aculab

floppy tape

RELATED ARTICLES FROM



NATGUG NEWS

Digital Tape Storage System for TRS-80 Level 2.

HARDWARE REVIEW - ACULAB FLOPPY TAPE

The day of the cassette is numbered!!! A relatively low cost alternative to disk is now available in the form of the high speed digital continuous loop tape.

The Aculab Floppy Tape unit costs £192.05 inclusive of vat (p&p extra). Compared to the cost of buying the expansion interface and disks which would set you back in the region of £600.00 this is obviously the way that the less affluent i.e. those of us who can't justify or even explain to the wife or tax man (or anyone else for that matter) the origin of this recent cash flow.

The system is adressed by what is now the standard (originated by it's recent predessor the less sophisticated Exatron Stringy Floppy) of SYSTEM /12345. The Aculab's main advantages (beside the fact that it's British!) being the resident directory and operating system being more flexible and does not require the operator to remembEr the location of files and how much space is left on the tape. The obvious cost in memory as a result of this refinement is 1.25K which leaves the user with 14K of available memory (approx.) for those who are worried about putting machine code programs onto floppy tape the directory can be relocated to clear low memory if this is required, so programs like Space Invaders or even Copsys or Pencil can be saved on the F-tape.

The control codes are prefixed by "@" (e.g. @SAVE, @LOAD, @RUN, @MERGE, @KILL and @NEW. Most are self explanatory with the exeption of @SAVE which can be appended to save machine code with auto start address. There are also two other commands (@PUT and @GET) to transfer data to and from tape when using the random access file capability which is the other main advantage over the Exatron system (point to note that Exatron have approached Aculab for the licence to market Aculab software and have also expressed interest in the operating systum so maybe for once an information flow will go the other way across the pond!).

To conclude, after having used the system to the extent that a 14K program will load every time in 14 sec. (an obvious pointer to the 9000 baud rate of transfer) without any problems of bad dumps or loads. Some problems have occured, though this is probably due to operator error. e.g. on saving a program if the closing quotes are omitted

the directory will not load the program if it is called with full quotes, so beware and read the manual. This however can be useful in protecting proprietry programs by using shifted letters etc.

Software is at this moment being amended by Aculab for use with the F-tape system and Hardings have said that they will support the software also Sofdware House have said the same. Hardings expect to be in the market soon after publication.

New tapes are available in various lengths from 10 feet to 50 feet with 5 foot and 75 foot in the pipeline. Typical usage is 30 sectors per 10 foot (one sector being 255 bytes), so a 20 foot tape is equal to 16k!

Library note: there are a few routines now available in the library that are of use to floppy owners e.g. READ (determines the start, end and entry of a system tape), NEWFLOS (for relocating directory). Hopefully we will have a Floppy page in the newsletter if we get away from Space Invaders, X-wins etc. for long enough to write it!

Adrian Hancock
Bob James

FLOPPY TAPE CORNER

ACULAB HAVE KINDLY PLACED A BASIC PROGRAM (DIRTOP/BAS) IN THE GROUP LIBRARY THAT WILL BE OF USE TO ALL ACULAB OWNERS AND THOSE THAT ARE CONSIDERING PURCHASING ONE OF THEIR UNITS. THIS BASIC PROGRAM IN EFFECT REMOVES THE PROTECTIVE BLOCK ON THE DIRECTORY AFTER IT HAS BEEN LOCATED IN THE AREA OF MEMORY OF YOUR CHOICE. THIS IS HIGHLY CONVENIENT IN THE CASE OF LONG PROGRAMS THAT UTILIZE UP TO 14K OF TEXT AND THE REST OF MEMORY AS VARIABLE OR STRING STORAGE. THIS HAS BEEN A BUG-BEAR UP UNTIL NOW, ESPECIALLY WITH GAME PROGRAMS THAT INVOLVE TRADING OR WAGING WAR IN INTERGALACTIC SPACE. SOME SERIOUS PROGRAMS MAY ALSO BE AFFECTED.

THE PROGRAM IS SUPPLIED ON CONDITION THAT IT REMAINS THE SOLE PROPERTY OF ACULAB AND THAT IT WILL NOT BE SOLD TO OTHER PARTIES. THE RESULT OF SUCH ACTION WOULD REFLECT BADLY ON THE GROUP AND WOULD ALSO JEOPARDISE FUTURE CO-OPERATION.

AS SOME OF YOU MAY HAVE NOTICED, HARDINGS AND KANSAS ARE SUPORTING THE SYSTEM. HARDING'S LATEST CAALOGUE LISTS THE PROGRAMS NOW AVAILABLE. HOWEVER THEY DID NOT HAVE A COPY OF 'DIRTOP/BAS' IN THEIR POSESSION AT THE TIME.

A GOO GUIDE IS THAT IF THE BASIC TEXT USES UP TO 14265 BYTES IT WILL FIT. IF IT UTILIZES THE REMAINDER OF THE MEMORY (IN A 16K SYSTEM) THE DIRECTORY WILL BE OVERWRITTEN IF 'ZAP HAD BEEN RUN. A JUMP TO 3000 HEX WILL REBOOT THE SYSTEM (SOFT) OR THE SYSTEM COMMAJD CAN BE USED TO REGAIN CONTROL OF THE OPERATING SYSTEM.

FOR THOSE WHO HAVE ALREADY PURCHASED THE ACULAB YOU WILL SOON BE ABLE TO PURCHASE A COMPATIBLE ZEN EDITOR/ASSEMBLER MODIFIED BY LAURIE SHIELDS. EXTENDED BASIC IS ALREADY AVAILABE ON FLOPPY AS IS MON3.

ACULAB ARE WORKING ON A FLOPPY BASIC WITH MOST OF THE DISK BASIC FEATURES, PLUS A FEW EXTRA LIKE A REVERSE ACTING &H FUNCTIONENABLING DEC TO HEX TO DEC CONVERSIONS. INSTRING, RSET ,LSET AND CMD FUNCTIONS ARE OTHER GOODIES THAT WILL BE INCLUDED. AS YU CAN SEE, THE FLOPPY TAPE HAS COME OF AGE, AND NOW THAT A CHEAP 32K EXPANSION CAN BE BOUGHT FROM:-

MICROTEK INC,
9514 CHESAPEAKE DRIVE,
SAN DIEGO, CA 92123.
TEL 714-278-0633

FOR $159.50 (CODE MT 32B), A CHEAP ALTERNATIVE TO THE DISK IS NOW A FACT.

INCLUED BELOW IS THIS MONTH'S FREE GIFT!! ALAN POUND OF ACULAB (THE FATHER OF THE FLOPPY) HAS WRITTEN THIS PROGRAM - DIRTOP/BAS. IT IS A BASIC PROGRAM THAT WILL CREATE A MACHINE LANGUAGE PROGRAM. WHEN RUN THIS PROGRAM WILL ENABLE YOU TO MOVE THE DIRECTORY TO THE TOP OF MEMORY OF ANY SYSTEM FROM 16K TO 48K, WITH OR WITHOUT MEMORY PROTECT. HAVING ANSWERED THE MEMORY SIZE AND PROTECT OPTIONS THE PROGRAM WILL ASK WHETHER YOU WANT TO RUN THE PROGRAM IMMMEDIATELY OR NOT. IF YOU ANSWER 'YES' THE DIRECTORY WILL BE RELOCATED AS SPECIFIED AND DIRTOP/BAS WILL BE LOST. IF YOU ANSWER 'NO' THE PROGRAM WILL INSTRUCT YOU TO INSERT A TAPE WITH AT LEASTONE SECTOR FREE, IT WILL THEN TELL YOU THE NAME OF THE PROGRAM, E.G. 'DIRTOP16/ZAP' AND WILL THEN @SAVE IT AS A MACHINE LANGUAG ROUTINE WHICH CAN BE USED AS NEEDED. THE EXTENSION '/ZAP' INDICATES THAT THE DIRECTORY WILL BE OVER-WRITTEN BY AN INCOMING BAIC PROGRAM LONGER THEN 14K (ON A 16K MACHINE) AS EXPLAINED ABOVE. DIRTOP/BAS HAS BEEN GIVEN TO THE GROUP AND CAN BE COPIEDAS MANY TIMES AS NEEDED AS LONG AS NO-ONE CLAIMS THE COPYRIGHT.

```
100 '    *** DIRTOP/BAS ***
110 '    BASIC PROGRAM TO CREATE M/LANG UTILITIES TO MOVE
120 '    FLOPPY TAPE DIRECTORY WITH OR WITHOUT RESERVING
130 '    MEMORY, AND TO RESTORE NORMAL BEGINNING OF BASIC.
140 '
150 CLS:INPUT"HOW MUCH RAM IN MACHINE ( 16 / 32 / 48 )";M
160 INPUT"RESERVE MEMORY FOR DIRECTORY ( Y / N )";A$
170 FOR Z=0 TO 38 :READ Y :POKE 30720 + Z ,Y :NEXT Z
180 D=124
190 IF M=32 THEN D=188
200 IF M=48 THEN D=252
210 POKE 30720 + 11 ,D
230 IF LEFT$(A$,1) = "Y" THEN D = D-2 ELSE D = D+3
240 POKE 30720 + 21 ,D
250 INPUT"EXECUTE  IMMEDIATELY (  Y /  N )   * THIS  TROGRAM
WILL BE LOST *";B$
260 IF LEFT$ (B$,1)  = "Y" THEN POKE  16526 ,0  :POKE  16527
,120 :Z = USR (0)
270 PRINT"PREPARE TO  SAV PROGRAM - INSERT A TAPE INTO DRIVE
'0' WITH AT LEAST ONE FREE SECTOR."
275  E$="" :IF  LEFT$(A$,1)  >  <  "Y"  THEN  E$="/ZAP" ELSE
E="/SYS"
280 N$ = "DIRTOP" + RIGHT$(STR$(M),2) + E$
290 PRINT"PROFRAM WILL BE CALLED ";  N$ ;"PRESS SPACE BAR TO
SAVE PROGRAM."300 C$ = INKEY$ :IF C$ = "" THEN 300
310 @LIST :@SAVE N$ ,1 ,30720 ,30720
320 '
330 DATA 33,232,66,175,119,35,34,164,
64,33,0,12,34,173,65,37
340 DATA 46,225,119,33,255,122,34,214,
64,34,177,64,46,205,34
350 DATA 160,64,205,77,27,195,25,26
```

Bob James Adrian Hancock

SAVING GSF ON FLOPPY TAPE.

That excellent set of machine code sub-routines from Racet Computes called GSF is supplied on cassette and the manual which comes with it describes how to save the routines to disk. However, life is not so easy for those of us who still persevere with floppy tapes. The routines cannot be saved using SYSTEM/CMD under XBAS as they are not written as a contiguous block, consequently, rather more subtle techniques have to be used.

What follows is largely the work of Laurie Shields, who devised this technique for me some months ago, and I am very grateful to him for it. As with David Washford's contribution in the June Newsletter (p.4), the writing up of this technique has been prompted by conscience pricking as a result of Laurie's letter published in the April Newsletter.

Machine Code Programs can be saved to F-tape by the simple expedient of entering the following command from the keyboard:-

@SAVE"FILENAME/EXT",(sectors),(start address),(entry address)

(Where the entry address is optional, allowing the program to be @RUN if used.)

Each sextor is 256 bytes long and the F-tape ALWAYS records a complete 256 byte sector, NEVER a part of one. If the end of memory is reached before a complete 256 byte sector has been recorded then the "infernal machine" goes back to the beginning of memory and starts recording the ROM until all 256 bytes have been used up. (With interesting consequences on reloading!) In order to prevent this, it is vital always to begin recording 256 x no. of sectors to be recorded, addresses BEFORE the end of memory.

Unfortunately, in the case of GSF, the program is not an integral number of 256 bytes long and thus simply using the appropriate MEMORY SIZE address as the start address in the @SAVE procedure results in chaos on trying to reload. To prevent this, the start address to use must be the next integral of 256 before GSF starts (16K=29696, 32K=45568, 48K=61952) i.e. 254 for 16K, and 216 for the others, addresses lower than the memory size addresses quoted in the GSF manual.

When reloading it is important to remember that, on power up, the appropriate one of the values quoted above should be used in response to the MEMORY SIZE? prompt rather than the GSF manual value. If the latter is used, the routines will partially load over the stack which is only 50 bytes below the MEMORY SIZE address on power up, again causing chaos and not recommended!

Provided all the above is followed, GSF can be saved to, and loaded from, F-tape successfully in the normal way. However, following the rest of the cassette loading procedure given in the manual, i.e. entering CLEAR, does not appear to cause GSF to autostart when called from a BASIC program. The reason seems to be that the appropriate entry address has not been located in the USR memory addresses (16526 and 16527 - as described in the LEVEL II BASIC MANUAL). These entry addresses for GSF are given in the manual on the last page in the instructions for saving to disk under "cccc".

There are various ways to overcome the problem, the easiest being to include a POKE statement line in the BASIC program BEFORE the first call to GSF i.e.

POKE 16526,LSB:POKE16527,MSB

The MSB and LSB values are calculated from the "cc[illegible]" values given in

the GSF manual for the appropriate size of memory.

Alternatively, a small autostart machine code routine can be incorporated at the beginning of GSF before @SAVE-ing it using some of the spare 254/216 bytes created by having to save an integral number of 256 bytes. The assembler program for the routine is as follows:-

```
00100   ORG             ;Convenient starting point before
00110                   ;GSF according to size of memory.
00120   LD    HL,ccccH  ;Load HL registers with entry address
00130                   ;obtained from GSF manual
00140   LD    (16526),HL ;Load USR addresses with contents
00150                   ;of HL registers.
00160   JP    1A19H     ;Jump back to BASIC
00170   END
```

This can be converted into machine code using any of the assembler programs on the market. I do not own a copy of ZEN or MACRO-80 so I have no idea whether either, or both, is/are the "best thing since sliced bread". However, as part of the job lot which I acquired when I bought my TRS-80 second hand, was a copy of EDTASM which did the job perfectly well and joined the result to GSF.

Using:-

@SAVE "GSF",14,(start address),(entry address)

it is now possible to save GSF to F-tape and then auto start it using @RUN "GSF" before loading the BASIC program with which GSF is to be used. (N.B. In this case, "entry address"="start address"= ORG address in the above program.)

For those who do not have an assembler program at all, the machine codes corresponding to the above program are (in HEX):-

21 LSB MSB (of relevant entry address) 22 8E 40 C3 19 1A

This can be put into the appropriate memory locations prior to @SAVE-ing either by using a monitor program, or even the "M" facility of the humble T-BUG program (which I also acquired in my job lot). Or, again before @SAVE-ing, the data can be converted into decimal giving:-

33 LSB MSB (of entry address) 34 142 64 195 25 26

and the following BASIC program run:-

```
10 DATA (decimal values as above)
20 FOR I=1 TO 9
30 READ D
40 POKE ADDRESS+I-1,D
50 NEXT
```

(Where ADDRESS is the appropriate starting point for the size of memory being used.)

Laurie originally asked me to write this up so that even a beginner like me could understand it.

R. C. BAILEY

Thanks again Brian and all for another great weekend.
==========

It seems ages since I last wrote anything for the magazine. I suppose I could blame Leon for not being his old provocative self but that would be unjust (and anyway we don't know what he might be up to in the next issue).

Believe it or believe it not I penned that back in January and then never got round to finishing it off.

Less than one month later appears Leon's undated broadsheet plugging his usual outdated CP/M 8080 style of programming. Hasn't he learnt yet that whenever one writes general purpose routines, and this applies even more when macros are involved as their details tend to get forgotten, that any registers used ought to saved first and restored on completion. Now we all know that CP/M was written a long time ago before the Z80 was born and hasn't noticeably improved since, but there are other systems about that do use the IX and IY registers, and merrily jumping in waving macros in both hands like Maggie in the Falklands isn't going to do any good if those registers are being used by the system.

Just for the record Leon's macro ILD routine uses 9 bytes and clobbers the IX register's original contents, the same operation can be achieved in conventional style with just 6 bytes and all registers used restored on completion. In other words whilst recognising that as far as some people are concerned 'never mind the quality ... feel the width' seems to be the fundemantal parameter for their basis of judgement I prefer to be with those who appreciate the merits of simplicity in their philosophy. Now back to whatever it was that I was writing about before being so rudely interrupted (a proper assembler indeed . . . he ought to be strung up by his relocatables with his modules plastered with Bdos errors).

In the last magazine but one, someone (if only I could find my copy) was asking how to go back and forth between @Zen and Aculab Basic. Well it couldn't be easier; whilst in @Zen key G (for GO), 114 or 72H (for Address), just <enter> for breakpoint and there you are with Basic's READY > message. To get back from Basic key SYSTEM <cr> and then /23456 <cr>. It couldn't be easier and it is in the manual.

I was shamed the other week when Maxwell Stampe said he had to dig right back to a 1981 edition of the mag for the price of Zen so if Leon will pardon the advertising:

@Zen	for the Aculab	£22.50
Zen	for Level 2 cassette Model 1,Model 3 or Genie	£22.50
Zen	for the Colour Genie (with colour and sound)	£22.50
Zen83	for disk Model 1, Model 3 or Genie (all Dos's)	£44.50

All versions include a comprehensive monitor for manipulating memory, capability to assemble direct to memory and a debugging facility including breakpoints.
The latest cassette and disk versions include system tape reading editing and copying routines, global text replacement, cross reference listing, etc.

```
1 'BY ADRIAN C. HANCOCK,11/7/81    -20-
2 'DIRECTORY PEEK ROUTINE DEVELOPED BY ANTHONY ROBSON AND BOB JAMES
3 CLEAR1000
4 ON ERROR GOTO 41
5 CLS
6 DIMZ$(32,3)
7 PRINT"     *** ACULAB FLOPPY TAPE AUTO FILE SELECTOR UTILITY ***"
8 T=128
9 O=1
10 P=17408+((O-1)*16) : IF PEEK(P+15)=0 THEN 14
11 IF PEEK(P)<65 OR PEEK(P)>122 THEN PRINT : GOTO 16
12 C=C+1 : FOR N=PTOP+10 : Z$(C,0)=Z$(C,0)+CHR$(PEEK(N)) : NEXTN
13 Z$(C,1)=STR$(C) : Z$(C,2)=LEFT$(Z$(C,0),8) : Z$(C,0)=RIGHT$(Z$(C,0),3) : Z$(
,3)=Z$(C,2)+"/"+Z$(C,0) : PRINTàT,Z$(C,1);") ";Z$(C,3) : T=T+32
14 O=O+1
15 GOTO 10
16 PRINT"SELECT FILENAME BY NUMBER (OR 'L' TO àLIST)"; : IF C>9 THEN GOSUB 30 E
SE GOSUB 28
17 IF D$="L" : àLIST : GOTO 1
18 IF ASC(D$)<48 THEN 16
19 IF ASC(D$)>57 THEN 16
20 FOR C=0 TO 32 : IFVAL(Z$(C,1))=VAL(D$) THEN 22 ELSE NEXT C
21 PRINT"FILE NOT IN DIRECTORY" : GOTO 16
22 GOSUB 31 :GOSUB 36 : PRINT : PRINT"RUN, LOAD OR ABORT ";Z$(C,3);"?":GOSUB2
23 IF D$="A" THEN END
24 IF D$="R" : àRUN Z$(C,3)
25 IF D$<>"L" THEN 27
26 àLOAD Z$(C,3)
27 GOTO 22
28 D$=INKEY$ : IFD$="" THEN 28
29 RETURN
30 PRINT" HIT ENTER WHEN READY"; : INPUT D$ : RETURN
31 IF RIGHT$(Z$(C,2),1)=" " THEN 32 ELSE RETURN
32 Z$(C,2)=LEFT$(Z$(C,2),LEN(Z$(C,2))-1)
33 Z$(C,3)=Z$(C,2)+"/"+Z$(C,0)
34 GOTO 31
35 END
36 IF LEFT$(Z$(C,0),1)=" " THEN 37 ELSE RETURN
37 Z$(C,0)=LEFT$(Z$(C,0),LEN(Z$(C,0))-1)
38 Z$(C,3)=Z$(C,2)+"/"+Z$(C,0)
39 GOTO 36
40 END
41 IFERR/2+1=22 THEN A=PEEK(17396) ELSE A=0
42 IF A=16 THEN PRINT"SECTOR NOT FOUND" : RESUME 16
43 IF A=4 THEN PRINT"CHECKSUM ERROR" : RESUME 16
44 IF A=8 THEN PRINT"PARITY ERROR" : RESUME 16
45 IF A=160 THEN PRINT"INSUFFICIENT MEMORY" : RESUME 16
46 IF A=192 THEN RESUME 21
47 IF A=144 THEN PRINT"LOAD FILE FORMAT ERROR" : RESUME 16
48 IF A=0 THEN PRINT"ERROR ";ERR/2+1;"IN LINE ";ERL : END
```

Thats all for now, have fun with Autofile!

Adrian C. Hancock

ACULAB RECOVERY OF PENCIL OR SCRIPSIT FILES AFTER A CRASH.

I am now running @SCRIPSIT as elsewhere advertised and I have had my first ever system crash. Actually my system just locked up with a long article in memory. I found that recovery was really quite easy. If the reset button is pressed the TRS-80 will return to the usual ready prompt of BASIC. I loaded a monitor (Monitor 3 in this case) in a version which loaded over the @SCRIPSIT. I knew that by loading here I would not corrupt any of my precious text. I then scanned through the memory starting from the top of @SCRIPSIT with the A command which gives an ASCII dump of memory. When I found the text in memory just above the program end. I looked forHwhere the text seemed undamaged. This meant that I lost only two words at the beginning. I noted this address down on a bit of paper. I then continued through memory looking for the end of my text. When I found the end of my text I changed the next few bytes in memory to 00 in hex with the E command of Monitor 3. I also noted this address. Finally I used the monitor to jump to 3000H which is the entry point to the Aculab. I now placed a wafer in the drive and @LISTed then @SAVEd the memory area containing the text. The end byte set to 00 matches the Aculab end of file marker. Finally I ran @SCRIPSIT again and reloaded the text file that I had just saved completely successfully. The moral of this is that if you use @PENCIL or @SCRIPSIT don't panic, don't switch off but do have a wafer with a lower memory monitor available. I found that my text for @SCRIPSIT began at about 7140H although it might begin earlier than that. @PENCIL text will↓8robably begin at about 5B70H. I therefore suggest keeping a Monitor which loads well below either of these addresses.

Good luck on your recovery,

John Newgas.

P.S. I have just heard the Disc Electric Pencil has a warm start re-entry point at 5361H which equals SYSTEM /21345 from level 2 basic. This re-entry point returns you to the Electric Pencil with the Text Buffer intact. (Thank-you to Jeremy San). Does anyone know the equivalent warm start entries to @Scripsit and @Pencil?

FLOPPY TAPE CORNER

It would seem that the recent problems caused by the unreliability of the tape medium have at last come to an end. At last the full story has come out. It was apparently due to the move by Exatron from one side of the States to the other, coupled with a slightly naughty tape supplier. The tape was originally BASF video tape cut down to size and then put in cassettes. Then one of the suppliers received reject tape with dropouts and instead of cutting out the bad, incorporated this duff medium in the material sold to Exatron. Due to another supplier selling this duff tape as BASF video cassettes, BASF stopped supplying this type of tape to all buyers. Exatron, desperate for tape supplies, found Sonerex who started to supply their needs. However Exatron had not done their homework and the result was the loose coated directory smearing gunk we all came to love!!!!!!!!!!!!! Happily today the picture is clearer. The tape supplier now produces a type of tape that is as reliable as the original.

As regards the move from West to East this resulted in the mould for the cassette and components being broken. Exatron then proceeded to redesign the whole cassette. Again unfortunately the tolerances were not quite as exact as they should be and they didn't fit the drive!!!!!!! (maybe it was because they should have left Microlink's name on it). The tapes now available from Aculab have been certified, tested, and fitted in a drive and so far the ones we are using have not caused any problems.

It would be advisable however when using 'old' tapes to check the condition of the head before a write as you could still transfer gunk onto a perfectly good tape i.e. like any tape recorder you need to keep it clean.

Bob James

Adrian Hancock

FLOPPY TAPE CORNER PART 2

This month I have pleasure in presenting the long awaited review of X-Bas, the new Aculab Extended Basic for the F-Tape

X-Bas is a very economically written piece of software occupying only 2.5k of RAM on top of the normal Floppy Tape directory, making a total loss of 3.25k. This is a minor sacrifce compared with the gains in performance over the basic unit. X-Bas is akin to a small version of Disc Basic (which takes up 10k). The normally complicated F-Tape file handling is sorted out by X-Bas with a full set of disc-like random access file handling commands and the addition of sequential file handling capability. A number of useful extra commands normally only found in Disc Basic have also been included, making conversion of Disc programs to Floppy Tape very much easier and cutting down on those annoying L3 error messages. The final touch to this package is the fact that the first 1k is an overlay area which can contain either of two utility programs supplied with X-Bas or any other machine code utilities which you care to write or relocate to fit in this area, a simple CMD call gives instant access to these utilities.

The main features are described in more detail below:

Random Access Datafiles.
As the basic F-Tape stands the resident operating system only gives two commands for Random Access files, these being àGET and àPUT which simply get a record from tape and dump it into an I/O buffer or put a record onto tape from the I/O buffer respectively. The task of getting information into and out of the buffer and making sure that the buffer is full before àPUTting it on tape is all left to the user. Getting data out of the buffer into the Basic working area is also very convoluted involving a barely comprehensible set of VARPTR statements. All numerical variables have to be converted to strings before transferring to tape and this can be awkward where floating point variables are invoved.

X-Bas provides a set of commands (familiar to Disc users) which simplifies all these processes beyond all recignition. They consist of three statements:

FIELD : This allows the buffer to be split up into several areas of pre-defined length up to a total of 255 bytes.
E.G. FIELD 4 AS A$, 10 AS B$, 6 AS C$ etc. would allocate 4 bytes to A$, 10 Bytes to B$ and 6 bytes to C$ and so on until 255 bytes had been specified.

LSET : This puts a string into a specified field and pads out to the right with spaces if the string is shorter than the allocated number

of bytes for that field.

RSET : This is identical to LSET except that it pads to the left.

There are also three complimentary pairs of functions:

MKI$: This makes an integer variable into a 2 byte string ready to be transferred to the I/O buffer.

MKS$: Makes a single precision variable into a 4 byte string.

MKD$: Makes a double precision variable into an 8 byte string.

CVI : This is the opposite of MKI$ and converts a 2 byte string into an integer variable after it has been called back from the I/O buffer.

CVS : Converts a 4 byte string into a single precision variable.

CVD : Converts an 8 byte string into a double precision variable.

Transfer to and from the I/O buffer no longer involves a series of POKEs and VARPTRs. All that is now needed is for a variable to be assigned to a particular field of the buffer and then any value subsequently given to the variable is transferred to the buffer, and vice-versa.

Sequential Files.
These are almost the same as both the cassette and Disc commands for sequential files, making conversion of programs written for either system easily convertible.
There are six commands associated with this facility:

àSET "FILENAME/EXT:D",rec : This is a real time saver as it partially overcomes the sequential nature of the tape medium. Normally if when recording or reading a datafile the computer has to perform a function on the data in between subsequent write or read commands it will not return control to the Floppy Tape in time to catch the next sector of data as it goes past, the tape therefore needs a complete pass for each record slowing the reading time to at least that of a casstte. àSET re-formats the tape so that the sectors are numbered alternately I.E. 1 - 2 - 3 etc. where - represents an un-numbered sector which is then filled in on the next pass of the tape so that a 5 sector file would be numbered 1 4 2 5 3. The read/write commands

stop the tape after each sector has been read or written and the gap between consecutive sectors allows time for tape to stop and start in time to catch the next sector cutting the maximum number of passes needed to read a file to two.The variable 'rec' is the number of sectors to be àSET-1.

àOPEN "FILENAME/EXT:D",MODE : This opens a named file which has previously been created with the àSET command. MODE refers to whether the file is being opened for output (MODE=0) or for input (MODE=1). Two files may be open at any one time, one for input and one for output.

àCLOSE, MODE : All files must be closed after they have been used.

PRINT£,variables : As with cassette files this is the command which actually puts your list of data onto tape.

INPUT£,variables : This inputs the data list from tape.

LINE INPUT£,var$: This is like INPUT£ but instead of being followed by a list of variables in the same order as those recorded LINE INPUT£ is only followed by a single string variable name. This command will read everything on the tape up to an 0DH which is not followed by a line feed. This means that files can be read even when the format in which they have been recorded is not known, making for virtually crash-proof file reading.

Added Basic commands are as follows:

LINE INPUT"prompt string";var$: This can be used instead of the normal INPUT statement in a Basic program with or without a prompt string. It will accept any string input including punctuation (an important anti-crash factor) and also does not automatically print that annoying question mark unless you include it in the prompt, which makes programs look much neater

INSTR (n ,var1$, var2$) : The INSTR function searches for the first occurrence of string var2$ in string var1$ starting from the nth occurrence. This is a useful routine which has only been available in Disc Basic or as a subroutine in the back of the Level II manual until now.

MID$ (var1$,n,k) = var2$: This resembles MID$ in Level II

with the exception that it can now be used on the left side of an equation. The statement replaces k characters of string var1$ with var2$ starting at the nth character of var1$.

Hex Variables : Variables can be supplied in decimal or, if preceded by '&' in hex. Conversions can be carried out from dec to hex and back again very simply and it is surprising how often it comes in useful.

USR var1, (var2) : This is a modified user call allowing an address (var2) to be called as well as the name of the routine.This allows several different USR routines to be held in memory and called at will with the appropriate address.

CMD : This is possibly the most useful part of the whole package as it has an almost infinite number of functions depending on what is resident in the 1k overlay area when the command is given.

Overlays:
There are two overlay programs supplied with the package, these are RENUM/CMD and SYSTEM/CMD.

RENUM/CMD is a very versatile renumber and byte-saver routine allowing unwanted spaces and/or REMs to be removed as well as all the usual renumbering facilities. Removal of spaces in DATA statements is optional

SYSTEM/CMD is similar to Disc Basic's LMOFFSET and makes the saving of machine code programs much simpler than usual. The utility reads the program from tape into a buffer located at 5300H and then displays the run-time addresses for the program. Also displayed is the format of the command to be used to save the program including the number of sectors. This is in hex but the appropriate '&' signs are shown to perform the hex-dec conversion which occurs as the program is being saved. The clever part is that, regardless of run-time address, the program is always saved at 5300H with an entry point of 5300H. A module is tacked on the end of the program which moves it from the buffer area before running it. This means that only one address has to be memorised for most programs. SYSTEM/CMD will not work for all programs but these are in the minority. Where the utility really scores is the fact that in most cases a program which would normally overwrite the directory can now be loaded and run without having to relocate the directory first. Unfortunately this will still not allow the running of adventures from F-Tape on a 16k system, but it is nevertheless a very useful utility.

Although only two overlays are provided it is a simple matter

to create your own. A lot of utility programs can be relocated from their normal position at the top of memory down to the ovelay area at 4900H where they can be easily called by a CMD. Almost any machine code program of 1k or less which performs its own return to Basic can be placed in the overlay area. I have a relocated Lower Case Driver from an early newsletter and most of you will have seen our Screen Print (JKL) overlay in a previous Floppy Tape Corner.

As you can see X-Bas adds a lot to the basic Floppy Tape operating system and it seems to work perfectly in all respects. All in all it represents £20.00 or so well spent.

Adrian Hancock.

CHECKING THE SPEED OF AN ACULAB FLOPPY TAPE.

I have found that there can be some incompatability between different Floppy Tapes which have their drives running at different speeds.

It is possible to adjust the speed of the drive very easily by opening up the unit.

This program will give a speed index for your Aculab. To use the program to measure your Aculab-unit's speed use a wafer (with or without programs) that was originally formatted on your machine. More accurate readings are obtained with longer wafers. I normally use a 50' wafer.

```
10    DEFINT J,K,L
20    @LIST
30    J=INP(240)
40    IF J=127 THEN GOTO 100
50    K=K+1
60    GOTO 30
100   INPUT"HOW MANY SECTORS WERE LISTED IN TOTAL";L
110   PRINT"THE SPEED INDEX IS ";K/L
120   END
```

My Aculab shows a speed index of 21.85 on average for most tapes. I have some wafers which are slightly sticky or stiff, normally I can hear this when when they are running. The program shows up any changes in the running speed as they wear in. The Aculab will normally read wafers without trouble with a ±15% speed margin.

For another user to use my wafers and synchronise his machine to mine he/she should run the program with my wafer and adjust his machine speed to give my standard index.

This program only gives a relative speed measure. The measurements made on your own wafers will nearly always show the same index unless you have changed the drive speed. The index measures the change between the speed used for formatting (@NEW) and the speed when the program is run.

John Newgas.

SAVING THE VIDEO DISPLAY TO FLOPPY TAPE

This program enables the contents of the video screen to be saved on Aculab 'Floppy Tape' in one pass of the tape. As written it saves the contents of the memory map in 4 sectors of 256 bytes to drive number 0, under the filename "SCRNSAVE", the 3 character extension being used to take a page number. The program also enables the tape to be read back to the screen, and with minor adaptations could be used for many different purposes. To illustrate it's use, I include the 'patch' amendments to include it into the BASIC 'Text Editor/Word Processor' supplied by Messrs A.J.Harding (Molimerx).

Program listing 1 - assembly & machine language (@ZEN format)

```
                        ORG  7FD0H
                        LOAD 7FD0H
7FD0 C3E47F   START:    JP   RGSTRS ;LEAPFROG DATA STORE
7FD3 5343524E FLNM:     DB   "SCRNSAVE" ;PRESET FILENAME
7FD7 53415645
7FDB 000000   EXT:      DS   3      ;TEXT TO BE POKED FROM
                                    ;BASIC
7FDE 00       DRV:      DB   00H    ;DRIVE 0 SELECTED
7FDF 003C     LDADDR:   DW   3C00H  ;FIRST SCREEN ADDRESS
7FE1 0000     ENTADDR:  DW   0000H  ;NO ENTRY ADDRESS
              NRC:      EQU  04H    ;4 SECTORS PER SCREEN
              NRN:      EQU  00H    ;RECORD NUMBER 0
7FE3 00       FLG:      DS   1      ;FLAG 0=PUT 1=GET
7FE4 11D37F   RGSTRS:   LD   DE,7FD3H ;START OF FILENAME
7FE7 0604               LD   B,NRC  ;NUMBER OF SECTORS
7FE9 0E00               LD   C,NRN  ;OPERATIVE RECORD NO.
7FEB 2AE37F   CHOOSE:   LD   HL,(FLG) ;LOAD FLAG
7FEE 3E00               LD   A,0000H ; ZERO ACCUMULATOR
7FF0 B5                 OR   L      ;CHECK FLAG
7FF1 C2FA7F             JP   NZ,@GET ;IF 1 GOTO @GET
7FF4 CDD737   @PUT:     CALL 37D7H  ;CALL PUT ROUTINE
7FF7 C3FD7F             JP   BACK   ;GOTO RETURN
7FFA CDDD37   @GET:     CALL 37DDH  ;GET ROUTINE
7FFD C9       BACK:     RET         ;RETURN TO BASIC PROGRAM
                        END
```

The listing above is assembled for 16k, but can be relocated to any other desired position by re-assembly. The following steps are necessary to use it from a BASIC program:-

1 - set memory size to 32719
2 - POKE page number in ASCII into locations 32730-32733
3 - POKE ORG address into locations 16526-16527 (LSB first)
4 - routine to SAVE :

```
            a - POKE 0 flag into location 32739
            b - call routine by X=USR(0)
   5 - routine to GET :
            a - check correct page number
            b - POKE 1 flag into location 32739
            c - call routine by X=USR(0)
```

n.b. Note that with this simple routine no error checking is carried out, because firstly an error is not likely to be disastrous with only one screenful of data, secondly an error should be a rare occurence, and thirdly the author is a bit lazy that way!

Program listing 2 - Patching the routine into Messrs Harding's BASIC 'Text Editor/Word Processor'.

Remember to enter memory size 32719.
Only the affected lines are listed below (the full program is copyright and must be obtained from Messrs Harding).

```
110 POKE 16526,208: POKE 16527,127
125 CLS:GOSUB2000:A$="X":A=VARPTR(A$)
1000 CLS:GOSUB2000:POKE32739,1:X=USR(0):
     S=PEEK(15360):N=G:M=16384:GOSUB1310
1240 GOSUB1500:POKE32739,0:X=USR(0):GOTO230
1800 FORI=32720TO32765:READX:POKEI,X:NEXT:RETURN
1810 DATA 195,228,127,83,67,82,78,83,65,86,
     69,48,48,48,48,0,60,0,0,0,
     710 17,211,127,6,4,14,0,42,227,127,62,
     0,181,194,250,127,205,215,55,
     195,253,127,205,221,55,201
1900 CLS:INPUT"PREPARE TO @LIST FLOPPY DATA
     TAPE";X$:@LIST:FOR I=1TO 1000:NEXTI:RETURN
1920 <DELETE WHOLE LINE>
1930 <DELETE WHOLE LINE>
2000 PRINT:PRINT"<ENTER> PAGE NUMBER ( 0 - 999)":
     INPUTP$:IFVAL(P$)>999 OR VAL(P$)<0 THEN2000 ELSE
     P$=RIGHT$(STRING$(2,48)+P$,3)
2010 FORI=1TO3:POKE32730+I,ASC(MID$(P$,I,1)):
     NEXTI:CLS:RETURN
```

Description line by line :-

```
110 this tells TRS-80 where to find routine
125 this new GOSUB 2000 initializes page number to start
1000 sets page to GET, sends GET flag, and calls routine
1240 sends PUT flag, and calls routine
1800 reads machine language routine and shoves it into high
     memory
1810 this is program listing 1, converted into decimal
1900 prepares Floppy Tape at start of program
1920-1930 not required - keyboard debounce in A.F.T. ROM
2000 page number input sub-routine
2010 POKEs page number in ASCII into data store
```

* Hopefully Floppy Tape owners will be able to find their way through the above maze, and will be able to find their own uses for this routine.
* Incidentally may I take this opportunity to mention @ZEN, I think John Hawthorne and Laurie Shields have produced a program which is equal if not superior to the rivals, and for Floppy Tape owners who aspire to assembly language, I think it is a 'must'.

Clive Isbell,
Trewint House,
Chelmsford Road,
Felsted,
Essex.

FLOPPY TAPE CORNER

A recent interchange of information between the Medical Subgroup and the Southall Group with us at Stony Stratford has highlighted a concurrent inherent incompatibility that is to say; the different User Groups are using filename extensions that seem to have no bearing on program type.

Nick Robinson has suggested that we use a moderately standard extension code so that information exchange can be effected without any subsequent misinterpretation. Some of the codes are already standard as a result of recent workshops and thus will not need explaining, the differences have arrisen due to the arrival of X-BAS and @ZEN and a number of command file utilities. It is therefore advisable that such a standard nomenclature be initiated.

The following are suggested formats that might be ddopted:

Basic program	/BAS or /
Basic program >42 secs	/ZAP
X-Basic program	/XBS
M/C object code	/SYS
M/C source code	/SOR
M/C X-BAS utility	/CMD
M/C X-BAS offset module	/LMO
M/C relocated directory	/REL
Pencil file	/TXT
Sequential data file	/SD
Random data file	/RD

N.B. M/C object code is normal M/C programs as loaded by the standard system tape command whilst source code is that produced by EDTASM or @ZEN before assembly.

The above list is the result of consultation with the other Floppy user groups and hopefully it will be used as such. If new extensions arise please notify us and we will put them in the newsletter and simillarly any alternative extensions are welcome but we are aiming for some degree of uniformity comparable to that of the disc users.

Microtek

It is becoming increasingly apparrent that the Microtek is suffering from a lack of input buferring on the address lines. This is making it very susceptible to distance from the CPU. If when using it in conjunction with the Aculab the unit is placed on the first tee-piece of the extension bus and the Aculab taken away from the second the system

should work. I have been using my 16K Microtek for at least 3 months now with no problems although initially problems had arisen due to the unit being incorrectly connected to a Video Genie. The resultant damage was to chips U23,U28 (LS14's) and to chip U27 (LS10's) and also to the output buffers U1 and U2 (LS125's). These chips were replaced at a total cost of $2.56 and since then no problems have arisen except that the Tandy microtest does not like the RAM being added in the opposite order to the expansion interface and comes up with memory faults on non-existant memory! At present the Aculab is running on an additional 18" extension in addition to the bus without any problems. One other minor modification was that the RAM from the Tandy was exchanged with that in the Microtek.

So bearing this information in mind the faults that have occurred may be due to any of these factors and thus can be checked out for little cost. One other area of contention is that the U.K. Tandy PSU gives a higher voltage output than the U.S. standard and this may be a contributory factor, however as my system is running on a U.K. PSU without trouble it does not seem likely.

Program hint.

Try running this program:

```
10 INPUT "A";A$
20 IF A$="A" THEN 50 ELSE 100
30 PRINT "LINE 30": END
50 PRINT "LINE 50": END
100 PRINT "LINE 100": END
```

Now remove the word THEN from line 20 but leaving a space and run again. Observe the result.

Now remove the space from line 20 and run again, observe the result.

You will find that in order for an IF THEN ELSE loop to work with an implied THEN a space must follow the closed qoutes of the literal in question. This is a requirement of the Basic interpreter. So when you come to crush programs with X-BAS or another squash program make sure that you put your THENs in, you know it makes sense!

Patch for X-BAS CMD modules that won't fit into the 1K area.

POKE &4900, &C3 : POKE &4901, LSB : POKE &4902, MSB

LSB, MSB are the HEX address of the M/C routine to be patched. The CMD call will now call the routine as per X-BAS but remember you still have to memory protect the utility.

Bob James
Adrian Hancock

SAVING THE ACULAB DIRECTORY IN CASE OF DISASTER

ACULAB owners may find that if the the 3 sector directory at the front of each wafer is lost or will not load then normally the wafer contents will be unusable.

Alan Pound of Aculab Ltd. has pointed out that the Directory itself can be saved as an area of memory. This allows recovery of the wafer even with a corrupted front 3 sectors which will not @LIST or which is incorrect.

Immediately after a new wafer is formatted by @NEW proceed as follows:-

```
@LIST:@SAVE"DIR",3,17408
```

This sets a file named DIR as the first file on the wafer after the normal directory. It will therefore always start in the same place (relative sector No. 4).

Each Time another program is @SAVED on a wafer repeat the line

```
@LIST;@SAVE"DIR",3,17408
```

This will save the updated directory in DIR.

Should this wafer at any time not list then load another different wafer which also has a similar "DIR" file at the beginning an @LIST it. Now replace the faulty wafer in the drive but do NOT type @LIST, instead

```
@LOAD"DIR"
```

This will load the directory as saved in the DIR file into the normal directory location area. (4300h to 4800H). At this point the original normal wafer directory should be renewed.

Type:-

```
10 REM  DUMMY LINE   ENTER
```

Then

```
@SAVE"DUMMY":@KILL"DUMMY"
```

This will rewrite the front three wafer sectors which contain the normal directory as used in @LIST.

It is important that all the DIR files are in the same place on a wafer and have the same filenames. This is why I have standardised on having them first.

John Newgas.

Anthony has managed to modify JKL to run on the Aculab under XBAS. This enables the contents of the screen to be outputted to a line printer (graphics inluded with suitable printer). The program originally came from 80-US and was modified at a Sunday morning workshop on Brian's system. The program will run on a standard Aculab system altough it will have to be relocated to 32000D.

```
00100          ORG    0BF10H
00110 START    LD     HL,(401EH)
00120          LD     (DISPLY+1),HL
00130          LD     HL,CHECK
00140          LD     (401EH),HL
00150          JP     06CCH
00160 CHECK    PUSH   AF
00170          PUSH   HL
00180          PUSH   DE
00190          PUSH   BC
00200          LD     A,(3802H)
00210          CP     28
00220          JR     Z,JKL
00230          LD     A,(3810H)
00240          CP     28
00250          JR     Z,TWO34
00260 OUT      POP    BC
00270          POP    DE
00280          POP    HL
00290          POP    AF
00300 DISPLY   JP     $-$
00310 JKL      LD     HL,3C00H
00320          LD     B,64
00330          LD     C,16
00340 LOOP1    LD     A,(3840H)
00350          CP     4
00360          JR     Z,OUT
00370          LD     A,(HL)
00380          CP     128
00390          JP     M,CONT1
00400          LD     A,46
00410 CONT1    CALL   LPRINT
00420          INC    HL
00430          DJNZ   LOOP1
00440          LD     A,13
00450          CALL   LPRINT
00460          LD     B,64
00470          DEC    C
00480          LD     A,C
00490          CP     0
00500          JR     NZ,LOOP1
00510          JR     OUT
00520 TWO34    LD     HL,3C00H
00530          LD     D,16
00540 LOOP6    LD     E,0
00550 LOOP2    LD     B,64
00560          PUSH   HL
00570 LOOP5    LD     A,(3840H)
00580          CP     4
00590          JR     NZ,CONTA
00600          POP    HL
00610          JR     OUT
```

```
00620 CONTA   LD     A,127
00630         LD     C,(HL)
00640         CP     C
00650         JP     M,LOOP3
00660         LD     C,128
00670         SET    6,C
00680 LOOP3   LD     A,E
00690 LOOP4   CP     2
00700         JR     NZ,LOOP4A
00710         BIT    6,C
00720         JR     Z,LOOP4A
00730         LD     A,(HL)
00740         PUSH   AF
00750         JR     CONT4
00760 LOOP4A  OR     A
00770         RLA
00780         RLA
00790         PUSH   HL
00800         PUSH   DE
00810         LD     HL,INDEX
00820         LD     D,0
00830         LD     E,A
00840         ADD    HL,DE
00850         JP     (HL)
00860 INDEX   BIT    0,C
00870         JR     CONTB
00880         BIT    1,C
00890         JR     CONTB
00900         BIT    2,C
00910         JR     CONTB
00920         BIT    3,C
00930         JR     CONTB
00940         BIT    4,C
00950         JR     CONTB
00960         BIT    5,C
00970 CONTB   POP    DE
00980         POP    HL
00990         PUSH   AF
01000         JR     Z,CONT3
01010         LD     A,35
01020         JR     CONT4
01030 CONT3   PUSH   DE
01040 CONT4   PUSH   DE
01050         CALL   LPRINT
01060         POP    DE
01070         POP    AF
01080         CP     4
01090         JR     Z,CONT5
01100         CP     12
01110         JR     Z,CONT5
01120         CP     20
01130         JR     Z,CONT5
01140         LD     A,E
01150         INC    A
01160         JR     LOOP4
01170 CONT5   INC    HL
01180         DJNZ   LOOP5
01190         LD     A,13
01200         PUSH   DE
01210         CALL   LPRINT
01220         POP    DE
01230         POP    HL
01240         INC    E
01250         INC    E
01260         LD     A,6
01270         CP     E
01280         JR     NZ,LOOP2
01290         LD     BC,64
01300         ADD    HL,BC
01310         DEC    D
01320         LD     A,D
01330         CP     0
01340         JP     NZ,LOOP6
01350         JP     OUT
01360 LPRINT  EQU    3BH
01370         END    START
```

Bob James

Adrian Hancock

FLOPPY TAPE CORNER

```
 0 CLS:PRINT"THIS PROGRAMME KILLS FILE NAMES.
     PRESS ANY KEY TO CONTINUE":GOSUB23
 1 @LIST:CLEAR200:DIMF$(20):L=0
 2 O=1:DL=((PEEK(16814)*256)+PEEK(16813))
 3 P=DL+((O-1)*16):IFP)32767THENP=-1*(65536-P)
 4 IFPEEK(P+15)=0THEN16
 5 IFPEEK(P)(65ORPEEK(P))122THEN17
 6 Z$=""
 7 FORN=PTOP+10:Z$=Z$+CHR$(PEEK(N)):NEXTN
 8 FORN=1TO8
 9 IFMID$(Z$,N,1)=" "THEN12
 10 NEXTN
 11 N=9
 12 X$=LEFT$(Z$,N-1):Y$=RIGHT$(Z$,3):Z$=X$+"/"+Y$
 13 A$="":PRINTZ$;TAB(15)" DELETE YES OR NO";:INPUTA$
 14 IFA$()"Y"THEN16
 15 L=L+1:F$(L)=Z$
 16 O=O+1:GOTO3
 17 IFL=0THEN22
 18 FORN=1TOL
 19 PRINT"KILLING ";F$(N)
 20 @KILLF$(N)
 21 NEXTN
 22 END
 23 A$=INKEY$:IFA$=""THEN23ELSERETURN
```

```
0 CLS:PRINT"THIS PROG. CHANGES FLOPPY FILE NAMES
  AND READS M/C START AND ENTRY ADDRESSES":GOSUB 38
1 CLEAR200:@LIST
2 FORI=1TO2000:NEXT:CLS:O=1
3 PRINT@0,"FILENAME/EXT","START","ENTRY"
4 PRINT"******** ***","******","******":PRINT
5 DL=((PEEK(16814)*256)+PEEK(16813))
6 P=DL+((O-1)*16):IFP)32767THENP=-1*(65536-P)
7 IFPEEK(P+15)=0THEN11
8 IFPEEK(P)<65ORPEEK(P)>122THEN13
9 FORN=PTOP+7:PRINTCHR$(PEEK(N));:NEXT:PRINT"/";:FORN=P+8TOP+10:PRINTCHR$(PEEK(N
));:NEXT
10 PRINT"",PEEK(P+12)*256+PEEK(P+11),PEEK(P+14)*256+PEEK(P+13)
11 O=O+1
12 GOTO6
13 O$=" ":PRINT@960,"ENTER OLD NAME";:INPUTO$:IFO$=" "GOTO13
14 Z$=O$:GOSUB29:O$=Z$
15 O=1
16 P=DL+((O-1)*16):IFP)32767THENP=-1*(65536-P)
17 IFPEEK(P+15)=0THEN21
18 IFPEEK(P)<65ORPEEK(P)>122THENPRINTO$;" NOT FOUND":GOTO13
19 FORN=PTOP+10:IFPEEK(N)<>ASC(MID$(O$,N-P+1,1))THEN21
20 NEXTN:GOTO22
21 O=O+1:GOTO16
22 PRINT"ENTER NEW NAME";:INPUTW$
23 Z$=W$:GOSUB29:W$=Z$
24 FORN=PTOP+10
25 POKEN,ASC(MID$(Z$,N-P+1,1))
26 NEXTN
27 INPUT"CHANGE ANOTHER NAME (Y/N)";ZX$:IFZX$="Y"GOTO2
28 INPUT"READY TO WRITE - PRESS ENTER ";Z$:@SAVE"REM",1,32000:PRINT"DONE !":@LIS
T:END
29 M=LEN(Z$)
30 IFM=11THEN32
31 FORN=M+1TO11:Z$=Z$+" ":NEXTN
32 FORN=1TO11:IFMID$(Z$,N,1)="/"THEN34
33 NEXTN:RETURN
34 X$=LEFT$(Z$,N-1):Y$=MID$(Z$,N+1,3)
35 IFN=9THEN36ELSEFORI=NTO8:X$=X$+" ":NEXTI
36 Z$=X$+Y$
37 RETURN
38 A$=INKEY$:IFA$=""THEN38ELSERETURN
```

FLOPPY TAPE CORNER

Here is a little program which will resave difficult Floppy files:-

```
10 @LIST:INPUT"FILENAME/EXT";A$:INPUT"NO.OF
   SECTORS",N:FORX=0TO(N-1):@GETA$,X:?X;:@PUTA$:NEXT
```

Aculab recommend OUT 240,0 for stopping the tape rather than OUT 240,2.

Nick Robinson

USING THE ACULAB FLOPPY TAPE WITH LEVEL III BASIC

There is nothing more galling than finding that the software that you have that is cassette-based is incompatible with a newly acquired Floppy Tape system. Such incompatibilities included until recently Electric Pencil (now very competently modified by Laurie Shields), and Scott Adams Adventures. Here are some reasons why this software may be incompatible:-

1) the whole of available RAM is used, including the input/output buffer
2) the directory vector area (16813,16814) is over-written so that the system does not know where to look for the directory
3) the program uses a machine-language routine that overwrites the directory.

I got in touch with Alan Pound of Aculab, who gave me the following patch for Level III. He has given his permission for this patch to be freely used and exchanged between group members. Apparently A.J.Harding also uses this patch.

Load Level III under SYSTEM:-

```
*?LEV3
*?/114

POKE 17689,201
POKE 17690,0
POKE 17691,124
POKE 17695,201
POKE 17696,151
POKE 17697,25
POKE 17698,201

@SAVE"LEV3",21,17152,17152
```

Mark Brooker

FLOPPY TAPE SECTION

ALAN AND I HAVE AGREED, AS A STANDARD, A ZERO BYTE AS THE EOF MARKER FOR ALL SEQUENTIAL FLOPPY TAPE FILES.

AT PRESENT THIS INCLUDES ALAN'S EXTENDED FLOPPY BASIC (AMAZINGLY POWERFUL FOR THE AMOUNT OF RAM HE USES), THE FLOPPY @ZEN EDITOR/ASSEMBLER/FILEZAP AND THE FLOPPY ELECTRIC PENCIL.

TEXT FILES ARE TOTALLY COMPATABLE WITH EACH PROGRAM, PENCIL MAKES ASSEMBLER FILE EDITING A DELIGHT.

@ZEN IS NOW AVAILABLE (FROM YOURS TRULY) ON THE FLOPPY TAPE. IT HAS ALL THE FACILITIES OF THE ORIGINAL ZEN WITH A FEW EXTRA GOODIES I'VE PUT IN AS PART OF CUSTOMISING IT TO THE FLOPPY TAPE. MOST IMPORTANT IS PERHAPS THE POOR MAN'S SUPERZAP THAT WILL VERIFY FILE SECTORS AND ALSO GET AND PUT DAMAGED ONES.

AS SOME PEOPLE SAW AT CAMBRIDGE, I'VE ALSO GOT THE ELECTRIC PENCIL PATCHED TO THE FLOPPY TAPE AND IT PERFORMS VERY WELL. ON A 16K LEVEL 2 MACHINE IT MAKES A RATHER GOOD WORD PROCESSOR PACKGE AT AN ATTRACTRIVE PRICE.

I'VE NOT YET GOT ANY ARRANGEMENT WITH MICHAEL SHRAYER SO I CAN ONLY OFFER AT PRESENT THE FOLLOWING ARRANGEMENT TO ANYBODY WANTING A COPY:-

1. SEND ME A COPY OF YOUR ELECTRIC PENCIL.
   IT MUST BE A GOOD READABLE CASSETTE AS I WILL THEN

2. READ IT AND RELOCATE TO SUIT THE FLOPPY DIRECTORY

3. ADD THE FLOPPY FILE HANDLING COMMANDS

4. RECORD IT ONTO A 10FT FLOPPY TAPE

5. THE ONLY INSTRUCTIONS GIVEN WILL BE FOR THE NEW COMMANDS. IN OTHER WORDS I ASSUME THAT AN OFFICIAL COPY OF ELECTRIC PENCIL HAS BEEN BOUGHT INCLUDING THE INSTRUCTION MANUAL

6. THERE ARE TWO MODS TO PENCIL THAT ARE ALSO AVAILABLE:

TANDY LOWER CASE DRIVER THAT USES BREAK AS THE CONTROL KEY AND CLEAR FOR THE BREAK.

HIGH MEMORY PROTECTION THAT USES 4049H AS HIMEM TO PROTECT MACHINE CODE ROUTINES.

CURRENTLY @ZEN AND THE FLOPPY PENCIL PATCH ARE PRICED AT

£19.50 EACH.

NEWDOS 80 IS FANTASTIC

I'M JUST ABOUT TO START WORK ON INTERLINKING PENCIL AND SCRIPSIT WITH A TELEX PUNCH AND INTERFACE BOARD.

I DON'T THINK IT WILL RIVAL YOUR NETWORK BUT IT SHOULD BE FUN.

LAURIE SHIELDS

Here we go again folks! This month we bring you (with a little help from our friends e.g. Anthony Robson who wrote most of the programs), two unique and useful utilities arising from a good idea from Bob, who kept mucking his directory up and is too lazy to re-save the program,and from both of us getting fed up with writing down the start and entry addresses of machine language programs.

The first utility, DIRCHECK/BAS, when @RUN after @LISTing the tape will dump to screen the filenames with start and entry addresses for machine code programs (BASIC returning zeros and data files returning one zero and 256).

The second utility is, wait for it, RENAME/BAS!!! This allows you to rename programs already @SAVED, without calling them into memory first. First @LIST then @RUN RENAME/BAS. This will give a display similar to DIRCHECK/BAS but will then ask you for the filename to be changed. You are then asked for the new filename. Once this is entered the program will @SAVE and @KILL a dummy program called "REM". This is a ploy to make the F-tape save a new directory including the new filename (the main program will be lost). The only limitation on this utility is that there must be at least one free sector for the dummy program.

Thats all for now. Tune in next month for the next exciting instalment of the Floppy Tape saga.

Adrian Hancock
Bob James

1) If you have difficulty finding a file, type:

```
10 <ENTER>
@SAVE "REM":@LIST
```

This will write a new directory, and may help.

2) If you can find the directory, but can't load a program (length N sectors) try:

```
FOR X=1 TO N:@GET "FILENAME/EXT",X:?X;:NEXT
```

If a number is missing, a sector is bad.

3) The following may be useful:

```
OUT 240,(drive 0, or 241 drive 1 etc.)

1=RUN
2=STOP
3=PARK
4=WRITE (no RUN)
5=WRITE and RUN
```

Use is as follows:

```
@LIST "INSTRUCTIONS":OUT 240,2
```

If the INSTRUCTIONS program then calls the main program, it saves endless cycling of the tape.

Nick Robinson

(* I hope the above makes sense - Nick is a doctor, and has been trained, like most doctors, to write rather illegibly! Could all members please ensure that their handwritten contributions do not resemble the random meanderings of an ink-sodden spider. LFH *)

PURGE - A FLOPPY-TAPE PURGE PROGRAM

```
1 @LIST:CLEAR 200:DIM F$(20):L=0
2 O=1
3 P=17408+((O-1)*16):IFPEEK(P+15)=0 THEN 15
4 IFPEEK(P)<65 OR PEEK(P)>122 THEN 16
5 Z$=""
6 FOR N=P TO P+10:Z$=Z$+CHR$(PEEK(N)):NEXT N
7 FOR N=1 TO 8
8 IFMID$(Z$,N,1)=" " THEN 11
9 NEXT N
10 N=9
11 X$=LEFT$(Z$,N-1):Y$=RIGHT$(Z$,3):Z$=X$+"/"+Y$
12 A$="":PRINT Z$;TAB(15)" DELETE YES OR NO";:INPUTA$
13 IF A$<>"YES" THEN 15
14 L=L+1:F$(L)=Z$
15 O=O+1:GOTO 3
16 IF L=0 THEN21
17 FOR N=1 TO L
18 PRINT "KILLING ";F$(N)
19 @KILL F$(N)
20 NEXT N
21 END
```

RENAME - A PROGRAM TO RENAME FLOPPY-TAPE FILES

```
1 CLEAR200:@LIST
2 O=1
3 P=17408+((O-1)*16):IFPEEK(P+15)=0 THEN 7
4 IF PEEK(P)<65 OR PEEK(P)>122 THEN 9
5 FOR N=P TO P+10:PRINT CHR$(PEEK(N));:NEXT N
6 PRINTPEEK(P+14)*256+PEEK(P+13);PEEK(P+12)*256+PEEK(P+11)
7 O=O+1
8 GOTO 3
9 O$=" ":INPUT"ENTER OLD NAME";O$:IFO$=" ":END
10 Z$=O$:GOSUB25:O$=Z$
11 O=1
12 P=17408+((O-1)*16):IFPEEK(P+15)=0 THEN 16
13 IFPEEK(P)<65 OR PEEK(P)>122 THEN PRINT O$;" NOT
FOUND":GOTO9
14 FOR N=P TO P+10:IFPEEK(N)<>ASC(MID$(O$,N-P+1,1)) THEN 16
15 NEXT N:GOTO 17
16 O=O+1:GOTO 12
17 INPUT"ENTER NEW NAME";W$
18 Z$=W$:GOSUB25 :W$=Z$
19 FOR N=P TO P+10
20 POKEN,ASC(MID$(Z$,N-P+1,1))
21 NEXT N
22 ZX$=""
23 INPUT"REPEAT";ZX$:IFZX$="YES"GOTO2
24 INPUT"READY TO
WRITE";ZX$:IFZX$="NO"GOTO34ELSE@SAVE"REM",1,32000:PRINT"DONE
```



SUPPORTED BY YOU AND PRODUCED AT WWW.TRS-80.ORG.UK

WHERE 8 BITS AND 16K STILL RULE OK