296 to 27295 should be poked as (xxxxx-6873),YY.

)

Listing 1.

```
 10 CLEAR VAL "26999": LET dos=VAL "15363": LET im=VAL "86": LE
    T ei=VAL "251"
100 CLS : PRINT "RAM-IMAGE SPLITTER  1986 BL MENU"
110 PRINT '"1 SAVE SCREEN$ "''"2 GO TO DOS"''"3 SPLIT + COPY 48
    k"''"4 SPLIT + COPY 16k"''"5 INTERRUPT MODE: IM ";"1" AND i
    m= VAL "86";"2" AND im=VAL "94"''"6 INTERRUPT: ";"EI" AND e
    i=VAL "251";"DI" AND ei=VAL "0"''"7 TEST COPY";TAB VAL "16"
    ;"STOP GO TO BASIC"
120 LET A$=INKEY$: IF a$=" STOP " THEN RANDOMIZE USR VAL "0"
130 IF a$<"1" OR a$>"7" THEN GO TO VAL "120"
140 GO TO VAL a$*VAL "100"+VAL "100"
200 LET ix=VAL "32768": LET t=VAL "1": LET s=VAL "0": LET l=VAL
    "27": GO SUB VAL "1000"
210 GO SUB 4000: GO SUB 5000: LET ix=VAL "16384": LET n$=u$+"0"
    : LET l=VAL "6912": GO SUB VAL "1200"
220 RUN
300 RANDOMIZE USR VAL "15360"
310 RUN
400 LET mem=VAL "38240": GO TO VAL "510"
500 LET mem=VAL "5472"
510 GO SUB VAL "3000"
540 GO SUB VAL "2100": GO SUB VAL "2000": GO SUB VAL "4000": LE
    T ix=VAL "27296": LET l=mem:  LET n$=v$+"1": GO SUB VAL "12
    00"
550 LET ix=VAL "16384": LET l=VAL "4040": LET n$=v$+"2": GO SUB
    VAL "1200"
560 RUN
600 IF im=VAL "86" THEN LET im=VAL "94": GO TO VAL "100"
610 LET im=VAL "86": GO TO VAL "100"
700 IF ei=VAL "251" THEN LET ei=VAL "0": GO TO VAL "100"
710 LET ei=VAL "251": GO TO VAL "100"
800 CLS : LET l=VAL "22": PRINT "1 TEST 48k COPY "''"2 TEST 16k
    COPY "
805 PRINT ''"INTERRUPTMODE: IM ";"1" AND im=VAL "86";"2" AND im
    =VAL "94";''"INTERRUPT: ";"ENABLED" AND ei=VAL "251";"DISAB
    BLED" AND ei=VAL "0"
810 LET a$=INKEY$: IF a$<>"1" AND a$<>"2" THEN GO TO VAL "810"
820 IF a$="1" THEN LET l=VAL "150"
830 LET ix=VAL "27136": LET t=VAL "3": LET s=VAL "10": GO SUB
    VAL "1000"
840 GO SUB VAL "3000"
850 GO SUB VAL "2000"
860 GO SUB VAL "1600"
870 RANDOMIZE USR VAL "16384"
```

Listing 1 continued.
```
1000 GO SUB VAL "1500": POKE VAL "27005",INT (ix/VAL "256"): POK
     E VAL "27004",ix-VAL "256"*INT (ix/VAL "256"): POKE VAL "27
     007",t: POKE VAL "27009",s: POKE VAL "27011",l: RANDOMIZE U
     SR "27000": RETURN
1200 RANDOMIZE USR dos: REM: ERASE n$CODE
1210 RANDOMIZE USR dos: REM: SAVE n$CODE ix,l
1220 RETURN
1500 RESTORE VAL "9000": FOR a=VAL "27000" TO VAL "27044": READ
     b: POKE a,b: NEXT a: RETURN
1600 RESTORE VAL "9100": FOR a=VAL "16384" TO VAL "16422": READ
     b: POKE a,b: NEXT a: RETURN
2000 LET ix=VAL "16423": LET t=VAL "2": LET s=VAL "11": LET l=VA
     L "16": GO SUB VAL "1000": GO SUB VAL "1600": RETURN
2100 LET ix=VAL "27136": LET t=VAL "3": LET s=VAL "10": LET l=VA
     L "150": GO SUB VAL "1000": RETURN
3000 LET ix=VAL "22200": LET t=VAL "0": LET s=VAL "0": LET l=VAL
     "1": GO SUB VAL "1000"
3010 REM IF PEEK 22200=36 THEN LET im=94
3020 LET sp1=PEEK VAL "22209": LET sp2=PEEK VAL "22210"
3030 RETURN
4000 INPUT "ENTER NEW NAME: "; LINE v$
4010 IF LEN v$<1 OR LEN v$>7 THEN GO TO 4000
4020 INPUT "INSERT DESTINATION-DISK THEN PRESS ENTER: ";
4030 RETURN
5000 RESTORE VAL "9200": FOR a=VAL "40000" TO VAL "40011": READ
     b: POKE a,b: NEXT a: RANDOMIZE USR 40000: RETURN
8999 STOP : REM LOADER 27000,45
9000 DATA VAL "205",VAL "107",VAL "60",VAL "33",VAL "0",VAL "0",
     VAL "22",VAL "0",VAL "30",VAL "0"
9010 DATA VAL "62",VAL "1",VAL "6",VAL "1",VAL "14",VAL "0",VAL
     "245",VAL "197",VAL "213",VAL "205"
9020 DATA VAL "214",VAL "46",VAL "209",VAL "193",VAL "28",VAL "6
     2",VAL "16",VAL "187",VAL "204",VAL "161"
9030 DATA VAL "105",VAL "241",VAL "61",VAL "40",VAL "2",VAL "24"
     ,VAL "231",VAL "205",VAL "124",VAL "60"
9040 DATA VAL "201",VAL "30",VAL "0",VAL "20",VAL "201"
9099 REM RUNNER X,39
9100 DATA VAL "243",VAL "237",im,VAL "33",VAL "39",VAL "64",VAL
     VAL "17,VAL "0",VAL "91",VAL "1"
9110 DATA VAL "160",VAL "15",VAL "237",VAL "176",VAL "49",sp1,sp
     2,VAL "241",VAL "237",VAL "79"
9120 DATA VAL "241",VAL "237",VAL "71",VAL "241",VAL "225",VAL "
     209",VAL "193",VAL "217",VAL "8",VAL "253"
9130 DATA VAL "225",VAL "221",VAL "225",VAL "225",VAL "209",VAL
     "193",VAL "241",ei,VAL "201"
9199 REM LDIR 32768-16384
9200 DATA 33,0,128,17,0,64,1,0,27,237,176,201
```

)

MACHINE CODE AND BETA DOS. BY HENDRICK BROOTHAERS.
FOR 4.XX DOS.

Here is another interesting article from Hendrick Broothaers in Belgium. Part one is in this issue and the second part will follow in issue 5. The first part contains an explanation of useful Beta DOS routines and part two has examples of how to use them from machine code.

These routines are only for version 4.xx through 5.xx. All the routines can be accessed with a unique CALL address. The CALL address is 3BFD (15357). When this address is CALLed the number of the requested routine must be in the C register. The contents of some other registers and some DOS variables provide parameters and control over the routines.

In order to gain access to these routines the DOS must be switched on, this is done by a CALL 3C06 (15366) , followed by a PUSH HL. (the PUSH HL puts a RETURN address on the stack which will automatically switch the DOS off when we leave our code via a RETurn) The next thing to do is to set the necessary registers (if this was not done before), load the C register with the desired routine number and do a CALL 3BFD (15357) followed by a RETurn.
Here is an explanation of the DOS variables used by the routines.

5CD1/2 (23761/2)	Auto run line number. Used when a BASIC file is written to disk using routine 12.
5CD2 (23762)	Must hold the array character when loading a DATA file (character or number array) with routine 14.
5CD7 (23767)	Buffer address for a read or write of one sector.
5CD8 (23768)	Used in routine 14 if 5D10 is different from zero.
5CDD (23773) to 5CE4 (23780)	Filename. (8 characters).
5CE5 (23781)	Filetype (B) BASIC (C) CODE (D) DATA.
5CE6 (23782)	Total length for BASIC.
5CE7 (23783)	Load address for CODE or DATA.
5CE8/9 (23784/5)	Program length for BASIC or Byte length for CODE or DATA.
5CEA (23786)	File length in sectors.
5CEB/C (23787/8)	Sector and track where file starts on disk.
5D0F (23823)	Number a file has in the catalogue, (used by routine 10).
5D10 (23824)	Controls subroutine 14 (see text).

DESCRIPTION OF THE ROUTINES.
The routines are numbered Ø to 2Ø (or in HEX ØØ to 14).

ROUTINE Ø (# ØØ)

The currently selected drive is restored to track Ø and the break key is checked.

ROUTINE 1 (# Ø1)

Selects the drive who's number is in the A register.

ROUTINE 2 (# Ø2)

Positions the head on the track number that is in the A register on routine entry.

ROUTINE 3 (# Ø3)

Store A register in 5CFF (238Ø7) (= sector for read or write).

ROUTINE 4 (# Ø4)

Store HL register in 5DØØ (238Ø8) (= buffer address).
Note: (routines 3 and 4 have no specific stand-alone use, they are called from within other routines).

ROUTINE 5 (# Ø5)

READ from disk. Any sector/track and number of sectors:

```
Registers on entry: B = number of sectors to read
                    DE = track/sector to read from
                    HL = buffer address for read data
```

ROUTINE 6 (# Ø6)

WRITE to disk. Any sector/track and number of sectors.

```
Registers on entry: B = number of sectors to write
                    DE = track/sector to write to
                    HL = buffer address for write data
```

ROUTINE 7 (# Ø7)

CATalogue to stream whose number is in the A register.

ROUTINE 8 (# Ø 8)

Read file info from track zero to DOS variables 5CDD to 5CEC (23773 to 23788). For file who's number is in the A register. (16 bytes are moved).

ROUTINE 9 (# Ø9)

Write file info from DOS variables 5CDD to 5CEC (23773 to 23788) to disk. For file who's number is in the A register. (16 bytes are moved).

ROUTINE 10 (# 0A)

Search disk catalogue for file who's name and type are in the DOS variables 5CDD to 5CE5 (23773 to 23781). On completion address 5D0F has the number the file has in the catalogue or FF if the file is not found in the catalogue.
ROUTINE 11 (# 0B)

SAVE a non-basic file. File name and type must be in DOS variables 5CDD to 5CE5 (23773 to 23781), DE = length in bytes HL= start address.
ROUTINE 12 (# 0C)

SAVE a BASIC file. On entry: File name and
type must be in DOS variables 5CDD to 5CE5 (23773 to 23781).
Memory 5CD1 (23761) (LO)-5CD2 (23762) (HI) must have the autorun line number.
ROUTINES 13-15-16-17 (# 0D-0F-10-11)

These routines are one and the same. They are used from within DOS as an exit from all the other routines.
ROUTINE 14 (# 0E) (explained in detail later).

LOAD (BASIC-CODE-DATA). The operation depends on the contents of locations 5CD2 (23762),5CD7 (23767),5D10 (23824) and the registers A - HL - DE. The file name and type must be in DOS variables 5CDD to 5CE5 (23773 to 23780).
ROUTINE 18 (# 12)

ERASE the file who's name and type are in variables 5CDD to 5CE5 (23773 to 23780).
ROUTINE 19 (# 1 3)

LDIR memory to DOS variables 5CDD to 5CEC (23773 to 23788) HL is memory pointer.
ROUTINE 20 (# 14)

LDIR DOS variables 5CDD to 5CEC (23773 to 23788) to memory. HL is memory pointer.

Details on routine 14 (# 0E)

Routine 14 is the most complex routine, it is used to LOAD a file I will explain the different cases. You must ALWAYS make sure that the file name and type are in DOS variables 5CDD to 5CE5 (23773 to 23780) before CALLing routine 14, otherwise a "NO FILE" message is generated and the operation is aborted.

)

For a BASIC file:

5D10 = 00 A register = 00 then CALL routine 14
-there must be enough room below RAMTOP
-any previous BASIC is wiped out

For a DATA file (number or character array):

5D10 = 00 A register = 00 HL register = 0000
5CD2 = array name expl. 81 for number array "a"
C1 for character array "a$"
the routine DIMensions the array before LOADING the data.

For a CODE file:

5D10 has following controls over routine 14 :
5D10 = 00 = LOAD a complete file
5D10 = FF = LOAD one sector only
5D10 different from 00 and from FF = write one sector to disk.

-with 5D10 = 00

-A register = 00 LOAD the file to the address it was SAVED from
-A register = 03 LOAD the first sectors of a file to the address in HL register. D register = number of sectors to load
-A reg. not 00 and not 03 = LOAD the file to the address in HL

-with 5D10 = FF

LOAD the sector who's number is in the L register to the address specified in loc's 5CD7-5CD8.
example: if L = 5, the fifth sector of the file is LOADED.

-with 5D10 different from 00 and from FF

WRITE one sector to a file. As above the sector number must be in the L register and the memory address in loc's 5CD7-5CD8 before routine 14 is CALLed.

This concludes part one of this article, the next BDUC newsletter will have the second part with examples of how to use the routines from machine code.

B D U C, 2, DOWNHAM AVENUE, RAWTENSTALL, ROSSENDALE,
LANCASHIRE, BB4 8JY. ENGLAND. PHONE:010 44 706 218354

B.D.U.C.

BETA DISK USERS CLUB

BETA DISK NEWSLETTER NO. 5

Hello again. First, some news about a new product for the Beta. It's a little bit different than anything previously available. A German software house called Individual Software has produced a replacement EPROM for the Beta, it's called VISION and it can be used with 3.xx and 4.xx Beta interfaces.

The EPROM contains the Beta DOS software as normal but additional code has been implemented to produce a program that provides a new operating environment for Beta DOS, it's basically a desktop package and provides a means of driving Beta DOS from pull down menus and pointers. The pointer may be controlled via the keyboard, Kempston mouse or keyboard mouse (a device produced by Individual, which simulates a joystick and is plugged in a joystick interface).

Most functions can be carried out with single keypresses from menus. The technical information states the package is capable of supporting upto four drives and is able to drive printers. The price of the package as a 16K replacement EPROM is £15 plus £2 P & P. Individual also produce a range of other Beta related software packages. The newsletter will hopefully have a review in the next issue.

More information can be obtained direct from Individual Software, Volker Marohn, Am Beilstueck 30, 4600 Dortmund 50, West Germany or see the enclosed information sheet.

More about new products. Myrmidon Software has just released an updated version of 'The Last Word' word processor package, to be marketed by Trojan Products. Nick Buckingham who runs Myrmidon tells me it's a much improved version with many new and revamped features including a re-written Beta DOS handling routine. Also, the previously separate extension software has been integrated into the main package. The specification certainly looks impressive. Myrmidon originally wrote TLW for the now defunkt Saga Systems who collapsed several months ago. Myrmidon is providing support for TLW users.

Another liquidation was announced recently that of Kempston (of joystick interface fame). Kempston also produced a disk interface known as the KDOS but support for this interface was never strong, fewer software houses supported the KDOS than Beta DOS, although versions of Art Studio using KDOS were produced.

SPLITTER PROGRAM MODIFICATION FOR 5.XX TRDOS. BY M.J.SMITH BDUC.

Here are the modifications for using Bernhard Lutz' splitter program as featured in the last issue. The instructions for creating the main program should be followed but before the program is saved the following modifications to the listing should be included.

1.Load your splitter basic from the last issue add/edit lines as follows.

```
  10 LET DOS = VAL "15619"
 300 RANDOMIZE USR VAL "15616"
8999 STOP: REM LOADER 27000,20
9000 DATA 0,0,0,33,0,0,22,0,30,0,6,1,0,0,0,14,5,205,19,61,201
```

2.Delete lines 9010-9030, these are no longer required.

3.Save as described.

VERSION 4.XX DISASSEMBLY. (27000=6978H)

```
6978   CD6B3C    CALL  3C6B      ;switch DOS on  (15467)
697B   210000    LD    HL,0000   ;buffer address
697E   1600      LD    D,00      ;track
6980   1E00      LD    E,00      ;sector number
6982   3E01      LD    A,01      ;length
6984   0601      LD    B,01      ;number of sectors
6986   0E00      LD    C,00      ;
6988   F5        PUSH  AF        ;save value
6989   C5        PUSH  BC        ;save value
698A   D5        PUSH  DE        ;save value
698B   CDD62E    CALL  2ED6      ;read disk (11900)
698E   D1        POP   DE        ;return value
698F   C1        POP   BC        ;return value
6990   1C        INC   E         ;sector counter+1
6991   3E10      LD    A,10      ;load A reg,sectors/track.
6993   BB        CP    E         ;if sector=16
6994   CCA169    CALL  Z,69A1    ;jump to track counter
6997   F1        POP   AF        ;return value
6998   3D        DEC   A         ;
6999   2802      JR    Z,699D    ;all done switch DOS off
699B   18E7      JR    6984      ;do another sector
699D   CD7C3C    CALL  3C7C      ;switch DOS off (15484)
69A0   C9        RETURN          ;end
69A1   1E00      LD    E,00      ;zero sector counter
69A3   14        INC   D         ;track+1
69A4   C9        RET             ;return
```

The modification for 5.XX DOS is quite simple. In the version 4.XX disassembly above the program counts the tracks read and increments the sector counter when 16 tracks have been processed. For version 5.XX the DOS takes care of this process and the coding is now simplfied. The DOS routine is called with the total number os sectors to read in the B register.

VERSION 5.XX DISASSEMBLY. (27000=6978H)

```
    6978    00          NOP                 ;make space
    6979    00          NOP
    6980    00          NOP
    6981    210000      LD    HL,0000       ;buffer address
    6984    1600        LD    D,00          ;load D reg track number
    6986    1E00        LD    E,00          ;load E reg sector number
    6988    060100      LD    B,00          ;number of sectors
    698B    00          NOP                 ;make space
    698C    00          NOP
    698D    0E05        LD    C,05          ;routine 5,read data
    698F    CD133D      CALL  3D13          ;do it
    6992    C9          RET                 ;finish
```

Remember the magic file must be located on track 1 sector 0 for this program to operate correctly this is achieved by saving the snapshot on a clean disk.

Another enhancement would be to have the program automatically save all four combinations of interrupt mode and interrupts enabled or disabled. It would then be much simpler to find the split version that works. The program could use suffix A to D thus.

File A IM1, interrupts enabled.
File B IM1, interrupts disabled.
File C IM2, interrupts enabled.
File D IM2, interrupts disabled.

This feature could be added as another subroutine, but memory space for the basic part is limited so other features like the menu section could be overwritten. In this version there would not be any requirement to select options from a menu since the program would be automatic. Anybody willing to do the mods?

FILE UNDELETE.TRDOS 5.XX. BY M.J.SMITH.BDUC.

This short program enables files to be recovered from a disk if they have been previously ERASED from the disk catalogue, however files cannot be recovered if a MOVE has be executed since this operation repacks and totally deletes the program from the disk.

The code to call routine number 5, read data (see last issue) is loaded by line 30 from the data in line 40.
The data for the files is contained on track 0 sector 0 and the catalogue is 16 sectors long (4096 bytes). This is read by the code and the data is stored in buffer address 32000. The loop in lines 120 to 160 searches for the deleted file marker in this case 1 and replaces it with another marker that the DOS recognises as a legal name for display. The whole buffer is now written back to track 0 sector 0 using routine number 6.
The recovered file will now begin with the character ! and may be loaded normally. The program checks the disk catalogue for the erased file marker and returns with the message "No Files" if an attempt is made to undelete files from a disk with no files tagged with the erased marker 1.
The recovered file may be renamed and the MOVE function may be used as normal. The code is similar to that for the splitter program above. The information for the routine is POKED from within the program.
There may be a problem with undeleting files if the same characters appear in a filename after character 2. Also the program will not undelete the last deleted file in the directory, since TRDOS does not rewrite track zero, it just forgets that the last file was ever created and uses the file space when accessing the disk for saving.

Listing 1.

```
  10 CLEAR 29999
  20 LET F=0: LET P=0: LET N$=""
  30 CLS : PRINT AT 10,0;"FILE UNDELETE"''"READING CATALOGUE...P
     LEASE WAIT"
  40 RESTORE : FOR A=30000 TO 30020: READ D: POKE A,D: NEXT A
  50 DATA 0,0,0,33,0,0,22,0,30,0,6,1,0,0,0,14,5,205,19,61,201
  60 LET B2=32000: LET T=0: LET S=0: LET L=16
  70 LET HI=INT (B2/256): LET LO=B2-HI*256
  80 POKE 30004,LO: POKE 30005,HI
  90 POKE 30007,T
 100 POKE 30009,S
 110 POKE 30011,L
 120 RANDOMIZE USR 30000
 130 IF P=1 THEN POKE (B2+244),0: LET P=2: GO TO 210
 140 FOR X=B2 TO B2+(L*256) STEP 16
 150 FOR I=0 TO 7
 160 LET N$=N$+CHR$ (PEEK (X+I))
 170 IF (PEEK (X))=1 THEN POKE (X),33: LET F=1
 180 NEXT X
 190 IF F=0 THEN GO TO 260
 200 PRINT '"FILES UNDELETED"''"WRITING CATALOGUE...PLEASE WAIT"
```

```
200 PRINT '"FILES UNDELETED"''"WRITING CATALOGUE...PLEASE WAIT"
210 POKE 30016,6
220 RANDOMIZE USR 30000
230 IF P=1 OR P=2 THEN GO TO 270
240 LET B2=32000: LET T=0: LET S=8: LET L=1: POKE 30016,5: LET
    P=1
250 GO TO 70
260 PRINT '"NO FILES"
```

DOS ROUTINES - PART 2 EXAMPLES. BY HENDRICK BROOTHAERS.

This is part two of this article, it contains three examples of how to use the DOS routines from machine code for 4.XX DOS.

1. The following example send a catalogue to the screen.

EXAMPLE ONE

```
CALL   3C06       ;switch DOS on
PUSH   HL         ;put the DOS off return address on the stack
LD     A,02       ;select stream 2  (screen)
LD     C,07       ;set for DOS routine 7  (catalogue)
CALL   3BFD       ;call DOS to execute routine 7
RET               ;return via  DOS  off  address
```

2. The next examples is a combination of three things.

This code must be entered with the DE register pair pointing to a memory area that holds a file name and type.
First the filename is moved to the DOS variables using routine 19, now the disk catalogue is searched to see if the file exists on the diskette using routine 10, if the file is not in the catalogue then address 5D0F contains FF, if the file exists, address 5D0F contains the number of the file in the catalogue.

Finally the information for the file is moved from track zero to the DOS variables. The data moved has the file length, sector, track and is moved by routine 8

EXAMPLE TWO

```
CALL   3C06        ;switch DOS on
PUSH   HL          ;push return address
EX     DE,HL       ;make HL point to the file name
LD     C,13        ;set for DOS routine 19 (13H)
CALL   3BFD        ;CALL routine 19, move filename to DOS vars
LD     A,43        ;character "C" code 67 (43H)
LD      (5CE5),A ;set file type in DOS variable for file type
LD      C,ØA     ;set for DOS routine 1Ø (ØAH)
CALL    3BFD     ;CALL routine 1Ø, search file in disk catalogue
LD      A,(5DØF) ;get result in A register
CP      FF       ;check if file exists
RET     Z        ;return if file does not exist
LD      C,Ø8     ;set for DOS routine 8
CALL    3BFD     ;CALL routine 8,move file info to DOS vars
RET              ;return via DOS off address
```

3. This example shows how routines may be called from within other routines.
Lines 1 to 1Ø are a routine to LOAD a file from disk,when the routine is entered at line 1, the file is loaded to memory starting at AØØØ. When the routine is entered at line 2, the file is loaded in memory starting at the address held in the DE register on entry.

EXAMPLE THREE

```
 1 LD    DE,AØØØ   ;set DE to buffer address 4096Ø (AØØØ)
 2 CALL  3CØ6      ;switch DOS on
 3 PUSH  HL        ;save return address
 4 CALL  32        ;CALL routine to select drive
 5 EX    DE,HL     ;get buffer address in HL register pair
 6 XOR   A         ;clear the A register
 7 LD    (5D1Ø),A  ;set 5D1Ø to Ø (see routine 14 information)
 8 LD    A,1       ;A register=1 (load to address in HL)
 9 LD    C,14      ;set for DOS routine 14 to load the file
1Ø JR    24        ;go execute routine 14 and return via DOS off.
```

Lines 2Ø to 25 catalogue the disk to the screen.

```
2Ø CALL  3CØ6      ;switch DOS on
21 PUSH  HL        ;save return
22 LD    A,Ø2      ;select stream 2=screen
23 LD    C,Ø7      ;set for DOS routine 7=catalogue
24 CALL  3BFD      ;execute routine in C register
25 RET             ;return
```

Lines 3Ø to 42 selects the drive from the number in the memory location A123 number is in memory location A123, then calls the catalogue routine.

```
3Ø CALL  3CØ6      ;switch DOS on
31 PUSH  HL        ;save return ad dress
32 PUSH  DE        ;save value
33 PUSH  BC        ;save value
```

```
33 PUSH  BC        ;save value
34 PUSH  AF        ;save value
35 LD    A,(A123)  ;load A register with drive number
36 LD    C,Ø1      ;set for DOS routine 1=select drive
37 CALL  24        ;go select drive (via the call in line 24)
38 CALL  22        ;do a CAT
39 POP   AF        ;get value back
4Ø POP   BC        ;get value back
41 POP   DE        ;get value back
42 RET             ;return
```

This is the last part of the Machine code and Beta DOS article. Thanks again to Hendrick Broothaers in Belgium.

FLOPPY DISK DRIVE- USER LINKS. BY M.J.SMITH. BDUC.

The Beta interface is capable of supporting multiple drives which are handled automatically by the interface firmware and floppy disk control chip. If multiple drives are used some knowledge of floppy drive link options is required, especially if the user wishes to upgrade from a single drive.

Here is an explanation of the links for most types of drive, these are based around the Mitsubishi MF503A, 5.25" double sided drive, (similar to the drives supplied by TR), but are also applicable to most drives of Japanese origin.

1:Drive Select.
These links determine the physical identity of the drive, **for** the first drive, logical unit Ø or drive A link DSØ is **made.** For the second drive logical unit 1 or drive B, DS1 is made.

Drive 1	DSØ - IN	Drive 2	DSØ - OUT
	DS1 - OUT		DS1 - IN

BETA CONFIG:Drive A - DSØ IN   Drive B - DS1 IN

If MX is in place the drive will react to any drive select command. Hence would normally be out in most applications.
BETA CONFIG: MX - OUT

2:Motor Commands.
These links determine various functions for the drive as follows:

)

Motor starts by motor on command.	MM - IN	MS - OUT
Motor starts by drive select.	MM - OUT	MS - OUT
Motor starts by motor on or drive select.	MM - OUT	MS - OUT
Motor starts by IN USE latched by drive select. pin 4.	MM - IN IU - IN	MS - IN

BETA CONFIG: MM - IN **MS - OUT**

3:Status Commands.

Standard Ready.	2S - OUT	DC - OUT
Hold Ready.	2S - IN	DC - OUT
Disk Change. (Reset by drive select)	2S - IN	DC - IN
Termination resistor. Used in last physical drive.	TD - last drive.	

BETA CONFIG: 2S - OUT DC - OUT TD - IN (last drive)

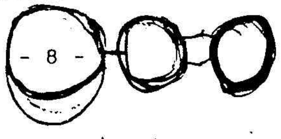
4:Indicator.

LED will not light.	IU - OUT	IS - IN
LED will light with drive select.	IU - OUT	IS - OUT
LED will light with IN USE signal.	IU - IN	IS - OUT
LED will light with the logical sum of the drive select and IN USE signals.	IU - IN IL - OUT	IS - OUT

BETA CONFIG: IU - OUT IS - OUT IL - OUT

Please note this information is intended as a guide only, you should consult the manufacturer's specification sheets to verify correct link settings.

B D U C, 2, DOWNHAM AVENUE, RAWTENSTALL, ROSSENDALE, LANCASHIRE, BB4 8JY. ENGLAND. PHONE:010 44 706 218354



Eletor —

B.D.U.C.

BETA DISK USERS CLUB

BETA DISK NEWSLETTER NO. 6

Welcome to this the last issue of the BDUC newsletter. I am sorry it has come to this,but activity and support for the Beta are almost non-existent here in the UK. It is possible that some company may take up the Beta again but I think that is most unlikely since the Plus 3 seems to be quite popular despite it's 3" drive. Most shops sell the Disciple or Swift Disc system.

I hope v3.00 and v4.00 users found the Individual Software literature of interest. This issue features an article about a comprehensive track zero utility program. It's another excellent program by Hendrick Broothaers. Unfortunately, the program is too large to publish in these pages since the program has a Basic listing and two large code blocks. The program may be obtained from BDUC by sending a disk stating format required and sufficient postage for return. The article makes for a good read so here it is.

TRACK ZERO UTILITY - BY HENDRICK BROOTHAERS, BELGIUM.
V3.0,V4.03,V4.1,V4.11 TRDOS

Z R U T I L D O C U M E N T A T I ON

Updates:
15-03-86
- For a bug in option 7 in the second menu. It did not update the number of free sectors left, when a file was recovered at the end of the disk.
- For a bug when DOS version 3.0 is installed. Option (5) read/write is from/to the wrong track, because version 3.0 uses other values for type of format in the DOS variables.
- Changed the file type of the TRACK 0 file to CODE, so that DOS versions which do not have the file handling options can run zrutil.

22-03-86
- For new option (9), in the second menu to disable the automatic setting of disk format type in case track zero is unreadable. Select manual format type setting instead.

08-04-86
- Adjust format type for DOS 4.X versions. When 40 track **drives** are used a special format code has to be set in the DOS "**format** variable.

13-04-86
- Add option (6) CAT to printer. This comes to BASIC at line 9615 which is a call to initialize the printer code (KEMPSTON "S"). Line 9620 is the actual CAT to printer. The other lines with the POKES direct the output to the printer.
19-04-86
- Update of this text file. Last three paragraphs added.

Why is there a "zrutil" program.

I am convinced that every one of you has known those bad times when after a lot of programming work, and storing your efforts on a disk, you find out at some later time that your disk for some unknown reason is not readable anymore. If it comforts you, it happens to all of us. The purpose of this article is not to go into the reasons why these things happen, but how we can minimize the damage and recover from such a disaster.

Wouldn't it be nice when track zero became scrambled that we had another disk with a backup of the one that screwed up. Unfortunately, usually we do not have that backup disk and it always seems to be that the backup does'nt have the latest files on it.

To overcome these problems I came up with the idea of keeping a copy of track zero elsewhere on the disk. First I thought about the last track on the disk, but this is not a good thing to do for different reasons.
-First of all is it the weakest track because it has the highest data density.
-Then there is the danger of overwriting the last track when the disk becomes nearly full.
-Also the last track is a different track number depending on the disk format, this would make the program more complicated and I wanted to make a relatively easy to use program.

I decided that a good place to do it is track one, it has none of the drawbacks mentioned above, and you can still do whatever you want with your disk, except use track one. If you want to use this idea, I have written a program called "zrutil" which really stands for track zero utility program. It will do the job for you. To begin with it will take time and effort to adapt your existing disks, but believe me, you will be pleased you made the effort, the day your next crash happens.

What can "zrutil" do for you.

First, I must tell you that the program is compatible with the following DOS versions: 3.0 - 4.03 - 4.04 - 4.1 - 4.10 - 4.11

(and maybe others!). It checks which version is installed and adapts itself automatically to that version. If your DOS version is not compatible, zrutil can be adapted with some POKES. See beginning of BASIC section.

This section is not intended as a full guide but serves as a condensed set of user instructions. Option 7 of the main menu brings you in the "track zero safety features". Here you can use option (4) to reserve track one for a copy of track zero. The disk must be empty for this option, otherwise the option is not executed since it only works on an empty disk.
Option (1) can be used to clean up your disk but BE CAREFUL. THIS OPERATION WIPES OUT THE CONTENTS OF A DISK. The easy way is to use option (6) which does it all in one go. It wipes out the disk, and reserves track one. Once you have done this your disk is ready to receive new files.

When you have changed something on the disk,(added or deleted files, move, etc.) you can use option (2) which makes an updated copy of track zero to track one. Option (2) has to be done WHENEVER YOU CHANGE SOMETHING ON THE DISK, in order to keep an EXACT copy on track one. This means a little effort every time, but it's worth it since track one must be maintained to enable recovery of files.

Now comes the good part;
For some reason your track zero contains rubbish. All you have to do is do option (3) which copies track one back to track zero, and your disk is good again.

Yes, but what if disk errors occur in track zero?
No problem at all, do option (1) to reformat track zero ONLY and do option (3) to copy one to zero, and there you go again with a usable disk. It should be obvious that track zero has to be formatted in the same format as the disk was formatted before.

If the disk cannot be formatted and track zero remains unreadable, it is I M P O R T A N T that you use option (9) to manualy set the disk format type. This disables at the same time an automatic check by the program for the disk format. Whenever you use one of the options to access track zero the automatic checking for format type will become active again.
Even when the worst happens and track zero is totally unusable (due to physical damage for example) you can still save your valuable files by using option (5) of the main menu. You probably have to do option (9) to disable the reading of track zero (to get the format type) and to manually set the format type of your disk.

You can now read track one and decode the file information which you can then use with option (5) to read the file into memory and then write it back to another disk via option (5) again or via option (8) and the DOS SAVE command.
For BASIC programs you will have to find the program length and set the system variables accordingly. You could also copy track one to track zero of another disk and copy all other tracks to that disk.
There are a few other goodies in "zrutil".
Back to the main menu now. Option (3) allows you to change the name of a file. This might not look very useful at first, but it allows you to recover deleted files and to fix any names that have funny characters in them and could mess up the disk.
(like a zero in the first position of the name). If the file you changed was a deleted file, the deleted file count will be updated automatically. When you use option (3) you only change the memory buffer, to update the disk you must use option (4).
When you do a CAT, any deleted files at the end are not shown in the CATALOGUE printout. Use option (8) of the second menu to look at any deleted files at the end.
Note that only files who can succesfully be recovered are shown.
To recover these files you have to use option (7). This option shows one file at the time. It will make the first deleted file visible so that it can now be recovered like any other deleted file. The files can only be recovered in the order they are shown. If you want to recover the third one only, you have to recover 1,2 and 3 and delete 1 and 2 later. This is because the free sector count must be handled correctly.
Do not forget to update the disk with option (4) of the main menu.
Option (5) allows you to read or write anywhere on the disk.
Even if your track zero and one are bad, this can be used to recover some files. It is less easy of course but it is useful.
Remember that if track zero is unreadable to use option (9) to manually set the format type.
Option (9) provides a handshake with your own basic by bringing you to basic LINE 100. You can put your basic at line 100 and do whatever you want to do. To go back to the zrutil main menu just do a GOTO 9225.
In case you want to use this, it might be helpful to know that the track zero data from option 2 is stored in memory starting at location 47500 for a length of 9 sectors. The useful sector 8 (count starts at zero) information starts at address 49773
Please note that all the "recover file" options use the catalogue info in memory as input and output. This means that you have to do option (2) of the main menu first to get the catalogue info into memory, and after you have done any changes to the catalogue in memory, you have to do option (4) of the main menu to save your changes on track zero.

)

DISK RENAME - BY M.J.SMITH,BDUC.
V5.XX TRDOS

This short program enables the title of a disk to be changed without re-formatting of the disk, and avoids unnecessary copying to other disks just to change the disk title. The program copies track Ø sector 8 to the buffer, the new name is entered and the new name is written to the buffer area and then the whole sector is written back to the disk. Routine 6 called from line 200 is the write a sector(s) routine in this case only 1 sector is written back to the disk. The machine code is very similar to the undelete code in the last issue.

```
 1Ø CLEAR 29999
 2Ø DIM B$(9)
 3Ø CLS
 4Ø PRINT AT 8,Ø;"DISK RENAME"
 5Ø INPUT "NEWNAME:"; LINE B$
 6Ø RESTORE
 7Ø FOR A=3ØØØØ TO 3ØØ2Ø
 8Ø READ D
 9Ø POKE A,D
1ØØ NEXT A
11Ø DATA Ø,Ø,Ø,33,Ø,Ø,22,Ø,3Ø,Ø,6,1,Ø,Ø,Ø,14,5,2Ø5,19,61,2Ø1
12Ø LET B=31ØØØ
13Ø LET T=Ø
14Ø LET S=8
15Ø LET L=1
16Ø LET HI=INT (B/256)
17Ø LET LO=B-HI*256
18Ø POKE 3ØØØ4,LO
19Ø POKE 3ØØØ5,HI
2ØØ POKE 3ØØØ7,T
21Ø POKE 3ØØØ9,S
22Ø POKE 3ØØ11,L
23Ø RANDOMIZE USR 3ØØØØ
24Ø LET A$=""
25Ø FOR I=B+245 TO B+252
26Ø LET A$=A$+CHR$ (PEEK I)
27Ø NEXT I
28Ø PRINT '"       DISK:";A$
29Ø PRINT "RENAMED AS:";B$
3ØØ PRINT
31Ø FOR T=1 TO 8
32Ø POKE (B+244+T),CODE B$(T)
33Ø PRINT (B+244+T);" ";B$(T);" ";CODE B$(T)
34Ø NEXT T
35Ø POKE 3ØØ16,6
36Ø RANDOMIZE USR 3ØØØØ
```

DISK VERIFY - BY M.J.SMITH. BDUC
V5.XX TRDOS

This small program may be used to verify each track of a disk and each side in turn. It may be used when a disk is perhaps suspect or damaged. This program will not recover lost data but will confirm if checksums are intact. The program could even be modified just to check track Ø. This program uses routines 21,22 and 23, previously undocumented. The 'standard' format is employed with machine code poked into locations 30000 to 30026 data buffer at 32000.

```
 10  CLEAR 29999
 20  RESTORE
 30  FOR A=30000 TO 30026
 40  READ D
 50  POKE A,D
 60  NEXT A
 70  DATA 0,0,0,33,0,0,22,0,30,0,6,1,0,0,0,14,21,205,19,61,201,1
     4,0,205,19,61,201
 80  LET B=32000
 90  LET Z=0
100  LET C=0
110  LET HI=INT (B/256)
120  LET LO=B-HI*256
130  POKE 30004,LO
140  POKE 30005,HI
150  POKE 30022,22
160  LET E=USR 30021
170  LET TT=80
180  FOR T=0 TO TT-1
190  PRINT AT 4,0;"SIDE:";Z;" T:";T+1,"E:";C
200  POKE 30007,T
210  LET E=USR 30000
220  IF E<>0 THEN LET C=C+1
230  NEXT T
240  IF Z=0 THEN LET Z=1: POKE 30022,23: GO TO 160
250  PRINT C;" ERRORS"
```

TRDOS ROUTINES - BY M.J.SMITH. BDUC
V5.XX TRDOS
This article describes four extra DOS routines present in the v5.xx version. These are not documented anywhere, they are listed below in a similar format to the twenty routines featured in issue number 4.

ROUTINE 21 (# 15)

Verify currently selected track, range 0 to 79.
Before entry set B=1 (number of tracks),A=track,D=track,E=sector
CALL 3D13, on return BC register pair has error code else 0 which signifies OK.
(See the verify program in this issue)

ROUTINE 22 (# 16)

Selects side 0 for double sided systems. Default side for single sided drives. Set C=16 then CALL 3D13

ROUTINE 23 (# 17)

Selects side 1 for double sided systems.
Set C=17 then CALL 3D13

ROUTINE 24 (# 18)

This routine checks the directory of the drive currently specified and returns the current drive status, codes returned are displayed as error message 0 to 12. Alternatively, the BC register pair contains the value of the error code.

BETA BASIC 3.0

I currently have a disk data handling utility which runs under Beta Basic 3.0. If anyone is interested please send a disk stating your disk format and enclose return postage.

F U T U R E   C O N T R I B U T I O N S

If anyone has any material for any future publications they may still send it to BDUC. I hope to complete a "Beta Disk Users Guide" by Summer 1988. This will have material from these newsletters and other material that was not included for various reasons.
It was dissapointing to find that only 7% of members wished to re-subscribe to the newsletter in 1988! I'm afraid with this level of support I had no alternative but to wind up BDUC.
Thank you for your support during 1987 and best wishes for 1988.

1793 FLOPPY DISK CONTROLLER

```
                 ---------------
                |                |
 -------7-|DALØ         TG43|-29------
 -------8-|DAL1         DDEN|-37------
 -------9-|DAL2        R CLK|-26------
 ------10-|DAL3           WG|-30------
 ------11-|DAL4           WD|-31------
 ------12-|DAL5        EARLY|-17------
 ------13-|DAL6         LATE|-18------
 ------14-|DAL7      VFOE/WF|-33------
          |          RAWREAD|-27------
          |                 |
          |                 |
 -------1-|VBB           HLD|-28------
          |                 |
 -------5-|AØ   1793        |
          |                 |
 -------6-|A1            HLT|-23------
          |                 |
 ------40-|VDD              |
          |              SSO|-25------
          |                 |
 ------21-|VCC              |
          |             STEP|-15------
 ------22-|TEST             |
          |                 |
 -------3-|CS          DIR C|-16------
          |                 |
 ------20-|VSS              |
          |            READY|-32------
          |                 |
 -------4-|RE           TRØØ|-34------
          |                 |
 -------2-|WE               |
          |               IP|-35------
          |                 |
          |             WPRT|-36------
          |                 |
 ------38-|DRQ              |
 ------39-|INTRQ            |
 ------19-|MR               |
 ------24-|CLK 1MHZ         |
          |                 |
           -----------------
```

B D U C, 2.DOWNHAM AVENUE, RAWTENSTALL, ROSSENDALE,
LANCASHIRE, BB4 8JY. ENGLAND. PHONE:010 44 706 218354