EDMONTON 99'er COMPUTER USERS' SOCIETY

99'er Online

DECEMBER 1984

P.O.Box 11983
Edmonton, Alberta
Canada T5J 3L1

TO: ⌜   ⌝

⌞   ⌟

99'er ON LINE is the news letter of the Edmonton 99'er
Computer User's Society published ten times a year. All material
contained in this news letter may be published in other news
letters provided that source and author are identified unless
otherwise stated. We welcome correspondence from all TI User
Groups and will extend source credit courtesy in **99'er ON LINE.**

CORRESPONDENCE: News letter editor: BOB PASS, 59 LABELLE CR,
ST. ALBERT, ALBERTA, CANADA   T8N-2G6. All other correspondence:
EDMONTON 99'er COMPUTER USER'S SOCIETY, P/O BOX 11983, EDMONTON,
ALBERTA, CANADA   T5J-3L1

OFFICERS: PRESIDENT--BILL CANNON,   VICE PRES--PAUL HELWIG,
TREASURER--EVAN SMITH, SECRETARY--SUSAN LIVINGSTON

DISCLAIMER: All information published in this news letter
is, for the most part, the fruits of the labors of amateurs;
therefore, we cannot guarantee that the information presented is
always correct.

REGULAR MEETINGS: Regular meetings of the Edmonton User's
Group are held on the second Tuesday of each month on the 3'th
floor of the General Services building of the University of
Alberta from 7:00 till 10:00 PM and are open to all members in
good standing. Non-members may attend their first meeting free
of charge.   The Executive Committee meets monthly. Members may
attend these meetings as observers or to address a particular
issue. Arrange with one of the officers listed above if you wish
to attend.

ADVERTISING: Commercial advertising space is available in
this news letter at the following rates: FULL PAGE--$20.00.
HALF PAGE--$15.00. 1/4 PAGE--$10.00.   Discuss your commercial
needs with Paul Helwig at the next meeting or write to the P/O
Box above. Members may advertise their personal computer related
items for free but are asked to limit their ads to about 20
words. Mail your ads to the EDITOR'S ADDRESS or hand it to him
at the general meeting; newsletter deadline 15'th of the month.
MEMBERSHIP FEES: FAMILY---12 MONTHS, $20.00, 6 MONTHS,
$15.00. STUDENTS---12MONTHS,$15.00, 6 MONTHS, $10.00.

nova·COMPUTERWARE

52 AIRPORT ROAD
EDMONTON, ALBERTA T5G 0W7

(403) 452-0372

TI-99/4A

SOFTWARE & HARDWARE

THIRD PARTY PRODUCTS

Extended Basic $139.95
Disk Fixer $49.95
Super Sketch $89.95

Sanyo Electronic Typewriter
4E-355 (TI-Compatible) $899.99

Teac half-height drive $295.95

10% Discount
on Educational Software

OPEN 1:00 - 5:00
TUESDAY - SATURDAY

OTHELLO : A REVIEW

by: Francis K. Gaston

A highly popular board game is now an intriguing computer disc game on the TI 99/4A. This is a game requiring a contest of strategy, skill, and concentration. All basic moves and counter-moves are the same as the board game.

Othello features eight levels of play, two players against each other, or one player against the computer.

In the computer mode, there are five aids to the player during the game:
1: change difficulty up or down,
2: legal move display,
3: computer advice (where it would move if it were to play your turn),
4: switch sides (if you thought #3 was cheating, what's this?),
5: resume game play.

Also available is a special game entry option. In this mode you may create your own game by placing the starting pieces anywhere on the board. This provides a greater challenge than your normal Othello game.

Overall, the game is ideal for someone intrigued by computer "intelligence" programming. It's only downfall is the length of time between moves by the computer in the higher levels. Levels 1 to 4 do not display this problem and is much faster in it's decisions. Anyone who has a vested interest in logic games such as chess would find this game just as challenging. It does not require the elaborate and complex moves of chess so the game is ideal for all ages. From the original price of $69.95, this module was obtained at a price of $19.95 from Eaton's (in Saskatoon -Ed) -- truly a saving unworthy of it's performance!

MISTAKES

by: Bob Pass

Once again a gremlin got into last month's news letter! The article written by Tom Hall under the title "Little Gem" contained an error in the program listing in line 5. A complete & error free listing is as follows:

```
1 DISPLAY AT(5,3)ERASE ALL:"
INPUT FILE": :TAB(10);"DSK1.
"; :TAB(4);"OUTPUT FILE" : :
TAB(10);"DSK1."

2 ACCEPT AT(7,10)SIZE(-15):I
$ :: ACCEPT AT(11,10)SIZE(-1
5):O$ :: OPEN #1:I$ :: OPEN
#2:O$,VARIABLE 163

3 LINPUT #1:L$ :: IF LEN (L$
$)=78 THEN LINPUT #1:M$ :: L
$=L$M$

4 S=POS(L$," ",1):: ON ERROR
7 :: N=VAL(SEG$(L$,1,3))::
ON ERROR 6 :: A=INT(N/255)::
A$=CHR$(N-A6):: PRINT : : L$

5 PRINT #2:CHR$(A);A$;CHR$(1
31);SEG$(L$,S+1,255-S);CHR$(
0):: GOTO 3

6 PRINT #2:CHR$(255);CHR$(25
5):: CLOSE #2 :: END

7 ON ERROR 6 :: RETURN 3
```

FROM THE NEWS ROOM

NEXT MEETING: The next meeting will be held on Tuesday, February 12'th at 7:00 PM in room 349 of the General Services Building, U of A campus. The building is located on 116 Street north of the Jubilee auditorium.

IMPORTANT NOTICE: A general buisiness meeting will be held during the next meeting to address two issues affecting the operation of our user's society. First, we must decide if our group should direct funds towards the purchase of assets such as a telephone answering device, computer equipment, etc. Second, should we begin to collect a once only initiation fee from all existing and future members? These fees would be used to provide capital funds required to purchase fixed assets. If you have any thoughts to offer on these topics then be sure to attend the next meeting and exercise your rights as a member. (Bring your membership card please).

PROGRAMMING CONTEST: At the December meeting, a contest was announced inviting members to develop a program that would print our logo and letter-head on a dot matrix printer. The program may be written in any of the languages supported on the TI 99/4A and the finished letter-head should occupy no more than the top two inches of the page. All entries are to be submitted to Paul Helwig by the March meeting. Our president has PROMISED a suitable award.

SCREEN DUMP PROGRAM: Michal Jaegermann has written an excellent screen dump utility which he has made available to all our members. The program requires a printer with graphics capability and any one of the following: MiniMemory, or Editor/Assembler and 32K memory, or Extended Basic and 32K memory. Please bring along your own disk or tape. Michal has asked that those members wanting a copy to make a suitable donation to our Equipment Kitty in lieu of a payment to him.

NEWSLETTER: Did you get your December issue of the newsletter? If you did, then you must belong to some other user group! Your trusty editor (me) had to let his helpers go north to help Santa with some last minute things like waxing the sleigh, getting a speeiling checker for Bob Pass, etc. With the Christmas rush, I just could not get a news letter ready in time for the January meeting. I will send out a make up letter in August.

Do any of you have any written material that you would like to see published? Come on, sure you do! Scribble something down and give it to me. I'll see that you become a published author in Canada, USA, England, Australia, and a few other countries if you can also give me a translation in Swahili. I would like to thank all of the people who have contributed excellent articles for this news letter; there is no way that I could do this on my own. Their contributions have made our newsletter so much more interesting to more people than what I could have done alone. But this does not let the rest of you off the hook. Let's get some more ideas from more of you and develop more interest in more members who will be stimulated more and they will get more ideas which (if they send them to me for publication) will give you more ideas on how to more out of your investment. So be greedy and send me some items for the news letter!

MODEM GROUP: Attention all members with modems. If you would like to establish contact with the rest of the world via your modem, fill out the form on the cover sheet of this news letter and get it back to us. We

will prepare a list of all members who respond and circulate the list to all respondents so that you can get in touch with each other to establish a communications network. We will have a few goodies for those who participate in this "SIG" (special interest group). To get started, Tom Hall will help get things rolling. Please fill out all areas of the form and return it by mail or at the next meeting.

A PRINTER SETUP PROGRAM
By Tom Hall

The following is a program which I wrote to simplify the process of setting up the printer prior to a print job. The program provides you with a menu of all the major functions ever likely to be needed in most printer applications. The program is written for my ROLAND PR-1010, but is completely compatible with the GEMINI-10X.
The program is self-explanatory, so enjoy!'

```
100 CALL CLEAR :: OPEN #1:"P
IO"

110 DISPLAY AT(2,6):"PRINTER
SETUP M:  :: DISPLAY AT(5,
1):"PITCH (10/12)? 10"

120 DISPLAY AT(6,1):"   . WID
E/ELONGATED? N" :: DISPLAY A
T(7,1):"EMPHASIZED? N"

130 DISPLAY AT(8,1):"COMPRES
:E"? N" :: DISPLAY AT(9,1):"
  .LE-PRINTING? N" :: DISPL
AY AT(10,1):"ITALICS? N"

140 DISPLAY AT(11,1):"LINE P
ITCH (6 OR 8)? 6"

150 ACCEPT AT(5,16)VALIDATE(
"012")SIZE(-2)BEEP:Z$ :: IF
Z$="" T--: 120 ELSE P=V AL(Z
$)

160 IF (P<>10)*(P<>12)THEN 1
50

170 ACCEPT AT(6,21)VALIDATE(
"YN")SIZE(-1)BEEP:DBL$ :: IF
DBL$="Y" THEN DBL=14 ELSE DB
L=20

180 ACCEPT AT(7,13)VALIDATE(
"YN")SIZE(-1)BEEP:EMP$ :: IF
EMP$="Y" THEN EMP$="E" ELSE
EMP$="F"

190 ACCEPT AT(8,13)VALIDATE(
"YN")SIZE(-1)BEEP:COMP$ :: I
F COMP$="Y" THEN COMP=15 ELS
E COMP=18

200 ACCEPT AT(9,18)VALIDATE(
"YN")SIZE(-1)BEEP:DP$ :: IF
DP$="Y" THEN DP$="G" ELS E D
P$="H"

210 ACCEPT AT(10,10)VALIDATE
("YN")SIZE(-1)BEEP:IT$ :: IF
IT$="Y" THEN IT$="4" EL SE I
T$="5"

220 ACCEPT AT(11,23)VALIDATE
```

("53")SIZE(-1)BEEP:LP$

230 DISPLAY AT(13,1):"ALL CO
RRECT? Y" :: ACCEPT AT(13,14
)VALIDATE("YN")SIZE(-1)BEEP
:YN$

240 IF YN$="N" THEN 150

250 PRINT #1:CHR$(27);"3";
260 IF P=12 THEN PRINT #1:CH
R$(27);"P";CHR$(0);

270 IF LP$="3" THEN PRINT #1
:CHR$(27);"0";CHR$(8);

280 PRINT #1:CHR$(DBL);CHR$(
27);EMP$;CHR$(COMP);CHR$(27)
;DP$;CHR$(27);IT$;

290 END

BASIC PROGRAMMING: -CASSETTE DATA FILES

by: Bob Pass

Some of you may not be aware that you can use your cassette recorder to do more than just load or save programs. Your cassette can also store data files which can be read into the console by a running program, modified by the user, and saved for later reference. By learning to use the basic commands OPEN#, INPUT#, PRINT#, & CLOSE# you can open up new horizons with your TI 99/4A by being able to save & recall data from cassette.

One important point to get clear first is the concept of "Buffers". The word "buffer" is used to describe an area of computer memory (or hardware) that is used to temporarily store data that is to be written into or out of the computer. Buffers are required whenever the computer must talk or listen to another device which does not operate at the same speed or in the same manner as the computer does. For example, since you cannot type at computer speed, the keyboard on your machine uses a buffer to pass information to the processor. Similarly, a cassate tape recorder simply cannot handle data at computer speeds; consequently the computer must use a buffer to transfer information to the device. Briefly, a buffer is a block of memory of fixed size which is referenced by a numerical tag (you can have more than one buffer available). When data is to be transfered, the computer will load the buffer with data untill it is full. Then the same buffer is read by a device service routine at a speed wich is compatible with the output device. When the buffer is emptied, more data is written into it untill the data transfer is complete. An important point to realize is that the transfer of data from the buffer to the external device is automatically done only if the buffer is full. If the buffer is only partially loaded when your application program ends, this data could be lost unless you instruct the system to close all open files (buffers). This will cause the system to finish dumping the buffer to the cassette. The last data item is always an end of file marker.

When data is read back into the computer, the process is reversed with the computer looking for the end of file mark so that it knows when to stop reading the buffer and shut down the external device.

As mentioned earlier, the buffers have a numerical

tag. In TI basic you can specify a tag from 1 to 255 with each buffer being distinct from any others by the tag number. Buffer number 0 is reserved for system use and is, in fact, the keyboard (and screen) buffer mentioned earlier. You can use more than one buffer at a time for different purposes; however the number of buffers that are open at the same time is limited to a default of 3. If you need more than three open buffers, use the CALL FILES(n) command where "n" is any number from 1 to 9. Note that this will limit you to a maximum of 9 open files or buffers at a time. The CALL FILES command must be used in the following way:

```
NEW
CALL FILES(n)
.C.
```

Now load your application program in the usual way and you will have the required number of files or buffers available. CALL FILES may not be used within a program; it must be entered in the command mode. Consequently, any program requiring more than three buffers must have the appropriate CALL FILES executed first. Each buffer that has been reserved occupies 518 bytes of RAM (except the first which takes up 1052 bytes) so it is wise to keep the required number of buffers as low as possible to conserve memory space.

Below is a short program that will allow you to set up & maintain a short telephone number list and save it to tape for later recall and/or modification. It can be easilly modified to hold more data as you see fit. There is one important thing that should be mentioned:

DO NOT USE A PROGRAM TAPE TO STORE DATA!

You wouldn't be the first to overwrite a program with a data file. It would be wise to keep your data files on separate tapes, preferably one file per tape to avoid confusion.

OPEN# This command prepares the system to transfer data to an accessory device. The buffer number (the TI manual calls buffers "files") is specified by you as well as the device name (such as CS1) to which the data is to be written. Additionally, you must specify the structure of the data file to be written on the cassette. Untill you have become thoroughly familliar with the TI User's Reference Guide and you have gained some experience working with cassette files, always specify "CS1", INTERNAL, SEQUENTIAL, FIXED for your file structure. Further more, you must tell the system the size of the data strings to be written (so that it will know how to read the data back later) by placing 64, 128, or 192 after the FIXED notation. You must plan the maximum length of each data item to be stored; if, for example you chose FIXED 64 in the OPEN# statement and then wrote a data item 70 characters long, the last 6 characters would either be lost or would overflow into the data of the next character string producing an unwanted concatenation or "trashed file". On the other hand, if your string was only 60 characters long, the system would automatically pad the string out to 64 with dummy characters which are stripped off when the data is recalled. Line number 260 from the program below contains the OPEN# statement. Notice that the size is 64 and that further more the data entry routine does not check for strings longer than 64 characters (see lines 420 to 460). By playing with this fact, you will be able to see what happens if you enter a very long string, saving the data to tape, and then reading it back.

CLOSE# This statement will cause the computer to empty the specified buffer number of pending data by completeing the transfer sequence. See lines 500 and 520. To prevent corrupted files, always close your

opened files under program control. Treat the OPEN# and CLOSE# statements like matched bookends. Do not place any statements between them that could cause a transfer of the program control out of the program block defined by these two statements. If you experience a program error message during a file transfer sequence, do not use FCTN QUIT as this will cause all data in the buffers to be lost. Instead, type BYE, RUN, NEW, OLD, SAVE, or LIST or else EDIT a line number; either of these actions will cause the buffers to be closed properly.

PRINT# This causes the system to transfer (print) data **FROM** the computer **TO** the device identified by and in the format specified by the OPEN statement whose buffer number corresponds to that of the PRINT# statement. See line number 500.

INPUT# This statement is the opposite of the PRINT# statement. It reads data **INTO** the system **FROM** an external device. The buffer number must match the corresponding OPEN# which conditions the system as to what format the incoming data will be in. See line number 280. Notice the comma in this line and also in the PRINT# statement (line 500). This is a data element separator which tells the system where the end of each data block is located; ie, when to pad the string out to the size specified in the "FIXED" parameter of the open statement. If you used a semi-colon (;) here, the two data elements would be joined together.

I encourage you to enter this program and experiment with it. Once you understand how it works, I am sure you will find many more applications of this concept. For further reading, refer to your User's Reference Guide, pages II-118 through II-136.

```
      THIS LISTING IS IN THE SAME
      FORMAT AS IT WILL APPEAR ON
      THE SCREEN UPON ENTRY.
      THIS IS AN ASTERISK (*),
      THIS IS A ZERO (0), AND THIS
      IS A LETTER O.
      THE FOLLOWING LINE IS A ROW
      OF ALTERNATING DASHES AND
      SPACES.
      - - - - - - - - - - - - -
100 REM **************
110 REM * PHONE LIST *
120 REM **************
130 ** TI BASIC
140 ** REQUIRES CASSETTE
RECORDER AND
150 REM CASSETTE CABLE.
160 REM DEMO OF CASSETTE
170 REM DATA STORAGE.
180 DIM NAME$(10),PHONE$(10)
190 CALL CLEAR
200 PRINT " PHONE LIST"::
::
210 PRINT "1-READ FILE FROM
TAPE"::"2-REVIEW AND ENTRY O
F DATA"::
215 PRINT "3.  SAVE FILE TO T
APE"::"4.  QUIT"::::::
220 INPUT CHOICE
230 IF (CHOICE>4)+(CHOICE<1)
=-1 THEN 190
240 ON CHOICE GOTO 250,320,4
70,540
250 REM READ TAPE FILE
260 OPEN #1:"CS1",INPUT ,INT
ERNAL,SEQUENTIAL,FIXED 64
270 FOR N=1 TO 10
```

```
280 INPUT #1:NAME$(N),PHONE$
(N)
290 NEXT N
300 CLOSE #1
310 GOTO 190
320 ** ENTER DATA IN FILE
330 CALL CLEAR
340 PRINT "WHICH RECORD NUMB
ER"
350 INPUT : Y
360 IF (ENTRY>10)+(ENTRY<1)=
-1 THEN 330
370 CALL CLEAR
380 PRINT "ENTRY NUMBER ",EN
TRY," IS:":NAME$(ENTRY)::
390 PRINT "WHOSE PHONE # IS:
":PHONE$(ENTRY)::
400 PRINT "1-ENTER NEW NAME"
:"2-ENTER NEW PHONE NUMBER":
410 PRINT "3-TRY ANOTHER REC
ORD":"4-EXIT REVIEW AND ENTR
Y MODE"
420 INPUT CHOICE
430 IF (CHOICE>4)+(CHOICE<1)
=-1 THEN 370
440 ON CHOICE GOTO 430,450,3
70,190
450 INPUT "NAME? ":NAME$(ENT
RY)
460 GOTO 370
470 INPUT "PHONE #? ":PHONE$
(ENTRY)
480 GOTO 370
490 REM SAVE FILE TO TAPE
500 OPEN #1:"CS1",OUTPUT,INT
ERNAL,SEQUENTIAL,FIXED 64
510 FOR N=1 TO 10
520 PRINT #1:NAME$(N),PHONE$
(N)
530 NEXT N
540 CLOSE #1
550 GOTO 190
560 END
```

A TI FORTH DISK FIXER

By: Tom Hall

The FORTH listing which follows is a utility which allows direct access to the data portion of any sector on a TI disk. Simply type the listing which appears at the end of this article into 3 consecutive empty screens on any FORTH disk and load it. What follows immediately are some instructions on the use of the utility. If you have any questions about its use, don't hesitate to get in touch with me.

The initial input is of the form <sctr#> FIND, where <sctr#> is the number of a sector, >0 thru >2CF. If you wish to reference the disk in drive 2, you may preceded the FIND command with the word DR1. The first input upon loading the utility may be a decimal number, but once the word FIND is executed, hex mode is invoked. What follows is a brief explanation of the major commands possible with this utility.

BM

This command should be used only when immediately preceded by "0 FIND". Use of BM displays the bit-map table of sector zero, showing in binary form the used and available sectors on the disk. A "1" indicates a used sector, a "0" an available sector. Since an offset from the beginning of the last sector located with FIND is computed, use of BM in any other sector will give

meaningless results.

ALT

This word allows data to be altered in memory, and then written back to disk. The format of the command is:

<addr> <wcount> ALT

where <addr> is the starting address and <wcount> is the number of words to be altered. The command can be aborted by simply striking the enter key with missing or invalid data under the cursor. This will cause an error message to appear but will leave the remainder of the addresses specified in the ALT command unaltered.

W

This command will write the current sector (the last one retrieved with the FIND command) to disk. Two points should be kept in mind when using the W command: (1) The command cannot be used again until another FIND command has been issued, or you will get an error condition; (2) the command is actually writing the current screen (4 sectors) to disk, so be sure that no changes have been made in any of the other 3 sectors associated with the current screen unless they are to be written to disk as well. If you wish to make sure that no undesired changes are made, the NEW command (which is simply EMPTY-BUFFERS) can be issued, followed by another FIND command. This will flush all buffers without writing anything to disk, and reload the desired sector from the disk.

FREAD

This word is for use in a sector which contains a directory entry. It will show the sectors on the disk which the file occupies, and yields the information in much the same format as the DISKO program, i.e. "<#> <cnt>" where "<#>" is the number of the first of a series of sectors containing the file data, and "<cnt>" is the number of consecutive sectors so occupied starting with sector <#>.

SCAN

May be useful when scanning the actual contents of a file, since the input for this command is the same information provided by the FREAD command, and in the same order. For instance, if the FREAD command displays the information "22 6", this means the file information starts in sector >22 and continues for 6 consecutive sectors. If you wanted to view this information, you could enter the command "22 6 SCAN", and you would be automatically shown the complete contents of the 6 sectors starting with >22. A built-in PAUSE feature allows momentary halting and resumption of scrolling with the press of any key, and FCTN 4 aborts the command.

THRU

A variation of the previous command, instead of a beginning sector number and count, this command takes the argument list of "<bgn#> <end#>", where "<bgn#>" is the starting sector number, and "<end#>" is the ending sector number. So, the command "22 28 THRU" would list out the contents of every sector beginning with >22, up to and including the contents of sector >28.

LOOK

When you want to scan a larger portion of the disk at one time, the LOOK command can be implemented. Like THRU, it takes a beginning and ending range, but instead of sector number, the LOOK command expects a beginning and ending screen number. The FIND feature works in this and all commands which produce a listing to an output device.

SOME ADDITIONAL NOTES

With the built-in base conversions of this utility, it is possible to alter the bit-map table with relative ease, a feature which is quite handy when, say, reconstructing a blown directory. One of the words invoked in the BM routine described earlier is a word called REV, which simply takes each bit of a byte, and reverses its order in the byte, from right-to-left, which is how the bit-map table is constructed. For instance, if you look at an initialized disk, you will see that the first word of the bit-map table reads "0300". In binary notation this becomes "00110000". Since we know that a newly-initialized disk has 2 sectors used -- Sectors 0 and 1 -- then in order to have these 1's in the right position, it is necessary to reverse the bits in each byte of the word separately. This makes "0011" into "1100", and "0000" is still "0000". The word REV does this quite conveniently.

Suppose the first line of your bit-map table looks like this:

11000000 00000000

This indicates that only the first two sectors of the first 16 sectors on the disk are used. Let's say that you determine that sector 3 should also be marked as used, so you want that line in the bit-map table to look like this:

11100000 00000000

The command to accomplish this would be as follows:

BIN 11100000 REV HEX .

This will print out the hex value of the word which would be written at that particular address in order to make the desired change to the bit map table. If the "." is omitted, then the appropriate value would be left on the stack and could be manually loaded into the required address with a "!" or "C!" command. Since the BM command also prints the addresses of each word pair in the bit-map table, the proper address to change can very easily be found.

```
( TI FORTH DISK UTILITY - By Tom Hall / Screen 1)
( 0 CLOAD ADDR ) 0 VARIABLE ADDR 0 VARIABLE ACC -1 VARIABLE OLD
0 VARIABLE SA
: ERR CR CR ." ERROR!" 7 EMIT ;
: Y -1 WARNING ! ' ERR CFA ' (ABORT) ! ; BASE->R HEX
0 VARIABLE SCN 0 VARIABLE SCTR
: LOC 9 0 GOTOXY ." CURRENT SECTOR: " SCTR ? CR CR ;
: FIND HEX DUP SCTR ! PAGE LOC 4 /MOD BLOCK DUP SCN ! SWAP 100
  * + DUP ADDR ! 100 DUMP ;  : NEW EMPTY-BUFFERS ;
: THRU 1+ SWAP DO I FIND 7FFF 0 DO LOOP PAUSE IF LEAVE ENDIF
  LOOP ; : 2U BASE->R 2 BASE ! U. R->BASE ;
: L PAGE LOC [ HEX ] ADDR  100 DUMP ;
: TEXT1 PAD 40 BLANKS PAD HERE - DUP ALLOT MINUS SWAP WORD
  ALLOT ; : HU BASE->R 10 BASE ! U. R->BASE ;
: STORE 2 SPACES TIB  30 EXPECT 0 IN ! DO 20 TEXT1 PAD NUMBER
  DROP SP I 1 MOVE U. 2 +LOOP ;  -->
```

```
( DISK UTILITY - By Tom Hall ; Screen 2)
: FIX 1 SLA OVER + SWAP DO CR I U. 2 I OVER + I STORE +LOOP ;
: ALT OVER 8 DUMP FIX ; : BIN 2 BASE ! ;
: W CR CR 8 SPACES ." WRITING SECTOR " SCTR  . UPDATE FLUSH ;
: LOOK HEX 1+ SWAP DO I DUP CR CR ." DUMPING SCREEN #" . CR CR
  BLOCK 400 DUMP PAUSE IF LEAVE ENDIF LOOP ;
: SCAN OVER + SWAP DO I FIND 7FFF 0 DO LOOP PAUSE IF LEAVE ENDIF
  LOOP ; : TU BASE->R A BASE ! U. R->BASE ; : S 4 SLA ;
: BP2 BASE->R 2 BASE ! 0 <# # # # # # # # # BL HOLD #> TYPE
  R->BASE ;
: REV 0 ACC ! 8 0 DO DUP 1 AND ACC  2 * + ACC ! 1 SRL LOOP
  DROP ACC  ; : Q 5 SPACES ;
: BM PAGE SCN  DUP 38 + DUP 59 + SWAP DO I DUP CR Q . C REV
  BP2 I 1+ C REV BP2 PAUSE IF LEAVE ENDIF 0 I SCN  - 38 - 2 /
  8 . 2 +LOOP DROP ;
  -->
```

```
( DISK UTILITY - By Tom Hall ; Screen 3 )
: INTL SCTR  4 /MOD DROP 100 * SCN  1C + + SA ! -1 OLD ! ;
: TRPL SA  3 DUP SA +! OVER + SWAP DO I C LOOP ;
: INTR BEGIN TRPL >R 10 /MOD R> 10 * + DUP OLD  - SWAP OLD !
  >R 100 * + -DUP WHILE . R> . REPEAT R> DROP ;
: FREAD INTL INTR ;
R->BASE
;S
NOTES:
PAGE is simply a word that clears the screen.
BM above is for a 2-sided drive; if your drive is 1-sided, use
the following modified BM:
: BM PAGE SCN  DUP 38 + DUP 2C + SWAP DO I DUP CR Q . C REV
  BP2 I 1+ C REV BP2 PAUSE IF LEAVE ENDIF 0 I SCN  - 38 - 2 /
  8 . 2 +LOOP DROP SCN  64 + DUP CR Q . C REV BP2 ;
```