THE OFFICIAL NEWSLETTER OF THE

TI-99/4A

APRIL 1986 NO. 7

FOR THE RECORD

by Ed Bittner
Recording Secretary

The March meeting of the West Penn 99'ers was opened by the newly elected president Scott Coleman promptly at 7:00. The initial discussion centered around the possibility of locating the meeting at alternative places. No conclusion was reached but it was stressed that not only convenience was important but also cost of rental. At the last executive committee meeting (Feb 25),yours truly volunteered to seek out the source of a newspaper article offering DS/DD diskettes at a very low cost. Ken Farr did indeed receive diskettes from this company and vouched as to there reliability. Scott asked for a motion to proceed with the purchase of 200 of these diskettes for club use and resale (as well as purchase of 50 cassettes (C-60) in conjunction with the PUG). John will(forth)ingly so moved and it was seconded and a unanimous vote ensued to go ahead with the purchase with club funds.

The club will now have DS/DD diskettes (which may also be used as SS/SD) for sale for the unbelievely low price of $.80 each in lots of 5 or more to club members. No firm price was established for larger quantities. J. Willforth reported that no new information was yet available regarding liability insurance for the group and Jan Travers gave the treasurers report.

Clyde Colledge reported on the status of the club library. It was generally agreed that Quality (programs that run) was much more important than Quantity (programs that don't). It was also felt that programs which get transferred to cassette should be capable of being run from cassette. Secondly, the library will not be of the type from which a member can borrow the library's only copy. Further discussion for the distribution of cassette information is necessary. Clyde promised that the library committee would meet and get back to us.

A short note on FREEWARE. Freeware may be distributed by the club but it is the users responsibility to send fees to the authors of such programs. Typically the names and addresses and fees are listed in the Freeware program.

Following the offical business, John Willforth did a nice demo of a RAM DISK (@$150 in kit form) which allows for a 90K storage when used in the SS/SD mode but can be upgraded to DS/SD easily (so John says'). With the RAM DISK access to programs is at a minimum, often loading programs faster by a factor of 10. It also is battery backed which brought a lot of ooze and ahz from the members.

Chuck Strink demoed a custom cursor program (I swear not !) and a "Text Flasher" program (Not concealed in a raincoat). He did distribute hard copies of these as well as a hard copy of his calculator program. A $1.00 raffle was held with two prizes being awarded, the Popeye Cartridge and a diskette of popular games. The meeting was immediately followed by the Basic course (Chuck Strink) and a Crash course in assembler by Clyde Colledge and Gene Kelly.

Errorroneously submitted,

Scoops Bittner

BASIC BASICS

by Charles Strink

One of the most helpful debugging tools in your programming kit is the TRACE command.

The TRACE command lists the line numbers of TI BASIC statements befor the statements are excuted. The line numbers are printed on your screen. If the program also prints information on the screen, you will see your program's information mixed in with the TRACE line numbers.

Once you have activated the TRACE command, you can deactivate it by entering a UNTRACE command.

If you use TRACE as a command, it will print the line numbers for every statement and if you are TRACING a large program this can be quite confusing to read. If you must TRACE a large program I suggest you use BREAK and CONTINUE as your screen fills with line numbers.

With a small program or only a section of a larger program, use TRACE and UNTRACE as statements within the program. You will then see only those statements that are excuted between the TRACE and UNTRACE statements.

UNTIL NEXT TIME.....
.....HAPPY COMPUTING

OFFICERS

PRESIDENT	SCOTT COLEMAN	(412) 271-6283
V. PRES.	CHUCK STRINK	(412) 668-2811
SECRETARY	ED BITTNER	(412) 864-4924
COR. SECY	GENE KELLY	(412) 829-0469
TREASURER	JAN TRAYERS	(412) 863-1575
LIBRARIAN	CLYDE COLLEDGE	(412) 828-3042
EDITOR	JOHN WILLFORTH	(412) 527-6656

WEST PENN 99'ERS TREASUREP'S REPORT FOR MARCH 1986

BALANCE AFTEP FEB. MEETING		$439.04
AMOUNT PAID OUT IN FEBRUARY		
CHECKS	$ 4.60	153.19
POSTAGE	22.00	
EQUIPMENT	126.59	
(2 SUPER DISKS)		
(1- 4 A/TALK)		
BALANCE AS OF MARCH 16.'86		$285.85

YES WE HAVE DISKS!!!!!!

AS OF THIS WRITING, WE HAVE DISKS, AND WILL HAVE MORE FOR THE MEETING. THE DISKS ARE DOUBLE SIDED DOUBLE DENSITY (DSDD), AND CAN BE USED ON ALL DISKETTE DRIVES THAT CAN BE CONFIGURED ON THE TI-99 COMPUTER. IF YOU CAN'T MAKE THE MEETING, AND WOULD STILL LIKE TO GET SOME, THE PRICE IS $8.00 FOR 10 DISKETTES WITH HOLE SAVERS, ENVELOPS, AND WRITE PROTECT TABS. PLEASE SEND $1.50 FOR EACH 10 DISKETTES ORDERED, SO WE CAN SEND THEM OUT TO YOU.
ANY QUESTIONS CONTACT SCOTT COLEMAN AT (412) 271-6283

NEXT MEETING....MONDAY, APRIL 21,1986.....
7:00 P.M., FOLLOWED BY BASIC AND ASSEMBLY SIGS AT APPROX. 8:15. DOOR PRIZE AND RAFFLE.

"POWER" EDITOR FOR TI-FORTH

by Michal Jaegermann

To LCT - the author of an original editor

In this article you will find a new editor for TI-Forth. But why bother if the existing editor is a quite decent one? Well, you will find in this new editor a lot of useful features which will make it into a quite a powerful tool which will assist you not only in creation but also in debugging your Forth programs.

First of all it sports an autorepeating keyboard - which is useful by itself. But you will find also overtype and insert modes and a limited but very convenient form of "cut-and-paste". On the top of it you will find an ability to single step through your source screen, with a continuous stack display and a possibility to execute any Forth word without leaving the editor. This last feature is a "real Forth" in the sense that you have not only full control over results of your actions, but also full responsibility. So be careful! In addition, this new editor adds to the system less than 2K of editor specific compiled code. So it is approximately of the same size as the old editor. Too good to be true? Read on

One more thing - while writing this editor I tried to be as compatible, with the old one, as possible. So you do not have to "unlearn" very few old habits in order to switch. Hope that this sounds attractive.

If you are still wondering about which editor of the two supplied I am talking about here then the answer is "both". You will find that THE SAME editor is used in both modes with two extra screens taking care of mode dependent display details and minimal differences in compilation of a couple of words.

You will find the code in screens following this article. (Editor's note: as there are 10 screens to the complete series, and space is limited, I must split up the screens over three or more newsletters. If you prefer, take advantage of the offer made later on in the artical. Sorry for any inconvenience folks.) Here is a description on how to use it.

System Requirements

First, the bad news. You need some extra words to compile the editor.

Good news - not too much and really handy on its own. Actually really necessary is only one: CMOVE) - move up memory contents. It requires on stack a starting address, a target address and a count in bytes. It will not do anything if a count is not a positive. For speed reasons definition is in code.

```
HEX CODE CMOVE) C09? , C07? , C03? , A0C? ,
                A04? , 060C , 0.. , 0b.? ,
                11?? , 0456 , 10FA , 045F ,
```

You will also need a .S for a stack display. It loads on a one of the -DUMP words but you may extract its definition from somewhere else if you wish. Three others are "convenience" words - if you do not like them - edit the source and forget about them. They are:

```
: \ IN C/L + C/L MINUS AND IN ! ; IMMEDIATE
 (for comments - nearly standard)
: 2DUP OVER OVER ;
 (quite obvious)
: AT GOTOXY ;
 (since I am alergic to GOTOXY)
```

And about the 64 column display. Since I have problems with an overscan, I moved the 64 column editor screen to the right. This is done by modification of SMASH on screen 65 of the system disk. In this code you will find (once only!) an entry 2000 , . Replace it with 2000 , . This will cause end-of-lines to disappear, but I think that this is a smaller problem. If you are lucky enough not to have an overscan problem then leave SMASH alone but remove 2+ from 64-column .CUR. By the way:

CLIST (n --) (list contents of a block n) may be defined as

```
: CLIST ( n -- ) BLOCK L/SCR 0
DO I C/L * OVER + C/L I SMASH .... LOOP DROP ;
```

and you do not really need CLINE and CLOOP.

The editor will trap all non-printable characters with one exception DEL, (hex 7F). If this is going to bother you add in EDI loop, right after RKEY the following:

' DUP 7F (... '. This will remove the problem.

If you find that the sensitivity of a keyboard does not suit your taste - play around with a delay loop in BLINK and constants imbedded in RKEY. They are not exactly independent but try it yourself.

After you have done all of the above you may load your new editor (you typed it in before - didn't you?) and try now to use it.

Starting and Leaving Editor

As usual, 20 EDIT will bring on your screen the contents of block 20 with an edit cursor in a home position. EDI will work also as usual. WHERE brings you to a location of a LOAD error.

One extra - ER (EditResume) recalls not only the last screen but also the last cursor position. So you will be back where you left the editor the last time. Once in the editor ctrl-E will switch to the previous screen (at home positon), and ctrl-X to the next one. Fctn-9 to get out.

Entering Text

Editor will come up in an overtype mode. So whatever you are typing replaces the text under cursor. Fctn-2 toggles between overtype and an insert modes. While inserting a new text pushes an old text to the right. Whatever spills over the right margin is lost.

Marking and Unmarking Text

Think of it this way. You have always exactly two marks. If they are invisible, then they are at the end of the current line and at your cursor position. Ctrl-Z puts a visible mark where your cursor is. The first one will replace (as mark) a cursor position, the second one - end of line. If you will try to put a third mark on a screen - the second one will be replaced with a new one. Visible marks are stored on stack, so if you have to move the first one, SWAP them (how to do it - later). Ctrl-U will erase all visible marks from your screen.

Deleting and Inserting In One Line

Fctn-1 will delete one character. Remember - it autorepeats
Fctn-3 deletes the whole current line and all subsequent text moves up. Deleted line is stored in an delete buffer.
Ctrl-8 opens a blank line over the cursor. Old last line is lost. So everything is like one would expect.
Fnct-7 removes all text between marks (visible or invisible) and replaces it with blanks. Up to 64 characters of removed text are stored in a delete buffer. If there is more - they are lost.

The action of fctn-8 depends on the editor mode. In overtype mode it acts as ctrl-8 but it moves text from the delete buffer into an opened line. While in insert mode - it inserts text from the delete buffer, WITHOUT leading and trailing (but one) blanks in time, on the cursor position. Old text moves to right. A right margin spillover is lost. Reread section on marking and experiment until you feel comfortable.

Do not hold fnct-7 too long, since it will clobber your delete buffer with freshly created blanks.

Additionally, you may Yank text to the delete buffer with ctrl-Y. This will put away text between marks for subsequent fctn-8 inserting WITHOUT removing the original from a screen. The same limitations as above will apply here.

Moving Around

Usual "arrow keys" will work - fnct-S, fctn-X, fctn-E, fctn-D. But also you have a "terminal style" controls. i.e. ctrl-H left, ctrl-J down, ctrl-K up and ctrl-L right. They are handy when single stepping through a source. Moreover ctrl-R will move to the right in one word steps. No key for a similar movement backwards.

Singlestep and Executing Forth

Ctrl-W will execute (if possible) any word on your screen which is pointed by the cursor. The cursor will advance to the next word and below your edit screen you will see a display of the stack. Great for debugging.

Ctr.-Q will do the same with two words (try 0 CLOAD EE ! . Do not try to execute compiling words like DO or IF since you will get an error. No big deal. ER will put you where you have just been, but the stack will be lost.

You may also hit ctrl-. to run an internal interpretor. You will be put below the edit screen and you may type there up to 80 characters of Forth to execute. Upon enter you will return automatically to the editor and the current stack will be displayed below. The editor part of a screen is frozen and will not scroll even if you dump the whole memory. This means that if you make an error then to unfreeze the screen, you have to return to editor (ER is ok) to get out later with fctn-9.

This is interpretor is even flexible enough to start a compilation (say with a colon definition), to return to editor, to do some editing and to resume a suspended compilation later. I am not advising you to use it normally in that way, but this is a great way to see for yourself how the compiler security is implemented and what DO is putting on stack to tell LOOP where to branch.

You will find out that in particular you may, using ctrl-., call editor itself - executing, for example, EDIT or ER. I would advise you not to do that. Reason is that you are storing on a return stack a return address. So, once you would like to get out and hit fctn-9 you will return... back to editor (previous instance). If you do that many times getting back to FORTH may take a number of fctn-9's. Ctrl-. and QUIT will always save the day.

How to Move Big Blocks

The editor above of course can be extended and made more powerful. But a goal was to make it convenient, nice and not too big. For example, one may add a big delete buffer and rewrite deleting and inserting a little bit to get a full "cut-and-paste". But instead of doing this I am using the ctrl-. feature on those unfrequent ocasions when I need more extensive capabilities. For example - how to move a block of five lines from screen number 23 into some other location on screen 37. Type 23 EDIT. In the editor, hit ctrl-. and once outside EDITOR #CUR. The word #CUR from the EDITOR vocabulary returns an address in an edit buffer which corresponds to a current cursor position. You will see it on stack. Now ctrl-9 and 37 EDIT. Once back in the editor, put your cursor on the position where you would like to see your block and type ctrl-. #CUR C/L 5 * CMOVE <enter>. You will find yoursylf back in the editor on the very beginning of an imported block. To mark the block as an update, just retype one character on screen. FLUSH will save all changes to disk. Another useful word, for such operations, from the EDITOR vocabulary is ROOM. See source screens for details.

How to Save Typing

Send us your disk in a self-addressed mailer (with proper postage included). If not in CANADA you may buy at your post office the proper amount of things called "Coupon Reponse International". You will get back a modified TI-Forth system disk with this editor and many other handy utilities included. It is really worth it. Our address is:

EDMONTON 99'er COMPUTER USER'S SOCIETY
P.O. BOX ....., EDMONTON, ALBERTA
CANADA ... ...

To make life easier - here is:

A Short Reference Guide

```
fnct-1  delete character
fnct-2  toggle overtype/insert modes
fnct-3  delete line
fnct-5  swap windows in text mode / home in bit-map mode
fnct-6  move right
fnct-7  delete between marks ( one or both may be defaults )
fnct-8  insert text from PAD
        * in overtype mode inserts a new line moving other
          text down
        * in insert mode insert contents of PAD on the cursor
          position shifting text on the line to the right
fnct-9  leave editor
fnct-(S,X,E,D) move cursor to the (left, down, up, right)
ctrl-8  open blank line
ctrl-(E,X)  get the (previous, next) screen
ctrl-W  execute one Word pointed by cursor - display stack
ctrl-Q  execute two words pointed by cursor - display stack
ctrl-R  move Right one word
ctrl-Y  Yank - store text between marks (64 max) in edit buffer
ctrl-(H,J,K,L)  move (left, down, up, right) - terminal style
ctrl-Z  mark cursor positon
ctrl-U  Unmark - replace marks by defaults (cursor, end-of-line)
ctrl-.  escape from editor to execute Forth. All Forth
        available. After <enter> or 80 characters typed returns
        automatically to the editor with a display of the
        stack. If you get an ERROR type ER to return back to
        the editor. Otherwise an edit screen will be frozen.
```

Source Screens

Referenced screen numbers are as on my disk. Remember to change them in LOAD statements if you put them in some other location.

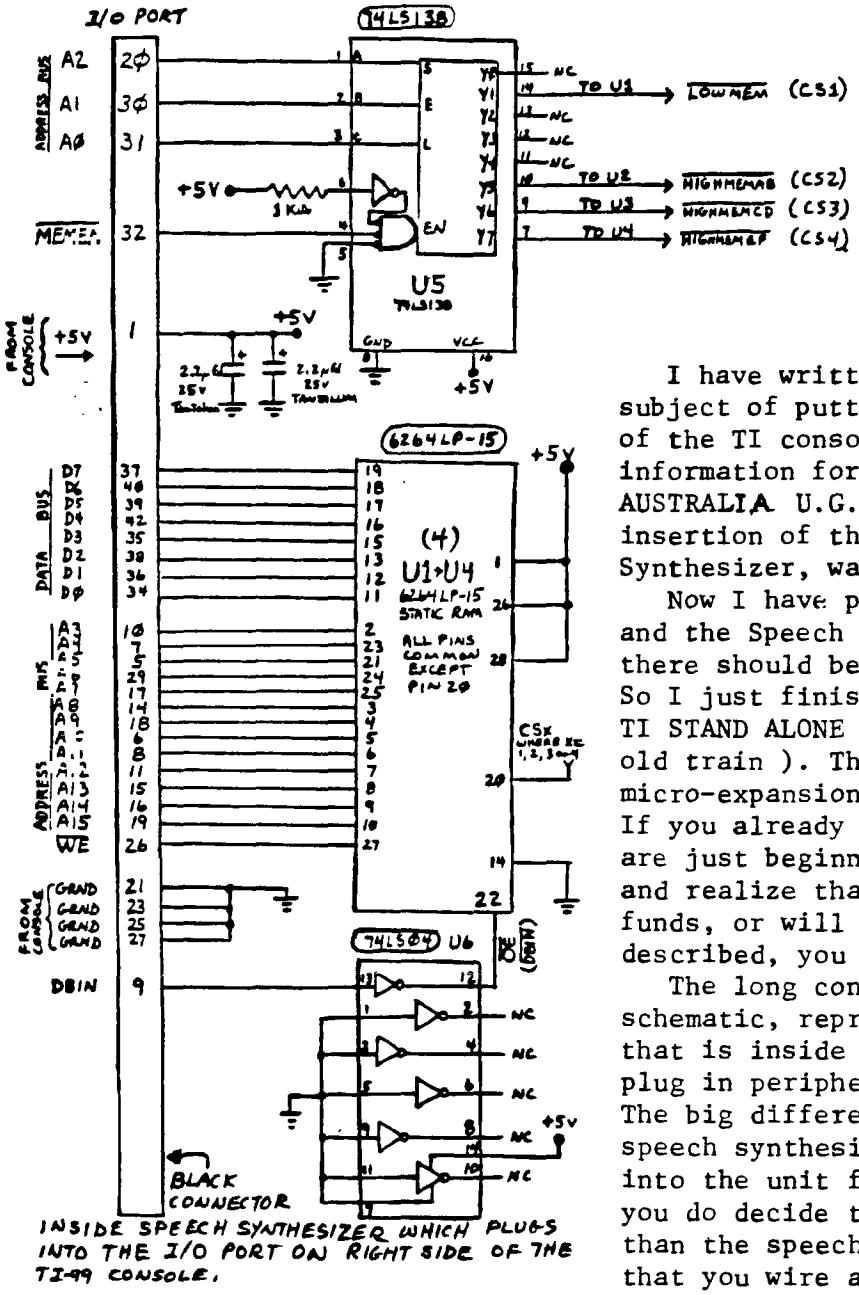
```
SCR# 20

( SCREEN EDITOR - 40 column display 20SEP85#Michal Jaegermann )
( 0 CLOAD EDIT ) BASE @ HEX     21 CLOAD  RANDOMIZE
VOCABULARY EDITOR IMMEDIATE EDITOR DEFINITIONS        3 WIDTH !
0 VARIABLE S_H              0 VARIABLE CLE             0 VARIABLE INS
: BLINK CURPOS @ DUP VSBR SWAP 1E OVER VSBW C0 0 DO LOOP VSBW ;
     18 LOAD    \ load cursor positions and auto-repeating key
: HLIST  0 0 AT SCR @ ."  SCR # " . CR CR CR
    L/SCR 0 DO I 3 .R CR LOOP ;
: LINE. DUP SCR @ (LINE) DROP S_H @ 1D * +
    SWAP 28 * 7C +  23 VMBW ;
: UP#  DUP 3 + SWAP DO I 0A .R LOOP ."  " 4 2 AT ;
: +.0  3 0 DO ." +....0...." LOOP ;
: CLIST L/SCR 0 DO I LINE. LOOP ;
: LLIST  4 1 AT        1 UP# ." ...." +.0 ." +" CLIST ;
: RLIST  4 1 AT ." 3" 4 UP# ." 0...." +.0        CLIST ;
-->
```

```
SCR# 21

( SCREEN EDITOR - 40 column display 20SEP85#Michal Jaegermann )
: ELIST  0 SCRN_START ! BASE->R DECIMAL HLIST S_H @
           IF RLIST ELSE LLIST ENDIF    R->BASE ;

: .CUR     COL S_H @ 2DUP 6 * 22 - MINUS >
    IF 0= 19 ELSE -4 ENDIF SWAP
    IF 1 S_H @ - S_H ! ELIST ENDIF - CUR C/L / 3 + AT ;
: ED>    2F8 SCRN_START ! PAGE 0 IN ! ;
: /ED    0   SCRN_START !  ELIST .CUR ;
: /INS/ 8F1 6 INS @ 1 OVER - INS ! 9C * 30 + VFILL ;
: FLIP  S_H @ -3A * 1D + +CUR .CUR ;
: BOX    NOP 2F8 DUP CURPOS ! C8 20 VFILL 8F1 6 84 VFILL
            R> DROP R> DROP ;  \ NOP is forward reference holder

    19 LOAD     \ load a generic (TI-Forth) editor
BASE ! ;S
```

WOULD YOU LIKE TO SEE THE LIGHT ?????

SCOTT COLEMAN TELLS ME THAT WHEN YOU PUT THE NEW SWITCHER POWER SUPPLY INTO THE OLDER TI CONSOLE (SILVER/BLACK), THAT MOST ... ...ERS DO NOT HAVE THE Light Emitting Diode, THE THING THAT SHOWS A RED LIGHT ON THE FRONT OF THE CONSOLE WHEN YOU TURN IT ON, AND AS A RESULT THERE IS NO ... INDICATOR PRESENT, AFTER THIS INSTALLATION.

SCOTT TELLS ME THAT THE OLD LED MAY BE REMOVED FROM THE OLD BOARD AND REINSTALLED IN THE NEW BOARD, BUT THE OLD RESISTOR WHICH HAS TOO HIGH A RESISTANCE, MUST BE DISCARDED, AND ... WITH A VALUE OF 330 OHMS + OR - 100 OHMS MAY BE USED. THE LED SHOULD BE PUT INTO THE NEW BOARD IN EXACTLY THE SAME PHYSICAL LOCATION, BUT THE RESISTOR IS NOW INSERTED IN THE CIRCUIT ABOUT 1 1/2" INTO THE BOARD BEHIND THE LED. FOLLOW THE ONE SOLDER RUN ON THE BOTTOM OF THE BOARD THAT DOES NOT GO ALONG THE EDGE OF THE BOARD, AND YOU WILL SEE WHERE ONE CAN BE INSERTED. .... GOOD LUCK! JFW/WPUG



INSIDE SPEECH SYNTHESIZER WHICH PLUGS INTO THE I/O PORT ON RIGHT SIDE OF THE TI-99 CONSOLE.

32 KiloByte MEMORY EXPANSION FOR INSIDE THE SPEECH SYNTHESIZER (OR ANY PLACE YOU WANT TO PUT IT).

by JOHN WILLFORTH
(based on ideas from the WESTRAILIA, and the CEDAR VALLEY USERS GROUPS)

I have written up several articles on the subject of putting 32K of static RAM inside of the TI console. I believe that most of the information for this came from the WESTERN AUSTRALIA U.G., and the work leading to the insertion of the same memory into the Speech Synthesizer, was done by the CEDAR VALLEY U.G.

Now I have put memory into both the console and the Speech Synthesizer. I thought that there should be no place you couldn't stick it. So I just finished putting it into the OLDE TI STAND ALONE DISK CONTROLLER (part of the old train). This made a nice quiet, sort of micro-expansion system (without RS232/PIO). If you already have a full blown system, or are just beginning to get int a disk system, and realize that you either don't have the funds, or will not need anymore than that just described, you should read on.

The long connector on the left of the schematic, represents the large 44-pin conn. that is inside the speech synth., or any other plug in peripheral ie: Stand-alone Disk Cont.. The big difference, however, is that ONLY the speech synthesizer carries pins 1,2,43, and 44 into the unit from the console. Therefore if you do decide to put memory into any other unit than the speech synthesizer, I would recommend that you wire across that unit, in other words you should run a wire from pin 1 on the console connector to pin 1 on the output end of that unit, where the 2nd unit from the console might be plugged in, and do the same for pins 2, 43, and 44. This will enable you to put the very small speech synthesizer out on the end, instead of between the 2 much larger units (console and Disk Controller). There is only one lead that is involved here that is a must, and that is the pin 1, since I have stayed with using the +5 VDC from the console, rather than tapping it from the +5 Volt source in the unit where this is installed.

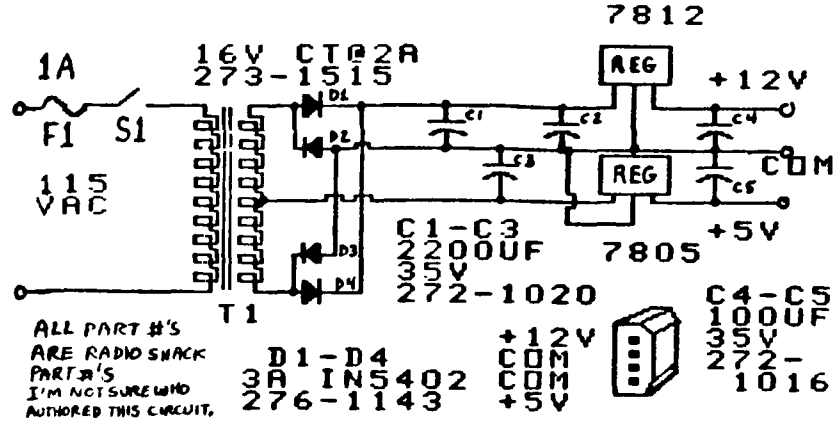
If you have the documentation on the RAM chip, you may be confused by the reverse order of the address lines. DON'T WORRY, just wire the chip up as I have indicated, and if you do your part correctly, it will work. I've done nearly 20 of these installations in the console and the speech synthesizer, and in a stand alone disk controller, and as far as I know, they are all working. If you want the more simple instructions, on how to install this same memory into your console, (which is what I prefer) just contact me, by sending a stamped , self-addressed envelop, and I will send the instructions. Have fun! JOHN WILLFORTH RD#1 BOX 73A JEANNETTE, PA 15644 , or call after 9:00 PM, (412) 527-6656

VCR TITLE SCREEN PROGRAM IN BASIC

by John Hedstrom

This is a simple program to title VCR tapes. It allows six lines of text (at rows 5,8,11,14,17,20) and 28 characters per line. To skip a line, press ENTER To print a quotation mark, type in 3 of them consecutively. The program will center each line horizontally and will then draw a border which looks like a filmstrip. The border uses the cursor which is ASCII code 030 and it remains black.) The computer can be hooked up to the VCR via the VHF antenna input (with modulator cable and VCR set to tuner) or via the Video In input (with monitor cable and VCR set to line) and the title screen can then be taped for any length of time.

```
100 REM  VCR Title Screen
110 REM  by John Hedstrom
120 REM  December 3, 1984
130 CALL CLEAR
140 INPUT "Screen Color?_":S
150 INPUT "foreground Color?
_":F
160 INPUT "Line #1?_":L1$
170 INPUT "line #2?_":L2$
180 INPUT "LINE #3?_":L3$
190 INPUT "LINE #4?_":L4$
200 INPUT "LINE #5?_":L5$
210 INPUT "LINE #6?_":L6$
220 T1=(30-LEN(L1$))/2
230 T2=(30-LEN(L2$))/2
240 T3=(30-LEN(L3$))/2
250 T4=(30-LEN(L4$))/2
260 T5=(30-LEN(L5$))/2
270 T6=(30-LEN(L6$))/2
280 CALL CLEAR
290 CALL SCREEN(S)
300 FOR C=1 TO 12
310 CALL COLOR(C,F,1)
320 NEXT C
330 PRINT TAB(T1);L1$::::TAB(
T2);L2$::::TAB(T3);L3$::::TAB(
T4);L4$::::TAB(T5);L5$::::
TAB(T6);L6$::::
340 CALL HCHAR(1,1,30,32)
350 CALL HCHAR(24,1,30,32)
360 CALL VCHAR(1,1,30,24)
```



DISK DRIVE POWER SUPPLY

DISPLAY/VARIABLE 80 to PROGRAM CONVERTER

article written by Al Kinney

Here is a nifty program to convert text that has been stored as DIS/VAR 80 back into a runnable program. (The origins of the program are uncertain to me, but, here it is!)

Now why, you might ask, would you want to do that? Glad you asked!! And now, I'm gonna' tell ya!! If you use COMPUSERVE, there is a Special Interest Group (SIG) for TI home computers, with over 1000 members!! These wonderful folks are constantly trying to out-do each other by putting excellent Public Domain software into the Down Load area of the SIG. Think of it a Mail Box, and the files are NOT bills.

Now, the only problem you have, is that for lots of very technical reasons, the files are stored as text in the DIS/VAR 80 format. In the "OLDEN" days, whenever you downloaded a file, you had to sit and type the darn thing in, and the way I type, I created more errors than I could fix!! Now, all that drudgery is gone, and by simply running the program listed below, you can recreate the program in the MERGE (DIS/VAR 163) format' Then, by removing the exclamation (REM) marks from each line, you will have a MERGEable file.

This also give you another way to edit existing files you may already have. Let's say you have a program, and you want to make a LOT of changes to it! First, you would simply load the program into memory, as usual, by entering "OLD DSK1.NAME". When loaded, you would then enter "LIST "DSK1.NAME/I"". That would cause the program to be written to the disk in DIS/VAR 80 format. The reason for changing the name should be obvious, to prevent over-writing the original file! Now, instead of flailing through the entire program, line by line, looking for variable XYZ, you can use TI-WRITER or EDITOR/ASSEMBLER, and do "Global" search and replace's! Those features are described in the respective manuals. After you have modified the program as needed, you simply run the conversion program, which rewrites the file with a NEW name. When it is finished, type "NEW", and "MERGE "DSK1.PROGRAM/2"", and proceed as before!

```
100 !!!!!!!!!!!!!!!!!!!!!!!!
110 !!  TRANSLATES FROM    !
120 !! DIS/VARBO TO MERGE  !
130 !!       FORMAT        !
140 !!!!!!!!!!!!!!!!!!!!!!!!
150 !
160 !Use a "Full Screen"
170 !Editor to create Ext-
180 !ended Basic programs
190 !
200 !Create a file using TI-
210 !Writer \\ Make sure you
220 !disable the word wrap
230 !mode and limit the line
240 !length to 80 characters
250 !
260 !
270 CALL CLEAR
280 DISPLAY AT(3,7)BEEP ERAS
E ALL:"!!!TRANSLATE!!!"
290 DISPLAY AT(7,5):"DIS/VAR
 80 FILENAME:"
300 ACCEPT AT(9,5)SIZE(15):I
N$
310 DISPLAY AT(12,5)BEEP:"ME
RGED OUTPUT FILENAME?:"
320 ACCEPT AT(14,5:SIZE(15):
OUT$
330 OPEN #1:IN$
340 OPEN #2:OUT$,VARIABLE 16
3
350 LINPUT #1:L$
360 S=POS(L$," ",I)
370 ON ERROR 490
380 N=VAL(SEG$(L$,I,S))
390 ON ERROR 440
400 A=INT(N/256)
410 A$=CHR$(N-A*6):: PRINT L$
420 PRINT #2:CHR$(A);A$;CHR$
(131);SEG$(L$,5&I,80);CHR$(0
)
430 GOTO 350
440 PRINT #2:CHR$(255);CHR$(
255)
450 CLOSE #2
460 PRINT :"ENTER ""NEW""
AND THEN ""MERGE""THE TRANSL
ATED FILENAME:":"OU
T$: : :
470 PRINT "REMEMBER TO REMOV
E THE     LEADING ""!"" IN
EVERY
LINE
  ": : : :
480 END
490 ON ERROR 440
500 RETURN 350
```

REMOTE CONTROL DISPLAY

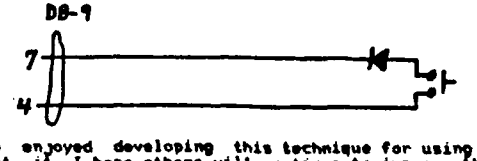
As I began preparing for a music demonstration at an Atlanta User's Group meeting I began to think of ways I could use the computer itself to help me present my program. The display features of the TI99 would make it very easy to present several lines of information, stop for discussion, then change any of the lines or clear the screen and present new information. Graphics, sounds, and color changes could also be used to emphasize certain parts. I could also use programs from the club library to show how different programs are written and their effectiveness.

I started out using a simple "Press any key to continue" routine, and wrote my demonstration program to pause after each segment. This worked well, but it restricted me to the keyboard, so I created a "Magic Button", a simple pushbutton switch connected to the Joystick port which provided me with room to move around. I could point to the display, move toward the group for discussion, and then change the display or execute an instruction with the press of the "Magic Button." The button is electrically the same as the FIRE button of Joystick#1, and is read by CALL KEY(1,K,S).

I wrote a subroutine "HOLD"(XBasic) to insert at end of each called program I used. Then I could insert the instruction "CALL HOLD" anywhere in the program to pause and discuss what was happening, or about to happen. The subroutine listed below will recognise any key on the keyboard or the "Magic Button"(FIRE Button #1), and contains a latch so that once the button is pressed, the program will not continue until it is released. This prevents accidently skipping short segments of program by holding the button too long.

```
30000 SUB HOLD
30010 CALL KEY(5,K,S)
30020 IF S=0 THEN 30050
30030 CALL KEY(5,K,S)
30040 IF S=0 THEN 30090 ELSE 30030
30050 CALL KEY(1,K,S)
30060 IF S=0 THEN 30010
30070 CALL KEY(1,K,S)
30080 IF S<>0 THEN 30070
30090 SUBEND
```

The construction of the Magic Button was done with parts from a local "surplus" store. The button itself was an 8 pole double throw momentary contact switch with 7 poles cut off. The long base of the switch made an excellent handle. Almost any switch will do, and the remote switch on a cassette microphone will do very nicely. The diode is a general purpose low power type (IN914, IN4001, etc.) The DB9 plug is somewhat more difficult to find, but if you have a broken Joystick you can use the plug from it. I used a 14 foot silver telephone cord because it is small and flexible. This is how the button is wired.



I have enjoyed developing this technique for using the TI99, and learning more about it. I hope others will continue to improve this and find other uses for our "little 99". Together we'll show Lubbock that while we may be orphans, we ain't giving up yet!

Jim Hubbard, A9CUC

PROGRAMMING PAGE - Tom Wynne

Our TI99/4A speaks many languages. Here is a simple typewriter program written in BASIC, PASCAL, FORTH, AND ASSEMBLY LANGUAGES. These all can be terminated by pressing FCTN 4 (CLEAR). These programs just print the characters you type onto the screen.

BASIC

```
100 REM TYPEWRITER PROGRAM
110 ROW=3
120 COL=1
130 CALL CLEAR
140 WRT$="  TYPEWRITER PROGRAM IN BASIC"
150 FOR I=1 TO LEN(WRT$)
160 PIECE=ASC(SEG$(WRT$,I,1))
170 CALL HCHAR(1,I,PIECE)
180 NEXT I
190 CALL KEY(0,CHR,S)
200 IF S=0 THEN 190
210 CALL HCHAR(ROW,COL,CHR)
220 COL=COL+1
230 IF COL=33 THEN 240 ELSE 260
240 COL=1
250 ROW=ROW+1
260 IF ROW=25 THEN 270 ELSE 190
270 END
```

FORTH

```
3 VARIABLE ROW
1 VARIABLE COL
: TYPEWRITER 3 ROW !
             1 COL !
             TEXT 3 1 GOTOXY
             ." TYPEWRITER PROGRAM WRITTEN IN FORTH"
             BEGIN
               COL @ ROW @ GOTOXY
               KEY
               EMIT
               COL @ 1 + COL !
               COL @ 41 = IF
                 1 COL !
                 ROW @ 1 + ROW !
               ENDIF
             ROW @ 25 = ?TERMINAL OR UNTIL ;
```

PASCAL

```
PROGRAM TYPEWRITER;

VAR ROW,COL: INTEGER;
    CHR      CHAR;

BEGIN
  ROW:=3;
  COL:=01
  GOTOXY(0,1);
  WRITE('TYPEWRITER PROGRAM ');
  WRITELN('WRITTEN IN PASCAL');
  REPEAT
    GOTOXY(COL,ROW);
    READ(CHR);
    COL:=COL+1;
    IF COL=40 THEN
      BEGIN
        COL:=0;
        ROW:=ROW+1
      END
  UNTIL ROW=24
END.
```

ASSEMBLY (Assemble with the R option)

```
        DEF  START       TYPE 'START' UNDER PROGRAM NAME
        REF  VSBW,VMBW,KSCAN,VWTR
STATUS  EQU  >837C
KEYADR  EQU  >8374
KEYVAL  EQU  >8375
SPACE   BYTE >20
TITLE   TEXT ' TYPEWRITER PROGRAM WRITTEN IN ASSEMBLY '
START   MOV  R11,R10     SAVE RETURN ADDRESS TO GO BACK TO E/A OR BASIC
        LI   R3,>F018    >F018 VDP REGISTER 1 VALUE
* WRITE TO VDP REGISTERS TO GET 40 COLUMN TEXT MODE
        MOVB R3,@>83D4   MOVE VALUE OF VDP REGISTER 1 TO >83D4
        LI   R0,>01F0    VALUE TO BE PUT INTO REGISTER 1 >01 >F0 TEXT MODE VALUE
        BLWP @VWTR       WRITE THE VALUE TO REGISTER
        LI   R0,>07F0    >07 REGISTER 7 >F0  >F WHITE ON >0 BLACK
        BLWP @VWTR       WRITE THE VALUE
        LI   R0,>0401
        BLWP @VWTR       SET CHARACTER TABLE SO EITHER EXB OR E/A CAN BE USED
* CLEAR SCREEN
        CLR  R0
LOOP    MOVB @SPACE,R1   LOAD R1 WITH SPACE CHARACTER
        BLWP @VSBW       WRITE THE SPACE
        INC  R0          ADD 1 TO ADDRESS
        CI   R0,960      IS THE ADDRESS 960? (ROW 24, COL 40)
        JNE  LOOP        NO, JUMP BACK TO LOOP
* PUT TITLE ON TOP OF SCREEN
        LI   R0,0        COLUMN 1 (ADDRESS 0)
        LI   R1,TITLE    GET ADDRESS OF FIRST LINE OF TEXT
        LI   R2,40       40 CHARACTERS TO WRITE
        BLWP @VMBW       WRITE IT
        LI   R7,80       CURSOR POSITION ROW 3 COLUMN 1 (ADDRESS 80)
        LI   R1,0        KEYBOARD SCAN 0 (ENTIRE KEYBOARD)
        MOVB R1,@KEYADR  ADDRESS USED TO HOLD KEYBOARD SCAN NUMBER
        CLR  R1
* MAIN KEY INPUT
AGAIN   MOV  R7,R0       LOAD REGISTER 0 WITH CURSOR POSITION
        LI   R1,>1E00    CURSOR CHARACTER
        BLWP @VSBW       WRITE THE CURSOR TO SCREEN
        BLWP @KSCAN      GET KEY INPUT
        MOVB @STATUS,@STATUS
        JEQ  AGAIN       NO KEY PRESSED
        MOVB @KEYVAL,R1  R1 KEY VALUE
        CI   R1,>0200    IS FCTN 4 PRESSED?
        JEQ  LEAVE       YES, LEAVE PROGRAM
        MOV  R7,R0       CURSOR POSITION
        BLWP @VSBW       WRITE CHARACTER TO SCREEN
        INC  R7          INCREMENT CURSOR POSITION
        CI   R7,959      IS CURSOR AT BOTTOM OF SCREEN
        JEQ  LEAVE       YES, LEAVE PROGRAM
        JMP  AGAIN       NO, GO BACK AND CHECK FOR KEY PRESS
LEAVE   LI   R0,>0400    RESET CHARACTER TABLE TO E/A OR BASIC
        BLWP @VWTR       WRITE TO REGISTER 4 WITH 0
        MOV  R10,R11     PUT RETURN ADDRESS BACK IN R11 TO RETURN
        RT               RETURN TO BASIC OR E/A
        END
```

SPRITE ONE LINER
by Barron Bartlett

Want to frustrate and amaze your Atari, VIC-20, Color Computer friends. Just type in the following in the command mode with Extended Basic.

```
CALL CLEAR :: CALL SCREEN(5) :: CALL MAGNIFY(2) :: FOR I=1 TO 28 ::
CALL SPRITE(#I,64+I,16,80,80,3*I,8) :: NEXT I :: FOR J=1 TO 5000 ::
NEXT J
```

Hit ENTER and watch all 28 sprites do their tricks. If you want to see it again, simply hit function REDO then ENTER again as many times as you wish.

132 COLUMN LISTER
Excerpts from Ed York's Column - CBM-BAY 86 Newsletter

The following program comes to us from Jim Peterson o Tigercub Software. Jim submitted the program i response to a challeage which appeared a few issue back. The challeage was to be able to list a progra to the printer in condensed print while defeating th 80 character per line default. Well, Jim has done i again! The program requires that you have both extende basic and a disk drive. The program also requires tha you 'list' the program to the disk, in order to create DIS/VAR 80 file. I know that the program will not wor on a Prowriter, due to the variable line length in lin 220. NOTE: The section of line 210 that reads 'PRIN #1:CHR$(15);CHR$(27);CHR$(78);CHR$(3)' sets condease print and skip over perforation at three lines from th bottom of the page. I wish the program would work wit the prowriter, and maybe someone can send me either correction or a new modified listing. The problem whe using this program with a Prowriter, is that should th line exceed 136 characters (which is the maximum numbe of characters per line in condensed print on Prowriter) then you do not get a line feed, thus you get printin atop printing. However, the problem does not exist whe using it with an Epson. I hope someone can send me corrected version. How about it Jim ? Do you have an ideas on how to make it work for the Prowriter ?

```
100 ! 132 COLUMN LISTER
110 !       BY
120 ! Jim Peterson
130 !       of
140 ! TIGERCUB SOFTWARE
150 !
160 CALL CLEAR :: CALL SCREEN(5):: FOR A=0 TO 14 ::
    CALL COLOR(A,14,5):: NEXT A :: DISPLAY AT(12,1):"
    CONDENSED PROGRAM LISTER"
170 DISPLAY AT(16,1):" WRITTEN BY: Jim Peterson": : :
    :" TIGERCUB SOFTWARE ": : :" PRESS ANY KEY TO
    BEGIN"
180 CALL KEY(0,B,C):: IF C=0 THEN 180 ELSE CALL CLEAR
    :: DISPLAY AT(2,1):" CONDENSED PROGRAM LISTER":
    :"IN ORDER TO USE THIS PROGRAM"
190 DISPLAY AT(6,1):"YOU SHOULD HAVE RESERUENCED":
    :"YOUR PROGRAM (STARTING WITH ": :"LINE 100 AND IN
    CREMENTS": :"OF 10) AND ALSO LISTED IT TO"
200 DISPLAY AT(14,1):"THE DISK (WHICH WILL CREATE": :"A
    DIS/VAR 80 FILE). HAVE THE": :"TWO ABOVE
REQUIREMENTS BEEN": :"PERFORMED ? Y"
210 ACCEPT AT(20,13)VALIDATE("YN")SIZE(-1):A$ :: IF
    A$="N" THEN CALL CLEAR :: END ELSE DISPLAY
    AT(12,1)ERASE ALL:"PLEASE ENTER THE FILENAME:"
220 DISPLAY AT(14,1):"DSK1." :: ACCEPT
    AT(14,6)SIZE(10):B$ :: OPEN #1:"PIO",VARIABLE 255
    :: PRINT #1:CHR$(15);CHR$(27);CHR$(78);CHR$(3)
230 OPEN #2:"DSK1."&B$ :: D=100 :: LINPUT #2:C$ ::
PRINT #1:C$;
240 LINPUT #2:C$ :: E=POS(C$," ",1):: IF E=0 THEN PRINT
    #1:C$;:: GOTO 240 ELSE IF SEG$(C$,1,E-1)
    ()STR$(D)THEN PRINT #1:C$;:: GOTO 240
250 PRINT #1: :: PRINT #1:C$;:: D=D+10
260 IF EOF(2)THEN CLOSE #1 :: CLOSE #2 :: END :: ELSE
    GOTO 240
```

****POWER SUPPLY****

Whilst I was scrounging around at Radio Shack at Lindale Mall, I came across a very interesting item. It is a Coleco plug in power supply for the Adam computer, Radio Shack 8277-1022. This power supply plugs right in the power socket on the wall, and has about a 4 foot connection cable. The end of the cable has the same mating connector that is presently on the back of our TI. Well, I had to see what this thing could do.

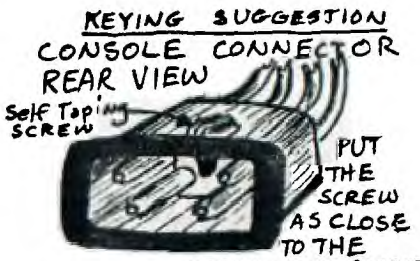
My thoughts are to completely remove the internal power supply from the console and use only this wall mounted unit. This would result in greatly reduced heating in the console, and more room for goodies to be installed. The only drawback is the off-on switch would be gone, and there isn't one on this Coleco power supply. This isn't a big problem for me because I use a plug strip with a switch and surge suppressor to run my computer. If things didn't turn out, I could always put things back the way they were, and use this plug in supply for other projects.

For you TECH WEENIES: I cut the sealed plastic case open to find out what makes this thing tick. It consists of a two linear foldback current-limited regulators with massive aluminum heat sinks for the 5 and 12 volt supplies, and a 3 terminal heat sunk 79M05 regulator for the -5 volts. The main secondary is fused, and there is thermal limiting included. The +5 volts is rated at 0.9 amp, +12 at 0.3 amp, and -5 volts at 0.1 amp. The outputs are reverse polarity protected. The regulators used are LM723 precision voltage regulators. Quite impressive for $4.95! I have drawn the schematic out and included it (opposite this page) for those that wish to use this bargain power supply. - Gary D. Bishop

EDITORS NOTE:

I have talked to Gary Bishop several times about this power supply since he actually put one into use. There are a couple things that should be noted, if you plan to go ahead with this project.

* You should key the recepticle on the back of the console where the connector from the power supply enters the TI-99. This new supply is at D.C. potential at this point and not A.C.. Since you removed your internal D.C. supply, and wired this new supply directly (internally) to the CPU board, the act of accidentaly plugging an old T.I. A.C. Transformer, will immediately spell doom with a capital "D" for the console, by putting A.C. on the D.C. circuits!
* Don't forget to remove power from the unit when you go on vacation !

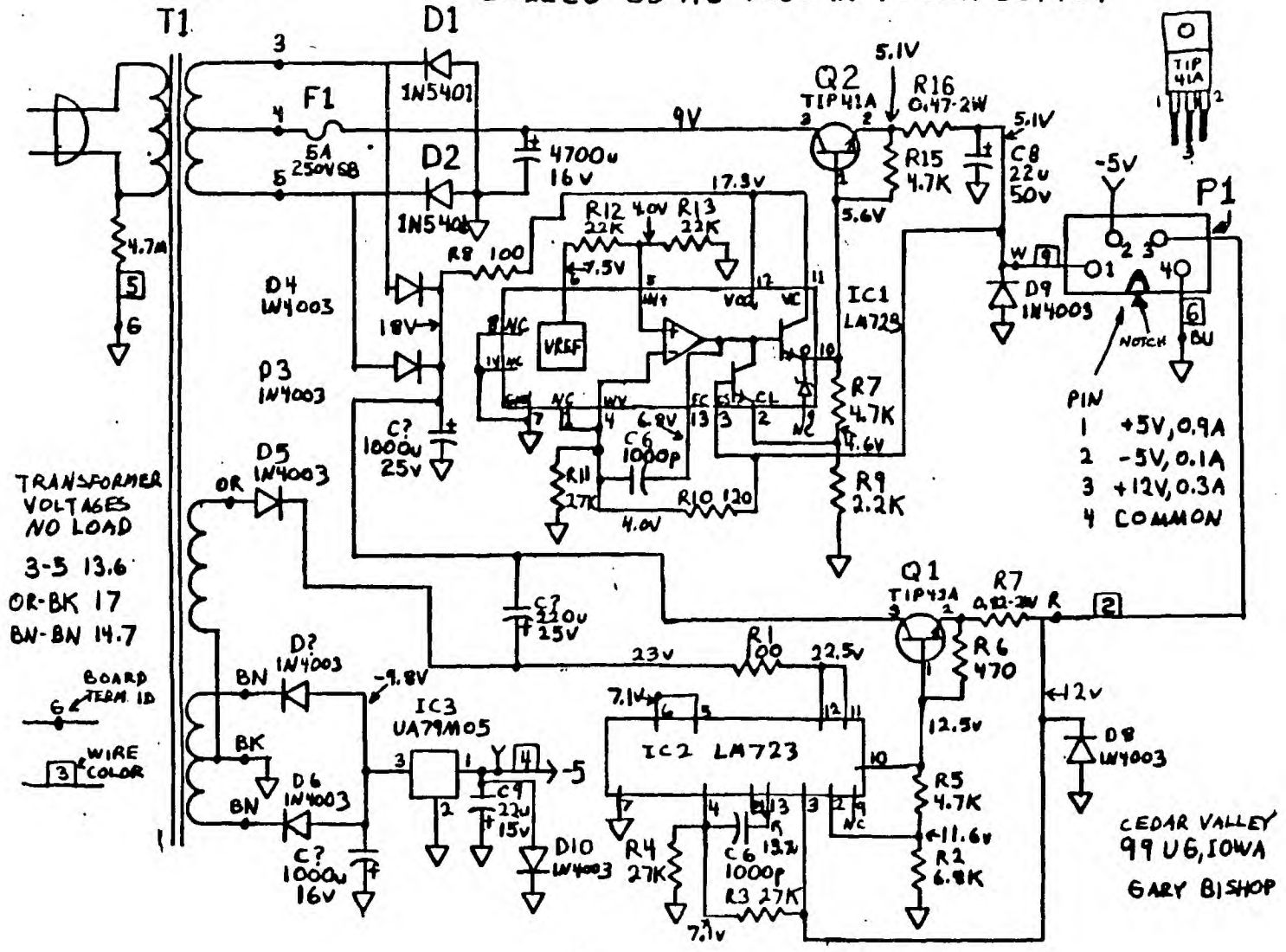


THEN.....

NOTCH THE PLUG FROM (P1) THE COLECO SUPPLY WITH A KNIFE SO THAT THE RUBBER CONNECTOR CAN SNUGGLY ENTER THE CONSOLE RECEPTICLE ABOVE. * PUT THE SCREW BETWEEN THE 2 WIDLY SPACED PINS *.

NOW..... A standard T.I. Transformer Power Supply connector should not enter.

SCHEMATIC OF RADIO SHACK 277-1022
COLECO 55416 PLUG IN POWER SUPPLY



DISK DRIVES BY JIM NESS

From LEHIGH 99'ER

It's funny (at least to me), but there are lots of people who seem to know lots of stuff about their computers, and all those tiny chips, and how the bits and bytes are handled. And there seems to be next to nobody that knows anything about disk drives, and how they work. Sensing this huge gap in man's knowledge, I decided to figure out what makes them tick.

The great thing about disk drives is that they can find files buried randomly within a huge field of data, and they do it pretty fast. Actually, they can do it so fast because it's not at all random.

The mechanical concept is not all that complicated. A small motor spins at 300 rpm (at least in this country, with its 60 hz power supply), and there is a tiny stepping motor attached to a read-write head. A stepping motor is a common item in indexing applications, where you want a motor to move a precise distance and stop on a dime. The read/write head is just a smaller version of what you have on a cassette recorder.

The stepping motor "steps" the head from track to track on a diskette. The tracks are concentric circles, not a long spiral as you would have on an album.

All of this is ultimately controlled by the disk software with your computer. Usually this is located in ROM within the machine. In most machines, the ROM is only sophisticated enough to load in the official Disk Operating System (DOS) which is located on the disk in the drive when the machine is turned on. The DOS contains all the file handling software, copying software, etc, and because it is on disk, it can be easily modified and/or updated as time goes by.

Our friends at TI decided to put the whole thing in ROM, which has a few bad side-effects. First, it makes it hard to update and improve the software, which is located in the Disk Controller Card. Second, although the machine is a 64k machine, just like all the others, TI has set aside so much memory for special purposes, that there is only 32k left to play with. They set aside 8k for cartridges, 4k for disk drive, 4k for RS232/PIO cards, 4k for the Operating System, (can't complain about that one), and 8k for various interfaces (speech, sound, VDP). Ok those are all good applications to have, but if you don't use them, you still can't use that memory for other things.

Anyway, all of the controlling software for the TI/99/4A is located in the ROM card, as I said. This software tells the step motor when to step to the next track, when to return to the beginning, etc.

There is no standard for how a computer keeps track of data. In the case of TI, there is a directory of existing files, and a map of where they are located, at the beginning of each disk. These files are not necessarily all in complete groups. If you delete a 12 sector file from a disk, there is a 12 sector gap recorded in the map. Then if you add a 20 sector file, the software will put the first 12 sectors in the gap, and put the rest in the first available spot. When you ask for a file that is broken up this way, you can hear the disk head scooting along to read each individual segment.

Because the disk drives themselves are pretty standard, there are a few things that don't change. For instance, there are 40 tracks per inch in most 5 1/4" systems (There is a new 96 TPI system around, not TI compatible). And most systems only use 35 or 40 of the available 48 tracks. There are either 9 or 18 sectors per track (single or double density). Each sector holds 256 bytes of data. And the standard design allows 250,000 bits per second to be written.

Wow, you say, 250k! That is about 25k bytes per second, right? How come I can not load a 25k pgm in one second then? Two reasons. First, as I said, the transfer of data is actually controlled by the ROM software in the TI99/4A. And to be as good as it is, it had to be a little bit slow. Not REAL slow (anyone ever use a C64 disk drive?), but not as fast as it could be. The second reason also has to do with software, but it is a universal problem associated with single density storage.

The major difference between single and double density storage is the way in which the data is coded. In order for the software to keep track of where the read head is located on a particular track, there are clock or synch bits laid down with the data bits. In the old fashioned single density format, a synchbit was laid down ahead of each "0" bit, so there were never two "0" bits in a row. That kept the software from getting lost if there were a lot of "0" bits in series. Putting all those synch bits on the disk took up a tremendous amount of space that should be used for data.

So, some genius came up with a way of encoding the clock bits in with the data bits, so that no unnecessary space was lost. Wala, double density storage was born! And double density, as used with the CorComp software, is said to increase transfer speed by at least 80%, mostly because the number of bits to transfer is cut way down.

So much for the exciting story of double density versus single density. How about double sided versus single sided? Well, obviously, it requires two read/write heads in the drive. Did you know that when reading a disk, the software reads, first, a track from side one, then the opposing track from side two, and continues back and forth?

The disk head needs something to keep the disk stationary against it. In a single sided drive, there is a small arm holding the back side of the disk against the head. In a double sided drive, that arm would be in the way of the back side read/write head, so the solution was to use two heads, directly across from one another, to hold the disk in place. In order to keep them across from one another, they alternate reading or writing as I said above. Very interesting, right? So if you wreck one side of a dbl sided disk, you can kiss the whole thing goodbye.

```
10 REM *ZLOTTO-CHIG V4 #9
20 REM *[PRTR MANDATORY!!]
30 REM *4/15/85 Rel 1.0
40 REM *AUTHORS:Jerry Novak
50 REM *and Jerry Tvrdik
60 REM *Set-up Menu
70 CALL CLEAR :: CALL SCREEN
(3)
80 DISPLAY AT(2,2):"Program
ZLOTTO:"
90 DISPLAY AT(3,9):" Generat
e 6 Random Numbers  Per Gam
e Set-4/15/85 R1.0"
100 DISPLAY AT(7,5):"[Printe
r MANDATORY]"
110 DISPLAY AT(14,14):"By";T
AB(10);"Jerry Novak "
120 DISPLAY AT(16,10):"Jerry
 Tvrdik"
121 DISPLAY AT(18,5):"CHICAG
O TI USERS GROUP"
130 DISPLAY AT(24,2)BEEP:"Pr
ess any Key to Continue"
140 CALL KEY(0,K,N):: IF N=0
 THEN 140
145 CALL CLEAR
150 GOSUB 520
160 DEF R=INT(RND*44)+1
170 INPUT "-Number Games to
Play-?":A
180 REM *Your PRTR(PIO,ETC)
190 OPEN #1:"PIO"
200 PRINT #1:"  -";A;"-";"ZL
OTTO Cards"
210 PRINT #1:"--------------
----------"
220 CLOSE #1
230 RANDOMIZE
240 DIM D(6)
250 FOR L=1 TO A
260 FOR M=1 TO 6
270 D(M)=R
280 FOR Z=1 TO M-1
290 IF D(Z)=D(M)THEN 270
300 NEXT Z
310 NEXT M
320 GOSUB 330 :: NEXT L :: E
ND
330 FOR I=1 TO 6
340 FOR J=1 TO I-1
350 IF D(I))D(J)THEN 370
360 TEMP=D(I):: D(I)=D(J)::
D(J)=TEMP
370 NEXT J
380 NEXT I
390 FOR X=1 TO 6
400 PRINT D(X)
410 IF D(X)<10 THEN PRINT "
 ";
420 NEXT X
430 GOSUB 460
440 PRINT " " :: RETURN
450 REM *Your PRTR(PIO,ETC)
460 OPEN #1:"RS232:BA=1200"
470 FOR X=1 TO 6
480 PRINT #1:D(X);
490 IF D(X)<10 THEN PRINT #1
:" ";
500 NEXT X :: CLOSE #1 :: RE
TURN
510 REM *Your PRTR(PIO,ETC)
520 OPEN #1:"RS232.BA=1200"
530 PRINT #1:"* Super ZLOTTO
 Numbers *"
540 PRINT #1:"**************
**********"
550 CLOSE #1 :: RETURN
```

```
190 OPEN #1:"RS232.BA=1200"

460 OPEN #1:"RS232.BA=1200"

520 OPEN #1:"RS232.BA=1200"
```

CHANGE THE ABOVE 3 LINES
TO USE A PARALELL PRINTER

RUN THE ABOVE VERSION (RS232)
WITHOUT A PRINTER
AND THE NUMBERS
WILL PRINT ON THE SCREEN
BUT WILL SCROLL OFF
THE TOP OF THE SCREEN
IF YOU RUN MORE THAN 3 GAMES

SAMPLE PRINTOUT

```
* Super ZLOTTO Numbers *
**********************
  - 6 -ZLOTTO Cards
----------------------
 5   17  30  31  37  38
 2   9   16  19  23  44
 15  22  23  25  33  43
 7   10  21  22  33  41
 4   10  13  14  36  43
 1   10  14  15  34  43
```

File Transfers on TIBBS By John Pearce

File transfers is another way of saying upload/download and that is the subject of this article. There are three communications programs in general use in the TI-99 community: Terminal Emulator II cartridge, 4A/Talk, and Fast-Term. Each of these programs is capable of doing TE2 protocol file transfers. The remainder of this article covers the essential points of doing file transfers with the TIBBS bullstin board using the TE2 protocol.

Just to be certain the terminology is straight, let's define upload and download with respect to the caller and TIBBS. An upload is a file transfer from the caller to TIBBS. A download is a file transfer from TIBBS to the caller.

Terminal Emulator II Cartridge

The first step is to select the communications configuration. For TIBBS, you may either choose the default option (menu choice 3) or you may select individual parameters (menu choice 2). If you take the latter choice, you may select either 110 or 300 baud, either even or odd parity, and select full duplex. To download a file, you will need either the file number from the download file list or a file name, such as TUTOR4. To initiate a download from TIBBS, you select D for download in the TIBBS File Transfer section (<U>pload/download at TIBBS' main menu). TIBBS will ask for a file number. Enter the file number or file name and press ENTER. If you make a mistake while typing, use CTRL-H (backspace) to correct your error as FCTN-S will not generate a backspace character. TIBBS will then send information which initiates the transfer. After some preliminary information, the screen will clear and TE2 will display the message HOST HAS STARTED FILE TRANSFER. After a short delay, the message PLEASE ENTER DEVICE NAME TO OUTPUT DATA TO is displayed. The response to this is DSK1, DSK2 or DSK3 (without a period) then press ENTER. The next message is PLEASE ENTER FILE NAME TO OUTPUT DATA TO. This is the file name (up to 10 characters) of the file you are downloading. Type in the file name and press ENTER. The advisory message FILE TRANSFER IN PROGRESS is displayed along with record count, block count, and retry count. When the transfer is finished, the message SUCCESSFUL FILE TRANSFER is displayed. The screen is cleared and TIBBS will display the File Transfer prompt. If the file transfer is aborted, TE2 will display such a message.

To upload a file to TIBBS, select U for upload. TIBBS will request a file name for the file to be uploaded. Type in a file name up to 9 characters long and press ENTER. TIBBS will clear the screen and display a copyright notice. You must press CTRL-4 to start the transfer. TE2 will display the message ENTER DEVICE NAME TO TRANSMIT DATA FROM. This should be DSK1, DSK2 or DSK3 (again without the period). Then the message PLEASE ENTER FILE NAME TO TRANSMIT DATA FROM is displayed. Type in the file name you wish to upload. The transfer will be initiated and the status information will be displayed as for the download. When the transfer is completed, the SUCCESSFUL FILE TRANSFER message is displayed. If it is necessary to abort either upload or download, use CTRL-3. After aborting a transfer, it may be necessary to press ENTER to force TIBBS to display the File Transfer prompt. There are three general reasons why a transfer would be aborted: the caller pressed CTRL-3, there was a disk error, or the disk was full. The last two errors can occur at either the caller's machine or at TIBBS. TIBBS will usually not permit an upload to start if there is less than 15 sectors available on the upload disk drive.

4A/Talk

Configuring 4A/Talk is straight forward. You may select either 110, 300 or 1200 bits per second for baud rate (depending on your modem). The parity selection must be odd parity (PA=0) and data bits must be 7 (DA=7). An example might look like "RS232.PA=0.DA=7" for a 300 baud modem on port 1. Unlike TE2, it is necessary to specify odd parity for communicating with TIBBS. The entries to and responses from TIBBS are the same for 4A/Talk and Fast-Term as for Terminal Emulator II. To initiate a download from TIBBS, you select D for download. Enter the file number or file name and press ENTER when requested. If you make a mistake while typing, you may use CTRL-H (backspace) or CTRL-S to correct your errors. 4A/Talk generates a backspace character when FCTN-S is pressed. TIBBS will then send the command which initiates the transfer. After some preliminary information, the screen will clear and the TE2 file transfer status display will be shown. The control portion of the screen looks like this:

```
Filename: DSK1.
Retries:
Record Number: 0
Sectors Remaining: 0
```

The cursor is positioned at DSK1 waiting for you to enter the name of the file. You may change the 1 to 2, 3 or 4 as needed. The Record Number counts from 1 to 5 while the Sectors Remaining starts with the sector size of the file and counts backward to zero. When the transfer is complete, a success message is displayed at the top of the screen. To upload a file to TIBBS, select U for upload. When requested, type in a file name up to 9 characters long and press ENTER. TIBBS will clear the screen and display a copyright notice. You press CTRL-1 to select the TE2 transfer menu then select choice 1 - send a file. The TE2 file transfer status display (as above) will be shown. Enter the file name to upload and press enter. The file transfer will be initiated. The Record Number will count from 1 to 5 as with the download and the Sectors Remaining will count down to zero. When the upload is completed, a success message will be displayed at the top of the screen. To abort either upload or download, press CTRL-X.

Fast-Term

The configuration of Fast-Term can be easier than either the TE2 cartridge or 4A/Talk. You can create a separate configuration file for TIBBS and specify that file when the program starts. Again, you must specify odd parity. Fast-Term will automatically set 7 data bits. You must also enable the TE2 protocol by pressing FCTN-Shift-T all at the same time. If done correctly, the message "TE2 Protocol On" will be displayed on the screen. To download a file from TIBBS, you should first press FCTN-N and enter the name of the file. Then enter D for download to TIBBS. Enter the name of the file when TIBBS requests it and press ENTER. Fast-Term displays a small file transfer status block similar to that displayed by the TE2 cartridge. The status block shows the record count, block count, and retry count. When the transfer completes successfully, you will hear the chimes and a completion message will be displayed. The reason for entering the file name with FCTN-N before entering the D for download to TIBBS is to avoid having TIBBS "time-out" and abort the transfer. To upload a file to TIBBS, select U for upload and enter the file name when requested by TIBBS. Then press FCTN-N and enter the name of the file to be uploaded and press ENTER. Then press FCTN-, (comma) to begin the transfer. The file transfer status block will be displayed showing the record, block, and retry counts. When the transfer completes successfully, you will hear the chimes and a success message will be displayed. The reason for entering the file name (FCTN-N) after giving TIBBS the file name is to avoid starting the file transfer (FCTN-,) before TIBBS is ready. After receiving the file name, TIBBS does 3 to 5 seconds worth of housekeeping to get ready for the upload. If the file transfer initiation message arrives from Fast-Term while TIBBS is still doing housekeeping, the message will be lost. To recover, abort the transfer (FCTN-4) and start again.

Summary

The TIBBS procedures and displays are the same regardless of which communications program you use. However, the procedures with the communication programs do vary. There is no great mystique to doing uploads and downloads. There are two things to be careful about: selecting odd parity and enabling TE2 in Fast-Term. In both 4A/Talk and Fast-Term you must select odd parity. The parity selection in the TE2 cartridge seemingly makes no difference. In Fast-Term you must enable the TE2 protocol before initiating the transfer.

If you have questions or problems, please feel free to leave me a message on TIBBS and I'll answer it as quickly as I can.

Fixing Blown disks

by Terry Atkinson

No. NJ 99'er Users Group Newsletter

If you have had a disk drive for any length of time, chances are you have encountered such devastating messages as "disk not initialized" (when you know full well it is!), or "program not found" (when you know is supposed to be there!). Or, perhaps, you have accidentally deleted a program and now you want to get it back. All of the above are fairly easy to remedy, and I think this information will give you some idea on what to look for, and how to proceed.

Fixing the disk bit map-(AU0).

As mentioned in my last article, AU0 or Sector 0 contains the disk bit map and if the characters "DSK" are altered, you will be unable to catalog or copy the disk. Indeed, a "DISK NOT INITIALIZED" error will show up. You can, however, retreive programs and files individually, and transfer them over to another disk. That is, if you KNEW the names of ALL the programs/files on that disk. There is a far better way, and it eliminates the possibility that you "forgot" that a particular program was there.

Boot up your disk fixer and load sector 0 from a disk. ANY disk will do. Then write the good sector 0 to the bad disk. This restores AU0 on the bad disk, but the bit map is NOT correct. But this does not matter!. All you want to do is to be able to catalog and copy the disk using DM2 and now you can. Use DM2 (not FORTH) to copy the entire disk to a new disk. You may then initialize the bad disk. That's all there is to it. You've retrieved your programs intact.

Ruined bit maps may not be discovered until it is too late. Any new programs saved to a disk with a ruined bit map may write over older programs or data. Goodbye older program. There's nothing you can do about it.

Another possibility is that S0 has been damaged, perhaps by magnetism or a scratch on the surface. In this case, you'll quickly find out when you try to read/write sector 0. You won't be able to. Now you have a problem, but not insurmountable. The only "fix" for this is to copy all sectors from the bad disk to a good disk, sector-by-sector. A tedious chore to be sure, but at least you can get all your programs back. It will still be necessary to proceed as above to get your programs back, as the bit map on the new disk will not be correct. Now, I am not sure how FORTH would behave under this circumstance. I know FORTH will "choke" when it tries to copy a damaged sector, but whether or not it will continue to copy the "good" sectors and put them into their proper places on the new disk, is beyond me. I wouldn't chance it. Better to be safe than sorry, and stick to tried and proven methods. Of course, you could experiment'! If it works, let us all know. If some of you FORTH addicts out there could shed some light on the subject, your comments would be most welcome.

Fixing the Directory Link Map (AU1).

In my last article I said that S1 kept track (alphabetically) of all the programs/files on the disk. That's the sole purpose. Bad S1's could produce errors such that attempts to catalog the disk will produce a heading, but no programs, or maybe just 'some' programs will be listed. To fix this, though, is extremely simple. Here's how:

First, look at AU0. Read the bit map to determine which sectors between 2 and 33 inclusive ()2-)21) are flagged as used. Make a list of these sectors in a column. Now, load each of these sectors in turn, and examine the first 10 bytes of each. Copy the bytes down beside the relevant used sector. Determine th alphabetical order of these programs merely by reading the numerical values. The lower the number, the closer to the front of the alphabet it is. Now, produce a list of these sectors arranged alphabetically. Here's a short example:

Sector used	Hex Values in 1st 10 bytes ())	Program Name
2	4B 20 20 20 20 20 20 20 20 20	K
3	49 20 20 20 20 20 20 20 20 20	I
5	4C 20 20 20 20 20 20 20 20 20	L
6	41 20 20 20 20 20 20 20 20 20	A
A	41 42 20 20 20 20 20 20 20 20	AB

Re-arranging the above alphabetically by sector would produce: 6,A,3,2,5 which are going to form the directory link map in WORD

Next, copy sector 1 from ANY freshly initialized disk and write it to the bad disk. This is the easiest way to "restore" S1 to all zero's. Now, use the (A)lter command, and change the first, and each successive word to produce the alphabetical pointers. For example:

@@06 @@0A @@03 @@02 @@05 @@00

Note the @@00 at the end. The directory link map must be terminated with this value. Now, write this sector to the bad disk, and you're in business.

Retrieving an accidentally 'deleted' program

When you have a program in main memory, and type "new", the program is not erased. Only the pointers are changed, but the program is still in memory. A knowledgeable programmer could actually "unnew" a program, although not without difficulty.

The same applies if you "delete" a program from the disk. Only pointers are changed, and the program is still on the disk provided you have not performed a "save" since the deletion. Unlike main memory, retrieval of a deleted program from disk is extremely easy. I will tell you the easiest way to do it. As you gain more confidence with the use of the disk fixer, you will undoubtedly find other ways as well.

Locate the sector containing the deleted file's directory (between)2-)21). You can do this by using the "FIND STRING" command, or, if you disk fixer does not have this command, merely load them in one at a time and look for your "deleted" program's name in the first 10 bytes. Change the program name to "ZZZZZZZZZZ" (HEX code of course). Now write that sector back to it's proper spot. Load-in sector 1 and locate the first word containing @@@@ and replace it with the directory sector # of your deleted program. Ensure the next word is @@@@. Now, exit the DF and load the subject program as per normal. Exit the disk-fixer, and load the program as normal and save it BACK to the same disk under the same program name (ZZZZZZZZZZ). Why?. Because this will automatically update the disk bit map (AU0). Now use DM2 to change the program name back to it's original name and the task is complete. You have recovered your "lost" program.

It was my intent to give a couple of challenges (using the disk fixer) in the next article, but reviewing the first two articles, I realized that perhaps a little more detail could be given on how programs are stored on disk, and what all these wierd numbers are in the first sector of a given program. Watch for it in the next newsletter, and HAPPY HACKIN'.

***** *****

```
31996 ! PROGRAM TO EXTRACT
LINES FROM ONE PROGRAM TO BE
INCLUDED IN ANOTHER PROG
RAM
31997 ! ELIMINATES UNWANTED
LINES OR SEGREGATES PARTS OF
ONE PROGRAM TO REMOVE
31998 ! ROUTINES OR TO SAVE
31999 ! REKEYING ROUTINES
32000 CALL CLEAR :: CALL INI
T :: INPUT "Line numbers of
routine to  be saved:  F
irst, Last?     ":L,M :: G=2
56 :: CALL PEEK(-31952,H,I,J
,K)
32001 C=INT(M/G):: D=M-C*G :
: F=(J-G)*G+K :: FOR E=(H-G)
*G+I TO F STEP 4 :: CALL
 PEEK(E,A,B):: IF A=C AND B=
D THEN 32003
32002 NEXT E :: PRINT :"Line
";0;"not found!" :: STOP !@P
-
32003 H=INT(E/G):: I=E-(G*H)
:: H=H+G :: C=INT(L/G):: D=L
-C*G :: FOR E=E+4 TO F S
TEP 4 :: CALL PEEK(E,A,B)::
IF A=C AND B=D THEN 32005 !@
P-
32004 NEXT E :: PRINT :"Line
";N;"not found!" :: STOP !@P
-
32005 E=E+3 :: J=INT(E/G)::
K=E-(G*J):: J=J+G :: CALL LO
AD(-31952,H,I,J,K):: STO
P !@P-
32006 !@P-
```

BASIC PROGRAMMING
CASSETTE DATA FILES
by Bob Pass
Edmonton, Canada UG

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 #, and CLOSE #, you can open up new horizons with your TI-99/4A by being able to save and 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 written into and 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 cassette 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 compatible with the output device. When the buffer is emptied, more data is written into it until the data transfer is complete. An important point to realize is that the transfer of data from the buffer to the external device isdone automatically only if the buffer is full. If the buffer is only partially loaded when the application program ends, this data could be lost unless you instruct the system toclose all open files (buffers). This will cause the system to finish dumping the buffer to cassette. The last data item is always an END OF FILE (EOF) marker.

When data is read back into the computer, the process is reversed, with the computer looking for the End Of File marker so that it knows when to stop reading the buffer and shut down the external device.

The buffers have a numerical tag. In TI BASIC or EXTENDED BASIC, you can specify a tag from 1 to 255 with each buffer being distinguished from others by the tag number. Buffer number 0 is reserved for system use and is, in fact, the keyboard (and screen) buffer.

You can use more than one buffer at a time for different purposes, however the number of buffersthat are open at the same time is limited to a default of three (3). If you need more than 3 buffers open, use the CALL FILES(n) command, where (n) is any number from one to nine. NOTE: This will limit you to 9 open files or buffers at a time. The CALL FILES command must be used in the following manner:

NEW < ENTER> CALL FILES(n) < ENTER > NEW < ENTER>

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 of RAM), so it is wise to keep the required number of buffers as low as possible to conserve memory space.

There is one important thing that should be remembered:

DO NOT USE A PROGRAM TAPE TO STORE DATA !!!!!!!!!!!!!!

You would not be the first to overwrite a program with a data file. It is advisable to maintain your data files on a separate tape, preferably one file per tape, to avoid confusion.

OPEN #n - This command prepares the system to transfer data to an accessory device. The buffer number (n) (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 bewritten to or read from. Additionally, you must specify the structure of the data file to be written on the cassette. Until you have become thoroughly familiar 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. Furthermore, you must tell the system the size the data string to be written will be (so 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 you choose FIXED 64 in the OPEN # statement and then write a data statement 70 characters long, the last 6 characters would either be lost or would overflow into the next character string, producing an unwanted concatenation or a "trashed file". On the other hand, if your string was only 60 characters long, the system would automatically pad the string to 64 characters with blank characters, which are removed when the data is recalled.

PRINT #n - 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.

CLOSE #n - This statement will cause the computer toempty the specified buffer number (n) of pending data by completing the transfer sequence. Treat OPEN # and CLOSE # statements like matched bookends. Do not place any statements between that would 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 !!!!! This will cause all data in the buffers to be lost. Instead, type BYE, RUN, OLD, SAVE, LIST or else EDIT a line number. Any of these actions will cause the buffer to close properly.

EOF

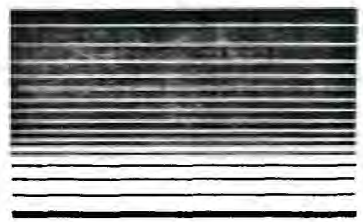
YE OLDE' TEST PATTERN
by H MacDonald

While programming the other day I came to a part I wanted to print to the screen. I've set up many screen formats and am sick of blue or light green. So I whipped out the QUICK REFERENCE CARD and opted #14 - Magenta. I finished the program and ran it just as my wife came into the room. "Oh what a pretty shade of red!", she said. "No, that's not red, that's Magenta!", I said. She said something like "mumble mumble", and left. So I looked Magenta up and fouund out it is really a shade of purple. (She told me later she thought magenta was red .) So the following program prints a Test Pattern like my Dear Ol'Dad used to use in the TV Reapir Shoppe when I was just a child in 1955 BC (before color).

I tried just putting it into a loop but the colors are not lined up in any way to compare them properly. Also, if you want to use inbasic, just start a new line every time you see ::. Run the program and use the test pattern to adjust the color "HUE" and "INTENSITY" controls on your TV or monitor. You too can have purple magenta or red. Orange red is nice too, olive green magenta is not so nice.

```
100 CALL SCREEN(16)
110 CALL CLEAR
120 A=1
130 CALL COLOR(0,4,4)::CALL
VCHAR(1,A+1,31,48)
140 CALL COLOR(1,3,3)::CALL
VCHAR(1,A+3,39,48)
150 CALL COLOR(2,13,13)::CALL
VCHAR(1,A+5,47,48)
160 CALL COLOR(3,10,10)::CALL
VCHAR(1,A+7,55,48)
170 CALL COLOR(4,9,9)::CALL
VCHAR(1,A+9,63,48)
180 CALL COLOR(5,7,7)::CALL
VCHAR(1,A+11,71,48)
190 CALL COLOR(6,8,8)::CALL
VCHAR(1,A+13,79,48)
200 CALL COLOR(7,6,6)::CALL
VCHAR(1,A+15,87,48)
210 CALL COLOR(8,5,5)::CALL
VCHAR(1,A+17,95,48)
220 CALL COLOR(9,2,2)::CALL
VCHAR(1,A+19,103,48)
230 CALL COLOR(10,15,15)::CALL
VCHAR(1,A+21,111,48)
240 CALL COLOR(11,16,16)::CALL
VCHAR(1,A+23,119,48)
250 CALL COLOR(12,14,14)::CALL
VCHAR(1,A+25,127,48)
260 CALL COLOR(13,12,12)::CALL
VCHAR(1,A+27,135,48)
270 CALL COLOR(14,11,11)::CALL
VCHAR(1,A+29,143,48)
280 GOTO 280
```

Mac



****

Below is a Extended Basic program that will keep your disk drives running until you push FCTN 4 (clear). Many disk drive cleaning kits require the drive to run for 30 seconds. Use this program and stop when the clean time has been reached.

```
10 CALL CLEAR
20 CALL SCREEN(13)::FOR C=1 TO 12::CALL COLOR(C,16,13)::NEXT C
30 DISPLAY AT(12,10):"CLEANING...."::DISPLAY AT(23,2):"(Hold FCTN
CLEAR to Stop)"
40 ON ERROR 60
50 GOSUB 70
60 GOTO 40
70 RUN "DSK1.B"
80 RETURN
```

Reprinted from June/July 1985 newsletter of the Wiregrass 99/4A Users Group.

(from MID ILLINOIS's MICRO)

****

SAPPHIRE SOFTWARE
P.O. Box 18124
Pittsburgh, PA 15236

X-xbasic E-editor/asm M-minimemory T-ti-writer

#	PROGRAM NAME	DESCRIPTION	CARTRIDGE	PRICE	CHECK
1)	PUB WRITER	TI-WRITER LOADER & FILES	XB	$10	____
2)	UPDATES	TI-WRITER AND MULTIPLAN	T,MP	$5	____
3)	TI-FORTH	LANGUAGE AND MANUAL	E,M	$15	____
4)	FORTH GAMES	SEVERAL GOOD FORTH PROGRAMS	E & FORTH	$10	____
5)**	TI-RUNNER	ARCADE GAME WITH 50 SCREENS	E,M	$10	____
6)**	BASIC COMPILER	COMPILES XBASIC SUBROUTINES	X	$10	____
7)**	TILE BREAKER	ASSEMBLY "BREAKOUT GAME"	X,E,M	$10	____
	BOXER	ASMB BOXING GAME(2 PLAYERS)	X,E,M		
8)**	D-STATION	ASSEMBLY SPACE BATTLE GAME	X,E,M	$10	____
	D-STATION II	CONTINUATION OF D-STATION	X,E,M		
9)**	STARGAZER I	MAPS CONSTELLATIONS	E,T	$10	____
10)**	STARGAZER 1,2,3	MAPS OUT DOZENS OF STARS	E,T	$10	____
11)	SCRABBLE	ASMB COLOR + 20,000 WORDS	E,T	$10	____
12)	MASS COPY 1 & 2	DISK COPIERS	X,E,M	$10	____
	DISKD	DISK SECTOR EDITOR	E,M		
	TEXD	TERMINAL EMULATOR	E		
13)	FAST TERM	BEST TERMINAL EMULATOR	E	$10	____
	DM1000V2.0	BEST DISK MANAGER	E		
	DM1000V1.0	DISK MANAGER	X		
	VDT	TERMINAL EMULATOR	E,T		
14)	SUPER DEBUGGER	ADVANCED DISSASEMBLER	E,M	$10	____
	TI-DIS-ASSM	TI's DISSASMBLER	E,M		
	MARTY's CATLSER	SEE OCT NEWSLETTER	E		
	MARTY'S DISSAM	BY MARTY KROLL JR	E		
	TICK	INTERRUPT DRIVEN CLOCK	X		
15)	MASS TRANSFER	ASSM FILE TRANSFER UTILITY	E	$10	____
	DISKD2	DISK SECTOR EDITOR	E		
	COMPACTOR	COMPACTS DIS/FIX 80 FILES	E		
	UNCOMPACTOR	UNCOMPACTS DIS/FIX 80 FILES	E		
	DISSASEMBER	YET ANOTHER DISSASMBLER	E		
16)	The disk manager	Good DM program	E	$10	____
	DM1000v3	Updated version	X,E		
	Fast-Term	Terminal Emulator	E		
	Fast-Docs	Docs for fast-term			
17)	Diagnostics1	Checks P-cards	M	$10	____
	Diagnostics2	Checks console	X		
	TI-runner edit	edit ti-run screens	X		
	Breakout	assembly game	E		
18)	File Transfer	Transfer Adventures	E	$10	____
	Clydes Loader	Load assembly in Xbasic	X		
	Spotshot	Assembly Game	E,X		
	Cubit	Cubert Assembly game	X		
19)	Funnel	Menu driven program that has a bunch of utilities like copiers,dms and others. Must have!	X	$10	____
20)	Printer disk	PRINTS PICTURES(3 DISK SET)	X & PRINTER	$10	____
21)**	OLD DARK CAVES &	Printed Manual	X	$10	____
22)	Mystery Disk #3	all assembly	E	$10	____
23)**	Road Race	Assembly Pole Position	X	$10	____
	Burger Builder	Assembly Burgertime clone	X		
24)	MYSTERY DISK	COMBO OF FORTH AND ASSM DSSD	X	$10	____
25)**	Midnight Mason	Assembly Game	X,E	$10	____

SAPPHIRE SOFTWARE
P.O. Box 18124
Pittsburgh, PA 15236

26)**	Micro Pinball	Assembly ->EXCELLENT!<-	X,E	$10	____
27)	Monopoloy	From Austrialia	E	$5	____
28)**	AsTIroids	Assembly+manual	E,X	$5	____
29)**	Arthropod	Assembly centipede clone	E,X	$5	____
30)**	Beyond Space	Assmb Game (Parsec+2)	E	$10	____
	Face Chase	Assembly Game	E		
31)**	Great Word Race	Assembly	E	$10	____
	Star Trap in 3D	Fantastice 3D space game	E		
32)	C-Language	With PRINTED manual	E	$10	____
33)	TI-Pilot	2-disks	E	$5	____
34)	Multiplan Tips	by Ted Andersen	Mp,Tw	$5	____
35)	Martys Disassblr	By Marty Kroll Jr.	E	$5	____
36)	Martys Cataloger	LATEST VERSION!!!!	E	$5	____
37)	Clydes Loader	Assembly into Xbasic	X	$5	____
38)	SuperBugII	By Edgar Dohman	E	$5	____
39)	Screen Dump	Uses Load interupt switch	E,X	$5	____
40)	Neatlist	Cross References Programs	X	$5	____
41)	Pr-BAse	Data-Base Program (DSSD)	X	$5	____
42)	Teckie BBS	set up your own bbs	X	$5	____
43)	TI-Writer Loaders	Editor-assembler version	E	$10	____
		Minimemory Verson	Minimem		
		Xbasic (includes Show Directory)			
44)	DOM Jan 1986	Music Programs	X	$5	____
45)	DOM FEB 1986	DONT MISS THIS ONE!!!	X,E,ToD	$5	____
46)	Dom March 1986	Miscellaneous Programs	X	$5	____
47)**	Spies Adventure	KBB vs CIA	X	$10	____
	Galactic BAttle	Inteplanentary war zone	X		
48)**	TI-TOad	Assembly Frogger Game	X,E	$10	____
49)	Jet Graphics	Utilitiyies (2-disks)	X	$5	____
50)	Universal Dissasembler	By Renee LeBlanc	Mini,E	$5	____
51)	Enhanced FOrth	By Renee LeBlanc	E	$5	____
52)	Mystery Disk #2	Games,Utilities ?????	E,X,M	$5	____
53)	DOM 4/86	ALL ASSEMBLY!!!! STEAL!!	E,T	$5	____
54)	New Membership			$17	____
55)	Membership Renewal			$12	____
56)	Newsletter Subscription only			$8	____

subtotal of order items----------->$ .00

Amount enclosed ----------->$ .00

FREE MYSTERY DISK #4 with every $35 purchase.(Assembly!)

Please print neatly.

Name _____ Date / /86

Address _____ PH( ) -

cITy _____ State _____ Zip _____

Orders processed as soon as possible, usually within 3 days. If you get a bac copy OF ANY PROGRAM, CALL ME!!! DON'T POUT, CALL CALL!

Mail to Sapphire Software
P.O. BOx 18124
Pgh PA 15236

** ALL ITEMS EXCEPT THOSE WITH AN ** AFTER THEM MAY BE ORDER BY ANYBODY, ANWHERE.

Thank you for your order!

Well you know that everyone should plan ahead, and I thought I did. But here I am on the last page and I still have some important things to tell you. FIRST we will have some of the cassettes and disks at the next meeting, initially these will deal with such things as TEACH YOURSELF BASIC and EXTENDED BASIC. Clyde Colledge and others have been working very hard to get together the programs you will have access to. Some very good public domain, some exceptional freeware programs, and we hope soon to negotiate some software purchases "FOR MEMBERS ONLY". Remember Chuck Strink and myself can help any of you who need a particular program, in the meantime. If you would prefer to write, my address is on the bottom of the article on memory.

DISK DRIVES
(from the A9CUG CALL newsletter, by their editor MARSHAL)

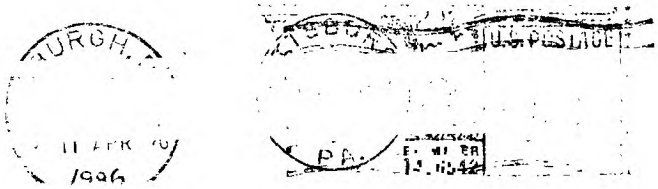
MANUFACTURER:	MODEL:	HEIGHT:	SIDES:	POWER REQ:
MPI	51	FULL	SINGLE	FULL
MPI	52	FULL	DOUBLE	FULL
SHUGART	SA-400L	FULL	SINGLE	FULL
SHUGART	SA-405L	FULL	DOUBLE	FULL
SHUGART	SA455-2	HALF	DOUBLE	HALF
TEAC	FD-FFA	HALF	SINGLE	HALF
TEAC	FD-55B	HALF	DOUBLE	HALF
TANDON	TM100-1	FULL	SINGLE	FULL
TANDON	TM55-1	HALF	SINGLE	FULL
TANDON	TM100-2	FULL	DOUBLE	FULL
TANDON	TM55-2	HALF	DOUBLE	FULL
Control Data Corp.	9409	FULL	DOUBLE	FULL
Control Data Corp.	9428	HALF	DOUBLE	FULL
QUME	142	HALF	DOUBLE	FULL
PANASONIC/MATZUSHITA	JA551-2N	HALF	DOUBLE	FULL
MITSHUBISHI	4851	HALF	DOUBLE	FULL
TOSHIBA	ND04D	HALF	DOUBLE	HALF

NOTE: WHERE THE DRIVE IS HALF HEIGHT AND THE POWER IS ALSO HALF, TWO DRIVES CAN "GENERALLY" BE USED IN PLACE OF ONE OF THE FULL HEIGHT DRIVE.

THERE ARE MORE, THAN THESE OF COURSE, BUT IN GENERAL THOSE THAT ARE COMPATIBLE WITH APPLE, COMODORE AND ATARI, WILL NOT WORK ON THE TI-99/4A. THOSE THAT WILL WORK ON THE P.C.'S, ARE GOOD. (EXCEPTION WOULD BE THOSE THAT ARE 96 T.P.I., OR SAY 80 TRACK DRIVES).

GOOD HUNTING!!!!! REWRITTEN AND ADDED TO BY JOHN WILLFORTH WPUG.

WPUG



JOHN WILLFORTH
R.D.#1 BOX 73A
JEANNETTE, PA
15644

WEST PENN 99ERS