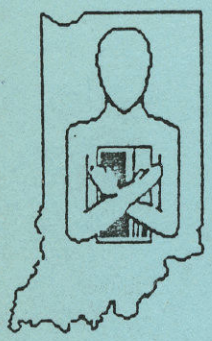
BEST OF

THE



HOOSIER USERS GROUP HOOSIER USERS GROUP HOOSIER USERS GROUP HOOSIER USERS GROUP HOOSIER USERS GROUP
USERS GROUP HOOSIER USERS GROUP HOOSIER USERS GROUP HOOSIER USERS GROUP HOOSIER USERS GROUP HOOSIER
GROUP HOOSIER USERS GROUP HOOSIER USERS
HOOSIER USERS GROUP THE HUGgers HOOSIER USERS GROUP
USERS GROUP HOOSIER People Helping People USERS GROUP HOOSIER
GROUP HOOSIER USERS GROUP HOOSIER USERS
HOOSIER USERS GROUP HOOSIER USERS GROUP
USERS GROUP HOOSIER USERS GROUP HOOSIER USERS GROUP HOOSIER USERS GROUP HOOSIER USERS GROUP HOOSIER
GROUP HOOSIER USERS GROUP HOOSIER USERS GROUP HOOSIER USERS GROUP HOOSIER USERS GROUP HOOSIER USERS

NEWSLETTERS

VOL I

TI-WRITER TUTORIAL
By Tom Kennedy

Now I want to cover the Text Formatter, which prints out the document. Most importantly, the special symbols, called Format Commands, that the formatter uses to alter the print-out of the document, which are installed in the Text Editor.

In other words, you put these commands into the text when you write it and as the formatter comes across them it changes the text accordingly but doesn't actually print the symbols.

There are six groups of formatter commands that are all applied in a similar manner. All commands must be in caps and must be on a line that starts with a period.

****

Text Dimension commands, as the name implies, move or shape the words in the document (margins, linespacing, right justify, etc.)

.FI	FILL	PUTS AS MANY WORDS ON A LINE AS WILL FIT.
.NF	NO FILL	CANCELS FILL.
.AD	ADJUST	ALIGNS THE TEXT TO THE LEFT AND RIGHT MARGINS. (RT. JUSTIFY
.NA	NO ADJUST	CANCELS ADJUST.
.LM n	LF MARGIN	SETS LEFT MARGIN TO "n".
.RM n	RT MARGIN	SETS RIGHT MARGIN TO "n".
.IN n	INDENT	CREATES AN AUTO-INDENT FROM LEFT MARGIN.
.LS n	LINE SP	SETS LINE SPACING TO "n" LINES.
.PL n	PG LENGTH	DEFINES NUMBER OF LINES TO A PAGE.
.BP	BEGIN PG	DEFINES FIRST LINE OF NEW PAGE.

Internal Format commands control the spacing of characters on a line.

.SP n	SPACE	SIMILAR TO THE TAB FUNCTION.
.CE n	CENTER	CENTERS NEXT "n" LINES BETWEEN MARGINS.

Highlighting commands control functions such as underline or bold and allow you to redefine characters to use them to send CTRL codes to the printer.

^	REQUIRED SPACE	JOINS WORDS TOGETHER WHEN REQUIRED TO PREVENT SPLITTING IN REFORMATING, UNDERLINE, ETC.
&	UNDERLINE	(UNDERSCORE) UNDERLINES ALL TEXT FOLLOWING UNTIL NEXT PACE.
@	BOLD	(OVERSTRIKE) RETYPES FOLLOWING TEXT FOUR TIMES.
.TL xx	TRANS-LITERATE	ALLOWS REASSIGNMENT OF ONE CHARACTER TO REPRESENT A NUMBER. OF CHARACTER VALUES TO SEND CODES TO THE PRINTER.
.CO t	COMMENT	SIMILAR TO REM IN BASIC--ALLOWS NOTES THAT DONT PRINT.

Page identification commands print notes in the upper or lower corner of each page, either headers or footers.

.HE t	HEADER	PRINTS TEXT (t) AND PAGE NUMBER AT TOP OF EACH PAGE.
.FO t	FOOTER	PRINTS TEXT (t) AND PAGE NUMBER AT BOTTOM OF EACH PAGE.
.PA	PAGE #	RESETS PAGE NUMBER IN .HE AND .FO

File management commands

.IF f	INCLUDE FILE	MERGES A FILE TO PRINT A DOCUMENT TOO LARGE FOR ONE FILE.

Mail Merge option commands are used to supply values to the variables in a letter that has been set up for the mail merge option

```
 .ML f   :MAIL LIST: IDENTIFIES VALUE FILE (f) FOR MAIL LIST.
 *n*     :VARIABLE : INSERTED IN TEXT AS VARIABLE FOR ASSIGNMENT FROM VALUE FILE.
 .DP n:t:DISPLAY   : PROMPTS YOU USING TEXT "t" TO ASSIGN TO VARIABLE (*n*).
         :  PROMPT :
**********************************************************************
```

The use of these commands in your text is what separates the word processor from a typewriter. They allow you to get the most out of your printer.

So, now you've written your document, and inserted all the format commands, now how do you print it out? First, save the document and exit the Text Editor. At the title menu, select Text formatter, (make sure the program disk is in the drive) and the screen will blank with the prompt "ENTER INPUT FILENAME". Enter the name of the file you just saved, (ex. DSK1.MYFILE) and hit enter.

Next, the prompt "ENTER PRINT DEVICENAME" appears after the file is loaded. If you use a serial printer, the device name would be RS232.BA=xxx with xxx being the baud rate. If you're using a parallel printer, the device name is PIO. Also, you must add either .CR or .LF to the end of the device name. This tells TI-Writer whether you're printer will handle the carriage return or the line feed. Check you're printer manual and the TI-Writer manual in detail to find out which you use.

The next prompt is "USE MAILING LIST". If you aren't printing "form letters" just hit enter to accept the default of N (NO).

Next is "WHAT PAGE(S)? (ALL). If you want to print the whole document, accept the default for all pages. Otherwise, you can print any of the pages or groups of pages.

The prompt "NUMBER OF COPIES: 1" tells how many copies of each page are to be printed.

The last prompt is "PAUSE AT END OF PAGE? N". The main purpose of this function is if you are using separate sheets of paper it will stop and wait for you to align the next sheet. Another use is to save a little paper. TI-Writer has an annoying habit of scrolling one whole blank page up before starting to print, which is not that big of a deal since what's one piece of paper worth considering how much you go through normally. But if you're just running test samples of type styles, or the like, you end up with a lot of white paper at you're feet. To prevent this, type "Y" and turn off you're printer. Now hit enter and turn the printer on, you should see "PRESS ENTER TO CONTINUE" (the software thinks one page has been printed). If not, turn the printer on and off again. Now you align the paper to the top of the page and hit enter and the printing begins. But if it's a long letter, you'll have to sit there and hit enter after each page so usually it's better to select the default when using continuous feed paper.

Now, about the Mailing List Option. Let's say you've written a form letter to send out to various individuals, maybe a resume'. You write the letter like normal, but when you come to a name or address or something that will change with each letter, you put in it's place a variable in the form of *n*, where n is a number to identify the order. So instead of starting off with:"Dear Mr. Smith" you would have "Dear Mr. ^1*" and so on. when you're all through with your letter, save it and purge the memory. Now you must create what is called a Value File, which is your mailing list where TI-Writer will draw the variables from. A value file consists of a list values to be inserted into the letter, listed one to a line, preceded by the number of the variable and ending with a

carriage return symbol. Groups of values must be separated by a line with just an asterisk and a carriage return. For example:

1 John Smith<cr>
2 123 STREET<cr>
3 Seattle, WA<cr>
*<cr>
1 Jane Doe<cr>
2 456 STREET<cr>
3 Seattle, WA<cr>

At the top of your letter you insert the .ML f command where f equals the filename of your value file. After selecting the mailing list option the computer will use this command to fill in the variables. If there is no .ML command in the letter then when you are prompted for "MAILING LIST NAME:" you supply the filename. This allows you to call on a number of files for different groups.

Another way to insert values is to use the Define Prompt command. With this command you do not insert a .ML comand calling a value file and instead you insert lines containing the format: .DP n:t - where n is the number of the variable and t is the prompt text. Now, when you come to the prompt "USE MAILING LIST?" you select "N" for NO and as the document is printed when a variable is encountered the printing stops and the text you chose appears on the screen asking you for the appropriate value. If you don't include a ".DP n:t" command in your text, the computer responds with "ENTER DATA FOR VARIABLE *n*" and it can get confusing trying to remember which item you're on. This method is handy for letters which you only want to print one copy at different times to different people.

Let me tell you, this is why I bought a computer. I'm sure we all went through that period of time before buying a computer when we would ask: "what am I going to use a computer for, anyway?". Well I decided there were two things I wanted to do: 1) Store files of data (recipes, albums, Etc.) and 2) Use my computer as a typewriter. I didn't know about TI-WRITER when I bought the 99/4A, but now I know that I made the best choice possible. I hope you will all find TI-WRITER as easy to use and as powerful as I have.

TI—WRITER TIPS

Q: Can TI-WRITER save a file in any format besides D/V80?

A: Yes, if you use the PF command to print a file, you can insert a "F" in front of the filename, as in: F DSK1.MYFILE. The F will cause the file to print in Display/Fixed 80 format.

Q: How do Transliterate commands work?

A: The Transliterate command is a special type of Format command that redefines any ASCII key value to equate to a string of character values. This is used to send specific code values to a printer in order to activate special fuctions. The format is....

".TL xxx:aaa,bb,ccc"

where xxx is the key to be redefined, and aa,bb,cc, etc are the subsequent code values being sent. You will have to check your printer manual to see which codes do what.

THINGS TI NEVER TOLD YOU!

by Jim Ellis

While trying to find out why the copyright symbol loses the upper half when TI-Writer (tm) calls the CHARA1 program, I found out where they hid the definition for the cursor. It can be changed to anything that you can fit into the 8 by 8 grid that defines letters, numbers, sprites, etc. that you can display on the TI screen. The cursor is defined by 16 bytes just like the definition used in Basic's CALL CHAR statement. You supply the 16 bytes that describe what you want your cursor to look like. I have mine with my initials. This looks similar to the way CR appears on the screen when you press ENTER. That makes your copy more personal. Now I shall tell you where to change these 16 bytes to customize your program disk. You must have Disk Fixer (tm) or some similar program to do this. The figure below is of the 256 bytes that comprise the first sector of CHARA1. You can copy the CHARA1 file to a freshly initiailzed disk, thay way sector hex 22 or 34 decimal will contain the sector we are interested in modifying. The sector appears below wih the symbol ** in the 16 bytes of interest.

```
      ADR- 0 1 2 3 4 5 6 7 8 9 A B
      ----------------------------------------
      00- 00 00 08 00 07 FA 00 20 00 00 18 24
      0C- 24 18 00 20 00 08 18 08 08 1C 00 20
      18- 00 18 24 08 10 3C 00 20 00 18 24 08
      24- 24 18 00 20 00 14 14 1C 04 04 00 20
      30- 00 1C 10 18 04 18 00 20 00 08 10 38
      3C- 24 18 00 20 00 1C 04 08 10 10 00 20
      48- 00 18 24 18 24 18 00 20 00 18 24 1C
      54- 04 08 20 20 38 00 1C 10 1C 10 00 40
      60- 00 20 20 38 24 38 00 70 50 70 48 54
      6C- 1C 14 00 70 40 70 00 1C 10 10 00 20
      78- 00 18 24 3C 20 18 00 40 08 14 10 1C
      84- 10 10 00 40 40 40 18 24 24 18 00 20
      90- 20 20 28 08 08 08 00 40 40 58 24 08
      9C- 10 3C 00 40 40 58 24 08 24 18 00 40
      A8- 40 54 14 1C 04 04 00 40 40 5C 10 18
      B4- 04 18 00 40 40 48 10 38 24 18 00 40
      C0- 40 5C 04 08 10 10 00 40 40 58 24 18
      CC- 24 18 00 40 40 58 24 1C 04 08 00 40
      D8- 40 40 18 24 3C 24 00 40 40 50 10 1C
      E4- 14 1C 00 40 40 40 1C 10 10 1C 00 40
      F0- 44 44 04 1C 14 1C 00 ** ** ** ** **
      FC- ** ** ** 40
```

You may change the displayed cursor to your choice by inputting your own code as per the character definition of CALL CHAR located in the TI Basic or Extended Basic manuals. Use only the character definition, do not include the character number. When you have completed making your changes, save the sector back to disk. Now you are ready to copy your version of CHARA1 back to your original disk. The new cursor will appear in both the editor mode and the format mode of TI-Writer. (tm)

LIBRARY BITS

by Dennis Sherfy

Last month I wrote about PROW-CHART on BASIC-11. This month, I will show you how you can use the table printed by this program with the TRANSLITERATION feature of TI-WRITER.

The first thing you need to know is, why do You need TRANSLITERATION?. The Quick Reference Card which came with your TI-WRITER package list ASCII codes 32 to 127. But what about 1 to 31? And how do you get to these special characters above 127? This is where TRANSLITERATION comes in.

TRANSLITERATION is merely a long word meaning to change one key to another. If, while using TI-WRITIER, I type, .TL 65:66, this will cause my printer to print a capital B (ASCII 66) each time capital A (ASCII 65) is pressed. I have simply changed A into B. Obviously, no one would want to do this. But what if I want to enter the ELONGATED character mode on the PROWRITER? Transliteration allows me to do this. To select Elongated type, my printer requires CHR$(14), or ASCII 14. Lets go back to my original example. If I enter .TL 65:14, I will be telling my printer to convert to Elongated type each time capital A is typed. In order to get out of elongate type, my printer requires CHR$(15), or ASCII 15. I could enter .TL 66:15. This would tell my printer to convert to Elongated type when it sees an "A", then revert back to the normal type style when it encounters a "B". Now, let's get realistic. I need to use "A" and "B" frequently, so I should select two other keys that are not used so often. In my case, I do not use {-(123), }-(125), |-(124), \-(92), or `-(96) often. I could Transliterate (change) them to another character. If I enter .TL 123:14 and .TL 125:15, I will designate { and } to start and stop Elongated type. By typing Hoosier {User's} Group, the word "User's" would be in Elongated type. Transliteration is more powerful than this example. You can change a single key to represent several keys. .TL 124:27,84,51,50 will cause your Prowriter to shift to 1-1/2 line spacing when "|" is pressed. In this case "|" equals ASCII 27 (Escape), ASCII 84 (select custom line spacing), ASCII 51 (3), and ASCII 50 (2). This tells your Prowriter to change to line spacing of 32/144 inch.

To use the Greek characters or the graphic characters you must use transliteration to enter the special character mode, such as .TL 124:27,35. Then use transliteration to print each character you want.

There are two additional things you need to know to use transliteration. First, it only works with the FORMATTER. Second, you must put only one TL command on a line, followed by a carriage return.

The PRO-CHART will show you each of the available characters and tell you which ASCII number to use with your transliterastion commands.

Experiment.....and enjoy your printer and computer.

TI WRITER SPECIAL CHARACTER MODE by Dennis Sherfy

In the January '86 issue of our newsletter, I wrote an article about Transliteration. The companion to Transliteration is the Special Character Mode.

I couldn't understand the Special Character Mode from the instruction manual. One of the major benefits of user group membership is the help you can receive from fellow members. I am merely passing on what I've learned from others.

You can do many of the same things with either Transliteration or Special Character Mode. The logical question is, why have two methods? The answer is that there are benefits to each method. You should know both, and use the one that is most convenient for each application.

ADVANTAGES OF TRANSLITERATION:
-A series of keystrokes can be incorporated into a single keystroke.
-When you re-use a function several times, Transliteration will save time.
-You can use special printer characters, such as Greek and mathematical symbols which the Prowriter can produce. Probably other printers have similar capabilities. I have not been able to access these characters with Special Character Mode.

ADVANTAGES OF SPECIAL CHARACTER MODE:
-Keys are not tied up when you want to pass instructions to your printer.
-You may access your printer's printing options from either the Editor or the Formatter.

If you read our January newsletter, you know how to use Transliteration. Here is how you use Special Character Mode.

Special Character Mode is selected by pressing CTRL U. When you want to return to regular characters, press CTRL U again. When you enter the Special Character Mode, the cursor will change to the underscore character. Special characters which can be used in this mode are listed on page 146 of your TI WRITER manual. If you want to begin a new page, press CTRL U, SHIFT L, CTRL U. If you want to select elongated type on the Prowriter, you use shift out, which is a special character. It can be accessed by pressing CTRL U, SHIFT N, CTRL U. Many of my printer's features require "escape" plus another key. Escape is a special character. If I want to change to elite type, I use "escape" plus "E". Using Special Characteer Mode its CTRL U, FCTN R, CTRL U, E.

There are some commands that require numerical input as well as escape and a letter. An example is custom line spacing. My Prowriter will print with a line space of nnn/144 inch. If I want 1/2 inch line spaces, I enter 72/144. My printer manual tells me to enter "escape" plus "T", plus up to three numbers. I obtain 1/2 inch line spaces with CTRL U, FCTN R, CTRL U, T, 7, 2. (Commas are used to show separate keystrokes -- you do not use commas when entering Special Character codes)

Experiment with your printer. Determine which method works best for you. Something that I found to be usefull was a practice file. I typed several instructions with Special Character Mode, and described what result I should obtain, such as "This should print in elite type." When I got it to work, I tried another command. Finally, I saved my file so I could re-read it later when I needed to refresh my memory on how something was done.

Transliteration AND Special Character Mode can make printing more enjoyable. They're both worth learning.

TI Writer: New character set with control U

By Dennis Wood

From the day I started using TI-Writer I longed for a better character set and an easier way of controlling my printer's special functions. The TI-Writer upgrade gave us lower case characters, but at the expense of smaller, harder to read characters. What was needed was lowercase, but as large as possible, consistant with the 6x8 pixel size available. With this in mind, I designed a new set of upper and lower case characters. In order to get decenders for the lowercase characters and still retain the uppercase size, I merely shifted the uppercase characters up one raster line. Since all 24 lines of text are all shifted up by the same amount, there is no difference on the screen. As TI-Writer loads the CHARA1 file the uppercase characters shift up one raster line. By doing this I could now define lowercase characters with a one line decender. The final result is uppercase characters defined in a 5x7 matrix and lowercase characters defined in a 5x5 matrix with an additional row below for decenders.

While doing this I had to learn how to read, decipher and change the CHARA1 file. As a result, I also discovered how the ASCII characters from 1 to 31 were defined. For example, the little cr at the end of each paragraph is represented by the ASCII 13 character. The other characters are shown on page 146 of the TI-Writer manual. After reading William Bullock's fine article in the September 1985 MICROPendium I realized how to get to the objective I sought. What was needed was a set of characters that would indicate what action was desired of the printer, and a Transliteration file that would take that character and transliterate it into the desired printer codes. I took the ASCII codes from 1 to 31 and defined a set of characters that would represent desired printer actions, such as Underline or Bold or Italic. To insert these characters in a document it is only necessary to type CTRL U followed by a specific SHIFT or FCTN and a number or letter. When TI-Writer encounters this combination in a text file it sends that ASCII code to the printer. In order to get the printer to execute the desired function, it is necessary to transliterate that ASCII code into the sequence the particular brand of printer requires. There does not seem to be a standard set of codes, but most printers seem to be pretty close.

The following is a description of the special characters, the function they represent, what key to strike, and a transliteration file for a C. Itoh Prowriter Jr. Note: TI-Writer Transliteration software does not pass certain ASCII codes. These are:

These keystrokes are preceded by a CTRL U and followed by a CTRL U.

FUNCTION	CHARACTER	KEYSTROKE	TRANSLITERATION
NLQ On	Large L	FCTN R	TL 27:27,120,1
NLQ Off	Inverse L	Shift R	TL 18:27,120,0
Italic On	Large I	FCTN T	TL 29:27,52
Italic Off	Inverse I	Shift T	TL 20:27,53
Underline On	Large U	FCTN U	TL 31:27,45,1
Underline Off	Inverse U	Shift U	TL 21:27,45,0
Superscript On	Up Arrow	Shift E	TL 5:27,83,0
Subscript On	Down Arrow	Shift X	TL 24:27,83,1
Either Off	Inverse Arrow	Shift D	TL 4:27,84
8cpi	Small 08	Shift Y	TL 25:(not Itoh)
Pica (10cpi)	Small 10	Shift A	TL 1:27,80
Elite (12cpi)	Small 12	Shift I	TL 9:27,77

This article is reprinted from the Puget Sound 99ers of Lynnwood, Washington

Condensed On	Large C	Shift H	TL 8:15
Condensed Off	Inverse C	Shift K	TL 11:18
Expand On	Large X	Shift F	TL 6:27,87,1
Expand Off	Inverse X	Shift G	TL 7:27,87,2
Proportional On	Large P	Shift P	TL 16:27,112,1
Proportional Off	Inverse P	Shift O	TL 15:27,112,Ø
Bold On	Large B	Shift B	TL 2:27,71
Bold Off	Inverse B	Shift V	TL 22:27,72
Emphasized On	Large E	FCTN Z	TL 28:27,69
Emphasized Off	Inverse E	Shift Z	TL 26:27,7Ø
Backspace	Back Arrow	Shift S	TL 19:8
8 lpi	Small L8	Shift 2	TL Ø:27,48
7/72" line space	Small L9	Shift C	TL 3:27,49
6 lpi	Small L6	Shift Q	TL 17:27,5Ø
4 lpi	Small L4	Shift W	TL 23:27,65,18
3 lpi	Small L3	Shift N	TL 14:27,65,24

Reserved characters are:

Line Feed	Small LF	Shift J	TL 1Ø:
New Page	Small PA	Shift L	TL 12:
Carriage Return	Small CR	Shift M	TL 13:
Cursor	Rectangle	Shift 6	TL 3Ø:

A disk with the redefined upper and lowercase characters, and the above control characters as well instructions and examples for setting up an Include File (IF) for transliteration codes is available from the library or:

Dennis Wood
16709 SE 31st
Bellevue WA, 98008

Please send $5 to cover the cost of the disk, duplicating, and mailing.

TI-WRITER has been enhanced to provide true lower case letters, eliminate the form feed, and provides an RS232 default while using the Formatter. You will need to add the CHARA1 file to your system diskette and replace the EDIT1 and EDIT2 files. For printer default in the Formatter, replace the FORMA1 and FORMA2 files to get "RS232.BA=1200.LF". For "RS232.BA=4800.LF", replace the FORMA1 and FORMA2 files with the FORMA4800A and FORMA4800B files, while keeping the file names FORMA1 and FORMA2 respectively.

TI Writer Alternate Character Set

by Erwin von der Ehe

This article explains how to make your own character fonts for display on TI Writer. My interest developed because my daisy-wheel printer produces non-standard characters in place of some lesser used keyboard characters, while TI Writer displayed the character on the keyboard. At printing time I usually had surprises to edit out of my text. The procedure presented here allows redefinition of any character normally displayed by TI Writer. This technique can be used for the problem described above or for properly displaying other characters printed through the Transliterate command of the Formatter.

Requirements:

1) TI-99/4A, Disk System, Memory Expansion, TI Writer.
2)Editor/Assembler
3)DISKO disk editor, on MATUIG Disk #32
4) Updated TI-Writer Files, MATIUG Disk #71
5) Grid paper to help define characters - reference CHAR subprogram in your BASIC Reference Manual or a program such as Sprite Maker, MATIUG Disk #94

Procedure:

1) Make a back-up copy of your TI-Writer program diskette. Use this back-up for all further activities in this article. This is IMPORTANT!
2) Determine which character (ASCII 0 through 127) you want to redefine
3) Determine the character definition code for the new character. Note: In 40-column display mode, characters are 8 pixels high by 6 pixels wide. Use the left 6 columns of your 8x8 pixel character definition grid for each character, but define the character code based on the full 8x8 grid.
4) Boot DISKO, using Editor/Assembler option #3, Load and Run. File name = DSK1.DISKO. Program name = START.
5) Before we go too much farther, here's how to navigate in DISKO:

FCTN	Description
1	Display sector in HEX
2	Display sector in ASCII
3	Return to E/A from menu
4	Display previous sector
6	Display following sector
8	Rewrite disk sector to match screen
9	Return to DISKO menu
=	Quit - Go to Title Screen

6) After DISKO is running, put your back-up TI-Writer program diskette into DSK1. Use Option #2, Search for Existing File, to find the start sector of the file CHARA1, the character definition file. Note the starting sector number. Press <FCTN>9 for menu.
7) Select Option #1, Disk Sector Editor and select the starting sector of your CHARA1 file.
8) You will see the display labelled File Sector 1, shown at the top of this page. Note in the screens included with this article, I've inserted dots (•) between the character definition codes for your convenience in locating the character code you want to redefine. Also, I've added the ASCII number next to selected character codes. If you have the 9-sector version of CHARA1, don't worry that only 5 sectors are shown here: the others only contain 0's and serve no useful purpose that I can see.
9) Use the navigation functions to display the sector you wish to change. Use the arrow keys to put the cursor under the starting character of the code you want to redefine. Be sure to view sectors in HEX mode.
10) Type in the new code. Now is the time to review this screen to see if this is really what you want. If it isn't, type over incorrect code until it's right or return to the menu. If the code is correct, use <FCTN>8 to indicate your intention. Answer Y to rewrite the sector.
11) Boot up TI-Writer using the disk with your revised file and check your results.

Notes:

1) This procedure also works on the CHAR1 file of QS-WRITER®.
2) This procedure also works on the CHARA1 file of FUNLWRITER, MATIUUG Disk #112.
3) TI-Writer can only call CHARA1 character file, and it must be on DSK1. You may wish to keep several TI-Writer Program Diskettes, each with a different customized CHARA1.
4) Disk Fixer® or another disk sector editor can be used in stead of DISKO.

EJv
21 Apr 86

File Sector1: ASCII #:

```
0000080007FA•0020000018242418• 0
•0020000818080810•0020001824
18243C•0020001824082418•00200014
141C0404•0020001C10180418•0020
000810382418•0020001C04081010• 7
•0020001824182418•00200018241C
0408•20203801C101C10•00400020
20382438•0070507048541C14•0070
4070001C1010•0020001824 3C2018• 14
•00400814101C1010•004040401824
2418•0020202028080808•00404058
2408103C•0040405824082418•0040
4054141C0404•0040405C10180418• 21
•0040404810382418•004040 5C0408
1010•0040405824182418•00404058
241C0408•0040404018243C24•0040
4050101C141C•004040401C10101C• 28
•00404444041C141C•007070707070
7070•0040
File Sector2:
4C50101C1010•0000000000000000• 32
•0010101010001000•002828280000
0000•00287C28287C2800•00385430
18543800•00444C1830644400•0020
502054483400•0008102000000000• 39
•0008101010100800•002010101010
2000•0044287C28440000•0010107C
10100000•0000000000301020•0000
007C00000000•0000000000303000• 46
•0004081020400000•003C4C546444
3800•0010301010103800•00384408
10207C00•0038441804443800•0008
1828487C0800•0078407804443800• 53
•0038407844443800•007C04081020
2000•0038443844443800•00384444
3C047800•0000303000303000•0000
303000301020•0000102040201000• 60
•0000007C007C0000•000010080408
1000•0038
File Sector3:
440810001000•0038445458403C00• 64
•00384444 7C444400•007844784444
7800•0038444040443800•00784444
44447800•007C407840407C00•007C
407840404000•0038444044443800• 71
•0044447C44444400•003810101010
3800•0004040404443800•00444850
70484400•0040404040407C00•0044
6C5444444400•0044645454 4C4400• 78
•007C444444447C00•007844447840
4000•0038444454 4C3C00•00784444
78484400•0038443008443800•007C
101010101000•0044444444443800• 85
•0044444444281000•004444445454
2800•0044281010284400•00444428
10101000•007C081020407C00•0038
202020203800•0000402010080400• 92
•0038080808083800•001028440000
0000•0000
File Sector4:
00000000 7C00•0020100800000000• 96
•0000003848483C00•002020382424
3800•0000001C20201C00•0040041C
24241C00•0000001C28301C00•000C
103810101000•0000001C241C0438• 103
•0020203824242400•001000301010
3800•0008000808084830•00202024
38282400•0030101010103800•0000
007854545400•0000003824242400• 110
•0000001824241800•000000382438
2020•0000001C241C0404•00000028
34202000•0000001C300C3800•0010
103810100C00•0000002424241C00• 117
•0000004428281000•000000445454
2800•0000002418182400•00000024
241C0438•0000003C08103C00•000C
10102010100C•0010101000101010• 124
•0060101008101060•000020540800
0000•0000 127
File Sector5:(Partial)
000000000000•0000000000000000
00000000000000000000000000000000
etc.
```

This article is reprinted via the May, 1986 issue of HOCUS, Newsletter of the Milwaukee Area 99/4 Users Group of Wauwatosa, Wisconsin.

A HANDY DANDY TI-WRITER USERS REFERENCE GUIDE

SUBMITTED BY BOB STEPHENS

The following handy TI-WRITER commands are reprinted for the June issue of the 99'er News published by the TI Users Group of Will County, Romeoville, Il. This puts the most used commands on one page for handy access at your computer.

EDITOR COMMAND	FCTN	CTRL	EDITOR COMMAND	FCTN	CRTL	EDITOR COMMAND	FCTN	CTRL
Back tab		T	Ins. Blank line	8	O	Quit	=	
Beginning/line		V	Insert character	2	G	Reformat		2orR
Command/escape	9	C	Last paragrapph		6orH	Right arrow	D	D
Delete character	1	F	Left arrow	S	S	Roll down	4	A
Del. end of line		K	Left margin rel.		Y	Roll up	6	B
Delete line	3	N	New page		9orP	Screen color		3
Line #'s(on/off)	0		New paragraph		8orM	Tab	7	I
Down arrow	X	A	Next paragraph		4OrJ	Up arrow	E	E
Duplicate line		S	Next window	5		Word tab		7orW
Home cursor		L	Oops!		1orZ	Word wrap/fixed		O

Load files: LF (enter) DSK1.FILENAME (load entire file)
LF (enter) 3 DSK1.FILENAME (merges filename with data in memory after line 3)
LF (enter) 3 1 10 DSK1.FILENAME (lines 1 thru 10 of filename are merged after line 3 in memory)
LF (enter) 1 10 DSK1.FILENAME (loads lines 1 thru 10 of filename)

Save files: SF (enter) DSK1.FILENAME (save entire file)
SF (enter) 1 10 DSK1.FILENAME (save lines 1 thru 10)

Print Files: PF (enter) PIO (prints control characters and line numbers)
PF (enter) C PIO (prints with no control characters)
PF (enter) L PIO (prints 74 characters with line numbers)
PF (enter) F PIO (prints fixed 80 format)
PF (enter) 1 10 PIO (prints lines 1 thru 10)
NOTE: The above assumes PIO. DSK1.FILENAME, and RS232 are also valid!
To cancel the print command press FCTN 4.

Delete file: DF (enter) DSK1.FILENAME

Setting Margins and Tabs: (16 tabs maximum)
L - Left margin R - Right margin I - Indent T - Tab
Use ENTER to execute or COMMAND/ESCAPE to terminate command.

Recover Edit: RE (enter) Y or N

Line move: M (enter) 2 6 10 (moves lines 2 thru 6 after line 10)
M (enter) 2 2 10 (moves line 2 after line 10)

Copy: same as move except use C instead of M.

Find String: FS (enter) /string/ (will look for string in entire file)
FS (enter) 1 15 /string/ (will look for string in lines 2 thru 15)

Delete: D (enter) 10 15 (deletes lines 10 thru 15 in memory)

OOPS!	CTRL 1 (CTRL Z)	This can be a real lifesaver. It recovers, or "backs up" a function that you didn't mean to hit. Like if you goofed and hit "Delete Line" instead of "Insert Character", you just hit "OOPS!" and the line comes back.
Del Char	FCTN 1 (CTRL F)	This is the same as "DEL" in console BASIC. It deletes one character under the cursor and pulls the rest of the line up to fill.
Reformat	CTRL 2 (CTRL R)	This is used to close up the text after using Insert Character. It deletes all spaces between the cursor and the next word in the text. Then it draws all subsequent words up through the paragraph until it encounters a Carriage Return.
Ins Char	FCTN 2 (CTRL G)	In the Word Wrap mode (solid cursor), thirty two blank characters are inserted after the cursor and the bulk of the text is pushed down the line. After insertion of new text, you hit Reformat and any remaining spaces are removed. In the Fixed mode (hollow cursor), this operates the same as in console BASIC.
Screen Color	CTRL 3	This allows you to chose which of the five color combinations of text/screen you prefer. The default, for good reason, is white on dark blue. I find this hard on the eyes. I prefer to turn down the color on my monitor and use either black on green or black on light blue.
Del Line	FCTN 3 (CTRL N)	Deletes the entire line that the cursor is on, including the space of the line.
Next Paragraph	CTRL 4 (CTRL J)	This advances the cursor to the beginning of the following paragraph and puts the first line at the top of the page.
Roll Down	FCTN 4	This is called a "vertical block scroll", which means that the next 24 lines of text are shown. This is handy for scanning quickly down the text to get to some point.
Dupe Line	CTRL 5	This creates an exact duplicate of the line the cursor is on and places it directly below. Some have questioned its value in writing text, especially since the Move/Copy function can do the same, but this key makes it faster and easier to create repetitive lines such as a double row of asterisks under a title.
Next Window	FCTN 5	This is a "horizontal block scroll". It jumps across to display the next block of 40 characters, in increments of 20. For example, the screen starts out on column one to forty, then twenty to sixty, then forty to eighty.
Last Paragraph	CTRL 6 (CTRL H)	The opposite of "Next Paragraph".
Roll Up	FCTN 6 (CTRL B)	The opposite of "Roll Down".
Word Tab	CTRL 7 (CTRL W)	This moves the cursor down the line to the first letter of each word.

Tab	* FCTN 7 *(CTRL I)*	Just like on a typewriter, this moves the cursor to the next setting, defined using the Tab function on the command line.
New Paragraph	* CTRL 8 *	This places a Carriage Return symbol at the end of the line you're on and skips down to the next line. If you have preset an auto-indent (by using an "I" in Tabs), then it also indents over to the proper column.
Ins Line	* FCTN 8 *(CTRL O)*	Inserts a blank line above the line the cursor is on.
New Page	* CTRL 9 *	Inserts a blank line with a Np and Cr symbol at the beginning. This causes the printer to feed to the next page.
Command/ Escape	* FCTN 9 *(CTRL C)*	This is how you exit from the edit mode to get to the command line and the functions above it. It also is used to cancel a command already in progress.
Word Wrap	* CTRL 0 *	This switches from the "Word Wrap" mode to the "Fixed" mode. In Word Wrap when you reach the end of the line the cursor jumps to the next line. If you're in the middle of a word at the end of the line, the whole word you were on moves down too. This allows you to just type continuously without looking up to see when to hit enter. In the fixed mode, when you reach the end of the line your letters just pile on top of each other and you hit enter to move to the next line.
Line Numbers	* FCTN 0 *	This removes or displays the four-digit line numbers at the left side of the screen. The numbers are used for reference when manipulating blocks or lines of text, just like when you're editing a BASIC program. You need line numbers to refer to where changes will be made.
Quit	* FCTN = *	Quit is the same as in console BASIC. Use Quit option of the command line to safely exit TI-Writer.
Back Tab	* CTRL T *	The same as Tab except it backs up one setting.
Beginning of Line	* CTRL V *	Moves the cursor to the beginning of the line you're on.
Del End of Line	* CTRL K *	This is just like Delete Character (FCTN 1), except it takes out everything to the right of the cursor.
Home Cursor	* CTRL L *	This moves the cursor to row 1, column 1, on the screen only. Unfortunately, it doesn't move to the first line of text, which would be more convenient when you were at the end of a long document and wanted to jump to the top.
Left Mrgn Release	* CTRL Y *	Allows you to temporarily back-arrow beyond the left margi when it has been set past zero.

(EDITOR'S NOTE: The following should be read by those who are already using FUNLWRITER and by those who might consider using it. This article answers some of the questions about FUNLWRITER that are not made completely clear in the FUNLWRITER documentation, such as: How does one calculate the K values in the FW LOAD program? How do you know if your ASSM2 file has already been modified to run out of FW? What does RESET do before it QUIT's? How should one save a source file in DF80 format to disk from the E/A editor? The author is a member of the Front Rangers User Group in Colorado. We taken his article from the July 86 issue of Micropendium and have added our own editorial notes.)

A REVIEW OF FUNLWRITER 3.3 by Joe Nuvolini 303-596-6938

PERFORMANCE.....A+
EASE OF USE.....A+
DOCUMENTATION...A
VALUE.....A+
FINAL GRADE.....A+

I recently received an exceptional Fairware disk from two gentlemen in the Hunter Valley 99 User Group in New South Wales, Australia. Their names are Will and Tony McGovern and the disk contained version 3.3 of FUNLWRITER which, according to an earlier letter I received from Tony, will be the definitive version of this program. (ED. NOTE: Not all copies of FW v3.3 are alike. The FW authors continue to make small updates. The date on the documentation can tell which of two FW v3.3 disks is the most recent.)

This is by far the most versatile program I have seen for the 99/4A. It allows you to use TI-Writer(TIW) and the Editor Assembler(E/A) without their respective modules. The program autoloads from Extended Basic(XB) and will also load from the TI-Writer, Editor Assembler, or Mini-Memory modules. The disk contains TI-Writer, the Editor Assembler, Disk Manager 1000 ver 3.1 (a FREEWARE program of the Ottawa User's Group), a sector editor, an a FORTH loader. You can also load your assembly programs without the use of any modules except Extended Basic. To run the disk you need the console, 32K memory, the disk controler and drive. It also helps to have a second disk drive and a printer.

There is a file called -READ ME- and six FWDOC files that should all be printed using the TIW Formatter and, more important, read before you begin. When you're done with that copy the FUNLWRITER disk so you have a working copy and put the back-up away in a safe place.

Before running the program examine the LOAD program by LISTing it. Line 120 allows you to set the primary and alternate screen colors. Lines 130 and 140 set the default options for the PF option of the Editor(130) and the Formatter(140). Lines 160 through 190 allow you to enter the names of programs you want on the User's List option while lines 240 through 280 are the load commands for these options. You can set a value for K in line 210 that will be the default for the drive number that appears on the screen with DSK. DO NOT RESequence the LOAD program or you will destroy it. The FWDOC/LOAD file explains how to set up the User's List options and the various methods of loading FUNLWRITER.

Now select a method and let's load the program. The first thing you'll see is the title screen followed by the first menu (ED. NOTE: If you get tired of the title screen, press any key to go immediately to the first menu) with three selections: 1-TI-WRITER, 2-EDIT/ASSM, and 3-USER'S LIST. We'll cover option three, User's list, later in this review. If you select option 1 or 2 you arrive at the central menu, which has 6 selections. They are 1-EDITOR, 2-FORMATTER or ASSEMBLER, 3-DM1000, 4-UTILITY, 5-SWITCH, and 6-RESET. Selecting SWITCH changes option 2 to ASSEMBLER, c-COMPILER and back to FORMATTER so that you can switch between these functions. (ED. NOTE: Later versions of FW v3.3 also cycle through MODEM (your favorite terminal emulation program), DISK EDITOR (DISKO, a public domain sector editor), and USER LIST with the SWITCH option. From this menu, USER LIST only accesses user selected assembly language programs, not XB programs.) I might mention here that the fles C99B through C99E will load REL2 of the c-Compiler by Clint Pully. It loads from this menu using files C99B through E. This is the preferred method of entry. It may also be loaded from the program file loader, discussed later, by entering C99C at the filename prompt. You must have the rest of Clint Pulley's small-c files for this option to be of any use to the user. Pressing RESET places the current filename you have been working on into the mailbox so that if you leave TIW or E/A and to to another FUNLWRITER function, say DM1000, and then return to TIW, when you select the EDITOR or FORMATTER that filename wil be there for you to load or print. After selecting RESET the option six name changes to QUIT and pressing that option returns you to the Master Title Screen. We'll discuss option 4, UTILITY after we finish our discussion of TIW and E/A.

NEXT PAGE PLEASE

If you select option 1-TI-WRITER from the first menu you then can select the EDITOR or FORMATTER from the central menu. The EDITOR functions like the TIW editor with the three following improvements:

1. If the loader can find a filename in the mailbox it writes it into the LF/PF buffer, which otherwise shows DSKx when called up with x being the default disk drive set in the LOAD program.
2. The quit function remains disabled at all times while in the EDITOR.
3. The show directory(SD) function is an assembly routine that allows single key pageing through the files. Fractured files are indicated by an asterisk after he file length.

The FORMATTER is the same as TIW with the following improvements:

1. There is now a Quick Directory(QD) function here from any menu in the program. To access it you enter FCTN 7 (AID). It operates in the same way as the SD function in the EDITOR.
2. The FORMATTER will automatically display the last file used when it can find one. If no name is present then DSKx appears.
3. The FCTN 9 (BACK) key allows you to return to the FUNLWRITER central menu.

If you need to reload either the EDITOR or FORMATTER immediately after exiting them they do not need to reload from disk.

If you select the EDITOR when ASSEMBLER or c-COMPILER is in the second position of the central menu, a modified version of the TIW EDITOR is loaded which is suitable for use as a source code editor. Word wrap is disabled, E/A tab defaults are set, and no final tab record is written to disk. To write a DF80 file to disk you use the PF option using F DSKx.FILENAME as described in the TIW manual. The source Editor loads CHARA2 which is slightly different than the CHARA1 file that is loaded by the TIW Editor. This acts as a reminder to let you know which editor you are in.

The ASSEMBLER has some enhancements added. The filenames are visible on the screen while it is executing. You can use AID to give you QD, allowing you to check the filename on the disk. If a filename is found in the mailbox it is written as the source code filename and te object code is the same name with the last two characters removed. Also, R is automatically entered in the Options field of the ASSEMBLER as a default value.

UTILITY option 4 on the central menu, brings you an assortment of assembly file loaders called the Program Load Environment(PLE). This menu displays five options on the screen, but has a total of 8 options, the last three of which are entered in the blind. Option 1 is for loading TIW utility files like Dragonslayer's Spellchecker. Option 2 sets up a GPL environment for loading other self-contained program image files; while option 3 is the E/A "RUN PROGRAM FILE" function. It should be noted here that the program file loaders will support cassette files by entering "CS1" (see E/A manual for more information on this function). Option 4 is the E/A "LOAD AND RUN" function and handles object files, compressed or not, and even displays the DEF table so you don't have to remember the program execution name if the program does not auto start. Option 5 is RE-ENTER (1-3) and it allows immediate re-entry to a program without reloading it, assuming it is re-enterable. The invisible options (6,7, 8) allow other object code loadin options, but in the interest of brevity I will not go into them here. Information on these options can be found in the FWDOC-EASM file. Entering FCTN 9 from this menu returns you to the central menu.

Now we'll discuss the USER'S LIST, option 3 on the first menu. This menu has 9 options. The first 8 options can be user defined although the LOAD program comes set to run the Myarc disk manager as option 6, the sector editor DISKO as option 7, and a TI-FORTH loader as option 8. Option 9 is BACK and it will return you to the FUNLWRITER title screen. This menu is set in the LOAD program, as are the loaders. You can run XB programs as well as E/A program or object files from the USER LIST menu if the corresponding files are placed on the FUNLWRITER disk. (ED. NOTE: the predefined options 6-8 can, if not needed, be replaced with your own options.) The XB programs are called by a RUN "DSKx.FILENAME" statement. The E/A files are loaded using a CALL LINK("UTILA",FILENAME,K). The numeric parameter K is the same as would be entered from the PLE discussed earlier, ie. 3 for an E/A program file and 4 for a "LOAD AND RUN" DF80 object code file. I find this part of the program particularly useful as you can put your favorite utility programs on the FUNLWRITER disk and have them available. In addition to TIW and E/A I have Masscopy, Fast Term, 4A/Talk, PRBASE, DM1000, DISKO, the Ti-FORTH loader, and a program called Recall all available through FUNLWRITER. I rarely take it out of drive #1. I should mention that I am using a double sided disk to hold all that. You will be somewhat restricted as to what you can put on a single sided disk with the FUNLWRITER files. (ED. NOTE: Single sided drive users might consider using FW version 3.2 since this version leaves more room than v3.3 for USER LIST files.

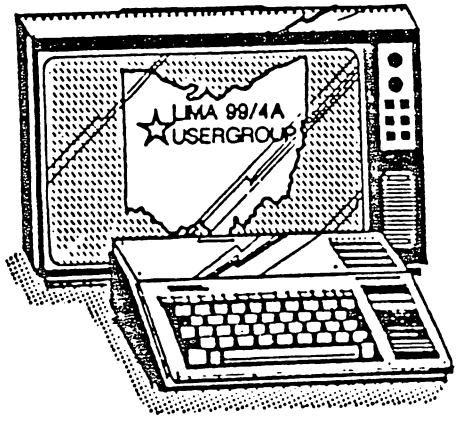
NEXT PAGE PLEASE

Bits, Bytes & Pixels

The main thing you loose with v3.2 is the ability to load FW from modules other than the XB module. You also loose the ability to do a QD from other places besides the EDITOR.)

There are several other files supplied with FUNLWRITER that deserve mention here. FWSAVE utility is for use with E/A to allow SAVING of any program loaded as an object file by FUNLWRITER into low memory. UPATCH is a patch that creates a file called UTIL1 once you have your screen color and printer defaults set in the LOAD program. UTIL1 is used to re-enter FUNLWRITER from several areas such as upon exiting DM1000. APATCH file is used to modify the ASSM2 file from your original E/A disk to work with FUNLWRITER. The ASSM2 file so created is 22 sectors long, 2 sectors longer than the original. It appears that the authors have already done this on the disk provided. FWRMM is for use with the Mini-Memory module to load the UTIL1 file into high memory.

A word here about Fairware... Will and Tony have set no price for this program but merely say "I can suggest only that you judge the program on its own intrinsic merits, perhaps measuring its worth by how much you use it as compared to other "fairware" or commerical programs that you use." I might suggest you do what our users group, the Front Rangers, did. We collected donations from the members of our users group who wanted the program and sent one international money order from the club to the authors. Be sure to include two disks when ordering your copy unless you have double sided capability as the DOC files are over 200 sectors long. Also be sure to enclose a couple of dollars postage as mail to and from Australia is not cheap! This is truely a fine piece o software. Let's make sure the authors are adequately compensated for their work. (ED. NOTE: The FW authors will NOT send FW directly to the United States unless a 'significant vote of confidence' is also included. Your disk and your postage paid return mailer are, by themselves, not enough. FW is available from user groups and from the Free Access Library 415-753-5581 at little or no cost.)



WRITE RIGHT
by Siles Bazerman

This time we are going to look at the TEXT EDITOR commands while in the Command Mode. Some of these have been covered before but a review seems appropriate at this time. Also, this will give you the complete list for reference. The initials and <ENTER> and other following material indicate what is to be typed.

Copy -Inserts a copy of line or consecutive lines in another spot in text buffer and renumbers lines, does not erase copied lines. C <ENTER>,start line NUMBER, stop line number, after LINE NUMBER,<ENTER>

Delete -deletes line or consecutive lines in text buffer. D <ENTER> first line number, last line number .

DeleteFile -Deletes file from disk. DF <ENTER> filename <ENTER>. (Once you use this the file is erased and cannot be recovered.)

Edit - Exits Command Mode and returns to Edit mode. E <ENTER>

FindString -Locates a word or phrase in the text buffer. FS <ENTER> /s/ where "s" is the word or phrase to be found <ENTER>

LoadFile (whole file) -Loads a file from disk into text buffer. LF <ENTER> DSKX.XXX <ENTER>

LoadFile (part of file)- Loads only part of a file into buffer. LF <ENTER> line number of first line to be loaded, space, line number of last line, space filename <ENTER>

LoadFile (merge whole file)- Merges a file on disk with contents of text buffer. LF <ENTER> line number in text buffer after which the file is to be merged, space file name of file on disk to be merged <ENTER>

LoadFile (merge part of file)-Merges part of file on disk with contents of text buffer. LF <ENTER>, line number in text buffer after which file is to be merged, space line number of first line of file to be merged, space, line number of last line to be merged, space, file name to be merged, <ENTER>

Move -Moves a line or block of consecutive lines from one point to another in the text buffer. M <ENTER> line number of first line to be moved, space, line number of last line to be moved, space, line number of line after which moved text is to be inserted <ENTER>. (Very useful for school papers as you can add whole sections later on at the end of the text and then place them where you

want instead of having to INSERT and REFORMAT.)

PrintFile -Prints the contents of text buffer. PF <ENTER>, device name (usually printer but can be disk), <ENTER>. To abort printing use FUNCTION 4. (This is the command to use for draft unformatted copy.) There are some additional PF choices; L, space, device name, (maximum line length is line number and 74 characters, columns 75 to 80 are not printed.); C, space, device name, (eliminates all control characters entered in Special Control Mode before file is printed.); F, space, device name, (Prints in FIXED 80 format.).

Purge -Clears text buffer (contents of buffer may be recovered by RecoverEdit). P,<ENTER>, Y for Yes or N for No, <ENTER>.

Quit -Exit program. Q <ENTER> S (save file), P (purge file), or E (exit Text Editor and return to main menu) <ENTER> For Save or Purge follow the prompts.

RecoverEdit - May recover all but first line of text buffer following Purge. RE <ENTER> Y or N (Yes or NO) <ENTER>.

ReplaceString -Replaces word or words with another word or words in text buffer. RS <ENTER> /string to be replaced/new string/ <ENTER>. Options in using this are: A (all) replace string in every subsequent instance; Y (yes) replace this string and find next instance; N (no) do not replace string in this instance but find next string; S (stop) escape this command. Will only search from where cursor is, to search whole text move cursor to first line of text. You may have to reformat after this command.

SaveFile (whole file)-Saves contents of buffer including Tabs to disk (also can be used to printer). SF <ENTER> filename <ENTER>

SaveFile (part of file)-Saves part of file to disk, tab settings saved only if saved part contains last line of buffer. SF <ENTER>, first line number to be saved, space, last line number to save, space, filename, <ENTER>

Showline - Locates line in text buffer by line number and locates it at top of screen. S <ENTER> line number <ENTER> (use 0 for line 0001, and E for last line).

ShowDirectory -Catalogs a disk on screen. SD<ENTER> disk drive number <ENTER>. To escape back to Command Mode press <ENTER> again.

Tabs -sets margins, tabs, paragraph indentations in text buffer . T <ENTER>; in appropriate column number type L for Left margin, I for paragraph Indent, T for Tab settings, and R for Right margin, blank out any unused prompts, <ENTER>.

This basically covers all the available commands. If you are not sure what to use then go to the Command Mode and look at the prompt line. Enter a choice and a new prompt line will appear, showing the commands for that section; enter a new command from the choices and a new prompt line will appear showing you what to enter for that command.

It's not very often that I get excited about a software program but I have to tell you about this one. How would you like a program that loads from Extended Basic, contains not only TI-WRITER but also EDITOR/ASSEMBLER, runs just about as fast as the cartridges, and more. We have one now.

This is a new "FREEWARE" program now in the group library called FUNLWRITER. It is a loader program in Extended Basic written by a group in Australia at Funnelweb Farm, and is an excellent addition to your software. It not only loads from menu Editor and Formatter, but under the Utility option loads both the Editor and Assembler,and also other utilities.

On the disk all the documentation and programs needed to fully support ALL commands and functions of both modules. Also on this disk is another FREEWARE program, Disk Manager 1000, Diskpatch (FREEWARE)a FORTH loader, and space for adding other options. Option 9 lets you load under four environments including "Run Program File"and "Load and RUN". This fills 358 sectors, or a complete SS/SD disk. It will run as written from any number drive (I don't know how, but it will). I have added in Fast-Term (FREEWARE), Dragonslayer Spellcheck (copyright), and one of Jim Swedlow's Label Programs on a DS/DD without filling it.

TK-WRITER REVISIONS

From the Jackson County 99'ers, come these tips on modifying Tom Knight's Extended Basic load program for TI-Writer.

Apparently, when going from the Editor to the Formatter, the LOAD program reloads the assembly program, not checking to see if it is still in memory. The resulting wait can be avoided by making the following modifications to the LOAD program:

```
100 CALL CLEAR :: CALL INIT :: CALL PEEK(-2043,A,B) ::
    IF A<>84 OR B<>75 THEN 108
102 CALL LOAD(16360,85,84,73,76,73,84,250,212,70,79,82,
    77,65,84,250,132,69,68,73,84,79,82,250,22)
104 CALL LOAD(8196,63,232) :: GOTO 110
108 CALL LOAD("DSK1.WRITER")
```

The second tip deals with the potential problem of selecting "SD" for Show Directory and losing your current work. If you are aware of this problem you can probably avoid it, however, there is a fix. Find the third sector of the EDITA1 program (using a utility program such as DISKO), and make the following modification:

ORIGINAL FORM

```
2D 54 53 48 3E 0F 2D 54 52 45 3E 2C 0C
2F CA 4D 20 3E 84 2F 42 53 44 3C D4 18
  2E 4C 53 46 00 00 2E 8A 4C 46 00 00
```

ALTERED FORM

```
2D 54 53 48 3E 0F 2D 54 52 45 3E 2C 0C
2F CA 4D 20 3E 84 2F 42 20 44 3C D4 18
  2E 4C 53 46 00 00 2E 8A 4C 46 00 00
```

This change in the program will return you to the command line should you happen to press "SD".

```
LFMIQVFLTRPXNOUYBUBTZHYAHPLAUHSY
FULMLJCQAMEHCRKZOCEYIKBHLKTPPCRE
IGCTZHJATEGAHIYUACCEPTATPMBYBOCC
WYWCMNLNLHLUNJABZOXAIFCFEUKKOMAH
BGKQDEMEHLZGUQAHJHWAIGCBHBKNZWIT
EIJOVADJCIJMNUCDBOTDMUVSGGWULCTX
VNEDRLUHJHMORDOTLTGUIOTQKAGPNFFC
OJQXETAWCHADYNGRSUMYNSDKRXPRHGLA
SYMRJRNORTIUGSZTTGPEQWPNZJCOXJHL
SDSOSJECZSMTYATIYYRLSQILLMGPNIVL
AXBEOFUAPAOMOXFYXRNCHNUMABCWBXDP
OGTRGYQLMMCNOEGXOLLRGBUMXYSBUTDE
GJLCHKALPDSABNSRHAIXGLUQFFUGUDCE
DBHDCYXIFLKBLRMJMJTNWWJUDBCSZNFK
IKRJAVPNKBZHRLEYPRGJPIJZODGNIAIT
IROTHAZIMYRRBXLAKZQSVUXIXTHQPNVS
BGSNVIMTEFMHMZJIKOJYGLTUIDJKAQGR
WEZWAEZHRZJZLLBYNJWCYDSMMCMSXJTS
BFMNMIZDADJLDDOYXKUCOCCCAVOHRTIY
MZCZUSBDSFVVGNISUTNIRPQUGNWDINZL
QIZNILLUESLMMUQZVTMYPAONEGDVELZL
DIWZTWRRAKHASKRJOHNOISREVLLACQPS
WVEGYOYHLCOETIRPSLLACDJODJRAVYPX
KCIXIYXVLMEALOZVADVMRWNJASOANBBY
IYYQDAOLLLACQEGSHWVHEQCOJRNIQITB
```

WORDSEARCH #5

WORD LIST

DISPLAYAT	ONBREAK
ACCEPTAT	ONERROR
IMAGE	PRINTUSING
CALLINIT	ONWARNING
CALLLOAD	DISPLAYUSING
CALLLINK	UALPHA
LINPUT	ERASEALL
SIZE	CALLJOYST
CALLVERSION	CALLSPRITE
CALLPEEK	CALLCHARSET

AN INTRODUCTION TO FILE PROCESSING WITH TI-BASIC AND EXTENDED BASIC

By R. K. Hallmark

Editor's Note: The following article was copied from the July, 1984 issue of "The Suncoast Beeper", Newsletter of the Suncoast 99er's of St. Petersburg, Florida.

Files, Records, and Fields:

File-A file is the way basic programs communicate data with external storage devices. Some typical external devices are:

Device	File Name
Cassette Recorders	CS1 or CS2
Disk Drives	DSK1 to 3
Parallel Port	PIO
Serial Ports	RS232/1 or 2

Record-A group of data items which are stored together in a file.

Field-A single data item in a record. Each variable in your program will occupy one field of a record.

A file consists of one or more records and these records consist of one or more fields. When the computer transmits data to one of the devices listed above it sends one record at a time.

File Attributes:

Files may be organized in a number of different ways depending upon the device being used and what the file is being used for. The organization and other charactheristics of a file are called its attributes. When data files are opened in BASIC (using an OPEN statement) you describe the file and its attributes to the computer. In the discussion below the order of the attributes and the terms used are the same as in your BASIC manual.

It is not always necessary to specify each of the attributes. In many cases the computer will select what is called a "default value" if you do not chose one. The default value that the computer will use depends upon both the device selected and the other attributes of the file.

A. File #-May be any number from 1 to 255. (File #0 is the screen for output and the keyboard for input.)

B. Device Name-See list above.

C. File Organization-the file organization refers to way individual records within the file can be accessed.

1. SEQUENTIAL-Data is read to or written from a file starting at the beginning and going through the file one record at a time, you cannot skip around. All files except disk files can only be sequential. Sequential is the default for file organization.
2. RELATIVE-True random access files. The records can be written to or read from in any order. Disk files can be either relative or sequential.

D. File Type-this refers to the format in which the data is stored.

1. INTERNAL-The data is stored in the binary form in which it can be most easily used by the computer. Files stored on cassette or disk should be internal format.
2. DISPLAY-The data is stored in ASCII format. This format is used for sending data to the parallel and serial interfaces. Display is the default for file type.

E. Open Mode-Describes whether the file may be written to, read from, or both.

1. UPDATE-The file may be both written to or read from. This is the default for open mode.
2. OUTPUT-The file may only be written to.
3. INPUT-The file may only be read from.
4. APPEND-Allows you to add additional records to the end of the file.

F. Record Type-Describes whether the file has FIXED or VARIABLE length records. All relative files have fixed length records.

1. Cassette Files-These files may be specified as fixed or variable but the computer actually uses fixed length records which may be 192, 128, or 64 bytes long. The default for cassettes is VARIABLE with a maximum length of 64 bytes. The computer actually uses FIXED with 64 as the length.
2. Disk Files-SEQUENTIAL disk files have the default length of 80 bytes and a maximum length of 254 bytes . RELATIVE disk files must be of fixed length and may be up to 255 bytes in length.
3. The printer ports (RS232 and PIO have the default record type of FIXED with a length of 80 characters. Other lengths may be used.

Sample Cassette Programs:

Data Input:

```
100 OPEN #1:"CS1",SEQUENTIAL,INTERNAL,
OUTPUT,FIXED 192
110 INPUT "NUMBER OF NAMES: ": NUMBER
120 FOR I=1 TO NUMBER
130 INPUT "LAST NAME: ":LNAME$
140 INPUT "FIRST NAME: ":FNAME$
150 INPUT "ADDRESS: ":ADDRESS$
160 INPUT "CITY: ":CITY$
170 INPUT "STATE: ":STATE$
180 INPUT "ZIP: "ZIP
190 PRINT #1:LNAME$,FNAME$,ADDRESS$,
CITY$,STATE$,ZIP
200 NEXT I
210 CLOSE #1
220 END
```

Data Output:

```
100 OPEN #1:"CS1",SEQUENTIAL,INTERNAL,
INPUT, FIXED 192
110 INPUT "NUMBER OF NAMES :":NUMBER
120 FOR I=1 TO NUMBER
130 INPUT #1:LNAME$,FNAME$,ADDRESS$,
CITY$,STATE$,ZIP
140 PRINT FNAME$: " ";LNAME$
150 PRINT ADDRESS$
160 PRINT CITY$;",";STATE$;" ";ZIP
170 FOR DELAY=1 TO 2000
180 NEXT DELAY
190 NEXT I
200 CLOSE #1
210 END
```

CALL KEY

THE CALL KEY PROGRAM IN EXTENDED BASIC IS A VALUABLE TOOL BUT I FOUND THAT IT HAS TAKEN ME A LONG TIME TO REALLY UNDERSTAND IT'S PROPER USEAGE. ONE THING THAT HELPED ME WAS THE FOLLOWING QUICK REFERENCE CHART WHICH I PASTED INTO MY EXTENDED BASIC MANUAL FOR QUICK REFERENCE. Joyce Corker, Waltham, Mass.

```
CALL KEY (0,KEY,STATUS)
   0=becomes whatever mode was used
     by previous CALL KEY statement
   KEY=is returned as ASC VALUE()
   STATUS=-1 if NO KEY is pressed

CALL KEY(1,K,S)
   Returns K values from LEFT SIDE of keyboard

CALL KEY(2,K,S)
   Returns K values from RIGHT SIDE of keyboard

CALL KEY=(3,KEY,STATUS)
   3=TI/99 4 MODE  (FORGIVING)
   K=only UPPER CASE letter values
     are returned even if a
     lower case letter is pressed
     in error BUT only works with
     FUNCTIONS 1-15  -  NO CONTROL KEYS

CALL KEY=(4,K,S) PASCAL MODE
   provides UPPER and LOWER case
   letter values
  FUNCTIONS 129-143   -  CONTROLS 1-31

CALL KEY(5,K,S)
   5=BASIC mode for TI/994A
   K=returns BOTH UPPER  LOWER
    CASE letter values BUT if
    upper case answers are
    asked for and lower case
    are returned, the lower case
    answers won't be accepted.

STATUS KEY CHANGES (EXCEPT IN '0' MODE)
   S=1  (NEW KEY PRESSED)
   S=-1 (SAME KEY PRESSED)
   S=0  (NO KEY PRESSED)
```

ACCESSING FUNCTION AND CONTROL KEYS AND ARROWS:

```
100 DISPLAY AT(3,3)ERASE ALL:"PRESS CONTROL KEY , "
110 FOR DELAY=1 TO 600 :: NEXT DELAY
120 CALL KEY(5,K,S)
130 IF K=128 THEN PRINT "CONTROL,COMMA PRESSED"
```

OR:

```
100 DISPLAY AT(3,3)ERASE ALL:"PRESS FUNCTION  RIGHT ARROW"
130 IF K=9 THEN PRINT "RIGHT ARROW KEY PRESSED"
```

OR:

```
100 DISPLAY AT(3,3)ERASE ALL:"PRESS FUNCTION 8"
130 IF K=6 THEN PRINT "FUNCTION 8 PRESSED"
```

CALL KEY COMBINATIONS

I ALSO KEEP A NOTEBOOK OF USEFUL (AND REUSEABLE) TIPS AND TRICKS FOR PROGRAMMING AND I'VE WORKED OUT A FEW USEFUL CALL KEY COMBINATIONS THAT I CAN PULL OUT AND USE WHENEVER I NEED THEM. OF COURSE, YOU CAN SUBSTITUTE LINE NUMBERS OR DIRECTIONS TO SUB PROGRAMS OR OTHER INSTRUCTIONS INTO THE LOGIC INSTRUCTIONS.

YES OR NO ANSWERS WITH CALL KEY 0

```
 80 CALL CLEAR
 90 PRINT "Y OR N? "
100 CALL KEY (0,K,S)
110 IF K=78 THEN PRINT "NO" :: STOP
120 IF K<>89 THEN 100 ELSE PRINT "YES"
130 STOP :: END
```

SPACE BAR OR CARRIAGE RETURN (ENTER) ANSWERS
WITH CALL KEY 5

```
100 DISPLAY AT(3,3)ERASE ALL:"PRESS SPACE BAR TO CONTINUE"
    : : "PRESS ENTER/CARRIAGE RETURN  TO PRINT"
110 FOR DELAY=1 TO 600 :: NEXT DELAY
120 CALL KEY(5,K,S)
130 IF K=32 THEN PRINT "SPACE BAR PRESSED" ELSE IF K<>13
    THEN 120
140 IF K=13 THEN PRINT "ENTER (C/R) WAS PRESSED"
150 STOP :: END
```

RIGIDLY CONTROLLED ANSWERS WITH CALL KEY 5

```
100 DISPLAY AT(3,3)ERASE ALL:"PRESS Y FOR YES" : :"  PRESS
    N FOR NO"
110 FOR DELAY=1 TO 600 :: NEXT DELAY
120 CALL KEY(5,K,S)
130 IF K=89 THEN PRINT "YES,YES" ELSE IF K<>78 THEN 120
140 IF K=78 THEN PRINT "NO,NO"
150 STOP :: END
```

ALPHABET ANSWERS THAT ARE FORGIVING OF WRONG CASE
ANSWERS WITH CALL KEY 3

```
100 DISPLAY AT(3,3)ERASE ALL:"PRESS R TO REPEAT": :
    "  PRESS P TO PRINT"
110 FOR DELAY=1 TO 600 :: NEXT DELAY
120 CALL KEY(3,K,S)
130 IF K=82 THEN PRINT "HERE YOU WOULD GO TO YOUR REPEAT
    SUBPROGRAM" ELSE IF K<>80 THEN 120
140 IF K=80 THEN PRINT "HERE YOU WOULD GO TO YOUR PRINT
    SUB PROGRAM"
150 STOP :: END
```

YES OR NO ANSWERS WITH CALL KEY 3
THAT FORGIVE YOU IF YOU ANSWER AN UPPER CASE QUESTION WITH
A LOWER CASE ANSWER.

```
100 DISPLAY AT(3,3)ERASE ALL:"PRESS Y FOR YES": :"  PRESS N
    FOR NO"
110 FOR DELAY=1 TO 600 :: NEXT DELAY
120 CALL KEY(3,K,S)
130 IF K=89 THEN PRINT "YES, YES" ELSE IF K<>78 THEN 120
140 IF K=78 THEN PRINT "NO,NO"
150 STOP :: END
```

"Getting Spritely With The 99/4A"

by Bill Cagle

I was thinking about an application for a software driven process controller. The current state of the art, has six mechanical timers, in a box that pass the timing control of the processes from one timer to the next and, at best, the timers are easy to damage. I believe this is a crude way to do business. So armed with an idea and the Extended BASIC manual I sat down and commenced to write some code. The first thing I did was to code some "bit" strings to make box es. I needed four corners, two sides, a top and a bottom line. This gave me th e characters to make a series of boxes to put the "big" sequence numbers in. The boxes were created with a loop that looks like this:

```
100 FOR I=8 TO 24 STEP 6
110 R=1 :: H=3 :: C=I+H :: W=H :: GOSUB 600
120 R=6 :: H=3 :: C=I+H :: W=H :: GOSUB 600
130 NEXT I
```

These variables are for (R)ow, (C)olumn, (W)ide, (H)igh. The sub at 600 looks like this:

```
600 CALL HCHAR(R,C,120,1) (This code prints an upper left hand corner.
610 CALL HCHAR(R,C+1,121,W) (This makes the top line)
```

The code continues till the box is printed on the screen and then it "RETURN"s to the loop to get the next set of R, C, W, and H to make the next box till all six are on the screen.

To create the "big" numbers I used code like this:

```
FOR I=1 TO 8 :: CALL CHARPAT(ASC(STR$(I)),N$(I)) :: NEXT I
```

This gets the bit pattern stored in the basic ROM and puts that pattern into th e strinc array called "N". Next you have to assign character numbers to the sprites with this:

```
FOR I=1 TO 6 :: CALL CHAR(I+125,N$(I)):: NEXT I
```

this assigns the character number of 126 to 132 to the "big" number sprites.

To get the big numbers on the screen, I used:

```
CALL SPRITE(#N,CHAR#,COLOR,DOT ROW, COLUMN)
```

and to get each sprite in the proper box, I used the "ON Y GOTO" statement. This was incorporated in a sub-routine like this:

```
420 GOSUB 8000

8000 ON Y GOTO 8010,8020,8030, etc.
8010 CALL SPRITE(#1,126,2,12,92) :: RETURN
8020 CALL SPRITE(#2,127,2,12,141) :: RETURN
```

this was continued until all six sprites (big numbers were defined and Y is the sequence currently being timed.

Next I had to produce a time keeping function. I used the ability of the VDP to keep a sprite in motion, once it is set in motion. This was accomplished by defining still another sprite and setting it in motion down the left side of the screen. The sprite was chosen to look like a sideways U and it slid down row of numbers. The speed was chosen by trial and error to give a speed of two lines per second. this would make the sprite pass a number every second and using the CALL POSITION(X,Y), X will have a down row value that will correspond to the number. This will let you compute the time from the value of X. I then wrote a routine that would continually check the value of X and when it was greater than the preset time for that segment, it would increment the sequence and start the next time interval. This would continue until all of the sequences are set up.

TI had made it very easy to put sprites and text on the screen at the same time. Maybe this will help some of you to start to play around with the extensive graphic command set which is built in Extended BASIC and 99/4A. If you try, you can do some tricky things with the screen and the sprites. I hope this will make you feel spritely!

32K MEMORY CHECKER

by Joe Nuvolini

Editor's Note: The following article was reprinted from the February, 1986 issue of "The Front Ranger", Newsletter of the Front Range 99er Computer Club of Colorado Springs, Colorado.

I recently picked up a second TI system at a bargain price but to my chagrin, I found that the 32K memory card was bad. With the help of one of our members, Darnell Denison, I found that I had a bad 4116 chip. Darnell wrote me a short program that told us which one it was. I took that program and refined it a bit and the program below will tell you exactly which 4116 chip is bad. Of course, you must key the program if your memory card is bad. Once keyed in the program will allow you to select the top or bottom row of chips and it will list the chips in the row selected by chip number, i.e., U35, U34,...etc. and tell you whether the chip is OK or BAD. Then you need only replace the chip and your card is fixed saving you $47+. Remember this program will only tell you if one of the 16 4116 chips is bad. If it is one of the other chips, this program will not work. Luckily, the chip that was bad on mine was a 4116 and it is now working just fine as I write this article. By the way, Bob Haggart was kind enough to give me a tube of 4116 chips that he had removed from a TRS 80 so if you should need one I have a good supply, and will be glad to give you one. Also, TI put those chips in to stay and they are a bear to get out. If you change one, I suggest you put a socket in to hold the replacement. The program listing follows:

```
110 PRINT "MEMORY EXPANSION CHECKER": : :
120 PRINT "SINCE PROGRAMS LOADED FROM DISK IN XB LOAD INTO THE
32KMEMORY, THIS PROGRAM SHOULD BE KEYED IN IF YOUR 32K CARD IS BAD!":
:
130 PRINT "TO USE THE INFORMATION PROVIDED BY THIS TEST ORIENT YOUR
MEMORY EXPANSION BOARD WITH THE TWO ROWS OF 4116 CHIPS AT THE TOP.":
: :
140 PRINT "ENTER: 1 TO CHECK TOP ROW OF CHIPS 2 TO CHECK BOTTOM ROW
3 TO END"
150 CALL KEY(0,K,S):: IF S=0 THEN 150
160 IF K<49 OR K>51 THEN 150
170 R=K-48
180 IF R=1 THEN A=-12288 ELSE IF R=2 THEN A=12287 ELSE IF R=3 THEN
CALL CLEAR :: END
190 IF R=1 THEN N=35 ELSE N=27
200 CALL CLEAR
210 IF R=1 THEN PRINT "TEST OF TOP ROW OF 4116'S." ELSE PRINT "TEST
OF BOTTOM RO W OF 4116'S"
220 PRINT :"READING FROM RIGHT TO LEFT..": :
230 FOR I=0 TO 7
240 IN=2 I
250 CALL INIT
260 CALL LOAD(A,IN)
270 CALL PEEK(A,D)
280 IF IN=D THEN PRINT "CHIP U";STR$(N);" IS OK" ELSE PRINT "CHIP
U";STR$(N);"IS BAD"
290 N=N-1
300 NEXT I
310 PRINT
320 INPUT "PRESS ENTER TO CONTINUE ":X$
330 GOTO 100
```

THINGS TI NEVER TOLD US

PREVENTING MEMORY LOSS

Losing a program from console memory can be a very frustrating experience, especially if the program had not been saved. One of the most common ways to lose a program, is by pushing the function plus (+) instead of the shift plus (+) key, which immediately erases everything in memory. I have had to tolerate this for the last two years with no way to prevent it. Now I have found this can be eliminated also if you have the extra memory and Extended BASIC by doing the following:

```
CALL INIT
CALL LOAD(-31806,16)
```

If you have any preschoolers and have noticed they have a habit of accidentally clearing the memory then this is a necessary routine to add to their software.

Editor's Note: This TINT comes to the Hoosier Users Group, courtesy of the Puget Sound 99ers of Lynnwood, Washington.

I do not know if this will be of any interest to anyone (probably everyone knows it already); however, for those of you who do not, using Function 3 (erase) key will give you a running clock in the lower left hand side of the screen when using the A-Maze-Ing solid state module. Likewise, the quickest previous time through the maze appears and remains until a person beats the time and then it is updated. I did not find any mention this in the manual.

Submitted by Donna Griffin.

Other quirks on the TI computer:

For new TI computer owners, there is an easier way to edit program lines in BASIC. First, type in the line number that you want to edit, and press Function X (the arrow down key). After you finish editing the line, press ENTER. It is easier to use than the EDIT command, especially when doing multiple line editing.!

For those of you who have not been told yet, there is a way to access the TI Test Mode on the Munch Man Disk or Module. To accomplish this, after selecting the Munch Man option and the title screen appears, hold down the shift key and enter the sequence 8 3 8. But be quick because you must do this before the computer begins to play the music. You will then select the speed (RND 0-2), then the screen level (SS 00-19), and then the number of Munch Men (MM 1-9). a real challenge, try level 20 (enter but you will probably need all 9 Munch Men to finish this screen alone!

by Jim Ellis

How many of you using Multiplan with two drives are bothered by having to set up the system after loading for DSK2? Well if you know how to use Disk Fixer or some other similar program here is the way to make Multiplan come up with DSK2 as the default drive for your files.

The following is a sector from Multiplan, it is the second sector of the MPINTR file. The byte I have left empty normally contains 31 (hex for 1). With your sector editor, change this byte to 32 (hex for 2), then save the sector back to your disk. When you load Multiplan you wont see it display DSK2 under the TRANS/LOAD OPTION, but it will look at drive 2 for your files.

I found this out by trying different things until I got the right results, because I was tired of having to setup the drive every time I loaded the system. Hope this helps someone.

```
3908 8320 3A28 8320 38B4 8320 38DA
8320 35E0 8320 3DCC 8320 3F4C FF00
0410 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 2EAA 0000
0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0005 4453 4B   2E20
0010 4453 4B2E 5449 4D50 2E4F 5645
524C 4159 000E 4453 4B2E 5449 4D50
2E4D 5048 4C50 0000 0000 0000 0000
0000 0000 0026 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000
0000 0000 0012 0628 5000 0000 000C
5253 3233 322E 4241 3D33 3030 2020
2020 2020 2020 2020 2020 2020 2020
2020 2020 2020 2020 2020 2020 FFFF
03E0 0412 24CC 0000 0000 0003 0000
FFFF 0492
```

THINGS TI NEVER TOLD US

Editor's Note: When creating this column I did not expect to see it included in the HUGger Newsletter often, but, little did I realize how much can be learned by experimenting.

This month's TINT's come from sources outside the Hoosier Users Group. They are handy when you are involved in a midnight debugging session and say OH-OH!

The first TINT comes from Jim Peterson of Tigercub Software; 156 Collingwood Ave., Columbus, Ohio. Jim has this to include from Tips from the Tigercub # 3:

Here's a lifesaver that was passed on to me verbally, so I don't know who to credit for discovering it....It's 2 A.M., you just got the last bug out of your new program, you sleepily put a new cassette in the recorder, type OLD CS1, hit enter and....ooooh!, you meant to type SAVE CS1!! But all is not lost - just type SHIFT E, hit ENTER, get an I/O error message and start over! (Editor's Note: If you have the Alpha Lock DOWN and type E, and hit ENTER, you will get the same results. The lower case "e" is not accepted when using this procedure. Look for more from Tigercub # 3 in the December Newsletter.)

And the second comes from the Pittsburgh Users Group newsletter "Peripheral": Let's say you're editing a program and instead of pushing Function 2 (Insert), you inadvertently miss 2 and hit Function 3 (Erase)! DO NOT HIT ENTER! Instead, push Function P (the quotation marks) and then press ENTER. LIST the program, and guess what -- you didn't lose the line after all! Editor's note: When typing in the Function P and ENTER, the computer will come back stating "INCORRECT STATEMENT". Since this is not a valid entry, your original data was not lost, it just became "temporarily invisible".)

Converting other Basic's

Rich Klein had an interesting article in the May 1986 issue of the Chicago Times newsletter concerning the use of BASIC and X-BASIC's DEFine statement. DEF allowes the programmer to define his/her own functions, for example: if you wanted to use PI in a BASIC program you could start off by typing

```
100 DEF PI=3.141592654
```

From that point on, anytime your program came across the function PI it would evaluate it as defined.

A similar situation could be used to define functions found in other BASIC's but not found in TI BASIC. For example, many BASIC's contain the functions LEFT$, MID$, RIGHT$. These could be handled in TI BASIC by defining them as follows.

```
100 DEF LEFT$(X$,Y)=SEG$(X$,1,Y)
110 DEF MID$(X$,Y,Z)=SEG$(X$,Y,Z)
120 DEF RIGHT$(X$,Y)=SEG$(X$,LEN(X$)-Y-1
,Y)
```

KIDS ## Do you want your children to learn to play the piano, but can't afford one right now? Here is an inexpensive substitute: the TI-99/4A piano.

You play only the bottom three rows, essentially all of the letter keys. You can play them with the shift key up or down. In one case you play whole notes, in the other one you play half notes, which repeat when you hold down the key.

The very bottom row plays noise tones when the shift key is locked down, very amusing to little children.

To keep the program simple, the screen is left blank. But that does not impede all you budding programmers to create some nice graphics to enliven this music program, and make it even more attractive to children.

This program was published in Nittinian, the Swedish newsletter for 99-ers, by an unknown author. The translation was done by Maurice E.T. Swinnen of the Washington DC Area 99-er Computer Club.

```
50 REM PIANO, NITTINIAN 84-2
100 CALL KEY(0,K,S):: IF S=0 THEN 100
110 IF K=45 THEN 100
120 IF K<44 THEN 100 ELSE IF K>46 AND K<
58 THEN 100 ELSE IF K>60 AND K<65 THEN 1
00
130 IF K>90 AND K<96 THEN 100 ELSE IF K>
96 THEN 200 !CHECK IF LOWER OR UPPER CAS
E LETTER HAS BEEN PRESSED
140 IF K=44 THEN CALL SOUND(-100,1568,0)
:: GOTO 100
150 IF K=46 THEN CALL SOUND(-100,1760,0)
:: GOTO 100
160 IF K=59 THEN CALL SOUND(-100,698,0):
: GOTO 100
170 IF K=58 THEN CALL SOUND(-100,1661,0)
:: GOTO 100
180 IF K=60 THEN CALL SOUND(-120,-8,0)::
 GOTO 100
185 REM UPPER CASE LETTERS ASCII=65->90
190 ON K-64 GOTO 210,220,230,240,250,260
,270,280,290,300,310,320,330,340,350,360
,370,380,390,400,410,420,430,440,450,460
195 REM LOWER CASE LETTERS ASCII=97->122
200 ON K-96 GOTO 470,480,490,500,510,520
,530,540,550,560,570,580,590,600,610,620
,630,640,650,660,670,680,690,700,710,720
205 REM UPPER CASE LETTERS=HALF TONES+NO
ISE
210 CALL SOUND(-120,466,0):: GOTO 100
220 CALL SOUND(-120,-6,0):: GOTO 100
230 CALL SOUND(-120,-1,0):: GOTO 100
240 CALL SOUND(-120,622,0):: GOTO 100
250 CALL SOUND(-120,156,0):: GOTO 100
260 CALL SOUND(-120,740,0):: GOTO 100
270 CALL SOUND(-120,831,0):: GOTO 100
280 CALL SOUND(-120,932,0):: GOTO 100
290 CALL SOUND(-120,311,0):: GOTO 100
300 CALL SOUND(-120,1109,0):: GOTO 100
310 CALL SOUND(-120,1245,0):: GOTO 100
320 CALL SOUND(-120,1480,0):: GOTO 100
330 CALL SOUND(-120,-4,0):: GOTO 100
340 CALL SOUND(-120,-5,0):: GOTO 100
350 CALL SOUND(-120,370,0):: GOTO 100
360 CALL SOUND(-120,415,0):: GOTO 100
370 CALL SOUND(-120,177,0):: GOTO 100
380 CALL SOUND(-120,185,0):: GOTO 100
390 CALL SOUND(-120,554,0):: GOTO 100
400 CALL SOUND(-120,208,0):: GOTO 100
410 CALL SOUND(-120,277,0):: GOTO 100
420 CALL SOUND(-120,-7,0):: GOTO 100
430 CALL SOUND(-120,139,0):: GOTO 100
440 CALL SOUND(-120,-2,0):: GOTO 100
450 CALL SOUND(-120,233,0):: GOTO 100
460 CALL SOUND(-120,-3,0):: GOTO 100
465 REM LOWER CASE LETTERS=WHOLE TONES
470 CALL SOUND(-100,294,0):: GOTO 100
480 CALL SOUND(-100,1175,0):: GOTO 100
490 CALL SOUND(-100,988,0):: GOTO 100
500 CALL SOUND(-100,349,0):: GOTO 100
510 CALL SOUND(-100,131,0):: GOTO 100
520 CALL SOUND(-100,392,0):: GOTO 100
530 CALL SOUND(-100,440,0):: GOTO 100
540 CALL SOUND(-100,494,0):: GOTO 100
550 CALL SOUND(-100,220,0):: GOTO 100
560 CALL SOUND(-100,523,0):: GOTO 100
570 CALL SOUND(-100,587,0):: GOTO 100
580 CALL SOUND(-100,659,0):: GOTO 100
590 CALL SOUND(-100,1397,0):: GOTO 100
600 CALL SOUND(-100,1319,0):: GOTO 100
610 CALL SOUND(-100,247,0):: GOTO 100
620 CALL SOUND(-100,262,0):: GOTO 100
630 CALL SOUND(-100,110,0):: GOTO 100
640 CALL SOUND(-100,147,0):: GOTO 100
650 CALL SOUND(-100,330,0):: GOTO 100
660 CALL SOUND(-100,165,0):: GOTO 100
670 CALL SOUND(-100,196,0):: GOTO 100
680 CALL SOUND(-100,1047,0):: GOTO 100
690 CALL SOUND(-100,123,0):: GOTO 100
700 CALL SOUND(-100,880,0):: GOTO 100
710 CALL SOUND(-100,175,0):: GOTO 100
720 CALL SOUND(-100,784,0):: GOTO 100
```

```
100 !#########################################################################
110 !
120 !   Program UPDATED BY WILLIAM M. LUCID, Original by MBP for use with
130 !   the MBP Analog to Digital board for the TI Expansion System.
140 !   This is a documentation program, suitability, use of this program
150 !   is at USER'S OWN RISK.
160 !   ONLINE information about MBP is available from Jerry McClusky TIBBS(tm)
170 !   bbs  300/1200 baud in Wichita, KS  316-681-3167.
180 !
190 !      Vcc (+5 vdc)
200 !       ¦-----------¦
210 !       ¦           ¦
220 ! LM335 ¦           ¦
230 !      .-.          > 10,000 ohm variable resistor
240 !      ¦ ¦          <
250 !      ¦ ¦ Adj    \ >   Output   10mV/ Kelvin
260 !      ¦ ¦---------><..................... A/D Channel 0 (Pin 6)
270 !      ¦ ¦         ' >                    degree
280 !      ¦ ¦          <
290 !      '-'          >
300 !       ¦           ¦
310 !       ¦           ¦
320 !       ¦...........¦..................... A/D Ground (Pin 16)
330 !          Ground
340 !
350 ! Program for use with analog to digital board for P-Box.
360 ! Device used to sense temperature is described in National Semiconductors
370 ! Linear Databook. One low cost devices, resistor and powered by a five volt
380 ! supply. LM 335 are NATIONAL'S semiconductors. Calibration may be needed.
390 ! Each sensor is capable of being calibrated individually.
400 !
410 !#########################################################################
420 CALL CLEAR :: CALL SCREEN(8):: CALL INIT :: DEF SET=X+6*INT(X/10):: DEF TIME
=X-6*INT(X/16):: DEF F=.4578313254 :: DIM WK$(7),MO$(12)
430 FOR DW=1 TO 7 :: READ WK$(DW):: NEXT DW
440 FOR DM=1 TO 12 :: READ MO$(DM):: NEXT DM
450 DATA Sun,Mon,Tues,Wednes,Thurs,Fri,Satur
460 DATA January,February,March,April,May,June,July
470 DATA August,September,October,November,December
480 CALL PEEK(-31158,X1,X2,D,X4,X5):: X=D :: D$=STR$(TIME):: X=X5 :: X5=TIME ::
L1$="Today is "&WK$(X1)&"day" :: L1=INT((32-LEN(L1$))/2):: L2$=MO$(X5)&" "&D$&",
 1985"
490 L2=INT((32-LEN(L2$))/2)
500 Z=TC*F :: CALL PEEK(-31164,X1,X2,X3,X4,X5):: X=X1 :: SEC$=STR$(TIME):: IF X1
<10 THEN SEC$="0"&SEC$
510 X=X3 :: MIN$=STR$(TIME):: IF X3<10 THEN MIN$="0"&MIN$
520 X=X5 :: HR=TIME :: M$=" am" :: IF HR>11 THEN M$=" pm"
530 IF HR=0 THEN HR=12
540 IF HR>12 THEN HR=HR-12
550 HR$=STR$(HR):: T1$=HR$&":"&MIN$&":"&SEC$&M$
560 DISPLAY AT(6,L1):L1$ :: DISPLAY AT(8,L2):L2$: : :TAB(10);"The Time Is": : :T
AB(11);T1$
570 CALL PEEK(-31088,TC):: CALL PEEK(-31072,TC)
580 DISPLAY AT(17,7):"Room Temperature" :: DISPLAY AT(19,10):USING "####.# F.":Z
590 IF X1+X3+X5=213 THEN 480 ELSE 500
```

KIDS KORNER

Alphabet Recognition was reprinted from "Bug-Bytes of Australia" via the December, 1984 issue of Topics, Newsletter of the LA 99'ers, P.O. Box 3547, Gardenia, CA 90247.

ALPHABET RECOGNITION

This program was written by L.K.TUTCHINGS to help his son learn the alphabet. It proved to be a big hit at his pre-school too. I think that all the pre-schoolers out there will love it too. It requires Extended Basic and if you have a speech synthesizer, you will also get speech with it.

```
100 CALL SCREEN(8)
110 FOR COL==3 TO 8 :: CALL C
OLOR(COL,2,1):: NEXT COL
120 DISPLAY AT(4,4)ERASE ALL
:"1 ALPHABET RECOGNITION" ::
 DISPLAY AT(6,4):"2 ALPHA AT
TACK"
130 DISPLAY AT(8,4):"3 CLOSE
 OF PROGRAM" :: DISPLAY AT(1
8,2):"PUSH No KEY OF YOUR CH
OICE"
140 CALL KEY(0,K,S):: IF S=0
 THEN 140 :: IF K=ASC("1"")TH
EN 150 :: IF K=ASC("2")THEN
470 :: IF K=ASC("3")THEN 850
 ELSE 140
150 DISPLAY AT(12,4)ERASE AL
L:"ALPHABET RECOGNITION" ::
FOR DE=1 TO 300 :: NEXT DE
160 CALL CLEAR
170 PRINT "THE IDEA IS TO PR
ESS THE KEY ON THE KEYBOARD
THAT       MATCHES THE LETTE
R THAT IS  GOING ACROSS THE
SCREEN"
180 PRINT "THE COMPUTER WILL
 LET YOU   KNOW IF IT IS COR
RECT. IF IT IS CORRECT THEN
ANOTHER    LETTER IS RANDOML
Y SELECTED"
190 PRINT "UNTIL YOU HAVE CO
RRECTLY GOT 40 RIGHT": : :"P
RESS ANY KEY TO START"
200 CALL KEY(0,K,S):: IF S=0
 THEN 200
210 CALL CLEAR
220 RANDOMIZE
 230 FOR A=1 TO 40
240 X=INT(RND*25)+65
250 CALL SCREEN(2)
260 CALL MAGNIFY(2)
270 FOR C=5 TO 8
280 CALL COLOR(C,16,2):: NEX
T C
290 FOR Q=4 TO 25 STEP 4
300 DISPLAY AT(24,Q):CHR$(X)
:: DISPLAY AT(1,Q):CHR$(X)::
 NEXT Q
 310 CALL SPRITE(#1,X,16,86,2
0,0,10)
320 CALL SAY(CHR$(X))
330 CALL KEY(0,K,S):: IF S=0
 THEN 330 :: IF K[]X THEN 36
0 ELSE 390
340 CALL DELSPRITE(#1):: NEX
T A
350 GOTO 100
360 DISPLAY AT(20,7);"WRONG
TRY AGAIN!"
370 CALL SAY("UHOH.THAT IS N
OT RIGHT, TRY AGAIN")
380 DISPLAY AT(20,7)SIZE(16)
:"                " :: GOTO
310
390 DISPLAY AT(20,12)SIZE(5)
:"RIGHT"
400 Z=INT(RND*5)+1 :: ON Z G
OTO 410,420,430,440,450
410 CALL SAY("GOOD WORK,GO S
OME MORE"):: GOTO 460
420 CALL SAY("THAT IS CORREC
T, CAN YOU DO IT AGAIN"):: G OTO 46
430 CALL SAY("RIGHT. GO AGAI
N"):: GOTO 460
440 CALL SAY("GOOD, WHY STOP
 NOW"):: GOTO  460
450 CALL SAY("YES.GO AGAIN")
460 DISPLAY AT(20,12)SIZE(5)
:"      " :: GOTO 340
470 CALL CLEAR
480 DISPLAY AT(12,8):"ALPHA
ATTACK" :: DISPLAY AT(20,2):
"WANT INSTRUCTIONS Y OR N?"
490 CALL KEY(0,K,S):: IF S=0
 THEN 490 :: IF K[]ASC("Y")T
HEN 550 ELSE 500
500 DISPLAY AT(2,2)ERASE ALL
:"THE OBJECT IS TO SHOOT DOW
N": :"THE ALPHABET IN ORDER"
510 DISPLAY AT(6,2):"USE THE
 JOYSTICK AND FIRING": :"BUT
TON.THERE ARE 52 BULLETS"
520 DISPLAY AT(11,1):"SO DO
NOT WASTE THEM...." :: DISPL
AY AT(13,5):"GO TO IT! GOOD
LUCK!"
530 DISPLAY AT(20,5):"PUSH A
NY KEY TO START"
540 CALL KEY(0,K,S):: IF S=0
 THEN 540 ELSE 550
550 CALL CLEAR
560 CALL SCREEN(2)
570 CALL MAGNIFY(1)
580 RANDOMIZE
590 FOR S=1 TO 26
600 R=INT(RND*120)+1 :: C=IN
T(RND*246)+10 :: CS=INT(RND*
15)+1
610 CALL SPRITE(#S,64+S,INT(
S/2)+3,R,C,0,CS)
620 NEXT S
630 CALL SPRITE(#27,94,16,17
0,128)
640 AMMO=52
650 FOR T=1 TO 26
660 FOR CL=3 TO 8 :: CALL CO
LOR(CL,16,1):: NEXT CL
670 CALL POSITION(#27,R,D)
680 CALL JOYST(1,X,Y):: Y=0
690 CALL KEY(1,K,S):: IF S=0
 THEN 730 :: IF K=18 THEN CA
LL SPRITE(#28,46,16,R,D,-25,
0):: CALL SOUND(100,-1,0)::
AMMO=AMMO-1
700 IF AMMO=0 THEN 820
710 CALL COINC(#1,#28,8,C)::
 CALL POSITION(#28,R1,D1)
720 IF C=-1 THEN 750 :: IF R
1[9 THEN CALL DELSPRITE(#28)
ELSE 710
730 DISPLAY AT(23,2):"AMMO="
;AMMO
740 CALL MOTION(#27,-Y,X*6):
: GOTO 670
750 CALL SOUND(250,-7,0):: C
ALL DELSPRITE(#1):: CALL DEL
SPRITE(#28)
760 DISPLAY AT(24,1+1):CHR$(
64+T)
770 IF T]26 THEN 790
780 NEXT T
790 CALL DELSPRITE(ALL):: CA
LL CLEAR :: CALL SCREEN(2)
800 DISPLAY AT(10,1):"WELL D
ONE WANT TO PLAY AGAIN" :: D
ISPLAY AT(12,11):"Y OR N" ::
 DISPLAY AT(14,1):"YOU HAD";
AMMO;"BULLETS LEFT"
810 CALL KEY(0,K,S):: IF S=0
 THEN 810 :: IF K[]ASC("Y")T
HEN 100 ELSE 550
820 CALL DELSPRITE(ALL):: CA
LL CLEAR :: CALL SCREEN(2)
830 DISPLAY AT(10,1):"SORRY-
OUT OF AMMO PLAY AGAIN" :: D
ISPLAY AT(12,11):"Y OR N"
840 CALL KEY(0,K,S):: IF S=0
 THEN 840 :: IF K[]ASC("Y")T
HEN 100 ELSE 470
850 DISPLAY AT(12,11)ERASE A
LL:"GOODBYE" :: CALL SAY("GO
ODBYE")
860 FOR DE=1 TO 1000 :: NEXT
DE :: CALL CLEAR :: END
```

POINT PLOTTING By Brian Richwine, 2107 La Rita Lane, Anderson, IN 46011

Point Plotting runs on Extended Basic and draws a diagonal line (0,0 to 47,47) and a horizontal line (0,12 to 63,12) then erases the horizontal line. These subroutines (Pl Unplot, GR)(lines 30000 to 31030) allows the user to plot points.

To initialize the graphics, call up subroutine 'GR'. This allows you to use the 'Plot' and 'Unplot' subroutines in this form:

```
Call Plot(X,Y)
Call Unplot(X<Y)
```

X-Range: (0-63) Y Range: (0-47)

Program:

```
10 CALL GR :: CALL CLEAR
20 FOR A=0 TO 47
30 CALL PLOT(A,A)
40 NEXT A
50 FOR A=0 TO 63
60 CALL PLOT(A,12)
70 NEXT A
80 FOR A=0 TO 63
90 CALL UNPLOT(A,12)
100 NEXT A
30000 SUB PLOT(X,Y):: A=INT(X/2+1):: B=INT(Y/2+1):: CALL GCHAR(B,A,C):: IF C<128 THEN C=128
30010 CALL HCHAR(B,A,128+(C-128)OR 2^(-(INT(X/2)<X/2)-2*(INT(Y/2)<Y/2)<Y/2))):: SUBEND
30100 SUB UNPLOT(X,Y):: A=INT(INT(X/2+1):: B=INT(Y/2+1):: CALL GCHAR(B,A,C):: IF  C<128 THEN C=1
28
30110 CALL HCHAR(B,A,128+((C-128)AND 15-(2^(-INT(X/2)<X/2)-2*(INT(Y/2)<Y/2))))):: SUBEND
31000 SUB GR
31010 DATA F000,0F00,FF00,00F0,F0F0,0FF0,FFF0,000F,F00F,0F0F,FF0F,00FF,F0FF,0FFF,FFFF
31020 RESTORE 31010 ::CALL CHAR(128,""):: FOR A=0 TO 14
31030 READ A$ :: CALL CHAR(129+A,RPT$(SEG$(A$,1,2),4)&RPT$(SEG$(A$,3,2),4)):: NEXT A :: SUBEND
```

SINEWAVE By Brian Richwine, 2107 La Rita Lane, Anderson, IN 46011

Sinewave runs on Extended Basic and will draw a sinewave until it runs out of characters to define. The amount of definable characters can be increased by changing C and D in line 10 to a lower initial value.

This program gives you the ability to plot graphs with X-Ranges of 0 to 255 and Y-Ranges of 0 to 191. The graphics are initialized on line 10. The values of X and Y given to the subroutines in lines 500 to 550 must be integers.

Program:

```
10 D,C=91 :: FOR A=D TO 143 :: CALL CHAR(A,""):: NEXT A
15 CALL CLEAR
20 FOR X=0 TO 255 :: Y=INT(SIN(X/40*PI)*38+100)
30 GOSUB 500
40 NEXT X
50 STOP
500 Z=INT((X+8)/8:: W=INT((Y+8)/8):: R=8*((Y+8)/8-W)+1 :: S=7-8*((X+8)/8-Z):: CALL GCHAR(W,Z,T)
510 IF T<D THEN T=C :: C=C+1 :: IF C<143 THEN RETURN ELSE CALL HCHAR(W,Z,T)
520 CALL CHARPAT(T,A$):: U=R*2+(S>3):: B$=SEG$(A$,U,1):: V=(ASC(B$)-48+(ASC(B$)>64)*7)OR(2^(S+(S
>3)*4))
530 B$=CHR$(48+V-(V>9)*7):: IF R=1 AND S>3 THEN A$=B$&SEG$(A$,2,16)ELSE IF R=8  AND S<4 THEN A
SEG(A$,1,15)&B$
540 IF U>1 AND U<16 THEN A$=SEG$(A$,1,U-1)&B$&SEG$(A$,U+1,16)
550 CALL CHAR(T,A$):: RETURN
```

THE ASSEMBLY LANGUAGE FORUM - STARTING OUT

By Vic Kelson

Hello, and welcome to what I hope will become a monthly feature of the HUG newsletter. As a HUGger for the last 9 months, I've noticed with dismay the lack of coordinated interest in assembly language. I do believe that there's interest, though, 'cause I've heard "boy, I'd like to learn assembly language, but I just never have" from several people. Here comes your chance -- it's not that hard... really!!!

About the "forum" -- as I said earlier, I want this to be a monthly feature. Most importantly, this is ALL OF OUR column. Please let me know of any ideas, corrections, comments, complaints, or IDIDITs (my word for the euphoric feeling when a sticky piece of code finally works) that you might have, either at the meetings, on the bulletin board, or through the mail. I am learning this stuff too, and I hope to help moderate an ongoing discussion about assembly language in the HUG. I will also be holding get-togethers at the meetings, so that anybody who has never tried assembly can learn from those who have. We'll start real simple, and gradually get more advanced as time goes by.

Enough of that. The first featured assembly program in the forum is (ta-daaa!) a fairly fast screen dump routine for TI BASIC. If you have a dot-matrix printer, you might have wanted to make a copy of that neat picture you drew on your screen, but it's real hard on the TI. There are screen dump utilities (callable from extended BASIC) on the market, one of which I saw at the last meeting. I even wrote one in BASIC, and they have one thing in common: THEY ARE S---L---O---WWW..... The routine in this article is a LOT faster. It is not ideal. In fact, I have an even better one already, but the purpose of the article is to show you how to speed up a BASIC program using assembly language.

The problem with a BASIC screen dump is making the characters come out right on the printer when you've got nifty redefined characters on the screen. You see, due to the difference between the tasks performed by the video generator in your console and the printer, the characters are described differently. For, say the letter A:

```
                7 8 8 8 8 7 0 0  <-- Printer definition
                F 8 8 8 8 F 0 0

the dots are    X . . . . X . .  82   \
zeroes, the     X . . . . X . .  82    \
X's are ones    X . . . . X . .  82     \ TI definition
                X X X X X X . .  FC     / see CALL CHAR in
                X . . . . X . .  82    /  the BASIC manual
                X . . . . X . .  82   /
                X . . . . X . .  82  /
```

The hex codes at the top are the codes that the printer wants to see in hi-res graphics mode to make the character. The codes at right are the codes which define the character in the TI. Problem: how do we make the TI codes into printer codes??

The answer turns out to be simple in assembly language, because of an operation we can't do in BASIC, Pascal, FORTRAN... or most other high level languages: the SHIFT operation. Here's how it works for a "left shift"; If you have the following byte (8 bits):

```
0 0 1 1 0 1 0 0  (34 hex or 52 decimal, if
                  you're keeping track)
```

A left shift of 1 bit position will move all the bits one space to the left, resulting in:

```
0 1 1 0 1 0 0 0  (note that the rightmost bit
                  is filled with a zero)
```

(This is 68 hex or 104 decimal, 2 times the original value. Will moving the bits 1 space to the right divide by 2?? Try it -- why does it work??)

But how does this help us to change rows to columns, you ask? Well, you might have wondered what happens to the leftmost bit.. It falls out of the byte into a CPU flag, called the "carry bit". We can test the carry bit in our assembly program, and put either a zero or one, according to the flag, into our result byte.

Here's the algorithm for getting the leftmost column from a set of 8 row definitions: (I know you're completely lost by now, hang on...)

```
set the result byte to 0.
for I=1 to 8
  shift the result byte 1 bit to the left.
  get row defining byte number I from the group of 8.
  shift the row defining byte 1 bit to the left.
  if a one fell out (carry bit set):
    add one to the result byte (put the "1" bit in there)
  put back the shifted version of the row defining byte.
next I.
save the result byte. (it now contains the leftmost column)
```

Easy, huh?? For an exercise, draw the character definition for the letter A shown earlier on paper, and do the above algorithm 8 times. REMEMBER to put the shifted defining bytes back, so that if you loop through 8 times, the 8 result bytes will be printer definitions shown above. It really IS easy, I promise.

The algorithm I've just shown is actually a small part of the code for the assembly language program segment for our routine (shown below). The rest is just telling the computer how to talk to BASIC. Now, about the actual routine:

The screen memory for BASIC is set up as 768 character positions (numbered 0 to 767), 24 rows of 32 columns. Each memory cell in screen memory holds a charcter code, which tells the computer which set of defining rows to use to draw the character. The assembly routine given below returns a string variable 8 characters long which contains the printer definition for a character on the screen.

BASIC USAGE:

Before using the PRTCHR routine in your TI BASIC program, you must load it into expansion memory by executing the statements:

CALL INIT to initialize memory
CALL LOAD("DSK1.BSCSUP") to get the support routines
CALL LOAD("DSK1.PRTCHR") to load out routine

All of the required files are supplied on the distribution disk (see below).

The BASIC statement which accesses our routine is:

nnn CALL LINK("PRTCHR",I,A$)

Where I is the character number (0 to 767) to get, And A$ is the returned string. The string variable can be sent to the printer in graphics mode.

You will be on your own for the BASIC program, depending on your printer. In the listings portion of the article, I have included a version for my Gemini-10X printer. This routine will dump a screen of data, EXACTLY as it appears on the screen (see demo of use with my video version of the codebreaker program). The dump will take about 2 minutes, compared to 15-20 (or more) minutes for a BASIC or Extended BASIC screen dumper.

I realize that this is a pretty heavy first example of assembly language, but the idea is to show what kind of performance inprovement can be gained by even a short routine, like this one. I hope I've whetted your appetite.

For all of these articles, the text of the article, along with source code of assembly and BASIC (if any) programs will be available in machine readable form. No, Virginia, I don't expect you to type them in yourself. Whenever possible, I will try to make extended BASIC versions available on disk, and Mini Memory versions on cassette. Copies will cost you $5.00 if you send me a disk or cassette, $7.50 if you don't supply the disk, $6.00 if you don't supply the cassette. You can get the current month's stuff at the meeting, or send your check to me at:

Vic Kelson
3284 Peggy Lane
Terre Haute, IN 47805
Home phone: (812)466-4031 (after 5pm)

If you want a book on TI assembly language, I suggest:

"Introduction to Assembly Language for the TI Home Computer", by Ralph Molesworth

and

"Fundamentals of TI-99/4A Assembly Language", by M. S. Morley.

My personal preference (for what it's worth) is the second book. It starts out real simply, and has zillions of (well, 47) good examples. The TI Editior/Assembler manual is also a good idea.

Well, that's all for our first meeting. I hope to see at least 4 or 5 (or more) people at the meeting in December. I'll demonstrate the screen dumper, and we'll start from scratch on learning TI-99/4A assembly language. I promise that it'll be real fundamental.

Next month: Another utility to be used with TI BASIC.

FUNDAMENTALS

OF

TI-99/4A ASSEMBLY LANGUAGE

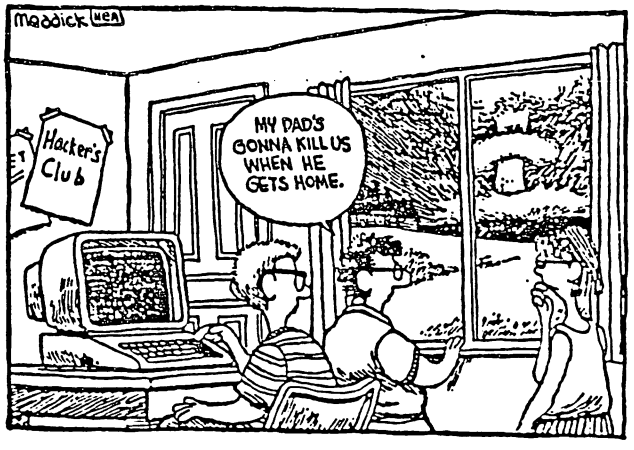
By Bill Jones

Tab Books recently introduced a new book for the TI: FUNDAMENTALS OF TI-99/4A ASSEMBLY LANGUAGE by M. S. Morley.

If you've been looking for a supplement to the Editor/Assembler manual, be sure to take a look at this one. FUNDAMENTALS was written using the classic approach to teaching assembly, but kept in mind the fact that most readers have no technical background in micro's.

Chapter one explains what assembly language is, chapter two details the 4/A, and the rest of the book is devoted to progressively harder exercizes that introduce each new instruction in the 9900's instruction set.

This is the best book I have seen yet for those of us who are exploring the potential of our machines using its own language. This book lists for $11.50 and should be available at Daltons or Waldens. On a scale of one to ten, I rate this book as an eight.



LIBRARY BITS
by Dennis Sherfy

If you have a full system, you ought to have DM1000. This is an assembly language program which will run from Extended Basic with the included "LOAD" program.

I have held off writing about DM1000 for several months because I have heard that there are bugs in the program. I think that some of them have been corrected. I'm told that one verified bug is that it will not properly function if you have a very large number of programs on a single disk. There may be other problems, but I am not aware of them. Now, the program.

DM1000 stands for Disk Manager 1000. The DM1000 disk contains two Display Variable 80 files which can be printed with TI-WRITER. These are the detailed instructions. The files are named DM1000DOC! and DM1000DOC2. DM1000 performs the same functions as Disk Manager cartridge, but it has at least two additional functions that are very usefull.

First, if you have deleted a file from a disk, but have not written over that file, you may recover it. When you delete a file, it is removed from the disk's directory, but the file still remains on the disk. The disk directory says that the sectors containing the program are available, and the file may be erased when you save the next program. If the file has not been written over by another program, you may salvage it with the "Recover File" option.

The second unique feature of DM1000 is it's ability to "sweep" a disk. This means that the disk's directory is wiped clean, and the disk can be used like a newly formatted disk. I used to reformat disks that I wanted to clear. Now I "sweep" them. Sweeping takes a fraction of the time it takes to reformat a disk.

While you can perform this next option with Disk Manager in two sweeps, DM1000 can do it in one step. This option is called "Move". When a file is copied onto a new disk, it is deleted from the orginial disk.

DM1000 has an option to copy a whole disk, in either sector by sector mode, or in "bit map" mode. In sector by sector mode, each sector on a disk is reproduced on the copy disk. In the "bit map" mode, only those sectors which are used will be copied. You can copy an entire disk with DM1000 in no more than 4 passes.

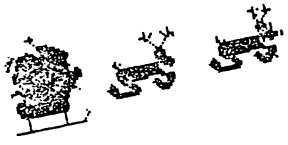
Other functions include adding or deleting file protection;adding or deleting disk protection; copying, renaming, deleting files, renaming, initializing, cataloging disks.

As far as I know, this program is available without charge. It was created by Bruce Caron of the Ottawa, Ont. TI-99/4A User's Group.

SAVE $$$ SAVE $$$ SAVE $$$ SAVE
SAVE $$$ SAVE $$$ SAVE $$$ SAVE

The Indianapolis Amateur Radio Convention (Hamfest) on Saturday July 12 and Sunday July 13 will feature the largest market of new and used electronic equipment including computer equipment and amateur radio equipment. The commercial building and the 4 flea market buildings will be open both days. The Hamfest will be at the Marion County Fairgrounds in S.E. Marion County on Troy avenue. Take the Southeastern Avenue exit from I 465 and follow the signs to the Marion Co. Fairgrounds. The Hamfest will start at 6 am on Saturday July 12 and close for the night at 5 pm. The Hamfest will reopen at 6am on Sunday and end at 4 pm on Sunday. There will be technical forums on Sunday. To reserve a place in the flea market buildings or in the commercial buildings, write the Indianapolis Hamfest, P.O. Box 11776, Indianapolis, IN 46201. See you at the Hamfest. Malcolm Mallette, WA9BVS

```
      TITL '* FAST SCREEN DUMP FOR BASIC *'
*
*   BY VICTOR A. KELSON   6 NOVEMBER 1984
*
      DEF  PRTCHR       THE SUBR NAME
*
*   CALLING SEQUENCE:
*         CALL LINK ("PRTCHR",INDX,STR$)
*
*   INDX - (INTEGER) NUMBER OF THE CHARACTER TO CONVERT TO PRINTER IMAGE
*         (0 TO 767) THE BASIC PROGRAM MUST HANDLE THE ROW,COL CONVERSION!
*   STR$ - (STRING) THE PRINTER IMAGE OF THE CHARACTER IN THE INDX POSITION
*
      REF  VMBR,VMBW,VSBR,VSBW
      REF  XMLLNK,NUMREF,STRASG,ERR
*
*   SYSTEM EQUATES
*
FAC   EQU  >834A
*
*   MY BUFFERS
*
CHRBFR BSS  >08          THE CHAR DEFN
STRBFR DATA >0800        THE STRING LENGTH
      BSS  >08           THE PRINTER IMAGE BUFFER
*
*   HERE WE GO! FIRST, GET THE CHAR POSITION
*
PRTCHR CLR  R0           ARRAY COUNT
      LI   R1,1          FIRST ARGUMENT
      BLWP @NUMREF       GET IT
*
      BLWP @XMLLNK       CONVERT TO INTEGER...
      DATA >1200         ...
      MOV  @FAC,R0       IS IT OKAY?? [0,767]
      CI   R0,767
      JH   ERROR         OUT OF RANGE. TELL BASIC
*
*   GET THE TI CHARACTER DEFINITION AND PUT IT IN MY BUFFER
*
      CLR  R1            MAKE THE LSB IN R1 ZERO
      BLWP @VSBR         GET THE CHAR NUMBER
      SRL  R1,8
      SLA  R1,3          MULTIPLY BY 8
*
*   R1 NOW POINTS TO CHAR PATTERN IN VDP RAM
*
      MOV  R1,R0         MOVE TO R0
      LI   R1,CHRBFR     POINTER TO RAM BUFFER
      LI   R2,8          NUMBER OF BYTES
      BLWP @VMBR         GET THE CHAR PATTERN
*
*   NOW,CONVERT THE CHARACTER TO PRINTER IMAGE
*
      LI   R5,STRBFR+1   SET UP THE OUTPUT PTR
      LI   R6,8          AND THE OUTER LOOP COUNT
```



```
*
*   THIS SEGMENT OF CODE SHIFTS THE BYTES IN THE BUFFERTO THE
*   LEFT AND CATCHES THE CARRY BITS IN R0. R0 IS THEN STORED IN
*   OUTPUT BUFFER
*
OUTLP  CLR  R0           CLEAR THE RESULT BYTE
       LI   R1,8         LOOP COUNTER
       LI   R3,CHRBFR    BUFFER PTR
       CLR  R4           FOR INSURANCE
LOOP   SLA  R0,1         SHIFT ANSWER TO LEFT
       MOVB *R3,R4       GET A BYTE FROM THE BUFFER
       SLA  R4,1         SHIFT THE BYTE...
       JNC  SKIPIT       ...DID A "1" FALL OUT?
       INC  R0           YES. PUT IT INTO THE RESULT BYTE
SKIPIT MOVB R4,*R3+      PUT THE SHIFTED BYTE BACK...
       DEC  R1           AND GO BACK TO GET ANOTHER,
       JNE  LOOP         UNTIL THE LOOP COUNTER GOES TO ZERO
*
       SLA  R0,8         PUT OUTPUT BYTE IN MSB OF R0
       MOVB R0,*R5+      PUT THE RESULT BYTE INTO THE BUFFER
       DEC  R6           AND LOOP ON THE NUMBER OF OUTPUT BYTES
       JNE  OUTLP
*
*   NOW, PUT THE OUTPUT BUFFER INTO THE BASIC STRING
*
RETRN  CLR  R0           ARRAY PTR
       LI   R1,2         ARGUMENT NUMBER
       LI   R2,STRBFR    THE STRING BUFFER ADDR
       BLWP @STRASG      THATS ALL.
       RT
*
*   ERROR HANDLING ROUTINE
*
ERROR  LI   R0,>1600     BAD ARG ERROR
       BLWP @ERR
       RT
*
       END
```

```
10000 REM   SCREEN DUMP ROUTINE
10010 REM   VAK, 11/14/84
10020 OPEN #99:"PIO.CR",OUTPUT,DISPLAY
10030 REM  DEMO FOR GEMINI-10X  PRINTER
10040 REM  SET UP CHAR POINTER
10050 PSCD=0
10060 REM  SETUP 1/8" LINEFEEDS
10070 PRINT #1:CHR$(27);CHR$(48)
10080 REM  LOAD THE UTILITIES
10090 CALL INIT
10100 CALL LOAD("DSK1.BSCSUP","DSK1.PRTCHR")
10110 REM  LOOP ON THE 24 ROWS
10120 FOR ISCD=1 TO 24
10130 REM  SET UP GRAPHICS MODE
10140 PRINT #1:CHR$(27);CHR$(75);CHR$(0);CHR$(1);
10150 REM  LOOP ON CHARS IN ROW (32)
10160 REM  GET THE PRINTER VERSION FROM OUR UTILITY
10170 REM  AND PRINT THEM ON THE PRINTER
10180 FOR JSCD=1 TO 32
10190 CALL LINK("PRTCHR",PSCD,ASCD$)
10200 PRINT #1:ASCD$;
10210 PSCD=PSCD+1
10220 NEXT JSCD
10230 REM  SEND A CR-LF TO THE PRINTER
10240 PRINT #1:CHR$(13);CHR$(10);
10250 NEXT ISCD
10260 CLOSE #99
10270 RETURN
```

THE ASSEMBLY LANGUAGE FORUM - NUMBERS, NUMBERS, NUMBERS

by Vic Kelson

Well, I'm back from a wonderful week in sunny Cancun, Mexico, where I didn't even once think about computers -- it was great. As promised, though, I will make these articles monthly from now on.

For the next several months, I will use the column to present some of the basic concepts of assembly language (from now on, referred to as AL) programming. This information will be slanted toward the TMS9900, but is mostly applicable to other machines. If you're a BASIC, FORTH, or other language user, don't worry, these articles are intended for beginners, and even BASIC users can apply some of this stuff.

I intend to give a discussion at the meeting each month to cover the material in the newsletter article. This will give you a chance to ask questions, and a chance to get a more complete discussion of the material. Hope to see you there.

THIS MONTH'S TOPIC: Number Representation and Number Bases

In order to learn AL, (or really, for almost any language) you must first understand the way a computer looks at numbers. This is the topic for this article. I'm sure that you're all familar with the binary system to some degree, so I won't talk much about decimal-binary conversions.

We'll start out with a look at the various units of memory. The smallest quantity of memory that we can deal with is the "bit". The word "bit" is a contraction of "binary digit". A single bit is either a 1 or a zero. The TI has several instructions which allow you to deal with individual bits.

The second unit of memory in the computer is called the "byte". A byte is a group of eight bits, and is the basic unit of memory used by computers such as the Commodore, Apple, and Z-80 systems.

The TI can deal with single bytes, but the basic unit of memory on the 9900 is a 16-bit "word". A word can be pictured like this:

```
MSB                          LSB

 0 1 1 0 0 0 1 1  0 0 1 0 1 1 0 1

 high order byte  low order byte
```

This word contains the hex value 632D, or 25,389 in decimal. The terms MSB and LSB stand for "most significant bit" and "least significant bit". The advantage of a 16-bit word over an 8-bit word is that you can handle larger numbers. An 8-bit word can handle values from 0 to 255, while a 16-bit word can hold values from 0 to 65,535.

BINARY ARITHMETIC

Arithmetic is very simple with binary numbers. There are only 2 binary digits (0 and 1), so here are the addition rules:

```
  0    0    1    1
 +0   +1   +0   +1
  0    1    1   10
```

Notice that in the last case, a "1" is carried into the next higher digit. This is handled just like decimal addition:

```
          1     11    111
 101    101    101    101
 111    111    111    111
   0     00    100   1100  (5+7=12! It works!)

step 1 step 2 step 3 step 4
```

You can see the carries in the addition problem above. One problem arises, though. A computer's word is 16 bits wide. What happens in this case:

```
  111 1111    1  1  (carries)
   0110110110001101
+  1110101101001001
  10101100011010110
  ^
```

We have a 17-bit long result!! In this situation, the carried bit (pointed to by the arrow) is thrown away (into the "bit bucket") and not included in the sum:

```
  111 1111    1  1  (carries)
   0110110110001101
+  1110101101001001
   0101100011010110 (result)
                 1 (bit bucket)
```

We'll get into the actual identity of the bit bucket next month.

Binary subtraction has similar rules:

```
  10    1    1    0
 -1   -1   -0   -0
  1    0    1    0
```

When handling larger numbers, though, for example, two 16- bit numbers, you must borrow, just like decimal subtraction:

Sample problem: 7-8=-1

```
  1111111111111     (borrows)
  ^0000000000000111
- 0000000000001000
  1111111111111111 (16-bit result)
```

You see, the arrow here points to a borrow from the 17th bit, which doesn't exist. In this case, the 16-bit result is returned, and the extra borrow falls into the bit bucket.

TWO'S COMPLEMENT NUMBERS

The last example raised an important question: What do all those "1"'s mean?? It seems strange to think that 7 minus 8 might be 65,535, doesn't it?

This problem is actually handled pretty simply. If we allow numbers to be either positive OR negative, then it seems obvious from the example that the number represented here is really -1. How can we generalize this result for all cases, and how do we represent negative numbers?

To allow for the use of positive and negative numbers (signed numbers), the TI (and most other machines) use a technique called "two's complement". Here's how it works:

1) The MSB (most significant bit becomes the "sign bit". If MSB=0, then the number is positive. If the MSB=1, then the number is negative.
2) To negate (change the sign of) a number, first invert all the bits in the number (form the logical or one's complement), and then add 1.

Does it really work?? Let's try it:

Example: Negate the number 27 to get the binary two's complement, and then add it to 27 to check and see if it adds up to zero.

27 decimal=0000000000011011

Inverting the bits.. (take one's complement)

one's comp=1111111111100100

Finally, add 1 to the result to obtain two's complement negative

two's comp=1111111111100101 (-27 decimal)

Now, add 27 to this...

```
 1111111111111111 (carries)
^1111111111100101
+ 0000000000011011
  0000000000000000 (16-bit result=0)
                 1 (carry into bit bucket)
```

It works! You should notice two important properties of two's complement numbers:

1) In two's complement numbers, the MSB is the sign bit. For positive numbers, the MSB is zero; for negative numbers, it is one.
2) Regardless of the sign of the number (positive or negative), the LSB tells whether the number is even or odd. This property can be used in your programs to make the even/odd test. Simply test the LSB (we'll do it in a later lesson) - even numbers have a zero in the LSB; odd numbers have a one.

USE OF SIGNED AND UNSIGNED NUMBERS

On the TI, in AL, it is possible to treat any 16-bit quantity as either signed (positive or negative 15 bit numbers, plus a sign bit) or unsigned (16 bit positive numbers). Some high level languages (FORTH, for example) also allow both signed and unsigned operations on 16 bit data. It is important to remember that the range of signed numbers is [-32766,32767], while the unsigned range is [0,65535].

Challenge: What decimal number does the following binary number represent?? (hint: negate it)

1000000000000000 (16 bit word)

HEXADECIMAL NOTATION

I hope that you've tried doing some binary arithmetic on your own, particularly negating and subtracting numbers (have you tried negating and adding to see if the result is the same as subtracting??). If you have, you've probably found that binary numbers are a real hassle. That's why we use hexadecimal notation. Why hexadecimal instead of decimal?? Look at the following table:

binary	hex	binary	hex
0000	0	1000	8
0001	1	1011	9
0010	2	1010	A
0011	3	1011	B
0100	4	1100	C
0101	5	1101	D
0110	6	1110	E
0111	7	1111	F

Using four bits, it is possible to form numbers in the range [0,15]. By using a number system with 16 digits, we can map the numbers from binary to hex easily:

Example: Convert to hex:

```
1001 1100 0011 0101  binary

 9    C    3    5    hex
```

This is done by looking up the four bit sections in the table. Try converting the binary to decimal - it's a lot harder, because you handle each bit separately. Hex to decimal is much easier. By the way, you can do the same mapping with octal (base 8) to binary. Hex is more convenient because there are exactly 16/4=4 hex digits per word.

IN CONCLUSION

I'm sure that many readers have seen most of this stuff before, but in order to introduce more users to AL, this is a necessary step. If you want to learn AL, (or advanced FORTH) or just to improve your other programming, you should practice, and become comfortable with the use of decimal, binary and hexadecimal numbers. If you have any questions, ask me or any of the other assembly or FORTH users at the meeting. I'm sure anyone would be happy to help.

Next month, I'll continue with the AL tutorial, by introducing the TMS9900's registers and flags. SEE YOU AT THE MEETING!! Vic...

THE ASSEMBLY LANGUAGE TUTORIAL - FLAGS AND REGISTERS

by Vic Kelson

Last month, we discussed the various number representations that we use in AL programming. This month, we'll go one step further, by examining the flags and registers of the TMS9900.

CPU REGISTERS

The TMS9900 registers can be represented like this:

```
PC  d d d d d d d d d d d d d d d d
WP  d d d d d d d d d d d d d d d d
ST  L> A> EQ C  OV OP X  -  -  -  -  -  .INT MASK.
```

(the d's are binary digits)

All of the 9900's registers are 16 bits wide. They all contain information which is important to the AL programmer:

PC - The "program counter". PC contains the "address", that is, the position in memory, of the next instruction to be executed.

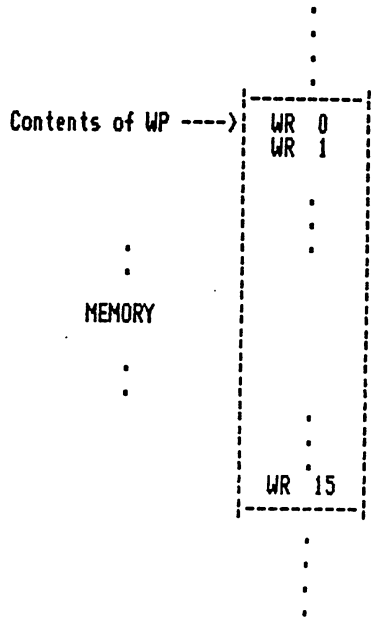
WP - The "workspace pointer". WP contains the address of a 16-word (remember, a word is 16 bits) block of memory, called the "workspace". These 16 words of memory are called "workspace registers" - we'll discuss them in a minute.

ST - The "status register". ST contains "flags" (single bits which we can use to control the execution of our program.

These three registers are the only registers which are actually on the TMS9900 chip - a rather unusual way for a microcomputer to be set up.

WORKSPACE REGISTERS

Here is a picture which describes what the WP register actually does:



WP contains a value which is the address in memory of the block of 16 "workspace registers". The AL programmer can use these workspace registers in a lot of very special ways. This is the largest difference between the TMS9900 and other micros, such as the 6502, 6800, and Z-80. Confused? We'll discuss this further in next month's article.

THE STATUS REGISTER AND FLAGS

I mentioned the ST register and its contents briefly above. Here is a more detailed discussion.

The 16-bit statuus register (ST) is broken up into the followng pieces:

NAME	BIT #	MEANING
L>	0	Logical greater than
EQ	2	Equal
C	3	Carry
OV	4	Overflow
OP	5	Odd Parity
X	6	Extended Operation
-	7-11	Reserved
INT. MASK	12-15	Interrupt Mask

These are the meanings and uses of the flags:

L> - LOGICAL GREATER THAN

This flag is used to make logical (or unsigned) comparisons of two 16-bit numbers. For example, if you compared the following two binary numbers:

```
0100 1101 1011 0010  = 4DB2 hex
0011 1110 1111 0000  = 3EF0
```

The first number is larger than the second, and the 9900 will set (make into a 1) the L> flag.
Comparing these numbers:

```
1100 0000 0111 1111  = C07F
0010 0000 0111 1111  = 207F
```

will also set the L> flag, because the 16-bit number C07F is larger than 207F.

A> - ARITHMETIC GREATER THAN

This flag returns the result of comparing 16-bit signed numbers. If you remember from last month, twos-complement negative numbers have the MSB set to 1. In the first example (above) the A> bit would have been set to 1, because the 15-bit value 4DB2 is greater than the 15-bit value 3EF0 (the sign bit is 0, so they're positive numbers).

In the second example (above), the first number, C07F, is negative (the sign bit is 1). In two's complement notation, C07F is LESS THAN 207F!! This is the difference between the L> and A> flags, and our first glimpse of the difference that the programmer sees in unsigned and signed numbers.

It won't be the last time we see this...

EQ - EQUAL FLAG

This one is easy. The TMS9900 sets this flag to one if the result of a comparison is that the numbers are equal, and 0 if they're not.

The L), A), and EQ flags are also set after arithmetic operations, by comparing the result to zero. This is a useful feature.

C - CARRY FLAG

Remember the "bit bucket" last month?? Well here it is. Whenever an addition or subtraction is performed, this flag is set to 1 or zero, according to the value in that "bit bucket". Look over last month's examples, and set the C flags fo them.

OV - OVERFLOW FLAG

This flag is used when doing twos-complement arithmetic. The flag is set to one if the sign bit (the MSB) of the result of an addition is different from the sign bit of the two operands (note that this is only done if the sign bits of the two operands are the same). Example:

```
  0111 0000 0000 0000  = 7000 hex
+ 0111 0000 0000 0000  = 7000
  1110 0000 0000 0000  = E000
```

If the numbers are twos-complement, this means that we added two positive numbers and got a negative!! This sets th OV flag.

The OV flag is also affected by subtraction, multiplication, and division operations. I recommend that you try to figure out what happens for these cases, and look it up to see if you're right. We'll talk about it at the meeting.

X - EXTENDED OPERATION

The X flag is set if the processor is executing an extended operation. Extended operations are beyond the scope of the tutorial, and will not be used here.

INTERRUPT MASK

The interrupt mask is used to tell the processor how to react if it is interrupted. Again, this is beyond the scope of the tutorial.

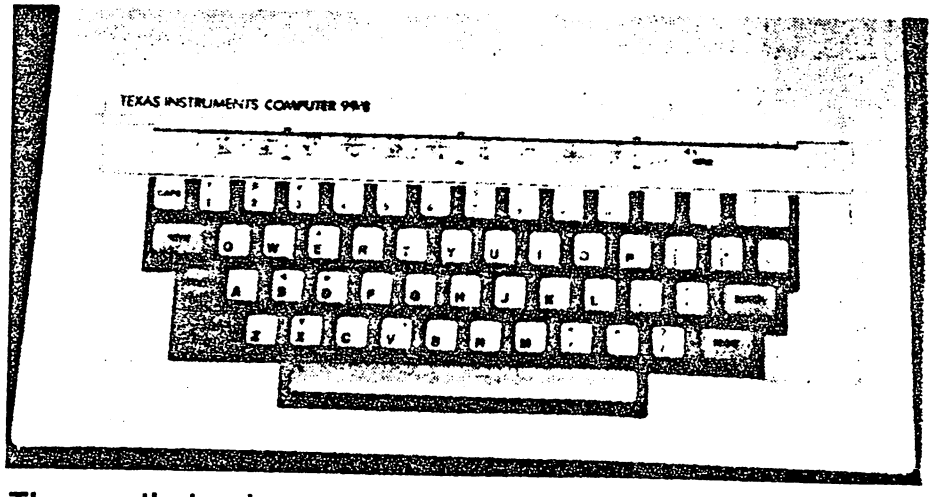
IN CONCLUSION

I hope that this article gets you started on the way to understanding the flags and registers of the TMS9900. I strongly recommend that you read sections 3.1 through 3.1.3.8 (pages 39-44) of the TI Editor/Assembler manual.

Next month, we'll discuss the TMS9900's addressing modes, and talk about how to move numbers around. We'll even write a small AL program!

The 99/8

Remember all of the rumors about a "new" computer TI was developing? Inside this Newsletter, read "At the Faire", which appeared in the December, 1984 issue of MICROpendium. Below is a picture of "The one that got away."



The one that got away

Pictured above is the TI99/8 home computer. Only 250 of the 64K RAM machines were produced before Texas Instruments made the decision to leave the home computer market. The machine features a built-in p-Code system, 64K of CPU RAM and 16K of VDP RAM. Program modules are loaded vertically into the top of the machine.(Photos courtesy of Dave Wakely) (Chicago Users Group)

KIBBIT INSTRUCTIONS

By Greg Goodwin

Kibbit is a TI FORTH graphics program which draws from the keyboard or joysticks. Please note: this is a revised version the Kibbit program which is saved on the FORTH DEMO disk in the HUGger Library.

THE KEY FUNCTIONS OF KIBBIT ARE:

CR: Clears the current work screen.
Space Bar: Aborts program.
Fire Button (or "Q"): Toggles Graphics modes. A sprite will appear on the tip of the line. The letter is the current mode.
D: Draw
U: Undraw
T: Dtog (If line undrawn else draw one.)

Either joystick or keys may be used in drawing (make sure the alpha lock is up!).

If you have any problems getting this version to run, leave me a message on the HUGbbs or thru the Users Group P.O. Box (also let me know what you think of it). A future version will include the saving of screens, and mixing with user programs.

Forth....Write.....Greg

WORDSEARCH 1

By Bill Cagle

```
COMPUTERANRJCLOSEWZNAFMQSDY
IUAPIMBJGKJNLTCXFFCPCDGOVIX
YPIFHRFOCLVYIBYTUPNILJEXXSE
EMMMWISDGITPXYQCWERWSSYFBKZ
PTLXJUEEFZGRVIEYUYXMOAUQYDN
XCIVBSTUGKIIECUSXOUMTIBNPRB
TJDGTAEEYGWNVEMULATORQRLTIN
DZFHECSMUHKTNBWDWQWBLUBJKVE
RMZKPRMBQJLETRAVMVSHORJLYEZ
ANBIHICFAQNVHITTUPTUOUYFUSN
NTULUJAUHHQVVXBNVKSAPWMXWPW
DBFKSLLBBBOYSYHKKDGPQCTXENN
OPQASRIKYOENBYDHRVZKCYEWRRP
MOVKBAIMTGFZDJGCOMPUTERDRPM
IEYHOKLMEVVNWXVFFOLNCAMGICR
ZRJMCKLLSFIPEUGQAWYSREIMZLD
ECUILAYVCLUFBTUFHAMUNCNBXYN
ZDMMGVUSICZATAAYTAVUCRAGTBM
KJOASSEMBLYJKKYDQQACNSLJUTR
AVLNQXTLIHAYJQBLPWCHUKGCKEO
BNINTERNALOCJUFMBUKVATBIHVK
XEIBZYJDNEINCWSICLEARMHBGTU
HESQMUZDKGPPCTIGUZXRXFVHUEC
ECQVNROTCESSIAPPENDWQWCAGLE
```

WORD LIST
TIGERCUB
CAGLE
BYTES
SAVE
DISKDRIVE
INPUT
COMPUTER
SECTOR
RANDOMIZE
PRINT
FOR
NEXT
TERMINAL
ASSEMBLY
EMULATOR
BASIC
GOSUB
CLEAR
CLOSE
OUTPUT
INTERNAL
UPDATE
APPEND
NUM

KIBBIT REVISED

By Greg Goodwin

```
( K I B B I T  by G. Goodwin)
50 VARIABLE JOY 50 VARIABLE JRT 0 VARIABLE SELECT
HEX 3800 SSDT : INIT_TURTLE
8050 2828 2828 1000 8A SPCHAR
8040 2424 2424 1800 8B SPCHAR
8040 201C 0808 0800 8C SPCHAR 22 22 5 8C 1 SPRITE ;
DECIMAL : KIBBIT INIT_TURTLE GRAPHICS2 BEGIN
1 JOYST CASE 04 OF -1 JOY +! ENDOF 252 OF 1 JOY +! END OF ENDCASE
CASE 04 OF 1 JRT +! ENDOF 252 OF -1 JRT +! ENDOF ENDCASE
18 = IF 1 SELECT +1 ENDIF SELECT @ CASE
0 OF DRAW 138 1 SPRPAT ENDOF
1 OF UNDRAW 139 1 SPRPAT ENDOF
2 OF DTOG 140 1 SPRPAT ENDOF
3 OF 0 SELECT ! ENDOF ENDCASE JRT @ JOY @ 1 SPRPUT
JRT @ JOY @ DOT ?KEY 13 = IF GRAPHICS 2 ENDIF
? KEY 32 = UNTIL TEXT ;
KIBBIT ;S
```

--NEW FORTH--

LOADS FROM EXTENDED BASIC AND OTHER CARTRIGES!

by Bill Jones

Most HUGgers know Greg Goodwin as something of a patriarch to the Hoosier Users Group's FORTH interest group. He is also a professional programmer who spends his days writing assembly programs on the 'big boys' and comes home at night to relax by writing programs in FORTH on his TI. His popular KIBBIT graphics program in FORTH is well known among us and his programs frequently appear on the bulletin board.

Greg didn't stop there though. He wasn't satisfied with programming in a language that only people with the Editor/Assembler (or the new CorComp disk controller) could use. He picked the FORTH source code apart and came up with the modifications that made it possible to load the language through several TI cartriges. Greg has been able to make FORTH load from Extended BASIC and TI Writer. Now anyone with a disk drive and memory expansion can use FORTH even if they only have one of these cartriges!

I'm told that two different modified versions are used, one that loads from Corcomp, Minimem and the Editor/Assembler, and one that loads from the Extended BASIC and TI-Writer cartriges. With Extended BASIC, loading is done the same way as any other assembly language program file, then you CALL LINK and you're in FORTH! Imagine that as an autoload program. Greg tricked TI-Writer into thinking that FORTH is one of its utility programs.

Since it would be nice to combine BASIC and FORTH programs on our bulletin board, I asked if it would be possible to switch back and forth between the two languages. He said although he hadn't tried it yet, a routine similar to FORTH's MON word could possibly cold-start Extended BASIC. One hitch to that is that both BASIC and FORTH expect to load from disk 1.

Greg decided to release this version of FORTH to the users groups, and says that the Hoosier Users Group may send one copy of the disk to each group that asks, for $5. He cautions that he is not giving it over for public domain, but sends it to each group along with a limited license to produce copies for members of that group. It contains the FORTH that TI released to public domain with some exceptions. It loads with other cartriges and it has a fix for a bug in the graphics mode that TI never fixed. He also added routines to allow speech and sound.

This new addition to our library will open FORTH up to a second generation of TI FORTH users who, I'm sure, will find FORTH as fascinating and useful as I and many other FORTH enthusiasts already have. Thanks Greg!

This disk is, of course, free for the copying to active HUG members; other clubs wanting a copy may send $5 to the Hoosier Users Group at the address below.

HOOSIER USERS GROUP DISK
PO BOX [redacted] 2222
INDIANAPOLIS, IN [redacted]
46206-2222

THE FORTH PAGE

by Greg Goodwin

This month: A catalog program.....

This program uses 3 screens and the type of formating used will allow for a regular print out to a printer if desired. To execute the Program just type: CATALOG. If you desire a print out type: SWCH CATALOG UNSWCH. The Float utilitys must be loaded to run this program. And if a print-out is desired then the Print utility must be loaded. Enjoy!!!!

This program, printed with TI Writer didn't print the @.

********** please make corrections as follows.

```
( SCREEN #1        by G. Goodwin  Catalog  9/24/84)
VOCABULARY DSR IMMEDIATE
DSR DEFINITIONS
BASE->R HEX
0 VARIABLE =TYPE=
0 VARIABLE [BUF] 26 ALLOT
PABS@10 + [BUF] 1400 FILE DIRECT
: OPEN
  DIRECT
  SET-PAB
  INTRNL
  RLTV
  INPT
  F-D" DSK1."
  OPN ;
  R->BASE -->
```

```
( SCREEN #1 Catalog 9/24/84)
PABS @ 10 + [BUF] 1400 FILE DIRECT
```

```
( SCREEN #2
: !LEN! 2 SPACES SEC @ . ;
: AV CR ." SEC= " TSEC @ . ." USED=" TSEC @ SEC @ - 2+ .
." AVAI=" SEC @ . CR CR
: FILE_TYPE =TYPE= @ CASE
05 OF ." PROGRAM" [SP] @ 3 + SPACES ENDOF ENDCASE ;
```

```
( SCREEN #3
=TYPE= ! C@ + 1+ COUNT OVER OVER FAC SWAP CMOVE STR FAC>
PAD COUNT >R DROP F->S =TYPE= @ IF . ELSE TSEC ! ENDIF
  FILE_TYPE PROT @ IF ." Y" ENDIF ;
  OPEN BEGIN RD DROP [BUF] C@ WHILE SCAT REPEAT CLSE CR CR CR ;
```

```
( SCREEN #2  BY G. GOODWIN )
BASE->R HEX 0 VARIABLE [SP] 0 VARIABLE TSEC 0 VARIABLE SEC
: !LEN! 2 SPACES SEC @. ;
: AV CR ." SEC= " TSEC  . ." USED=" TSEC  SEC  - 2+ .
: FILE_TYPE =TYPE=  CASE
00 OF_DUP 0 > IF AV ELSE DROP ENDIF ENDOF
01 OF ." DIS/FIX" !LEN! ENDOF
02 OF ." DIS/VAR" !LEN! ENDOF
03 OF ." INT/FIX" !LEN! ENDOF
04 OF ." INT/VAR" !LEN! ENDOF
05 OF ." PROGRAM" [SP]  3 + SPACES ENDOF ENDCASE ;
R->BASE -->
```

```
( SCREEN #3     BY G. Goodwin)
BASE->R HEX 0 VARIABLE PROT
: SCAT
  CR [BUF] COUNT DUP 10 SWAP - >R TYPE R> SPACES
[BUF] COUNT + DUP DUP COUNT FAC SWAP CMOVE FAC>
FDUP F0< IF 1 PROT ! F->S MINUS
ELSE 0 PROT ! F->S ENDIF
=TYPE= ! C + 1+ COUNT OVER OVER FAC SWAP CMOVE STR FAC>
PAD COUNT >R DROP F->S =TYPE=  IF . ELSE TSEC ! ENDIF
5 R> - SPACES + COUNT FAC SWAP CMOVE STR FAC>
PAD COUNT [SP] ! DROP F->S SEC !
  FILE_TYPE PROT  IF ." Y" ENDIF ;
: CAT 0 0 GOTOXY CLS
  OPEN BEGIN RD DROP [BUF] C WHILE SCAT REPEAT CLSE CR CR CR ;
FORTH DEFINITIONS : CATALOG DSR CAT ;  FORTH
R->BASE
```

```
**********************************************************************
*                  TI FORTH DISK COPIER  -  BY RON RUTLEDGE          *
**********************************************************************
          1         2         3         4         5         6
   ....+....0....+....0....+....0....+....0....+....0....+....0.....
 0 ( 6 PASS SINGLE-DRIVE DISK COPIER 7/27/84)
 1 ( enter COLD, 1 LOAD, and DUPLICATE )
 2 BASE->R DECIMAL -SYNONYMS 0 VARIABLE BIG 15358 ALLOT
 3 : ?# EMPTY-BUFFERS 0 BLOCK 10 + @ 256 1024 */MOD SWAP 0= 0= + ;
 4 : PAK CR ." PRESS ANY KEY" 52 GPLLNK KEY DROP CR CR ;
 5 : LMD ." LOAD MASTER DISK" PAK ; : LCD ." LOAD COPY DISK" PAK ;
 6 : DUPLICATE CLS 0 0 GOTOXY LMD 0 DISK_LO ! ?# DUP DUP
 7 DISK_SIZE ! DISK_HI ! LCD
 8 ." ...FORMATING COPY DISK..." 0 FORMAT-DISK
 9 0 DO CLS 0 0 GOTOXY LMD
10 I 15 0 DO DUP I + DUP . CR BLOCK BIG I 1024 * + 1024 CMOVE LOOP
11 CLS 0 0 GOTOXY LCD
12 15 0 DO DUP I + DUP . CR BLOCK BIG I 1024 * + SWAP 1024 CMOVE
13 UPDATE FLUSH LOOP
14 DROP 15 +LOOP 1 DISK_LO ! ; R->BASE
15
```

==========

This program, which I found on a California electronic bulletin board, will copy a single sided disk using one disk drive in six passes and a double sided disk in twelve passes. Thanks to Bob Utter for doing the debugging. To use this program do the following:

1) Select the Load and Run option in the Editor/Assembler and execute "DSK1.FORTH". (It would help to read chapters 2, 3, & 5 in the FORTH manual).

2) FORTH will display "BOOTING..." first, then a menu, then "TI FORTH" with a cursor underneath it. At this point enter the following commands:

   A) -EDITOR
   B) 1 CLEAR
   C) 1 EDIT

This will load a blank screen with only the position lines on top and the line numbers to the left. Now key in the screen shown above.

3) Next, press <FUNC> BACK. This will return you to command mode, then enter "FLUSH". This will save your screen to disk. **WARNING** It is very easy to mess up a FORTH disk so do your work on a copy, not on your master!

4) If you have a printer, you may print the screen by entering "-PRINT" followed by "0 TRIAD". If you haven't corrected SCR #72 do the following:

   A) Follow the procedure for editing SCR#1 except DO NOT enter the clear command. That will wipe out the entire screen. (Enter "72 EDIT").
   B) At the beginning of line 5 change PAB_ADDR to PAB-ADDR (_ to -)
   C) If you have a parallel printer, in line 4 change " RS232.BA=9600" to " PIO". Note the space preceeding either entry.

5) The FORTH copier may be used like this:

   A) If you were working in FORTH previously, enter "COLD".
   B) Enter "1 LOAD". If your screen is bad, FORTH will tell you now.
   C) Enter "DUPLICATE". FORTH will begin by asking for the master disk.

Note the following about the FORTH Disk Copier. First, it will copy anything regardless of the protection scheme as long as it uses sector boundries as defined by TI (ie, everything). Second, it will not write to a disk that has the write protect notch covered but it doesn't check to see that you put the correct disk back in either like the Disk Manager 2 does. Third, unless you are using a double sided drive you do not need to initialize your diskette, the copier will do it for you. Fourth, the entire disk will be copied regardless of how much is actually used on it. If you are having problems hang tight, there will be a demo at the next meeting!

Double Sided FORTH

By Jim Vincent

Midwest Cooridnator
99'ers Users Group Association

So you have double sided disk drives on your TI 99/4A. When you got your copy of TI FORTH from the user group you immediately tried copying it to a double sided disk using the Disk Manager II, didn't you? Didn't work so hot did it? Well, this article will show you what to do to make TI FORTH work with your double sided drives.

TI FORTH uses direct sector addressing to read/write screens to disk. Each screen is 1024 bytes or four single density sectors long. To be compatible with the Disk Manager module and normal file I/O for initial loading of the FORTH program, the FORTH disk uses the standard disk header and directory sectors. Since there are only three files on the disk, this means sectors zero through four are allocated to this overhead operation. The first file, FORTH (actually just a short assembly language loader program begins in sector 22 (hex) as usual. It is followed by the actual FORTH program file which is also handled via normal I/O routines and occupies up to sector 4C. SYS-SCRNS occupies all remaining sectors on the disk, for single sided thats 138 (hex). Thus the file SYS-SCRNS takes up not only the sectors from 4D to the end of the disk, but also has an extent that takes up sectors 5 to 21 (hex).

Now since TI FORTH uses direct sector addressing, it expects screen 3 (the boot screen to be in sectors C thru F. If you use Disk Manager to copy these three files from a single sided disk to a double sided disk, the Disk Manager is able to put the whole SYS-SCRNS file on disk contigously. No extents are required, thus what was at sector 5 to 21 is now at sector 168 to 184. It is any wonder FORTH acts strange? The boot screen only contains garbage!

To remedy this situation we must copy a single sided disk, sector for sector to a double sided one, and then doctor sectors zero and four to comply with the Disk Manager's standards. Then to use the extra capacity we will update a couple screens. Your first step however, must be to initialize a double sided disk using the Disk Manager II. Next load FORTH and set DISK_LO to zero. Load the -COPY screens and use the command FORTH-COPY to dupe your single sided FORTH to the double sided disk. If you don't have two drives, use Disk Manager to copy all three files but then use FORTH to copy screens 1 thru 9. Here's the technique:

```
n BLOCK UPDATE ( where n is screen number to be read from old disk)
FLUSH ( writes screen to new disk, move up to 5 screens at a time)
```

Now, edit screen 3 to add the following commands:

```
180 DISK_SIZE ! ( supports double sided capacity per disk)
360 DISK_HI ! ( supports two double sided drives)
```

Next time you boot FORTH it will recognize screen 175 as part of disk 1 and screen 185 as part of disk 2. Now let's fix the commands in the -COPY screens. Edit screen 39. The value 90 appears once in DTEST and twice in FORTH-COPY. Change all three occurances to 180. Next edit screen 40 with the following:

Line #	Current Contents	New Contents
3	168	2D0
5	2000	2028
5	12 + 26	12 + 0201 SWAP ! DUP 14 + 24
10	165	2CD
13	4016	802C

Now let's make our FORTH disk compatible with Disk Manager II. Here's the word you need to do it:

```
HEX
: DOUBLE-FORTH 0 BLOCK UPDATE DUP A+ 2D0 SWAP !
     DUP 10 + 2028 SWAP ! 12+ 0201 SWAP !
     1 BLOCK UPDATE DUP E + 2A0 SWAP ! DUP 1C +4D20 SWAP !
     DUP 1E + 2805 SWAP ! 20 + F029 SWAP ! FLUSH ;
DECIMAL DOUBLE-FORTH
```

Now that you have full use of your double sided drives I'd like to issue a couple of challenges.

1. Figure out how to alter the FORTH command FORMAT-DISK to format a double sided disk.
2. Alter FORTH to support CorComp's double density controller.

Good Luck !

DOUBLE-DENSITY FORTH

by J. W. Vincent

Editors Note: The following article was copied from the September, 1984 issue of "The National 99'er", Newsletter of the 99'ers Users Group Association; Bakersfield, California.

This article is intended for all TI FORTH users who have (or plan on having) double density and/or double sided disk capabilities. While the techniques described should work with any disk controller capable of double density, the author's CorComp 9900 Disk Controller Card is the only one that has been tested. The purpose of this article is to illustrate both how to access the additional screen capacity and how to modify the FORTH words and disk to be compatible with the new format and Disk Manager. Throughout this article lowercase letters used in a FORTH definition will indicate a variable value to be entered. The following terms will be used to refer to the various formats a FORTH disk may have.

```
 90 SCRN or SSSD - the original 90 screen single sided single density format
180 SCRN         - either a SSDD or DSSD disk when comment applies to both
360 SCRN or DSDD - a double sided double density disk
           SSDD - a single sided double density disk
           DSSD - a double sided single density disk
```

The first step is to use Disk Manager to format (initialize) a 180 or 360 SCRN disk. Next you must copy FORTH from the 90 SCRN disk to the new 180 or 360 SCRN disk. The disk copy feature of CorComp's Disk Manager will do this properly for you. If you have two drives, the FORTH-COPY word in the -COPY screens will also do it properly (do 0 DISK_LO ! first). However, if you are using TI's Disk Manager II, after copying the three files you must use FORTH to copy screens 1 to 9 because Disk Manager II puts them in the wrong place! To do this, enter the following for each of the nine screens.

```
n BLOCK UPDATE ( where n is the screen number to be read from old disk)
FLUSH          ( after inserting the new disk - note: up to five screens may be entered at a time)
```

Now edit screen 3 of your new disk and add the following commands:

```
x DISK_SIZE !  ( where x = 180 or 360 as appropriate)
y DISK_HI   !  ( where y = x times 1, 2, 3, or 4 depending on the number of drives you have)
```

Unfortunately, TI FORTH does not privide a method for configuring each drive individually. Therefore, the user must be conginzant of which screens are available on each drive when there are differences between them. At this point, FORTH can be booted and it will recognize the full capacity of your 180 or 360 SCRN disk. You can create, edit, list, and load from screens greater than 89. However, neither Disk Manager nor FORTH-COPY will recognize this disk as having more than 90 screens. To fix this problem you must modify the -COPY screens (39 and 40), the disk header (sector 0) and, the SYS-SCRNS file header (sector 4). First edit screen 39. Change the value 90, which appears once in DTEST and twice in FORTH-COPY to 180 or 360 as appropriate. Next edit screen 40 as follows:

```
Line  3 - change 168 to 2D0 for 180 SCRN or 5A0 for 360 SCRN
Line  4 - change 944 to 1244 for SSDD or DSDD (no change for DSSD)
Line  5 - replace entire line with:
          DUP 10 + 2028 SWAP ! DUP 12 + a SWAP ! DUP 14 + 24 0 FILL
          where a = 0201 for DSSD, 0102 for SSDD, or 0202 for DSDD
Line 10 - change 165 to 2CD for 180 SCRN, or 59D for 360 SCRN
Line 13 - change 4016 to C02C for 180 SCRN, or C059 for 360 SCRN
```

Next edit screen 33 to modify the FORMAT-DISK word to:

```
: FORMAT-DISK 1+ a 33616 ! 18 SYSTEM ;  ( where a = 258 for DSDD, 513 for SSDD, 514 for DSDD)
```

DOUBLE-DENSITY FORTH, cont'd.

Finally, you need to create a word that will modify the header sectors on your new dis This word only needs to be executed once since copies of this disk, once it's modified, will not require modification. Here is the way to do it:

```
HEX 0 DSK_LO !              ( removes disk fence)
: DD-FORTH 0 BLOCK UPDATE   ( read screen 0 and mark as updated)
    DUP  A + a SWAP !       ( a = 200 for 180 SCRN, 5A0 for 360 SCRN)
    DUP  C + b SWAP !       ( b = 944 for DSSD, 1244 for SSDD or DSDD)
    DUP 10 + c SWAP !       ( c = 2028 for all versions)
    DUP 12 + d SWAP !       ( d = 201 on DSSD, 102 on SSDD, 202 on DSDD)
       38 + C8 FF FILL      ( flag all sectors as in use)
       1 BLOCK UPDATE       ( read screen 1 and mark as updated)
    DUP  E + f SWAP !       ( f = 2A0 for 180 SCRN, 570 for 360 SCRN)
    DUP 1C + g SWAP !       ( g = 4D20 for 180 or 360 SCRN versions)
    DUP 1E + h SWAP !       ( h = 2805 for 180 SCRN, 5205 for 360 SCRN)
       20 + i SWAP !        ( i = F029 for 180 SCRN, F059 for 360 SCRN)
    FLUSH ;                 ( write modified screens to disk)
DECIMAL DD-FORTH            ( execute it)
```

Now your new high capacity copy of FORTH is fully compatable with Disk Manager, the FORTH format, copy, test, and header words and your double density and/or double sided disk drives and controller. Enjoy!

WORDSEARCH #2

Hidden in the following wordsearch are 20 TI FORTH words. Look for the answers in the June HUGger Newsletter, and Good Searchin!

WORD LIST

ABORT
BOOT
BUFFER
CREATE
COMPILE
DECIMAL
DIGIT
EDIT
EXECUTE
ERROR
FORTH
LOOP
MAGNIFY
PAUSE
RANDOMIZE
SCREEN
SMUDGE
SPRITE
UPDATE
VOCABULARY

```
XRDCQIIHCGBRPOOLVELNDHMRKMMCAUZA
DSZQBXWXINOAVFKXUWDYQXNWBEXDJVUM
GNISLCQOIMJUUQOMYKTPPDAURMVDNTLJ
KPMTBFFYRALUBACOVAGPSTVXCJUCAACA
LKNSKOQEFATUFJPWCUPPONINZNXMQBLP
YPMAWZIWEFZNMIIMDUFVMMOQEQRKCXMV
NANIPCNDJCDNHAHJEFCIBFTAFSXFAXLP
EPNIFSULZDSIMKGSOGFXXUXYBMOZKGSG
TBUVJDNFFFGLZYINACPTLGZDERTRXYWR
XOZEDBTIPJLUGQHAIVAFSUIDTHWQGEDX
NLANVJGAOTKKTUXEBFSRWGBHVHWDMXIK
QBVZPMULTWMDBWYQVOYIICOKFKVMREJU
VEDITSXHMIMBWFAXRNRTUROIEZJCUCEF
BWRIEXDYCEGDUMSYWYRTUIPMEFQJCUQI
BSALDOGPTKENBKIAJRSFJTPPPJPAXTWY
UVWOQPKYDDXYMSODXCLLTGFOLIHVFEIE
FZNSBSRIEJEJUHCDRRZQERRORZLWXXAX
FONYHLVBNEPCWXKNREFVXLBVJEIETGUG
EIXMLFFZMJEBIWOOIAJJAHVOIVBHPGCF
RNEERCSKOCNLIMCRATEHGUGUONSDWAQP
KVOOBGZCUJKRBLALQEUAFVCRGTQXABVF
VFZDOOSQUPDATENLEIWXIHRDRVXXANQZ
EVFSBNRHEISCDPIYKIQKXTGALVDHQXGN
ICWHUVSCAXRFSOIIMGLEZIMODNARJRFV
NMCSYOYMYKACTSPRITEHYTUHPWJRLRXJ
```

FORTH CORRECTIONS
PART I

Editor's Note: The following article is printed, in part, in the HUGger Newsletter via the August, 1985 issue of MicroPendium. These corrections were so extensive and due to space limitations, one or two corrections will appear each month. **By TOM FREEMAN**

CORRECTIONS TO THE FORTH SYSTEM DISK

I have found the following errors in the system disk as I received it (even th version with screens 58-59 dated OCT83).

The first is simple. Line 1 in screens 53, 54, and 55 contains the wor VDPSET2. This should be SETVDP2.

Second, lines 9 and 10 in screen 58 should be switched, and the new line should read:

VDPMDE @ 4 < IF SMTN 80 0 VFILL 300 ' SATR ! ENDIF

If (INIT ALL SPRITES) is on this line, it may be deleted as it is not compile anyway. Note the ' before SATR not !, which was in the "corrected" versio labelled 20OCT83 LAO on line 0. You may add 1NOV84 TSF to this if you wish

Third, in line 9, screen 59, between >R and SP@ should read:

8 SLA SWAP 00FE AND OR

Line 0 on this screen should read 20OOCT83 LCT).

CORRECTIONS TO DOUBLE FORTH

Thanks to Jim Vincent for publishing the Double-Sided Forth information and to others for reprinting it.

Unfortunately the disk with screens only on it does not in fact copy with the T Disk Manager. The problem is in bytes 15-17 (the sector access chain) anc possibly in byte 10, which contains the total number of records, in this case twic the number of sectors.

Thus in screen 40 make the following changes/additions to Jim's note:

line 11 change CA02 to 9A05 for 180 SCR, 3A0B for 360 SCR

line 12 change 2250 to 22D0 for 180 SCR and 360 SCR

line 13 change 1403 to 2A03 for 180 SCR, 5703 for 360 SCR (retain previous change)

For DD-Forth make the following addition between DUP E etc. and DUP 1C etc:

DUP 12 + j SWAP ! (j = 4005 for 180 SCR, E00A for 360 SCR)

And the following changes:

DUP 1E etc. (h = 5505 for 360 SCR)

20 + etc. (i = F056 for 360 SCR)

I do not own a double-density card, so I could not test these changes, but I an reasonably sure they work. I also decided to retain the original word DISK HEAD and made a new one as follows:

```
: 2DISK-HEAD DISK-HEAD 0 BLOCK DUP A + 2D0 SWAP ! DUP 10 + 202
SWAP ! DUP 12 + 201 SWAP ! 200 + DUP E + 2CD SWAP ! DUP 12 +
9A05 SWAP ! DUP 1C + 22D0 SWAP ! DUP 1E + 2A03 SWAP ! 20 + C02C
SWAP ! UPDATE FLUSH ;
```

These values are for DSSD drives. You can substitute the appropriate value for SDDD or DSDD.

For what it's worth, NONE of these is really necessary if you use Forth itself t copy the disks (or any mass copier) since it doesn't make use of the disk map. Bu it's nice to be complete! And you might give a disk to a friend who doesn't hav Forth set up yet.

WORDSEARCH # 3

There are 26 words relating to the BBS.

WORD LIST

MESSAGE MODULE
CHARACTER BUFFER
FILE MODULE
LINEFEEDS
ERRORS
BACKSPACE
SCROLLING
PRINTING
SCREEN
LOGOFF
SYSOP
HEADERS
ENTRY
HUGBBS
EDIT
PREVIEW
SAVING
BELL
CHAT
DOWNLOAD
GOODBYE
HELP
LIST CALLERS
DEFAULT OPTION
UPLOAD
TERMINAL EMULATOR

```
HUGBBSSINBJHXBIJWCECAPSKCAB
ZYPMCCVLITJKYIXCASYXYDDGKFE
GIWFPRTBWSRORREOOMWKWPMCCVL
OPMQNEYFTIUTIDFPREVIEWYJWCL
OTOAUESRELLACTSILSGDETWXTGX
DKIUFNCYPGBIMBXPWSYWTLELNRS
BTLRFVTCNOTTFRVWLAEVANKIVLL
YKJORAEMIOEMEGXERGJKGWTDKHI
EUVTPRKLOGOFFMDBXEQKQNTLNON
AZTAFLCSUFFMCUXUFMVMIMNSNTE
XBVLKDHUWDQSQSXSTOVROKPYOJF
IBQUVXANBGOUBCVSHDPDDQMTIDE
SZHMBFRUXZQMSROJAUNUUXZBTRE
YZKEHBAVUIEQEOAASLRHLHHDFVD
SGPLKTCHATTIPLAEIEXDEFYCOUS
OFBALSTGQRWQULIISCTHMKRTTJX
PNNNFREDCJBLOIVFDACJVRTXLLR
SIYISRRHOTTWANDYADVFOANRUIR
AWKMJHBYGWETJGLDYAMIXAENASQ
BASRMNUZXSNWMBTTADJSNAFMFTF
WPLEHOFIUFYLTUHEMORZHGWLEQG
FYGTQNFNCVEQOKZFAYLDSNTODZM
MJNOFFESFKIRVANTOPIPLVKJAEC
USREQSREDAEHMSDGIQANUUTIMEE
```

ADDITIONS AND MODIFICATIONS TO FORTH SYSTEM DISK

By TOM FREEMAN

I have found these useful on my system disk, in addition to those such as PAGE or SIZE which have been published elsewhere. For instance, how do you tell which base you are in? BASE @ . will always give you 10 since that is the value in the current base! The following will give the answer in decimal without changing the base:

```
: BASEINDEC BASE @ BASE->R DECIMAL . R->BASE ;
```

Those of you dumping information to a printer frequently may find typing SWCH and UNSWCH annoying. The following will make it easier:

```
( PRINTER WORDS TSF 3DEC84)
BASE->R        HEX
: PINDEX SWCH INDEX CR UNSWCH ;
: PLIST SWCH LIST CR UNSWCH ;
: PCR SWCH CR UNSWCH ;
: PVLIST SWCH VLIST CR UNSWCH ;
: P" 22 WORD HERE COUNT SWCH TYPE UNSWCH ;
R->BASE
```

The first three are obvious, but they do save typing time. PVLIST is a problem because it only uses 40 columns. This is probably OK since you won't use it often, but if you wish to change it, find the definition of VLIST on screen 43, copy it using PVLIST in place of VLIST and 4E in place of 25 near the end of the second line.

P" which of course replaces ." was a bit more complicated because of the way in which ." is used. I had to go back to the original definition to figure out where to put the SWCH and UNSWCH.

PDUMP to replace DUMP doesn't appear above because I felt it important to use the full 80 columns of the printer. The following will construct PDUMP: (apologies to Peter Geltner of Los Angeles, who gave it to me but has never published it)

1) make a new word DUMP10 exactly like DUMP8 except 37 in place of 1F in screen 42, line 7.
2) PDUMP is then exactly like DUMP except replace 8 with 10 in scr 43, line 3 and DUMP8 with DUMP10 in line 4.
3) Memory can be saved by defining the parts that are the same as new words, and using these in DUMP8, DUMP10, DUMP, and PDUMP before and after changes.

The Forth editors always give the screen number in decimal. I prefer to have it also in HEX if I'm using HEX. For the 64-column editor the following will do it: on screen 26, in the definition of SCRNO, between BASE->R and R->BASE replace what is there by DUP . BASE @ DECIMAL 16 = IF ." HEX...decimal= " . ELSE DROP ENDIF

For the 40-column editor go to LISTA on screen 34, replace DECIMAL with BASE->R, AFTER SCR # " insert the same code as in the last paragraph, then R->BASE after LOOP.

COLOR CHANGES IN FORTH

You may wish to change the foreground and background colors in text, graphics, multi, and split modes. I wish to use white on transparent, for instance, since I have a monochrome monitor, with the text area below the 64-column editor a contrasting black on white. The following information will tell you where to do it. XY will always refer to a foreground color of X and a background of Y (both in HEX, see your E/A manual).

1) TEXT mode—screen 51, line 9, where you see 0F4 7 VWTR change F4 to XY
2) text in GRAPHICS mode—screen 52, lines 6 and 10, where you see F4, use XY
3) screen text in 64-column editor—screen 54, line 6, 0F0 VFILL, 0F0->0XY (I left this unchanged, but 17 looks better on a color TV or monitor)
4) text in SPLIT mode—screen 55, line 6, 0F4 VFILL, 0F4->0XY
5) text in SPLIT2 mode—screen 55, line 11, 0F4 VFILL, 0F4->0XY
6) cursor in 64-column editor—screen 23, line 5 SPCHAR 0 F, F->X (F, th default is white, but it must contrast with the background of SPLIT)

WORDSEARCH # 4

There are 23 words relating to the BASIC and EXTENDED BASIC

WORD LIST

CALLCHAR
CALLCHARPAT
CALLCHARSET
CALLCLEAR
CALLCOINC
CALLCOLOR
CALLERR
CALLGCHAR
CALLHCHAR
CALLINIT
CALLJOYST
CALLKEY
CALLLINK
CALLLOAD
CALLLOCATE
CALLPEEK
CALLSAY
CALLSCREEN
CALLSOUND
CALLSPGET
CALLSPRITE
CALLVCHAR
CALLVERSION

```
DOSPGLHUDCALLHCHARBKJAFESMM
ICDNOLACCWQDRSEXHXCNVNERUJJ
CTALAJTAPRAHCLLACRYKEIUYESL
ALPLIMZSWXMFUTHXOISEWSKQNOE
LWKCLDRNOUJNXZGTIHRFGPXOLYC
LCNIOCLLACTCYRYJGCIBKGNVTAA
VMIUDUOGGUPNAGACSAQKDFJLLZL
CDLSDBBLGCLTWLSLPLJFEBALAXL
HULMGYXUOYLROVLKKLDAFJCZTUC
AMLSFICINRKUOALJGKQFVHNOGDH
RDAAZWAQNEDACDALOERRARJPOYA
XYCCALLGCHARSYCOEYLREDNICIR
FCWTAALKDXEHZVZUTQSXFPEXVRR
PQGEKBLTPDORGLBJLELTFDKEXAR
EAZGYYOEQCALLINITPZBOQEWEKE
TYUPQTAKILFMXRQSKTYVAPFLRNL
AIYSBCDXAMNOISREVLLACHCQOML
CUDLSGZUZUSBOHESZNHGFLWKBFA
OZFAXAFAOEWDVPHOFTDDLYYYWOC
LBHCECUSPWZFLJLWZJKAETHLGNA
LIAYMHLVABQLCDJEZNCJNRRQJPL
LKUPCLHEFXARDNUOSLLACMODNLL
AFSRAVYFKCDJIKOIBHVVQJFYGMA
COTCUPZELLZHSCEOETIRPSLLACC
```

This is the last of the series of FORTH corrections and updates as appeared in the August, 1985 issue of MICROpendium. Next month, Editor/Assembler changes.

BINARY SAVE OF YOUR SYSTEM DISK

Various people have given suggestions for this, starting with Craig Miller. The following may make it easier.

First, of course, decide which editor you want to use, then use it to make the changes above. Then change the definition of MENU on screen 20 to read 272 256 DO etc. This will give you garbage at the top of the screen when you first load everything, but will be useful later, as all of screen 20 can be used as the menu.

Next set up a blank screen that will load all your options in the order you wish. Here is mine:

```
( ORDER OF LOADING FOR BSAVE) BASE->R DECIMAL FORGET
-SYNONYMS
: TOM1 ; 51 LOAD 6 LOAD ( -SYNONYMS -TEXT -NEWORDS)
: TOM2 ; 57 LOAD 52 LOAD ( -GRAPH -GRAPH1)
: TOM3 ; 54 LOAD ( -GRAPH2)
: TOM4 ; 55 LOAD ( -SPLIT)
: TOM5 ; 45 LOAD ( -FLOAT)
: TOM6 ; 53 LOAD ( -MULTI)
: TOM7 ; 42 LOAD 39 LOAD 72 LOAD 89 LOAD
      ( -DUMP -COPY -PRINT -PWORDS)
: TOM8 ; 22 LOAD ( -64SUPPORT EDITOR)
: TOM9 ; ( BORDER FOR PROGRAMS WITHOUT OPTIONS)
: TOM10 ; 83 LOAD ( -BSAVE)
R->BASE
```

Notice how often my name appears (I like it). These are dummy defining words that provide borders so that I can FORGET just as much as I want. Notice that I also FORGET -SYNONYMS at the top so that words won't be duplicated. This also necessitated using xx LOAD directly, rather than the defined words. Now, once you have FLUSH'd everything, type COLD and when it is done, x LOAD, where x is the screen you used for the above, then insert a *copy* of the disk and type, in DECIMAL, ' TASK 21 BSAVE . (Include the last period.) This will print out the next available screen, and leave much of the disk for other things. Note that your original disk retains the order of loading, so that if anything happens to the BSAVE'd disk, it is easy to reconstruct. You can now edit screen 20 to take full advantage of your new MENU. My version is below, and includes other options that I placed after the BSAVE portion.

```
ALREADY LOADED: [FORGET BACK TO (XXX)]
(TOM1) -SYNONYMS -TEXT -NEWORDS
(TOM2) -GRAPH -GRAPH1   (TOM3) -GRAPH2
(TOM4) -SPLIT           (TOM5) -FLOAT
(TOM6) -MULTI (TOM7)-DUMP -COPY -PRINT
(TOM8) -64SUPPORT     - (TOM9) options
(TOM10) free, re-enter if forgotten
New words: forget with TOM1
BASEINDEC EB PAGE SIZE NEW BYE-flushes  2FORMAT-DISK
Printer words: loaded with -PRINT
P" PCR PLIST PVLIST PDUMP PINDEX
Available options:
-ASSEMBLER -CRU -BSAVE -TRACE -TESTHEAD -FORTHTRAN
-DOUBLECOPY (clears all memory first)  -DECOMPILE
-SEARCH -2FORTHCOPY
```

Note that I indicate how to FORGET each section. Also, anything beyond the 40th character on a line will be printed on a new line on your screen, so arrange this screen carefully.

Now all you need to do is edit screen 3 to BLOAD what you have. Here is mine:

```
( WELCOME SCREEN ) BASE->R HEX F0 7 8 SYSTEM 10 SYSTEM
0 0 GOTOXY ." Loading...TI FORTH" 10 83C2 C! DECIMAL 21 BLOAD
FORGET TOM10 ( eliminate BSAVE)
: -ASSEMBLER 34 LOAD ; : -TRACE   50 LOAD ; : -CRU   42 LOAD ;
: -FORTHTRAN 46 LOAD ; : -TESTHEAD 44 LOAD ; : -BSAVE 43 LOAD ;
: -DOUBLECOPY 48 LOAD ; : -DECOMPILE 51 LOAD ;
: -SEARCH 55 LOAD ; : -2FORTHCOPY 33 LOAD ;
1 VDPMDE ! 0 DISK_LO ! 180 DISK_SIZE ! 540 DISK_HI !
PAGE MENU  : TOM10 ; ( BORDER PRESERVING OPTIONS)
R->BASE
```

Now FLUSH again and you are ready. If you type COLD, you should get the whole thing back, with a nice neat menu on the whole screen.

SHORT FORTHTRAN

Here is the shortest FORTHTRAN I know (mainly because all comments are eliminated). Apologies to Mike Amundsen for this one, but it is modified to allow multiple screen transfers at once. When it is loaded the first time brief instructions are written on the screen. Note that in order to use quotes on screen I had to make a new word, appropriately called ", since this can't be put inside of ."

```
( SHORT FORTHTRAN 1DEC84 TSF) BASE->R DECIMAL
                : " 34 EMIT ; : INS1 CLS 0 9 GOTOXY ." ENTER
 FIRST SCREEN #, NUMBER OF SCREENS  TO MOVE, THEN:" CR CR ." DSK
-SCR OR SCR-DSK" ; HEX : INS2 ." TYPE INS1 TO REPEAT OR FORGET "
 " ; 0 VARIABLE FILBUF 50 ALLOT PABS 2 A + FILBUF 1900 FILE FILT
RAN  : SETFILE FILTRAN SET-PAB SQNTL DSPLY VRBL 50 REC-LEN ;
 DECIMAL     : PUTFILE   OUTPT FILTRAN OPN OVER + SWAP DO I
     BLOCK                CR CR ." FILE TRANSFER IN PROGRESS..
" CR CR 16 0 DO DUP FILBUF 64 CMOVE I REC-NO 64 WRT 64 + LOOP
 DROP LOOP CLSE BEEP ." DISK FILE COMPLETED. " CR CR INS2 QUIT ;
 : GETFILE INPT FILTRAN OPN OVER + SWAP DO I DUP BLOCK
  CR CR ." FILE TRANSFER IN PROGRESS..."
  CR CR 16 0 DO FILBUF 64 BLANKS
I REC-NO RD DROP DUP FILBUF SWAP 64 CMOVE 64 +
 LOOP DROP BLOCK DROP UPDATE LOOP CLSE FLUSH BEEP
 ." SCREEN COMPLETED"  CR CR INS2 QUIT ;      R->BASE      --)
( SHORT FORTHTRAN, P.2)  BASE->R
: INS ." ENTER FILE DESCRIPTOR WORD-USE THE FORM" CR
." F-D" " ."  DSKx.XXX" " CR CR ." THEN TYPE " ;
: SCR-DSK   CLS 9 3 GOTOXY ." SCREEN TO DISK TRANSFER"  9 4
    GOTOXY ." ========================="
   CR CR CR CR CR SETFILE BEEP INS " ." PUTFILE" "  CR QUIT ;
: DSK-SCR   CLS 9 3 GOTOXY ." DISK TO SCREEN TRANSFER"  9 4
    GOTOXY ." ========================="
   CR CR CR CR CR SETFILE BEEP INS " ." GETFILE" "  CR QUIT ;
INS1     R->BASE
```

FORTH DECOMPILER

Gives the Definition of FORTH Words.

by Bill Jones, Indy

Lots of times I've been testing out a program and created a test word in FORTH to try out what I'm working on. Since I usually enter it from the keyboard rather than store it on the disk, I sometimes forget how I invented the work. This little program can bring back the definition of the word I created or one that is already compiled in the system. It's a pretty neat trick that follows the words backward and gives them back to you. This particular procedure was written by Jim Vincent of the Milwaukee Area users group, but I found it listed in the L.A. 99ers group newsletter. I'm passing it along to the HUGgers just as it was listed. Jim's style is pretty compressed and I haven't made any attempt to expand it for clarity.

Also this month, I've been working on a routine to convert FORTH screens into display 80 files that TI-WRITER can read. If you've ever wanted to send FORTH code to someone else via modem you know how useful that can be! I will be presenting that program to the FORTH interest group at the next meeting and you'll see the listing here next month. It features special instructions to strip carrage returns that TI-WRITER inserts as well as the end of file at the end. Along with it, I'll be giving a good tutorial on how to make FORTH read and write regular disk files.

TI RELEASES FORTH SOURCE CODES

FORTH ASSEMBLER SOURCE CODE: These two disks contain the portion of TI Forth written in assembler. The disk contents is as follows:

PART 1: (ASMSRC) is the dictionary entry for the 250 or so primitives that are present when Forth is booted. This is loaded into memory at >A000.

PART 2: (DRIVER) is the code for the I/O system and support for Forth. It contains the disk and screen I/O, the allocation of user variable space, the stacks etc. Because it is more efficient to rearrange memory from the way that it defaults in the Editor/Assembler, this section also includes (in the UTIL* files) those portions of the E/A utilities that Forth requires and assembles them to different addresses. A small portion of code is also placed into the console RAM at>8300 for speed reasons.

Also on part 2 is the program called BOOT. After Forth is loaded using ASMSRC and DRIVER, Forth can (after loading the file words) save an image of itself to the VDP RAM and write this image to disk as a program file. BOOT is used to read this image and to reconstruct the Forth system from the image. When booting the Forth system in the normal manner, the file FORTH is the object code of BOOT and the file FORTHSAVE is the memory image of the system. Note that if the size of the system changes, BOOT will have to have some addresses modified to work correctly.

```
Screen 150
( FORTH word decompiler - JWV 7-NOV-84 ) BASE->R HEX
-FIND (.") DROP DROP 2- CONSTANT
ADQ -FIND LIT DROP DROP 2- CONSTANT ALIT -FIND BRANCH DROP DROP
2- CONSTANT ABRAN -FIND 0BRANCH DROP DROP 2- CONSTANT AZBRAN
 ( DITTO)
: .DQ 2E EMIT 22 EMIT 20 EMIT ;
: .Q 22 EMIT 20 EMIT ;
: DQ? DUP 2+ @ ADQ = IF .DQ 4 + DUP COUNT TYPE DUP C@ + DUP 2 MOD IF 1- THEN
THEN ;
: :? DUP @ 8334 = ; ( TESTS IF WORD STARTS WITH DOCOLON )
: ;? DUP @ @ 8340 = ; ( TEST IF WORD IS SEMIS )
: P? DUP6@ 2- = ; ( TEST IF WORD HAS @ AS A PRIMATIVE DEF)
: D? @ DUP @ 06A0 = SWAP 2+ @ 832E = AND ; ( TEST FOR DOES>)
: .ID 2+ NFA ID. ; ( PRINT WORD NAME )
: UN: -FIND IF DROP 2- :? IF CR .  : " DUP .ID BEGIN 2+ ;? IF ." ; " CR DROP
ELSE DUP @ :? OVER P? OR OVER D? OR IF .ID ELSE U.  THEN DQ? THEN DUP 0= UNTI
DROP ELSE ." Not a colon def'd word" THEN ELSE ." No such word" CR THEN ;

R->BASE ( ) ( ) ( ) ( )
```

Direct Disk Access-A FORTH Approach by Bill Jones

There have been several requests to continue printing articles on Forth programs so here's a small one dealing with a subject I've been working on for a couple of years now. I've published listings of programs to do disk interfacing. Recently, I investigated bypassing the DSR, talking directly to the controller chip for the disks. Two types of I/O (input/output) systems are used to communicate with the disks. The "official" I/O system of the 9900 microprocessor is the CRU or Communications Register Unit. The CRU bits are addressed much like memory, but using special instructions that turn off the memory and turn on communication. An alternative method is to use a memory address but connect a device to that location instead of memory. This kind of I/O is called "Memory Mapped I/O".

The disk system uses CRU bits to turn on the card, turn on the disks, select the side and so-forth. The status register on the controller, command register and read/write registers are memory mapped I/O at the end of the address space used for the controller card (hex 4000 thru 5FFF). Because Forth preshifts the CRU bits for you, CRU addressing begins at hex 880. From an assembler's point of view, those addresses appear at beginning hex 1100. Note, everything on the TI controller must be reversed from what manufacturers' documentation says. Because of the way the electronics of the card, all the bits are reversed. A command which would be binary 1011 would be 0100 instead. On CorComp and Myarc controllers those bits don't need to be flipped.

In the example program we are using two disk controller addresses that come on when we enable the card. Memory address 5FF8 writes a disk operation command to the disk controller chip. Address 5FFE is usually the place to write information to the floppy disk, but for a seek command, we must put the number of the track we want to seek to. Also for this example, we are using three CRU I/O bits. 880 will turn on the card and allow access to all addressed devices on that card. (The CRU bits are always active so you can turn the drives on and off even if the card is not on.) You will not be able to send a command to the controller chip unless the card is on.

My example demonstrates two commands of the controller chip, seek and restore. When the numbers for those commands are put into the command address and the disk is turned on, a head move operation will be performed. To turn on the disk, you must start a timer on the card running and then select the disk you want. In the example, TIME-ON starts the time by turning the timer off and back on again. LITE-ON selects DSK1. DSK-OFF clears all 16 (hex 10) possible CRU I/O bits used by the card. WARNING! IF THE DISK CONTROLLER CARD'S LIGHT IS ON AND YOU TRY TO DO A NORMAL DISK READ OR WRITE IN FORTH, YOU WILL LOCK UP THE COMPUTER! Always use CARD-OFF or DSK-OFF when you've finished experimenting.

Notice the INVERT instruction. On a TI controller card, all the commands to the controller chip and all data coming back must be reversed. If you are experimenting with a Non-TI card, make this a no-op instruction or remove it from the other words. A 40 track drive can seek between tracks 0 and 39, if you try to go beyond those limits, your drive will bump its head against the stop till the controller finishes its count. Don't do it. It's hard on the drive. Also, restore the head when you start and finish. The disk will be disoriented when you go between our demo system and the one Forth uses. Starting at track 0 makes it easier for the controller chip to reorient itself to the track it is seeking.

```
 0 ( FORTH DISK ACCESS continued...      WMJ 86 ) BASE->R HEX
 1
 2       5FF8 CONSTANT COMMAND-REG    ( STORE THE COMMAND HERE )
 3       5FFE CONSTANT WRITE-REG      ( STORE THE TRACK No. HERE )
 4       12   CONSTANT seek           ( COMMAND FOR SEEK )
 5        2   CONSTANT restore        ( COMMAND TO GO TO TRK0 )
 6
 7        : INVERT FF XOR ;      ( : INVERT ; FOR NON-TI CARDS )
 8 -CRU
 9 : CARD-ON    880 SBO ;          : CARD-OFF 880 SBZ ;
10 : LITE-ON    884 SBO ;          : TIME-ON 881 SBZ 881 SBO ;
11 : DSK-ON     CARD-ON  TIME-ON LITE-ON ; ( SET 3 CRU BITS )
12 : DSK-OFF    0 10 880 LDCR ;  ( CLEAR ALL CRU BITS )
13 : RESTORE    restore INVERT COMMAND-REG C! CARD-OFF ;
14 : SEEK INVERT WRITE-REG C! seek INVERT COMMAND-REG C! CARD-OFF ;
15                                                  R->BASE
```

To seek the head to a track, give the track number and type SEEK. If you used the RESTORE command when you began, the head should move smoothly across the drive to any track within range that you select. If you hear the head hit the stop, always restore to track 0 before continuing.

A final note: In full scale systems, commands would finish by checking status lines and then giving control back to you. Since that requires more detail than I wanted to cover here, I picked a quick and dirty way to make sure everything quit in good order. Turning the card off at the end of the instruction makes sure that you don't leave it on when you quit but may not give the timer enough time to finish your command. If you DSK-ON and give the seek or restore command in the same line, the timer should stay on long enough for the head to travel clear across the disk.

This example program comes from a set of Forth routines I am writing for my own disk operating system for my TI. If you would like a copy of the source code documentation of the segments on how to read and write sectors to the disk, where the control lines are addressed and how to use them, send me $4 and a self-addressed 9X12 envelope. The listing contains nearly a dozen screens of information about using the Western Digital 1771 floppy controller chip in the TI disk controller card. My address is: Bill Jones, 1305 N. Livingston Ave. Indianapolis, IN 46222.

Initialize disks with FORTH (double sided too) by Bill Jones, Indy ok

Have you ever wondered what would happen if you didn't have your disk manager cartridge? Mine came with a warning that said it could be destroyed by static. Without it, disks cannot be initialized and that means that the disks you have are all you can use. For me, the answer is FORTH. FORTH can do the system call that will initialize the disk, if your drives are single sided. Formatting the disk is only half the story though. A disk to be used in BASIC or other regular cartridges must also have initialization information written on it. (By the way, TI calls this the DX10 disk system.) FORTH can do this too, with special programming.

Working on this, I went one step further. The FORTH screens listed here can fully initialize a new disk either single or double sided without using the disk manager. Since FORTH can now be loaded by four different cartridges, most people with a complete system can use this program as an alternative way to initialize their disks. To use this program, the word FORMAT will format a disk when you use it in the form: n FORMAT, where 'n' is the disk number (beginning with zero). To initialize for BASIC, use the word INIT the same way. Both words operate on the assumption that the new disk will be made the same way as the system is currently configured. If DISK_SIZE is greater than 90, a double-sided disk will be created, otherwise it will be formatted and initialized as single-sided. You must be careful not to initialize a disk as double-sided if it was formatted as single-sided. INIT does not check boundaries to see if you gave it the right information. One last thing, these screens have to be loaded after -COPY since a word created there is used. I make sure this is done by pacthing in my code in place of the original FORMAT-DISK screen.

```
UNSWCH
SCR #90
  0 ( ONE/TWO SIDE DISK FORMATTER                          WMJ   25-JAN-85)
  1                                                         BASE->R DECIMAL
  2     -31923 CONSTANT TRACK#   ( FAC ADR OF TRACKS )
  3     -31919 CONSTANT SIDES    ( FAC ADR OF SIDES  )
  4
  5 : DSK-SIDES DISK_SIZE @ 90 > IF 2 ELSE 1 ENDIF ;
  6 : TRACKS    40 TRACK# C! ;
  7 : SIDE      DSK-SIDES SIDES C! ;
  8 : HOW-BIG   DSK-SIDES 2 = IF ." 2 SIDES"
  9             ELSE ." 1 SIDED" ENDIF ;
 10 : EXPLAIN   CR 14 14 GOTOXY ;
 11 : TALK      16 SYSTEM 10 12 GOTOXY
 12             ." FORMATTING DISK" EXPLAIN HOW-BIG ;
 13 : UNTALK    16 SYSTEM 7 0 GOTOXY ." FORMATTING FINISHED " ;
 14 : FORMAT    ( dsk ___ )
 15             TALK TRACKS SIDE 1+ 18 SYSTEM UNTALK ;       R->BASE
```

```
UNSWCH
SCR #91
  0 ( SETUP FOR DISK INIT   *LOAD AFTER -COPY*      WMJ   23-JAN-85 )
  1                                                      BASE->R HEX
  2
  3     ( MAP OF DISK HEADER INFORMATION )
  4         0 CONSTANT NAME!
  5         A CONSTANT SIZE!
  6         C CONSTANT SEC/TRK!
  7         D CONSTANT TAG!
  8        10 CONSTANT PROTECT!
  9        11 CONSTANT TRACKS!
 10        12 CONSTANT SIDES-DENS
 11        38 CONSTANT MAP!
 12
 13         0 VARIABLE TOP
 14         0 VARIABLE LO*
 15         0 VARIABLE OFF*                                    -->

SCR #92
  0 ( DICTIONARY FOR INIT                                WMJ   23-JAN-85 )
  1     : PUT>            TOP @ + ! ;
  2     : BYTE.PUT>       TOP @ + C! ;
  3     : FIND-BLOCK,     DISK_SIZE @ * OFFSET ! 0 BLOCK TOP ! ;
  4     : EMPTY-SECTORS.  TOP @ DUP 200 0 FILL 200 + 200 E5 FILL ;
  5     : ITS-NAME,       TOP @ !" DISK     " ;
  6     : SEC/TRK,        SEC/TRK! BYTE.PUT> ;
  7     : THIS-SIZE,      DISK_SIZE @ 4 * SIZE! PUT> ;
  8     : THESE           3 MAP!  BYTE.PUT> ;
  9     : TRACKS-SIDE,    TRACKS! BYTE.PUT> ;
 10     : SIDES?          DISK_SIZE @ 5A > IF 2 ELSE 1 ENDIF ;
 11     : #SIDES,         SIDES? 100 * 1+ SIDES-DENS PUT> ;
 12     : UNPROTECT,      20 PROTECT! BYTE.PUT> ;
 13     : "DSK",          44 TAG! BYTE.PUT> 534B TAG! 1+ PUT> ;
 14     : TO-FILL         DISK_SIZE @ 4 * 5A0 SWAP - 8 /  ;
 15     : START-USED      DISK_SIZE @ 5A / 2D * MAP! + TOP @ + ; -->

SCR #93
  0 ( INITIALIZE FORMATTED DISKETTE                      WMJ   23-JAN-85 )
  1     : USED-SECTORS    THESE START-USED TO-FILL FF FILL ;
  2     : AND-FILL        TOP @ EC + 14 FF FILL ;
  3 ( SAVE SYSTEM VARIABLES )
  4     : NOW,  DISK_LO @ LO* ! OFFSET @ OFF* ! 0 DISK_LO ! ;
  5     : -END- UPDATE FLUSH OFF* @ OFFSET ! LO* @ DISK_LO ! ;
  6     : WRITE ;
  7
  8 DECIMAL
  9
 10 : INIT NOW, FIND-BLOCK, EMPTY-SECTORS.                ( dsk ___ )
 11        WRITE ITS-NAME, "DSK", #SIDES, THIS-SIZE, UNPROTECT,
 12        40 TRACKS-SIDE, 9 SEC/TRK, AND-FILL USED-SECTORS
 13        -END- ;                                              R->BASE
 14
 15
```

THE FORTH PAGE

A look at Forths Inner Interpreter.

by Greg Goodwin

Although there is a lot of information that could be covered on the interpretion of a word we will look only at the actual execution of the 'word'. The Inner Interpreter is located at addresses 832e to 8346 on the TI. Address 833c is the DOEXEC routine. It is used by Forth words, and is called by Execute. For example the col definition of ZZ.
: ZZ ." Example I" CR ;

```
' ZZ CFA EXECUTE
^     ^  ^
FIND  CODE  EXECUTE WORD.
```

If we wanted to do the same thing in Forth Assembly we could enter:

```
CODE EXEC ' EXECUTE @() B,
        ( OR )
CODE EXEC *SP+ W MOV, 833C @() B,
```

A look at the code for Execute should help you to see that both examples do the same thing. Here is a full example.

```
BASE->R HEX ( EXEC-ASM DEMO )
  : ZZ CR ." Assembler Code Example." CR ;
CODE EXEC *SP+ W MOV, 833C @() B, CODE DING 3A06 @() BLWP, 0034 , NEXT,
: EXAMPLE ( Execute Forth Word in Code Word) ' ZZ CFA EXEC DING ; R->BASE
```

THE FORTH PAGE.....
.....Assembly Forth Mixing

By Greg Goodwin

Here is a short program that will allow you to run Program files created with the Ed-Asm in Forth. Note: the routines must be postioned above here, if it is a small program around FFC0 will work fine.

The assembly programs must not use the Forth Ram stack 8300-83FF. Some parts of that block may be used but consult the cpu ram pad map in the Forth users manual first. To create a program file of your Assemble program use the save utility on the part b disk. Check Ed-Asm manual for details.

the program.....

```
BASE->R HEX 0 VARIABLE BUFX 400 ALLOT
460 BUFX 1400 FILE SAVEIT
CODE XASM 06A0 , 7118 , 045F ,
: ATTRIBUTES ( DO PAB )
  SAVEIT SET-PAB ;
: VMOVE 2000 LD 1402 PAD 4 VMBR
1406 PAD 2+ @ DUP DUP HERE SWAP > IF ' XASM 2+ ! PAD @ VMBR XASM ELSE
 CR ." Cannot load to address " . CR ENDIF ;
: RUN ATTRIBUTES F-D" DSK1.FILENAME" VMOVE ." **DONE**" CR ; R->BASE
( TO EXECUTE TYPE RUN )
```

THE FORTH PAGE

By Greg Goodwin

Here is a small routine that will allow the user to input numbers in a Forth word! This word will require the Floating Point Routines to be loaded (-FLOAT).

```
HEX
: GET-REAL ( REAL NUMBER INPUT )
PAD F BLANKS 34 0 837C C! 0A SYSTEM
PAD 1+ F EXPECT PAD 1+ F -TRAILING 2DUP PAD C!
DROP VAL FAC> ;
: GET-INTEGER ( SAME AS REAL BUT FORCE INTEGER AND PUT ON STACK )
GET-REAL F->S ;
: EXAMPLE CR CR ." Enter Number->" GET-REAL CR ." YOUR NUMBER WAS=" FDUP F.
CR ." 5*" F.  ."=" >F 5 F* F.  ;
```

FORTH PAGE

by Greg Larson

Here's a quick and dirty Sector Dump I have been using for awhile. It will sector in two parts so you won't scroll past anything. To use, simply type the number of the sector you want to see, type "SD" and hit ENTER. The addresses shown in the listing are those of the disk buffers. The first address is used as an offset for the rest of the sector. The routine leaves the next sector number on the stack so that I could easily look at several sectors in succession. I imagine that this could be expanded to do the other functions of Disk Fixer, but I'm not sure that I'm up to doing it myself.

```
SCR #WHATEVER
 0 : SD ( Sector Dump , GBL 10/84 ) ( Sector # >>> Sector + 1 )
 1       DUP DUP CLS CR ." SECTOR NO. " . CR
 2       4 /MOD BLOCK SWAP 256 * + 128 OVER OVER DUMP + CR
 3       ." Print rest of Sector (Y/<N>)? " KEY CR 89 =
 4       IF 128 DUMP  ELSE DROP  ENDIF  CR EMPTY-BUFFERS 1+
 5 ;
```

THE HEART AND SOUL OF PERSONAL RECORD KEEPING

by Don Donlan

Those HUGgers who subscribe to the 99'er HCMagazine probably welcomed the 'CALL PEEKV and CALL POKEV' information related to the Mini-Memory Command module. With this article I would like to begin sharing some of the "secret" subprograms that are a part of the PERSONAL RECORD KEEPING module. A subprogram is an independent program which usually performs only one task or function. Because the subprogram performs only a single or rather narrow task, several subprograms are usually bundled together when making a program or, as in this case, a command module. Subprograms allow a programmer to perform tasks over and over again using the same code. A subprogram is like a meat grinder--it performs one task. But depending on what you 'feed' the meat grinder, you get different results. If I put pork in my 'meat grinder', I would get sausage; with beef, I get hamburger; with ham, I get the base for ham salad. Similarly, you can 'feed' a subprogram differing information to get different results. The information you 'feed' a subprogram is contained in PARAMETERS. These parameters are like serving trays that carry information back and forth between the main body of a program and the subprogram you select to use. Since you probably have more than one subprogram (or more than one task or function you wish to accomplish), you need to be able to select a particular subprogram. A name is given to each subprogram. In this way the computer knows which task or function to perform when you CALL the subprogram by name. Pages 71 through 90 of the TI 99/4A USER'S REFERENCE GUIDE that came with your computer tell about the color graphics and sound subprograms that are part of TI BASIC. You're probably already using the CALL CLEAR subprogram when you want to 'blank out' or erase the screen. Perhaps you've used the CALL SCREEN subprogram and passed along the required parameter (that is, a number from 1 to 16). The CALL SOUND subprogram has the ability to receive nine (9) parameters, but only three are required for the subprogram to do its job of producing a single tone. These subprograms are explained with examples in the USER'S REFERENCE GUIDE. They should give you some basic understanding of how subprograms work. You may want to start there.

For now let's return to the task at hand: defining and describing the subprograms that are available in the PERSONAL RECORD KEEPING cartridge. There are seven subprograms in the PRK command module. A program written in TI BASIC can use these subprograms if you plug in the PRK module and take option 1 (TI BASIC) BEFORE you try to load any program. I will go into more detail in following articles, but at this time I merely wish to list the seven subprograms, their parameters, and some comments on the function they can perform. In the final article of this series, I will be including a TI BASIC program that can be used on a system with a disk drive to access the records created by the PERSONAL RECORD KEEPING command module. This will provide greater flexibility in the use of this cartridge.

SUBPROGRAM NAME	PARAMETERS		DESCRIPTION OF FUNCTION
PREP SUBPROGRAM Code as: CALL P(V)	V	The number of bytes of characters you wish to reserve for use.	Prepares a work area or space that is to be used for the storing of information. Allocates a data area in VDP RAM.
LOAD SUBPROGRAM Code as: CALL L(V$,V)	V$ V	Data file name. Return code.	Loads a data file from a disk and indicates , in the return code, whether the subprogram successfully loaded the file.
SAVE SUBPROGRAM Code as: CALL S(V$,V)	V$ V	Data file name. Return code.	Saves a data file from the work area prepared by the PREP subprogram to a disk. Return code tells whether the SAVE subprogram did its job OK.

THE HEART AND SOUL OF PERSONAL RECORD KEEPING, cont'd

SUBPROGRAM NAME	PARAMETERS		DESCRIPTION OF FUNCTION
ACCEPT SUBPROGRAM Code as: CALL A(Y,X,W,C,V,L,H) or CALL A(Y,X,W,C,V) or CALL A(Y,X,W,C,V,F) or CALL A(Y,X,W,C,V$)	Y X W C V V$ L H F	Row number. Column number Field size or the number of characters Return code. Numeric variable. Character variable. Low value in a range of numbers. High value in a range of numbers. Field number of a piece of information within a PRK record.	Works much like the Extended BASIC 'ACCEPT AT' code. it 'captures' information from the keyboard as you key it and then echoes or displays that information on the screen for you to see. Various parameters may be used, depending on whether you want to 'read' numbers, numbers within a certain range, or a character string. The return code tells whether a valid, invalid, or function key was pressed. The F or field parameter can be used to validate data entered ONLY if your program has a second type of record--a header record--which it uses to cross-check the information you enter.
DISPLAY SUBPROGRAM Code as: CALL D(Y,X,W,V) or CALL D(Y,X,W,V$) or CALL D(Y1,X1,W1,V1, Y2,X2,W2,V2$, Y3,X3,W3,V3,...	Y X W V V$	Row number. Column number. Number of characters or field width. Number to display. Characters you want to display.	Similar to Extended Basic's 'DISPLAY AT' operation, this subprogram places either numbers or characters on the screen. Row number ranges from 1 to 24, while the column number ranges from 1 to 28. Multiple displays can be done or performed with one CALL D.
GETPUT SUBPROGRAM Code as: CALL G(R/W,REC,FLD,V) or CALL G(R/W,REC,FLD,V$) or CALL G(R/W,REC,FLD,MIS,V2) or CALL G(R/W,REC,FLD,MIS,V2$)	R/W REC FLD V V$ V2 V2$ MIS articles.	Read or Write code. Record number. Field number. Number written. Characters written. Number written. Characters read. 'Missing' return code.	Retrieves and places information from or into the work area defined by PREP subprogram. The subprogram identifies missing data (information not found) when reading a record. This allows the subprogram to tell the difference between a field that has never been entered ('missing') and a field that is all blanks or all zeroes. Saves space in memory by using this code. Values for these codes will be discussed in later
HEADER SUBPROGRAM Code as: CALL H(R/W,INFO,FLD,V) or CALL H(R/W,INFO,FLD,V$)	R/W INFO FLD V V$	Read or Write code. Header record item. Field number. Numeric variable. Character variable.	This is a core subprogram in PRK module. It is used to establish and maintain a kind of 'data dictionary'. This dictionary defines the kinds of information and the location of the information within the PRK records. Fourteen (14) kinds of information or header record items are stored here. This one record is the key to the entire 'data base' created by the PRK module. It gives the characteristics of each field within a record (name, type, size, decimals (if any)), storage space and position). The PRK file name, date, number of fields per record, the total number of records, the length of the header record itself, and the length of each data record are all a part of this one record.

Because of its central role in organizing the Personal Record Keeping command module, this is where we will begin our article in next month's newsletter.

THE HEART AND SOUL OF PERSONAL RECORD KEEPING

by Don Donlan (Part II)

The HEADER subprogram which resides in the Personal Record Keeping command module provides access to a "data dictionary". This dictionary defines the data that has been created by the PRK module. As I mentioned in the last article there are two formats or ways to code the "call" for this subprogram. For numeric information use [CALL H(R/W,INFO,FLD,V)] and to read/write character information use the format [CALL H(R/W,INFO,FLD,V$)]. If the "R/W" variable is "1" the subprogram will "read" information from the Header record. If the "R/W" variable is "0" (Zero), the subprogram writes information to the Header record. "V" and "V$" are used, depending on which kind of information you wish to retrieve (V for numeric; V$ for character). The "FLD" variable is sometimes ignored, sometimes required, depending on what number is in the "INFO" variable. "INFO" is a number between 1 and 14 which determines what Header record information you will read or write. A "FLD" number is ignored for the first 8 kinds of information; the rest require a "FLD" number. The fourteen kinds of information stored by the Header record are as follows:

1 = File Name up to 10 characters long.
2 = Day of the month (a number from 1 to 31).
3 = Month (a number from 1 to 12).
4 = Year (a number from 0 to 99).
5 = Number of fields per record (a very important number because it will determine how many times we will need to ask for information found for the information types 9 thru 14 shown below). This number is automatically incremented each time a new "highest numbered" field is defined
6 = Number of records in this PRK file (also automatically incremented each time a new "highest numbered record" is written).
7 = Size of Header record (length is automatically calculated and entered).
8 = Size of the Data record (this too is automatically calculated and stored).

Now we come to the Header record information that describes the individual fields within each data record. This sequence of information is repeated for each of the fields, up to the number of fields indicated in Header record information 5 (see above). If "FLD" is 1, the first Data record is defined; if 2, the second; if 3, the third and so on.

9 = Name of the data field (up to 10 characters long).
10 = Type of Data field: 1=character 2=integer 3=decimal 4=Exponential.
11 = Size of Data field. This depends on the type:
Character data fields are 1 to 15 bytes long.
Integer data fields are 1 to 10 bytes long.
Decimal data fields are 2 to 11 bytes long.
Exponential or scientific notation fields are 8 to 13 bytes long.
12 = Number of decimal places. For character and integer data, this is zero.
For decimal data, the number would range from 1 to "size" minus 1.
And for Exponential or scientific notation the number is 0 to 5.
13 = Amount of space required for the Data field (as set by Header subprogram).
14 = Position of this field within Data record (as set by Header

As you can see, with this kind of information you can reveal the data base structure of this PRK file. I don't know if you could use a BASIC program and the Header subprogram to write your own data base structure. It would be interesting to see what would happen. A typical Header record might look like this:

THE HEART AND SOUL OF PERSONAL RECORD KEEPING

by Don Donlan (Part II)

MYRECORDS_10_22_83_2_15_50_30_NAME_1_15_0_15_1_PHONE NO._1_15_0_15_16

I use the "_" character to set the "INFO" fields apart. Above would be header record for a file called "MYRECORDS" which was last used October 22, 1983. It says there are 2 fields for every Data record, 15 such records in the file. The length of the header record itself is 50 bytes; the size of each data record is 30 bytes. The name of the first of our two fields is "NAME", which will have up to 15 characters of information in it, so we'll reserve 15 bytes. This first field starts in position 1 of the Data record. Our second sample field is called "PHONE NO.", again a 15 byte character field that needs up to 15 positons in the record, so we'll start it in position 16. The sample Header record above could be the start of a phone and telephone number data base that was created by the PRK command module. If you have the PRK module you might want go ahead and make up such a sample file and store it on disk for use by the programs that we will be presenting at the end of these articles. That way you will be able to 'test things out' on your own.

Next month we will review the GETPUT subprogram, the utility that reads and writes the Data records created/retreived by the PRK command module.

PLATO Courseware Puts Twist into Traditional Studies.

PLATO educational courseware was developed with genius that would have made even the immortal Greek philosohper, Plato, smile.

This unique, computer-assisted learning system was developed at the University of Illinois the early 60's with funding from the State of Illinois and the National Science Foundation.

In 1962, William C. Norris, founder of Control Data Corporation (CDC) of Minneapolis, took an interest in the idea behind the PLATO system. And as a result, the company committed itself to the development of the system, and it pioneered the use of computer-assisted learning. The PLATO system has since been a proven learning tool in school systems, universities, businesses, industries, and governments worldwide for more that 20 years. Students of all ages have found the PLATO system to be an exceptional learning tool.

Through an exclusive agreement between Texas Instruments and Control Data, PLATO educational courseware in the areas of Basic and High School Skills will be available for use with the TI Home Computer.

The PLATO courseware designed for the TI Home Computer utilizes a particularly effective learning strategy which combines the use of tutorials and drills.

In a tutorial, the student is guided step-by-step through all the new information. The drills then allow students to evaluate the level of proficiency they gained from earlier tutorial work.

An entire integrated set of over 450 programs in 44 subjects for grades 3 through 12 will provide users with a comprehensive educational library.

PLATO Basic Skills courseware will be available for students in grades 3 to 8. The curriculum consists of mathematics, reading, and grammar. Users may choose from 64 courses.

Courseware also is available for students at the high school level. The PLATO High School Skills curriculum consists of math, reading, writing, science, and social studies. A total of 44 courses will be available in the High School Skills series.

Every PLATO program requires the use of the PLATO Interpreter cartridge. The Interpreter cartridge is packaged with survey diskettes designed to evaluate a student's basic skills, strengths and weaknesses in an objective way.

A Parent's Questionnaire is enclosed with every Interpreter cartridge. Parents with elementary age children can use the questionnaire to evaluate a child's skill level using a subjective means. The Interpreter cartridge has a suggested retail price of $49.95.

To operate PLATO courseware, users need a console and monitor (or television and adapter), the PLATO Interpreter cartridge, a Peripheral Expansion System, a Disk Memory System, and a Memory Expansion Card.

HEART AND SOUL OF PERSONAL RECORD KEEPING PART III

by Don Donlan

```
100 REM    INSERT THE P-R-K COMMAND MODULE AND USE THE SUBPROGRAMS THAT ARE
110 REM    RESIDENT THERE TO CREATE TWO SEPARATE DISK FILES: ONE IS THE
120 REM    P-R-K HEADER RECORD THAT DESCRIBES THE STRUCTURE OF THE DATA;
130 REM    THE OTHER IS A FILE OF THE DATA ITSELF.
140 REM BEFORE LOADING THIS BASIC PROGRAM.....................................
150 REM    EXECUTE THE FOLLOWING THREE BASIC COMMANDS:
160 REM             > CALL FILES(1)
170 REM             > CALL P(10000)  [To prepare a data area.]
180 REM             > NEW            [To clear out any old data.]
190 REM    Now load and run the following program.  Each line will be followed
200 REM    by a comment about the purpose of that statement.
210 REM ======================================================================
220 CALL L("DSK1.PRKFILE",C)  REM Load data file into reserved area.
230 IF C=0 THEN 650           REM Check error indicator: 0=failure, non-0=O.K.
240 OPEN #1:"DSK1.PRKHEADER",RELATIVE,INTERNAL,OUTPUT,FIXED
250 CALL H(1,1,0,F$)          REM Read the internal file name.
260 CALL H(1,5,0,F)           REM Read the number of fields per record.
270 CALL H(1,6,0,R)           REM Read the number of records in the file.
280 PRINT #1:F$,F,R           REM Write Header File record.
290 PRINT F$;F;R              REM Print this information on the screen.
300 FOR I=1 TO F              REM Begin a loop to write field definitions.
310 CALL H(1,9,I,F$)          REM Read the field name.
320 CALL H(1,10,I,T)          REM Read the field type.
330 CALL H(1,11,I,W)          REM Read the field width or size.
340 IF T<>1 THEN 370          REM Field is alpha (TYPE is not equal to 1)
350 S=S+W+1                   REM So add width + 1 to record size.
360 GOTO 380                  REM Then skip around next lines of code.
370 S=S+9                     REM With numeric field, add 9 to record size.
380 CALL H(1,12,I,D)          REM Read the number of decimal places.
390 PRINT #1,REC I:F$,T,W,D   REM Write this information to HEADER file.
400 PRINT F$;T;W;D            REM Print this information to the screen, too.
410 NEXT I                    REM Return to write remaining field definitions.
420 CLOSE #1                  REM Close the HEADER; open DATA file when done.
430 OPEN #1:"DSK1.PRKDATA",SEQUENTIAL,INTERNAL,OUTPUT,VARIABLE S+2
440 FOR I=1 TO R              REM Set up a loop to read records and print DATA.
450 PRINT I                   REM Print record number to the screen.
460 FOR J=1 TO F              REM Set up a loop to read fields in each record.
470 CALL H(1,10,J,T)          REM Read the field type.
480 IF T=1 THEN 540           REM Determine if field is alpha or numeric.
490 CALL G(1,I,J,C,D)         REM For numeric, retrieve data in a numeric var.
500 IF C=0 THEN 520           REM If 'missing data' code is not 0,
510 D=-9.999999999999E+127    REM enter a default value that can be tested for;
520 PRINT #1:D;               REM otherwise, print the numeric data to the file
525 PRINT D;                  REM and to the screen.
530 GOTO  590                 REM go around the lines that handle  alpha DATA.
540 CALL G(1,I,J,C,F$)        REM Retreive character data into a character var.
550 IF C=0 THEN 570           REM If 'missing data' code is not 0,
560 F$="?"                    REM enter a default value that can be tested for;
570 PRINT #1:F$;              REM otherwise, print the alpha data to the file
580 PRINT F$;" ";             REM and to the screen (with an additional blank).
590 NEXT J                    REM Go to the next field within the record.
600 PRINT #1:"@"             REM Finish by printing the entire record to file.
610 PRINT                     REM Finish pending print to the screen, too.
620 NEXT I                    REM Go on to process the next record in the file.
630 CLOSE #1                  REM Close the DATA file.
640 STOP                     REM Stop or End program execution.
650 PRINT "ERROR IN LOADING PRK FILE."     REM Error message if load fails.
660 STOP
```

HEART AND SOUL OF PERSONAL RECORD KEEPING, PART III, cont'd.

By Don Donlan

```
 10 REM   The following BASIC program takes the HEADER and DATA files
 12 REM   created in the previous program and converts them back into
 14 REM   PRK files which can be saved by the PRK save routine.
 16 REM
 18 REM   Before running the program, execute the following BASIC commands:
 20 REM
 22 REM            > CALL FILES(1)
 24 REM            > CALL P(10000)
 26 REM            > NEW
 28 REM
 34 REM
 36 REM   ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
100 OPEN #1:"DSK1.PRKHEADER",RELATIVE,INTERNAL,INPUT ,FIXED
110 INPUT #1:F$,F,R             REM Read file name, # fields, and # of records.
120 PRINT F$:F:R                REM Print this information on the screen.
130 CALL H(0,1,0,F$)            REM Write the file name to restored PRK header.
140 FOR I=1 TO F                REM Set up loop to create rest of PRK header.
150 INPUT #1,REC I:F$,T,W,D     REM Read field name, type, width, and dec. places.
160 PRINT F$:T;W;D              REM Print retrieved information to the screen.
170 CALL H(0,9,I,F$)            REM Write the field name to the PRK header.
180 CALL H(0,10,I,T)            REM Write the field type to the PRK header.
190 IF T=4 THEN 220             REM If scientific notation, (T=4) write no width.
200 CALL H(0,11,I,W)            REM Write the field width to PRK header.
210 IF T<3 THEN 230             REM For character and integer fields, do not
220 CALL H(0,12,I,D)            REM write the decimal places to PRK header.
230 NEXT I                      REM Go to next field in HEADER record.
240 CLOSE #1                    REM Close HEADER and open DATA file.
250 OPEN #1:"DSK1.PRKDATA",SEQUENTIAL,INTERNAL,INPUT ,VARIABLE
260 FOR I=1 TO R                REM Set up loop to read data and rebuild as PRK.
270 PRINT I                     REM Print the current record number to the screen.
280 FOR J=1 TO F                REM Set up loop to read the fields for DATA record
290 CALL H(1,10,J,T)            REM Recall what type of field you are about to get
300 IF T=1 THEN 380             REM If numeric (T<>1), then
310 INPUT #1:D;                 REM Read into numeric variable.
320 PRINT D;                    REM Print the retrieved data to the screen.
330 IF D=-9.9999999999999E+127 THEN 360   REM If default value, write null data
340 CALL G(0,I,J,D)             REM Normal data is written to PRK file.
350 GOTO 440                    REM Skip around alpha section and go to next field
360 CALL G(2,I,J,D)             REM This code indicates missing numveric data.
370 GOTO 440                    REM Skip around alpha section and go to next field
380 INPUT #1:F$;                REM Alpha data is read into character variable.
390 PRINT F$;" ";               REM Retrieved data is printed on the screen.
400 IF F$="?" THEN 430          REM Default value indicates missing data for field
410 CALL G(0,I,J,F$)            REM Normal data is written to PRK file.
420 GOTO 440                    REM Continue to loop for the next field in record.
430 CALL G(2,I,J,F$)            REM Indiate that character data is missing.
440 NEXT J                      REM End of field loop.
450 INPUT #1:F$                 REM Finish record by reading end of record "@".
460 PRINT F$                    REM Finish pending print to the screen.
470 NEXT I                      REM End of record loop.
480 CLOSE #1                    REM Close the DATA file.
490 CALL S("DSK1.PRKFILE",C)    REM Save the PRK file that has now been rebuilt.
500 IF C<>0 THEN 520            REM Check for error in trying to save PRK file.
510 PRINT "ERROR IN SAVING PRK FILE !!"   REM If error, flag message on screen.
520 STOP                        REM Stop or End the current program.
```

BEYOND PERSONAL RECORD KEEPING

by Don Donlan

Thanks to an inquiry on the bulletin board, I thought I would share with you the following program which sorts PRK records according to the first five fields in the order of determined by the 'key', which in this case is the first five character fields. I think the annotation should help explain what is going on in the program. I might caution you that since this is a BASIC program you won't see any increase in speed over the painfully slow sort that is already available in PRK. This simply lets you sort PRK records using any number of fields. It would be wise to have a printed or hand copied file structure available if you plan to modify this program to sort one of your files. In our example here, we use a file with 15 fields (5 character and 10 numeric), but you could modify that to suit your needs. Have fun!

```
1000 REM The following BASIC program BUILDS A PRK DATA FILE SORTS
1010 REM the file SORTWK. Then it restores the data to the PRK file.
1020 REM Requires use of Disk drive and diskette with enough room to
1030 REM build a relative record work file with which to do the sort.
1040 REM
1050 REM Before running the program, enter TI BASIC with the PRK module
1060 REM inserted in the console. Then execute these BASIC commands:
1070 REM > CALL FILES(1)
1080 REM > CALL P(10000)
1090 REM > NEW
1100 REM
1110 REM Load this program into memory (OLD DSK1.??????????) and RUN.
1120 REM ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1130 OPTION BASE 1 REM Set array index to begin at 1.
1140 DIM A$(15) REM Build and array to store default values.
1150 A$(1)="000000" REM Default values will be used to replace where
1160 A$(2)="00" REM no data has been entered into the PRK file.
1170 A$(3)="0000" REM There should be as many array/default values as
1180 A$(4)="0000" REM there are fields in the PRK record.
1190 A$(5)="0000"
1200 A$(6)="00000"
1210 A$(7)="0000.00"
1220 A$(8)="0000.00"
1230 A$(9)="00000"
1240 A$(10)="00000"
1250 A$(11)="00000"
1260 A$(12)="00000"
1270 A$(13)="000000.00"
1280 A$(14)="000000.00"
1290 A$(15)="0000"
1300 CALL CLEAR
1310 INPUT "ENTER NAME OF THE PRK FILE TO SORT >":FILE$
1320 CALL L(FILE$,C)
1330 IF C=0 THEN 1660
1340 CALL H(1,1,0,F$)
1350 CALL H(1,5,0,F)
1360 CALL H(1,6,0,R)
1370 S=0
1380 FOR I=1 TO F
1390 CALL H(1,9,I,F$)
1400 CALL H(1,10,I,T)
1410 CALL H(1,11,I,W)
1420 IF T<>1 THEN 1450
1430 S=S+W+1
1440 GOTO 1460
1450 S=S+9
1460 CALL H(1,12,I,D)
```

```
1470 NEXT I
1480 OPEN #1:"DSK1.SORTWK",RELATIVE R,INTERNAL,UPDATE,FIXED S+1
1490 FOR I=1 TO R
1500 RESTORE #1,REC I
1510 FOR J=1 TO F
1520 CALL H(1,10,J,T)
1530 IF T=1 THEN 1590
1540 CALL G(1,I,J,C,D)
1550 IF C=0 THEN 1570
1560 D=VAL(A$(J))
1570 PRINT #1:D;
1580 GOTO 1630
1590 CALL G(1,I,J,C,F$)
1600 IF C=0 THEN 1620
1610 F$=A$(J)
1620 PRINT #1:F$;
1630 NEXT J
1640 NEXT I
1650 GOTO 1680
1660 PRINT "ERROR IN LOADING PRK FILE."
1670 STOP
1680 GAP=2^INT(LOG(R)/LOG(2))-1
1690 PRINT "Beginning sort of "&STR$(R)&"records."
1700 FOR I=1 TO R-GAP
1710 FOR J=I TO 1 STEP -GAP
1720 REM According to PRK file structure, we INPUT 5 char.  10 numeric
1730 REM fields when we read the record.  Your situation may differ.
1740 INPUT #1,REC J:S1$,S2$,S3$,S4$,S5$,S6,S7,S8,S9,S0,SA,SB,SC,SD,SE
1750 INPUT #1,REC J+GAP:Z$,B$,C$,D$,E$,FX,GX,HX,IX,JX,KX,LX,MX,NX,OX
1760 REM For this file we build a 'key' made up of the 2nd-5th and 1st
1770 REM fields of the record.  You may build your sort 'key' differently.
1780 X$=S2$&S3$&S4$&S5$&S1$
1790 Y$=B$&C$&D$&E$&Z$
1800 IF X$<=Y$ THEN 1840
1810 PRINT #1,REC J+GAP:S1$,S2$,S3$,S4$,S5$,S6,S7,S8,S9,S0,SA,SB,SC,SD,SE
1820 PRINT #1,REC J:Z$,B$,C$,D$,E$,FX,GX,HX,IX,JX,KX,LX,MX,NX,OX
1830 NEXT J
1840 NEXT I
1850 PRINT GAP
1860 GAP=INT(GAP/2)
1870 IF GAP>0 THEN 1700
1880 PRINT "Restoring sorted data and saving as PRK file with name of ":&FILE$
1890 FOR I=1 TO R
1900 RESTORE #1,REC I
1910 FOR J=1 TO F
1920 CALL H(1,10,J,T)
1930 IF T=1 THEN 1970
1940 INPUT #1:D,
1950 CALL G(0,I,J,D)
1960 GOTO 1990
1970 INPUT #1:F$,
1980 CALL G(0,I,J,F$)
1990 NEXT J
2000 NEXT I
2010 CLOSE #1:DELETE
2020 CALL S(FILE$,C)
2030 IF C THEN 2050
2040 PRINT "Error in Saving PRK file."
2050 STOP
```

PERSONAL RECORD KEEPING & NAVARRONE'S DATA BASE MANAGEMENT

by Don Donlan

What I intend to accomplish in this article is a comparison that will give you some idea of the things to look for when deciding on a data base management tool for use in your record keeping. As most of you know I have been using the TI Personal Record Keeping cartridge for some time, but now Navarone Industries has come out with a new cartridge and diskette package called Data Base Management. Following are points of comparison to be considered if/when you want to invest in this kind of software for your needs.

PERSONAL RECORD KEEPING (PRK)	DATA BASE MANAGEMENT (DBM)
REQUIRED HARDWARE:	
PRK works solely within the console's memory, more precisely using about 10K of memory to store your data.	DBM uses both console and 32K memory so you need to have both memory expansion and a disk drive. Data uses 24K.
DATA RECORD:	
Maximum would be 15 fields of 15 characters each (165 bytes). Scientific notation may be used. Number of all records is determined by 10K divided by size of one record. Record definition is stored with data under one file name in PROGRAM format that can only be accessed after conversion. Records retrieved by number only.	Maximum of 255 characters; size/number of field(s) limited only by 255 record limit. Total number of records is determined by disk space available. Separate SETUP and DATA files are used used to define/store your records in a DISPLAY/FIXED format. Then other programs can easily read these records. Record key(s) must be chosen in SETUP.
SORT FEATURES:	
Sorts only one field at a time using BASIC routine that is very slow. It can perform mathematical transformations to calculate new data fields.	May read up to 6 fields in single sort using an ASSEMBLY routine that is very fast by comparison. No transformation of data, even on printout can be done.
PRINTOUT OF DATABASE:	
Report format or page(record) format are your only choices. No totals. Personal-Report Generator cartridge lets you define up to 127 characters per report line (compressed mode).	Records can be written with great deal flexibility; headings/totals possible. Maximum print width of 80 characters means that report lines are not as flexible to define.
EASE OF USE:	
PRK is fully menu driven, easy to use in a fixed screen display. The name of the field is used to prompt for the data. Field name is limited to 10 characters. No help prompts used. You can add new fields only with the use of Personal Report Generator. But only until 15 field maximum is reached.	DBM requires VERY CAREFUL reading of the user manual. There are multiple uses of FCTN keys which aren't obvious to a new user. Field prompts are not limited, except by screen size. Added help text available. New fields may be added without loss of data; new SETUP must be generated to access it.

I am very hard pressed to say which one is better because the plusses and minuses of both systems tend to balance one another out. The obvious choice is DBM for the large database user. For the occasional user, or one who keeps a lot of smaller lists, then PRK would be my recommendation; it doesn't requires as much of the user. And, for me, the math transformations make PRK unique. DBM, on the other hand, would be best suited for the advanced or full-time data base user who is looking for a tool to streamline their recordkeeping requirements and provide them with an easy means of building data files that can be used in more advanced programming applications. Neither system meets my 'ideal', but both have some very useful features worth your consideration.

PERSONAL REPORT GENERATOR

by Dennis Sherfy

Most people are familiar with Personal Record Keeping (PRK), especialy with the series of articles written by Don Donlan. Personal Report Generator, (PRG), a software cartridge from TI, is a valuable companion to PRK.

PRK is great for setting up files, adding and deleting records, and sorting files. However, it only allows you to print the data in three formats. Two of the formats are data tables. But what if you want to use the data for other purposes, such as printing address labels for your Christmas card list?

PRG is exactly what you need. PRG allows you to design reports in ANY format you wish. You can include text or spaces on any line, and print values from your data file within the text. Instead of numbers at the top of data columns, as produced by PRK, you can print titles above your columns. PRG does not print the data structure at the beginning of each printout, as does PRK. PRG prints field number 0, (the record or page number), only if you want it to do so.

In addition, PRG lets you add new fields to your file, lets you delete large numbers of records at one time, (such as delete pages 10-56), and combines two or more compatible files into one large file.

I have used PRG to print mailing labels. The "page" is defined as 6 lines long. This prints an address on each label, then skips to the proper starting position on the next label.

PRK allowed about 200 records per file, but my file was about 400 records in size. I entered the records in alphabetical order, creating two, 200-record files. With PRG, I was able to divide the files into four sets and merge the two low Zip-code sets and the two high Zip-code sets to form two new files which would print 400 labels in Zip-code sequence.

PRG is also used with the Statistics Module and has been available for about $12.

TI LOGO II LEARNING IS REAL FUN

by Walt Maes

The following article was copied from "Suncoast Beeper", Newsletter of the Suncoast 99'ers, 945 Montocello Blvd., North, St., St. Petersburg, Florida 33703.

Don was over to my house last night and I was having a problem with page 36 of the Logo II book. It shows you the commands to make a perfect triangle. Then it shows you six triangles with a common point and then six more triangles with the same point, but starting 30 degrees from the other. We started out just to make the six triangles from one point. We tried all kinds of angles and line lengths, and tested each one. You never seen triangles put into so many ways as we did. We built bridges, windmills, and many other things before we finally got the answer. After we did get the six triangles from one point we sat back to look at our work and found we had drawn a perfect 3 D cube.

The great thing about learning Logo is nothing is ever wrong, you have just found out another way to draw a picture. It maybe in modern art or classic art. That is for you to decid but it's fun to reach your goal. One of error messages is "Tell Me More", now how nice can a machine be?

If any of you have wondered how to stop at a given number of times like building stairs, that will go on endless if you use their commands and then try this:

```
TO STAIRS :SIZE :COUNT
REPEAT :COUNT [ RSQUARE :SIZE
MOVE :SIZE ]
END
```

As you know, size is the length of a line in the square and count is the number of stairs you want. Now you must have RSQUARE and move in your dictionary already to make this work.

To make the 3 D cube, enter this:

```
TO CUBE
REPEAT 3 [ TRI RT 120 ]
RT 60
REPEAT 3 [ TRI RT 120 ]
END
```

To make this work, you must have put TRI in your dictionary.

That's as far as I have gotten this month. Lets see what I learn by next month.

TI LOGO VARIABLES.

by George Paschetto

The following article comes to the HUGger Newsletter from the "Computer Bridge," Newsletter of the St. Louis Users Group via the LA 99'ers Topics, Gardenia, California.

To make your BOX procedure run a different size box each time, you can use a variable for the side. Type TO BOX and press enter ONCE. In the EDIT mode, the cursor appears after the procedure name at the screen's top for adding variables, so type the variable's name (S for SIDE) here. Now to use the variable, type :S (the colon is required when the variable is used instead of a number) where the 50 was. It should look like this:

```
TO BOX S
REPEAT 4 [FD :S RT 90]    FD :S will be the box's side.
END                       fctn 9 (back)
```

Now type BOX # (some number) and you'll get a box the size specified. If you type BOX and no number, or try to use the STAR procedure, the computer will come back with TELL ME MORE. BOX needs a number now in order to run, and any procedure that uses BOX now has to provide one. To use STAR now, you'll have to add.....

```
.....a variable              .....or a number.
TO STAR S                    TO STAR
REPEAT 6[BOX :S RT 60)       REPEAT 6[BOX 50 RT 60]
END                          END
```

fctn 9 (back) gets you back to the turtle.

Add this to change the number of boxes in the star:

```
TO STAR S TURNS
REPEAT :TURNS[BOX :S RT 360/:TURNS]
END
```

In turtle mode, type STAR # # 'enter'; the first number will be the box size, the second number will be how many boxes there are in the star. To see your procedures type "NOTURTLE" enter, then PA (print all). when done type "TELL TURTLE" again to return to the turtle.

If you have a particular interest or question concerning LOGO please write to George Paschetto, Apt. #5 255 University, Radcliff, KY 40160.

DID YOU KNOW?

Star Micronics, (producers of the Gemini 10 and 10X printers) has published an addendum to their Users Manual for the TI 99/4A users?

This free addendum contains a 12 page list of special instructions and programs for the TI 99/4A, along with DIP switch settings and tips for using italic style type plus info on graphis and other features.

A copy of this addendum may be obtained by writing to:

Cherie Maddocks, Technical Support
Star Micronics, Inc.
3 Oldfield
Irvine, CA 92714

This article was taken from Lincolnland 99 Lookout, Newsletter of the Lincolnland 99 Computer Group, Springfield, Illinois and from Topics, Newsletter of LA99ers Computer Group, Los Angeles/San Fernando Valley, California.

P-CODE CARD: POP GOES THE DIODE!

This article was taken, in full, from the 99/4 Users Of America, May, 1984 Newsletter.

We have found a 100% failure rate on the P-Code cards manufactured the fifty-second week of 1982. The card self-destructs whether you use it or not! As long as it is plugged into the peripheral cabinent it draws power, even though it may be turned off. This is due to a factory defective diode that does not allow current to drain off properly. If you have a P-Code card that has popped its cork, don't take it to your friendly exchange center and shell out $42 for a new one! Call TI CARES and request a new card! You don't have to pay for factory blunders, oversights, or what ever. The new cards have a germanium diode in them that corrects the problem.

USING LOGO II

Richard J. Bailey
68A Church Street
Gonic,N. H. 03867

If you want to get a new language for your computer that is easy to learn and fun to use no matter what your age-then LOGO is for you. TI LOGO has some limitations but as long as you realize what they are and work within them, you'll have a lot of fun.

LOGO is a language with built-in routines called 'primitives'. These primitives can be combined into larger routines called 'proceedures'. Once a proceedure is defined; LOGO allows you to use the proceedure as if it were a primitive. You can continue to combine primitives and proceedures until you have one proceedure that does exactly what you want it to do.

First some background. Long, long ago, in the age before video terminals and dot matrix printers (I heard my parents talk about that time!) the only way to draw complex figures was with expensive x-y plotters. The creaters of LOGO at M.I.T. designed a simple robot with wheels and an umbilical cord connecting it to the computer. The computer could direct the robot to 'walk' around on a piece of paper. By adding a ballpoint pen which could be raised and lowered, they could create the complex pictures and graphs they wanted. To protect the electronics of the robot they covered it with a semispherical 'shell'. Because of the shape of the cover and the speed with which it drew, they called it a 'turtle'. Even though this robot has been replaced by its video equivilence, the original names are still used.

One of the easiest ways to demonstrate LOGO is with turtle graphics. This is the facet of LOGO that most people are familiar with. To draw with the 'turtle' is quite simple once you get the hang of it. Commands can be entered in either the immediate mode or stored by creating a proceedure. To draw a box you could type:

FD 50 RT 90 FD 50 RT 90 FD 50 RT 90 FD 50

in the immediate mode. This will draw a line up (forward) 50 units, right 90 degrees, to the right (forward) 50 units, etc... until the box or square is drawn. To create a proceedure that does the same thing type:TO BOX <ENTER> then

REPEAT 4 [FD 50 RT 90] <ENTER>

Then FCTN 7 to exit the proceedure. Now any time you enter BOX, a square will be drawn.

More complex shapes can be created by the same method. As an example, the proceedure called 'SNOOPY' will draw Snoopy with Woodstock in his dog dish having a bubble bath complete with bubbles. This proceedure demonstrates the use of TILES and SPRITES with turtle graphics. The sprites were created with LOGO's built-in sprite editor and the tiles with the built-in character editor. I have not included the tile definitions so you will see the undefined characters if you enter this proceedure. You can redefine these characters yourself, or if you don't want to enter SNOOPY yourself, send me $5.00, your name and address, and I'll send you a disk with SNOOPY and a couple of other eye openers.

Now for some comments. The turtle graphics are but a small part of LOGO. Its sound, music, and list manipulative abilities are intriguing and are well covered in the manual that comes with LOGO. The TI implementation of LOGO needs more memory because you get an 'OUT OF INK' message when you draw to more than 1/3 of the screen area. You can get around this limitation by combining tiles for the background, adding sprites for movement, and drawing with the turtle on the same screen. You may also find that once a character is redefined you can't reset it to its original shape, you have to redefine it with the editor. One other annoying feature is that there is no way to clear memory. If you 'RECALL' more than one proceedure from disk, both will reside in memory. When you go to save, you save both proceedures under the name you think you're saving one under. If you work within these limitations, you will find you'll really enjoy LOGO.

PROCEDURES

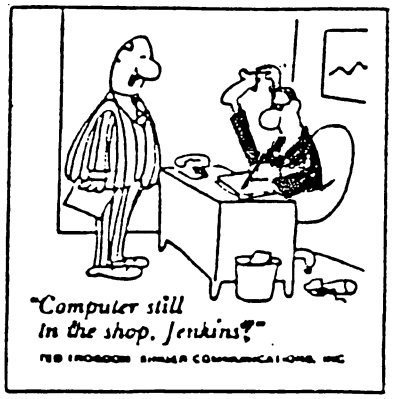
```
TO SNOOPY
; SNOOPY + WOODSTOCK
; * * * * BY * * * *
; RICHARD J. BAILEY
; 68A CHURCH STREET
; GONIC, N.H. 03867
TELL TURTLE HT ; SETUP
CS CB 15
SXY 71 34
LT 115
REPEAT 10 [FD 2 RT 1 ] ; SNOUT
REPEAT 6 [FD 4 RT 8 ]
SH 180
REPEAT 9 [FD 1 RT 20 FD 2 RT 20 ] ; NOSE
SH 330
REPEAT 10 [FD 4 RT 10 ]
REPEAT 3 [FD 3 RT 5 ]
REPEAT 4 [FD 2 LT 15 ] ; HEAD
REPEAT 7 [FD 2 RT 9 ]
REPEAT 5 [FD 3 RT 3 ]
REPEAT 6 [FD 2 RT 10 ]
REPEAT 3 [FD 2 LT 11 ]
REPEAT 15 [FD 3 RT 4 ] ; EAR
REPEAT 16 [FD 2 RT 10 ]
REPEAT 9 [FD 2 RT 2 ]
SXY 95 38 SH 0
REPEAT 2 [REPEAT 9 [FD 2 RT 20 ] FD 10 ] ; INNER EAR
SXY 56 60 SH 10
FD 1 RT 100 FD 5 RT 100 ; EYE
FD 1 RT 90 FD 5
SXY 88 26 SH 231
REPEAT 11 [FD 2 LT 7 ] ; NECK
RT 20
REPEAT 6 [FD 5 LT 1 ] ; BACK
REPEAT 13 [FD 1 RT 8 ] ; BOTTOM
FD 30 ; FOOT
REPEAT 10 [FD 1 RT 10 ]
FD 19
RT 90 FD 5 BK 5 LT 90
FD 6 RT 18
RT 90 FD 6 BK 6 LT 90
REPEAT 8 [FD 2 RT 18 ] ; TOE
REPEAT 5 [FD 4 RT 7 ]
SH 70
REPEAT 4 [FD 4 RT 8 ] ; LEG
SXY 50 -8
SH 10
REPEAT 16 [FD 2 RT 1 ] ; CHEST
REPEAT 6 [LT 5 FD 2 ] ; NECK
SXY 78 24 SH 110
FD 13 RT 90 FD 1 ; COLLAR
RT 90 FD 13
SXY 73 7 SH 191
REPEAT 8 [FD 4 LT 3 ] ; ARM
LT 35
REPEAT 3 [REPEAT 4 [FD 2 LT 45 ] RT 130 ] ; HAND
SXY 37 -41 SH 45
REPEAT 2 [REPEAT 18 [FD 1 RT 5 ] RT 130 ] ; TAIL
DISH
BUBBLE 1
END
```

```
TO DISH
SXY -79 -43 SH 15 ; DISH
FD 15 RT 75 FD 52
RT 75 FD 15 RT 105 FD 59
SXY -87 -44 SH 90
FD 250 ; BOTTOMLINE
PT 33 7 16 ; SNOOPY
PT 3 3 16
PT 79 9 16
PT 79 10 16
PT 80 11 16
PT 89 12 16
END

TO BUBBLE :ZZ
PT 33 15 16
PT 35 16 16
PT 36 17 16
PT 37 15 17
PT 38 16 17
PT 64 17 17
TELL 17
CARRY 5
SXY -55 -16 ; WOODSTOCK
SC 4 SS 8
TELL [1 2 3 4 ] ; HIDE BUBBLES
SXY -60 -28
SC 3 SS 8
TELL [6 7 8 9 10 11 12 13 14 15 16 ]
SS 8
SXY -55 -28
CARRY :SALL
EACH [SC RN SH -30 + ( RANDOM * YN / 3 )
                    SS ( 3 * YN / 2 ) ].

POP
MAKE "ZZ :ZZ + 1
IF :ZZ < 6 THEN BUBBLE :ZZ
SS 8 SC 16 CS
17 CARRY
END

TO POP
WAIT 55
MAKE "NUMB 6
LABEL: TELL :NUMB
SC 2
MAKE "NUMB :NUMB + 1
IF :NUMB > 16 THEN STOP
GO "LABEL
END
** DONE **
```



TI LOGO INTRODUCTION

by George Paschetto

Editor's Note: The following article comes from the "Computer Bridge," Newsletter of the St. Louis Users Group, via the August, 1985 issue of "TopIcs," Newsletter of the LA 99ers Computer Group.

LOGO is a programming language that does not use line numbers. It is organized into procedures that are named by the programmer. Typing in the procedure will cause it to run.

When you start up, a ? will appear on the left of the screen. Type TELL TURTLE 'enter". Now type REPEAT 4[FD 50 RT 90] 'enter' and the turtle will draw a box. (To erase a mistake before you press enter, backspace by holding the fctn key and pressing the 3 key.)

REPEAT 4[ ] will repeat whatever is in the [ ] 4 times.

FD n causes the turtle to go forward the given number.

RT n makes it turn right n number of degrees (90 is a right angle.)

The computer has been commanded but not yet programmed because what you typed is not stored in memory. If you want that command to run over again, you must make it part of a 'procedure.' I'll name it BOX. Type TO BOX 'enter' and your screen will change. This is the EDIT mode. TO BOX is at the top and the cursor is still right after BOX. Press 'enter' again and type REPEAT 4[FD 50 RT 90] (the end will move down.) Now to leave the EDIT mode, hold fctn and pres 9 (back.)

Want a box? Type BOX and press enter. The BOX procedure will run. Use the FD n or RT n commands to move the turtle around and the BOX again. The box is now part of the turtle's vocabulary. You can actually add to the language throuh your own procedures.

Another way to write a procedure:

DEFINE "BOX [[] [REPEAT 4[FD 50 RT 90]]]

The empty brackets in the beginning are necessary.

The procedure can also be called from another command or procedure. Type REPEAT 6[BOX RT 60] 'enter.' You'll get 6 boxes in a star pattern, (to run faster hide turtle with HT, ST after to show it.) It's the same thing as REPEAT 6[REPEAT 4 [FD 50 RT 90] RT 60.]

HANDY TIPS FOR THE TI99/4A COMPUTER

Here are a few tips for beginners (good for experienced programmers, too)

01) If you have the speech synthesizer and the TEII cartridge here is a trick for debugging programs. All you have to do is enter your program, type LIST "SPEECH" and hit enter. The computer will read your listing back to you!

02) If you want to disable the quit key (Fctn +/=) type in CALL INIT :: CALL LOAD(-31806,16) and then enter. You must have Extended Basic.

03) If you are going to save a program to tape and type OLD CS1 instead of SAVE CS1 don't panic. Press FCTN and E together then press <ENTER>. This will take you out of the tape loop.

04) You don't have to enter line numbers in TI Basic or Extended Basic. Before you start entering a program type in NUM n(1),n(2); where n(1) is the starting line number and n(2) is the desired increment.

05) In TI Basic you can edit a line with with the Edit command or with the FCTN key and either the E or X keys. To use edit, type EDIT n (n = line number). The other way is to enter the line number and the press FCTN X or FCTN E. This is the only edit method recognized by Extended Basic.

06) You can list programs to the screen in several ways. Try these; LIST, LIST n, LIST n-, LIST -n, LIST n-n.

07) If you want or need to renumber the lines in a program either to make it neater or make room for new lines you don't have to renumber them individually. Just enter the command RES n,n for Resequence (starting number, interval between lines).

08) When entering a listing in Extended Basic and several lines are very similar, you can save time by typing in the first line and hitting <enter>. Then press FCTN 8 (REDO). Change the line number and make the changes to the line as needed and hit <enter>.

09) Have you ever pressed ERASE by mistake and lost the whole line? Don't panic and don't hit <ENTER>. Instead press FCTN ? and <ENTER>. Your line will still be intact.

10) In Extended Basic type in RUN CS1. Follow the instructions on the screen. It will load the program and then run it automatically.

11) In Extended Basic you can use REM or ! to put documentation in a program that the program will ignore.

12) When you want to stop a listing on the screen in Extended Basic, just hit any key. To start the list again, strike any key.

13) You can add comments after a GOSUB or GOTO. They won't interfere with the program and you don't need rem or !.

14) With Extended Basic and a disk system, save a program under the name LOAD. When you start with this disk in the drive, it will load and run that program.

15) If you have the TEII cartridge and the Speech Synthesizer type in the program on page 37 of the TEII manual. Try entering strings of K's, Q's, U's, W's, J's, or X's for different sound effects. Try mixing them for interesting sounds.

THINGS TI NEVER TOLD US

SUBROUTINE SECRETS

Good programmers write their programs in small function blocks known as subroutines. By breaking their programs into small parts, it is easier to understand and fix them. But a program full of GOSUBs and GOTOs can be hard to follow on a TV screen. It would be handy to say where the GOSUBs and GOTOs are going so you wouldn't have to jump around in the program just to understand what its doing.

TI doesn't tell you, but you can do just that in console BASIC. After a GOSUB or GOTO you may type a REM statement followed by the name of that routine. The name must be no longer than the rest of the screen line, and the line MUST! end with a blank. (The REM statement will automatically add one more space after REM and will fill the line when you list it.) The line might look like: 100 GOSUB 200 REM DRAW LINE.

Editor's Note: The following T.I.N.T. comes courtesy of the Cin-Day Users Group, of Cincinnati, Ohio. I have tried this routine while keying in this page of the Newsletter, and it does work!

TI WRITER TRICK! by Ed Kennedy; Cin-Day Users Group, Cincinnati, Ohio

Here is a program suggestion for routine users of TI-Writer. The next time you want to include a program listing in a letter or document perform the following operation.

1. Load the program into memory using TI-Writer. Place a carriage return character at the end of each program line.
2. Save the program onto your diskette using the following:
   LIST "DSKn.(filename)"
   where n=1,2, or 3 for the disk drive to be activated.
3. Include in your letter or document's main file the following:
   .IF DSKn.(filename) (.IF - (Include File) is a Format Command which is used in the Text Formatter of TI-Writer.)

The key to this procedure is the second step. By LISTing to the disk you have saved the program in DIS/80 format which can be utilized by TI-Writer. Using this process you avoid needless typing and errors for inclusion of program listings in a letter or document.

```
100 CALL CLEAR
110 !FOR B=1 TO 24::PRINT::NEXT B
120 ACCEPT AT(5,1)SIZE(1)BEEP VALIDATE(UALPHA):A$(0+A)
122 !ACCEPT AT(5,1)BEEP:A$(0+A)
130 PRINT A$(A)
140 GOTO 120
```

The above program was inserted in this article using this procedure. For those fellow 99'ers with Extended BASIC try RUNning this program in order to get an interesting 32-column effect while using X-BASIC (28-column is normal).

A QUIRK

The program shown in the above article was submitted to us by Eric Costello. It appears to be a quirk in the computer. If you make a calculation in the array variable [A$(0+A)] the computer does not appear to be able to determine the end of the input 28 characters max or in this case it uses SIZE (1)). It lets you type 32 characters wide but is not stable enough to let you use that conveniently. Try it. Then take the exclamation marks out of 110 and 122 an put one in front of 120. This will let you do the same thing only 28 characters wide. See if you can figure out why it only goes 28 wide. We know.

16) If you have Extended Basic and 32K type this in as the last line of your program:
CALL INIT :: CALL PEEK(2,A,B):: CALL LOAD(-31804,A,B)
This will return you to the title screen when the program is ended.

17) When hooked up to a black and white tv use CALL SCREEN(15). This will disable the color generator and remove the vertical lines you may have seen

18) To speed up loading Infocom games, don't use Extended Basic. Use Mini- memory or E/A instead. To use these, select the load and run option and type DSK1.BOOT. When this is finished loading, press ENTER until you get
the program name, then type START. On the Mini-Memory, you'll get an error after BOOT loads, but keep pressing ENTER and proceed as above.

19) If you want to disable the keyboard for any reason type in CALL LOAD(-3257 2,128). You will have to turn off the console to regain control

BYTE-LINE

THE FOLLOWING PROGRAM WAS SUBMITTED BY STEVE WEINKHAME R OF THE SOLON GROUP. THIS LOAD PROGRAM WILL LET YOU RUN YOUR PROGRAM, CATALOG IT TO PRINTER, OR EVEN DELETE AN UNWANTED FILE...AND IT ONLY TAKES UP 10 SECTORS!

```
10 CALL CLEAR :: DISPLAY AT(
3,6):"RUN AUTOBOOT? Y" :: AC
CEPT AT(3,20)BEEP SIZE(-1)VA
LIDATE("YN"):XX$ :: IF XX$<>
"Y" THEN END
100 X$="1"
110 DIM A$(97),H(98),J(97),K
(97)
120 CALL INIT
130 FOR I=1 TO 5 :: READ TYP
E$(I):: NEXT I
140 DATA "DIS/FIX","DIS/VAR"
,"INT/FIX","INT/VAR","PROGRA
M"
150 CALL LOAD(-31806,16)
160 OPEN #1:"DSK1.",INPUT ,R
ELATIVE,INTERNAL
170 INPUT #1:A1$,U,U,V
180 Q,R,S=0
190 GOSUB 440
200 Q=Q+1
210 INPUT #1:A$(Q),H(Q),J(Q)
,K(Q)
220 IF LEN(A$(Q))=0 OR Q=97
THEN 230 ELSE 200
230 S=S+1 :: R=R+1
240 IF ABS(H(S))=5 THEN B$="
   " ELSE B$=" "&STR$(K(S))
250 T$=TYPE$(ABS(H(S)))&SEG$
(B$,LEN(B$)-2,3)
260 GOSUB 470
270 IF R=15 OR S=Q OR H(S+1)
=0 THEN 280 ELSE 340
280 R=0 :: GOTO 490
290 ACCEPT AT(24,22)BEEP VAL
IDATE(DIGIT," ")SIZE(-3):PRG
300 IF PRG=00 THEN 620
310 IF PRG=97 THEN 330 ELSE
IF PRG=98 THEN 340 ELSE IF P
RG=99 THEN 370 ELSE IF PRG=9
99 THEN 510 ELSE PROG$=A$(PR
G)
320 GOTO 340
330 GOSUB 680 :: DISPLAY AT(
23,1):"INSERT NEW DISK":"PRE
SS ENTER WHEN READY" :: ACCE
PT AT(24,23):AA$ :: CLOSE #1
 :: GOTO 160
340 IF PROG$<>"" THEN 360 EL
SE IF S<Q AND H(S+1)<>0 THEN
 230 ELSE 290
350 CLOSE #1
360 CALL PEEK(-31954,A,B)::
GOSUB 380 :: RUN "DSK1.12345
67890"
370 END
380 I=A*256+B-65536 :: CALL
PEEK(I,A,B):: I=A*256+B-6553
6
390 CALL LOAD(I+29,LEN(PROG$
)+5):: CALL LOAD(I+33,ASC(X$
))
400 FOR I=1 TO LEN(PROG$)::
P=ASC(SEG$(PROG$,I,1)):: CAL
L LOAD(I+34+I,P):: NEXT I
410 IF LEN(PROG$)<10 THEN 42
0 ELSE 430
420 FOR I=LEN(PROG$)+1 TO 10
 :: CALL LOAD(I+34+I,130):: 
NEXT I
430 RETURN
440 DISPLAY AT(1,1)ERASE ALL
:"DSK";X$;" -DISKNAME=";A1$:
"AVAILABLE=";V;" USED=";U-V
450 DISPLAY AT(3,4):"FILENAM
E  SIZE   TYPE":"
   ----  ----
460 RETURN
470 DISPLAY AT(R+4,1):USING
"##":S :: DISPLAY AT(R+4,4):
A$(S):: DISPLAY AT(R+4,14):J
(S):: DISPLAY AT(R+4,19):T$
480 RETURN
490 DISPLAY AT(21,1):"00 DEL
ETE A FILE":"97 READ NEW CAT
ALOG":"98 NEXT SCREEN 999 PR
INT CAT":"99 END      SELECTI
ON:"
500 GOTO 290
510 GOSUB 680 :: DISPLAY AT(
21,1):"ENTER DATE" :: ACCEPT
 AT(21,12):D$
520 OPEN #2:"PIO" :: PRINT #
2:CHR$(27);"B";CHR$(3);CHR$(
27);"A";CHR$(8);
530 PRINT #2:D$
540 PRINT #2:"DSK";X$;".";"-
DISKNAME=";A1$;"AVAILABLE=";
V;" USED=";U-V
550 PRINT #2:"    FILENAME
SIZE   TYPE":"    --------
---- --------"
560 COUNT=S :: S=1
565 FOR I=1 TO Q-1
570 IF ABS(H(S))=5 THEN B$="
   " ELSE B$=" "&STR$(K(S))
580 T$=TYPE$(ABS(H(S)))&SEG$
(B$,LEN(B$)-2,3)
590 PRINT #2:I;TAB(5);A$(I);
TAB(15);J(I);TAB(19);T$ :: S
=S+1
595 NEXT I
610 CLOSE #2 :: S=COUNT :: G
OTO 490
620 GOSUB 680 :: DISPLAY AT(
23,1):"DELETE WHICH FILE NAM
E?"
630 ON ERROR 650
640 ACCEPT AT(24,1)BEEP:DEL$
 :: GOTO 660
650 DISPLAY AT(24,1):"FILE I
S PROTECTED" :: FOR I=1 TO 5
00 :: NEXT I :: GOTO 490
660 DELETE "DSK1."&DEL$
670 CLOSE #1 :: GOTO 160
680 CALL HCHAR(21,1,32,125):
: RETURN
```

PRINTING FROM EARLY MODULES

Editor's Note: The following article was reprinted from the May, 1985 issue of The ROM Newsletter, Newsletter of the Users Group of Orange County, 17301 Santa Isabel Street, Fountain Valley, CA 92708 via the April, 1985 issue of "MINI-MAG 99".

"Kent Maxwell found a way to be able to print the Weight Control and Nutrition Module to a parallel printer. (Editor's note: TI produced these modules prior to the availability of an RS232 card with a PIO output, so their menus did not provide for same.)

The procedure is as follows:

1. When setting up your files, tell the computer that you will not be using a printer, then create your data files accordingly.

2. When reviewing the files, put in any fictitious printer device name (i.e., RS232/8) and the computer will indicate DEVICE NOT FOUND. At this time, enter PIO, and the computer will allow access to a parallel printer.

The procedure may vary slightly between the various early modules, but the key is to avoid a printer identification in the data unit process. Enter the device name when recalling and reviewing the previously entered data.

Kent Maxwell is an avid TI enthusiast who is employed with the VA Hospital in Sepulveda, California on the security force; and is also a member of Tex-Comp's Technical Consulting group."

SLOWING FAST DM-1000 V3.1 by Louis Guion (Courtesy of John Creviston)

Condensed from: NET 99er HCUG, newsletter.

The newest version of DISK MANAGER 1000 (DM1000, Version 3.1), suffers from the previously known defect of having any key which is depressed and held for only a moment, running away at breakneck speed and repeating several times before you can get your finger lifted from the key. Fast? Yes!, but a pain in the uh-huh! Well, you can fix it anytime you like, and set the repeat action just as fast, or as slow as you want to make it. You must have a single-sectoring program like DISKO or DISK+AID. Copy the file "MGR1" of DM1000 Version 3.1 to * NEWLY INTIALIZED*, *BLANK* diskette. You'll have only the one file. "MGR1" on the new diskette.

Now, load your DISK+AID and either go to sector >36, or have the program search for the HEX string of..... 06 03 16 F9 03 80 00 A0 FF 00 C0 1D. The important bytes are #s >42 and >43 which are the ... 00 A0 ... in the string above. >00 A0 is 160d, (decimal). You want to change this 160d, or 00 A0 to another value which will slow down the repeat key. The range of acceptable values will fall between 160d to 2000d or hex values >00A0 to >07D0.

TI TIP

by Dan Eicher

Here is a TIp for anybody that plays Infocom games.

If you don't like the time it takes for a game to load from disk, **don't** use Extended BASIC! Use your Mini-Memory or Editor/Assembler instead. To use these, select the load and run option and type DSK1.BOOT. When this is finished loading, press enter till you get program name, then type START.

On the Mini-Memory, you will get an error after BOOT loads but keep pushing enter and proceed as above. Here are some comparisons of the load time for the different modules:

Extended BASIC	3 minutes
Editor/Assembler	1.25 minutes
Mini-Memory	1.18 minutes

TAMING YOUR GORILLA
by Jim Ellis

Do you have one of those? It just won't do what you want it to! Well, perhaps I can help. After some time spent looking through the manual, (if you don't care what you call it), I found that I didn't know anymore about my son's gorilla than he did. I called Leading Edge some time back, and they sent me a program that supposedly would help, HELP! Recently, I purchased a book from Radio Shack that had been marked down, (the only time to buy), that contained information on their many various assorted printers. Oh, hadn't I mentioned we're talking about the Gorilla Banana printer? Well, it seems that many of the printers may have several things in common even though made entirely for someone else. To get on with why I'm writing this article, I have made some discoverys that may help some of you in your effort to use the Gorilla for something besides a paper weight. First off, I don't know if the serial and the parallel versions are the same or not. The one we have is the serial version. You can get what looks like a European/special characters set by using PRINT CHR$(n), where n has a value from 160 to 191. This I found by trial and error, but was pointed out in the book on R/S printers. If you are interested in all points addressable graphics, (why else did you buy it?), I discovered by studying the listing that Leading Edge sent me and the R/S book some interesting commands as follows:

CHR$(8) - Enter graphics mode
CHR$(15) - Reset to normal (10 cpi)
CHR$(28),CHR$(n1),CHR$(n2) - Repeat print data, where n1=number of desired character to be printed, and n2=ASCII value of the character desired printed.

For instance, PRINT CHR$(28);PRINT CHR$(200);PRINT CHR$(193), will print a double line useing the 1 and 64 dots, the 128 is added to tell the printer of the correct graphics code. It should not do anything if you gave it 65 only. Characters are 7 dots high by 6 dots wide giving 480 dot columns per 8.5 inch page. Now to determine the column number, which ranges from 128 to 255, we use the following format.

```
     1       X
     2      X X
     4     X   X
     8    X     X
    16   X       X
    32  X         X
    64 XXXXXXXXXXXXX
          /|||||||
         / |||||||
        /  |||||||
   same as   ||||||--64+128=192
right side   |||||---32+64+128=224
             ||||----16+64+128=208
             |||-----8+64+128=200
             ||------4+64+128=196
             |-------2+64+128=194
             --------1+64+128=193
```

Upon entering graphics mode the range of values needed to tell the printer what to print is from 128 (blank) to 255 (a column of 7 dots). If the printer has incremental line feeds, I have not found out that information as yet. To get you started here is a little program that will print out all of the possible dot column combinations.

```
10 OPEN #1:"RS232.BA=1200"
20 PRINT #1:CHR$(8);
30 FOR X=128 TO 255
40 PRINT #1:CHR$(X);
50 NEXT X
60 PRINT #1:CHR$(15);
70 PRINT #1:"NOW LOOK AT THE COLUMNS
CLOSELY TO SEE HOW THEY CHANGE."
80 CLOSE #1
```

You must use the proper statement when talking to your printer if you are using another basic, just make the proper changes to converse with your printer port and baud rate. I hope that this will help someone who has not been able to use these features before. I know how trying it has been for my son. If you have any questions just let me know on the HUGbbs or at the meeting.

DISK DRIVE SPECIFICATIONS
VERSION 1.1, SEPTEMBER 12, 1985
by Louis Guion, Startext 77536

MANUFACTURER	MODEL NUMBER	HIGH	SIDE DENS	TPI	BYTES	5 V PWR	12V PWR	ACCES TIME	MOTOR DRIVE	COMMENT
AlpsElectric	FDD2624BGI	1/2	DSDD	48	360K	.6A	.9A		Belt	Hi Pwr Reqmt
Canon	MDD211	1/2	DSDD	48	360K					
C.D.C.	9409	Full	DSDD	48	360K					
C.D.C.	9428	1/2	DSDD	48	360K					
Epson	SD521	1/2	DSDD	48	360K	.4A	.4A	6MSEC	Drect	O.K. in PBox
Hitachi	HFD505B	1/2								
Matsushita	JA551	1/2								
Matsushita	JA5551-2	1/2								
Micropolis	1115V	Full	DSQD							
Mitsubishi	M4851	1/2	DSDD	48	360K					
Mitsubishi	M4853	1/2	DSQD	96	720K	.5A	.7A			
MPI	B51	Full	SSSD	48	90K				Belt	Sold in PBox
MPI	B52	Full	DSDD							
MPI	501C-200	1/2								
MPI	502B-100	1/2	DSDD							
National	JA551-2	1/2	DSDD	48	360K					
Panasonic	JA551-2	1/2	DSDD	48	360K			6MSEC		
Qumetrack	142	1/2	DSDD	48	360K	.6A	.9A		Belt	Hi Pwr Reqmt
Qumetrack	142LX	1/2	DSDD	48	360K					
Qumetrack	542	Full		48						
Remex	RFD480	2/3	DSDD	48	360K				Drect	
Sanyo	FDA5200B/PC	1/2	DSDD	48	360K					
Sanyo	SM548D	1/2	DSDD	48	360K			6MSEC	Drect	
Shugart	400L	Full	SSSD	48	90K				Belt	Sold in PBox
Shugart	SA455	1/2	DSDD	48	360K	.6A	.6A	6MSEC		O.K. in PBox
Shugart	SA465	1/2	DSQD	96	720K					
Shugart	SA475	1/2		96	1.6M					For the "AT"
Siemens	FDD100-5	Full	SSSD	48	90K				Belt	Sold in PBox
Tandon	TM50-1	1/2	SSSD							
Tandon	TM55-2	1/2	DSDD	48	360K			6MSEC		
Tandon	TM55-4	1/2	DSQD	96	720K					
Tandon	TM65-2L	1/2								
Tandon	TM100-1	Full	SSSD	48	180K				Belt	
Tandon	TM100-2	Full	DSDD	48	360K				Belt	
Tandon	TM101-4		DSQD							
TEAC	FD55A	1/2	SSSD		180K			6MSEC		
TEAC	FD55B	1/2	DSDD	48	360K	.4A	.3A	6MSEC	Drect	O.K. in PBox
TEAC	FD55BV-06	1/2	DSDD	48	360K			6MSEC	Drect	No Hd Ld Sol
TEAC	FD55E	1/2	SSQD	96	500K			3MSEC	Drect	
TEAC	FD55F	1/2	DSQD	96	1M			3MSEC	Drect	
TEAC	FD55GFV-AT	1/2	DSQD	96	1.2M					For the "AT"
TEC	FB503	1/2	SSSD	48	90K				DRECT	2-DR. IN PBOX
Toshiba	5401		DSDD							
Toshiba	ND04D	1/2								
Toshiba	ND040T	1/2	DSDD							
Y.E.Data	YD580	1/2								

This information is intended to help TI-99/4A users in identifying disk drives that may be compatible with thier Peripheral Expansion Boxes and with their present disk systems. Since all information had been garnered from vendor advertisements, it is assumed to be correct, but must, none-the-less be used with caution due to transcription and other typographical errors.

If any reader can in any way add to the information presented, please do so by contacting the author at Startext MC 77536. Your help is appreciated!

PRINTER CONTROL CHARACTER/CODE

CROSS REFERENCE CHART

NAME OF PRINTER	EPSON MX80	PROWRITER	GEMINI-10X
CARRIAGE RETURN	CHR$(13)	CHR$(13)	CHR$(13)
LINE FEED	CHR$(10)	CHR$(10)	CHR$(10)
FORM FEED (TOP OF PAGE)	CHR$(12)	CHR$(12)	CHR$(12)
FORM LENGTH = ##	CHR$(27)&"C"&CHR$(##)		CHR$(27)&"C"&CHR$(##)
COMPRESSED MODE = ON	CHR$(15) [SI]	CHR$(27)&"Q"	CHR$(15) [SI]
COMPRESSED MODE = OFF	CHR$(18)		
DBL. WIDTH MODE = ON	CHR$(14) [SO]	CHR$(14) [SO]	CHR$(14) [SO]
DBL. WIDTH MODE = OFF	CHR$(20)	CHR$(15)	CHR$(20)
EMPHASIS MODE = ON	CHR$(27)&"E"		
EMPHASIS MODE = OFF	CHR$(27)&"F"		
DBL. STRIKE MODE = ON	CHR$(27)&"G"	CHR$(27)&"!"	
DBL. STRIKE MODE = OFF	CHR$(27)&"H"	CHR$(27)&CHR$(34)	
TO SET HORIZONTAL TABS	CHR$(27)&"D"	CHR$(27);CHR$(40); "###".	CHR$(27)&"D"&CHR$(##)
## = SPECIFIC TAB	&CHR$((1(=##(=80 OR 132)+128)	CHR$(27);CHR$(41);"###"	.... &CHR$(#)&CHR$(0)
END TAB SET STMT	&CHR$(0 + 128)	(ABOVE CODE CLEARS TAB)	
TO TAB ACROSS	CHR$(9 + 128)	CHR$(9)	
TO CLEAR ALL TABS		CHR$(27);CHR$(48)	
TO SET VERTICAL TABS	CHR$(27)&"B"		CHR$(27);CHR$(80);CHR$(##);
## = SPECIFIC TAB	&CHR$(1(=64 + 128)		CHR$(##); ... CHR$(0)
END TAB SET STMT	&CHR$(0 + 128)		
TO TAB DOWN	CHR$(11 + 128)	CHR$(11)	CHR$(11)
CANCEL PRINT & RESET	CHR$(24)	CHR$(24)	
DELETE DATA & RESET	CHR$(127)	CHR$(127)	CHR$(127)
LINE SPACING	CHR$(27)&CHR$(0(=#(2)		CHR$(27);CHR$(#)
	IF # = 0, SPACE=1/8	CHR$(27);"B"	IF # = 48, SPACE=1/8
	IF # = 1, SPACE=7/72	CHR$(27);"T";"14"	IF # = 49, SPACE=7/72
	IF # = 2, SPACE=1/6	CHR$(27);"A"	IF # = 50, SPACE=1/6
	(IF PREVIOUS SETTING)		
PRINTER ON [DC1]	CHR$(17)		
(DEVICE SELECT)			
PRINTER OFF [DC3]	CHR$(19)		
(DEVICE UN-SELECT)			
BACKSPACE		CHR$(8)	CHR$(8)
PICA FONT (10 CPI)		CHR$(27);"N"	CHR$(27);CHR$(66);CHR$(1)
ELITE FONT (12 CPI)		CHR$(27);"E"	CHR$(27);CHR$(66);CHR$(2)
PROPORTIONAL SPACING		CHR$(27);"P"	
FORWARD PRINT		CHR$(27);"f"	
REVERSE PRINT		CHR$(27);"r"	
UNDERLINING = ON		CHR$(27);"X"	
UNDERLINING = OFF		CHR$(27);"Y"	

Note: This table is intended to list a variety of useful printer commands for some of the most popular printers. This table is not guaranteed to be totally correct nor is this table intended to represent all of the possible printer commands for these printers. In actuality, these printers have graphic control commands which are not shown in this table. For exact definitions of your printer's control codes, please refer to printer manual.

TI 99/4A ERROR CODE REFERENCE CHART

The following error code listings were printed in the February, 1986 issue of HOCUS, Newsletter of the Milwaukee Area 99/4 User Group of Wauwatosa, Wisconsin.

EXTENDED BASIC

10 Numeric Overflow
14 Syntax Error
16 Illegal after Sbrtn
19 Name too long
20 Unrecognized Char
24 $/# Mismatch
28 Improperly used name
36 Image error
39 Memory Full
40 Stack Overflow
43 NEXT without FOR
44 FOR-NEXT nesting
47 Must be in Sbrtn
48 Recursive Sbrtn CALL
49 Missing SUBEND
51 RETURN without GOSUB
54 String Truncated
56 Speech $ too long
57 Bad Subscript
60 Line not found
61 Bad Line #
62 Line too long
67 Can't CONtinue
69 Command illegal in prgrm
70 Only legal in prgrm
74 Bad Argument
78 No program present
79 Bad value
80 Nil
81 Incorrect argument list
82 Nil
83 Input Error
84 Data Error
97 Protection Violation
109 File Error
130 I/O Error
135 Sbrtn not found

DISK MANAGER ERROR CODES

#:	First #	Second #
1:	OTHER	Rec not found
2:	SEEK/STEP	Cyclic Redundancy
3:	INPUT	Lost Data
4:	PRINT	Write protect
5:	NIL	Write fault
6:	NIL	No Disk Drive
7:	NIL	Invalid input
8:	NIL	
9:	Special Error Code for Comprehensive Test	

I/O ERRORS

#:	FIRST #	SECOND #
0:	OPEN	Device not found
1:	CLOSE	Write Protected
2:	INPUT	Bad Open Attribute
3:	PRINT	Invalid I/O Command
4:	RESTORE	Out of Space
5:	OLD	EOF
6:	SAVE	Device Error
7:	DELETE	File/Data Mismatch

TI BASIC ERROR CODES PERTAINING TO DISK SYSTEM

#:	FIRST #	SECOND #
0:	OPEN	Can't find specified Disk Drive
1:	CLOSE	Disk or program is Write Protected
2:	INPUT	Bad Open Attribute
3:	PRINT	Illegal Operation
4:	RESTORE	Disk full or too many files opened
5:	OLD	Attempt to read past EOF
6:	SAVE	Device Error
7:	DELETE	File Error
9:	EOF	

EDITOR/ASSEMBLER ERROR CODES

X.B. ERROR EQUATES

ERRNO	>0200	2	Numeric Overflow
ERRSYN	>0300	3	Syntax Error
ERRIBS	>0400	4	Ill. after Sbprgm
ERRNQS	>0500	5	Unmatched Quotes
ERRNTL	>0600	6	Name too long
ERRSNM	>0700	7	$/# Mismatch
ERROBE	>0800	8	Option Base Error
ERRMUV	>0900	9	Improperly used name
ERRIM	>0A00	10	Image Error
ERRMEM	>0B00	11	Memory Full
ERRSO	>0C00	12	Stack Overflow
ERRNWF	>0D00	13	NEXT without FOR
ERRFNN	>0E00	14	FOR-NEXT nesting
ERRSNS	>0F00	15	Must be in Sbprgrm
ERRRSC	>1000	16	Recursive Sbprgrm
ERRMS	>1100	17	Missing SUBEND
ERRRWG	>1200	18	RETURN without GOSUB
ERRST	>1300	19	String truncated
ERRRBS	>1400	20	Bad subscript
ERRSSL	>1500	21	Speech $ too long
ERRLNF	>1600	22	Line not found
ERRBLN	>1700	23	Bad line number
ERRLTL	>1800	24	Line too long
ERRCC	>1900	25	Can't Continue
ERRCIP	>1A00	26	Illegal in program
ERROLP	>1B00	27	Only legal in program
ERRBA	>1C00	28	Bad argument
ERRNPP	>1D00	29	No program present
ERRBV	>1E00	30	Bad value
ERRIAL	>1F00	31	Incorrect argument list
ERRINP	>2000	32	Input error
ERRDAT	>2100	33	Data error
ERRFE	>2200	34	File error
ERRIO	>2400	36	I/O error
ERRSNF	>2500	37	Subprogram not found
ERRPV	>2700	39	Protection violation
ERRINV	>2800	40	Unrecognized character
WRNNO	>2900	41	Numeric overflow
WRNST	>2A00	42	String truncated
WRNNPP	>2B00	43	No program present
WRNINP	>2C00	44	Input error
WRNIO	>2D00	45	I/O error

LOADER ERROR CODES

0-7 Standard I/O
8 Memory overflow
9 Not used
10 Illegal tag
11 Checksum error
12 Unresolved ref.

EXECUTION ERRORS

0-7 Standard I/O
08 Memory Full
09 Incorrect Statement
0A Illegal Tag
0B Checksum Error
0C Dup. Definition
0D Unresolved Ref.
0E Incorrect Statement
0F Program not found
10 Incorrect Statement
11 Bad Name
12 Can't CONtinue
13 Bad Value
14 Number too big
15 String-Number
16 Bad Argument
17 Bad Subscript
18 Name Conflict
19 Can't do that
1A Bad Line Number
1B FOR NEXT Error
1C I/O Error
1D File Error
1E Input Error
1F Data Error
20 Line too long
21 Memory Full
22- Unknown Error Code

RS232c ERRORS

OPEN:	00	Device cannot be opened
	02	Software Switch Error
	06	Hardware Error
INPUT:	24	Internal Data too large for buffer
	26	'CLEAR' pressed or Hardware Error
PRINT:	36	'CLEAR' pressed or Hardware Error
OLD:	50	Can't load from specified device
	52	Can't use software switch with 'OLD'
	54	Program too large to load
	56	'CLEAR' pressed or Hardware Error
SAVE:	60	Can't save to specified device
	62	See 02. Can't use with SAVE
	66	'CLEAR' pressed or Hardware Error
MISC:	43,73,83,93, Executing Illegal Command	

TI WRITER ERROR CODES

0 - Indicates Disk Controller not on; OR: Diskette not Initialized
6 - No Disk in Drive; OR: Is upside down; OR: Drive is not turned on
7 - No Disk in Drive
00 - Illegal use of LoadF, PrintF; OR: Error in using those commands
02 - No file in Diskette with Filename used
04 - Disk is full
06 - PrintF Command in progress was interrupted; OR: Disk Door was opened while Red Light was on.
07 - Invalid Filename (I.E. Name too long or using invalid characters)
15 - Invalid Disk Drive Number, or Device

KEY CODE / TOKENIZED BASIC CODE CHART

Compiled by Don Donlan

COMMAND	ASCII	Hex	Key
Marks EOL	0	>00	____
AID	1	>01	FCTN 7
BREAK/INTRUPT	2	>02	FCTN 4
DELETE CHAR.	3	>03	FCTN 1
INSERT	4	>04	FCTN 2
*QUIT/RESET	5	>05	FCTN =
REDO/REPEAT	6	>06	FCTN 8
ERASE A LINE	7	>07	FCTN 3
Cursor Left	8	>08	FCTN S
Cursor Right	9	>09	FCTN D
Cursor Down	10	>0A	FCTN X
Cursor Up	11	>0B	FCTN E
PROC'D	12	>0C	FCTN 6
Carriage Rtrn	13	>0D	ENTER
BEGIN	14	>0E	FCTN 5
BACK	15	>0F	FCTN 9
*DLEscape	16	>10	CTRL P
*DC1 (X-ON)	17	>11	CTRL Q
*DC2	18	>12	CTRL R
*DC3 (X-OFF)	19	>13	CTRL S
*DC4	20	>14	CTRL T
*NAKnowledge	21	>15	CTRL U
*SYNc idle	22	>16	CTRL V
*ETBlock	23	>17	CTRL W
*CANcel	24	>18	CTRL X
*End of Medium	25	>19	CTRL Y
*SUBstitute	26	>1A	CTRL Z
*ESCape	27	>1B	CTRL .
*File Separatr	28	>1C	CTRL ;
*Grp Separator	29	>1D	CTRL =
Cursor Char.	30	>1E	____
Edge Char.	31	>1F	____
Blank/Space	32	>20	Space
	33	>21	____ !
	34	>22	____ "
	35	>23	____ #
	36	>24	____ $
	37	>25	____ %
	38	>26	____ &
	39	>27	____ '
	40	>28	____ (
	41	>29	____)
	42	>2A	____ *
	43	>2B	____ +
	44	>2C	____ ,
	45	>2D	____ -
	46	>2E	____ .
	47	>2F	____ /
	48	>30	____ 0
	49	>31	____ 1
	50	>32	____ 2
	51	>33	____ 3
	52	>34	____ 4
	53	>35	____ 5
	54	>36	____ 6
	55	>37	____ 7
	56	>38	____ 8
	57	>39	____ 9
	58	>3A	____ :
	59	>3B	____ ;
	60	>3C	____ <
	61	>3D	____ =
	62	>3E	____ >
	63	>3F	____ ?
Null	128	>80	CTRL ,
ELSE.....	129	>81	CTRL A
::.....	130	>82	CTRL B
! (REM).....	131	>83	CTRL C
IF.....	132	>84	CTRL D
GO.....	133	>85	CTRL E
GOTO.....	134	>86	CTRL F
GOSUB.....	135	>87	CTRL G
RETURN.....	136	>88	CTRL H

COMMAND	ACSCII	Hex	Key
	64	>40	____ @
Upper Case..	65	>41	____ A
	66	>42	____ B
	67	>43	____ C
	68	>44	____ D
	69	>45	____ E
	70	>46	____ F
	71	>47	____ G
	72	>48	____ H
	73	>49	____ I
	74	>4A	____ J
	75	>4B	____ K
	76	>4C	____ L
	77	>4D	____ M
	78	>4E	____ N
	79	>4F	____ O
	80	>50	____ P
	81	>51	____ Q
	82	>52	____ R
	83	>53	____ S
	84	>54	____ T
	85	>55	____ U
	86	>56	____ V
	87	>57	____ W
	88	>58	____ X
	89	>59	____ Y
	90	>5A	____ Z
	91	>5B	____ [
	92	>5C	____ \
	93	>5D	____]
	94	>5E	____ ^
Underline	95	>5F	____ _
Grave Accent	96	>60	____ `
Lower Case..	97	>61	____ a
	98	>62	____ b
	99	>63	____ c
	100	>64	____ d
	101	>65	____ e
	102	>66	____ f
	103	>67	____ g
	104	>68	____ h
	105	>69	____ i
	106	>6A	____ j
	107	>6B	____ k
	108	>6C	____ l
	109	>6D	____ m
	110	>6E	____ n
	111	>6F	____ o
	112	>70	____ p
	113	>71	____ q
	114	>72	____ r
	115	>73	____ s
	116	>74	____ t
	117	>75	____ u
	118	>76	____ v
	119	>77	____ w
	120	>78	____ x
	121	>79	____ y
	122	>7A	____ z
	123	>7B	____ {
	124	>7C	____ ¦
	125	>7D	____ }
	126	>7E	____ ~
Delete Char.	127	>7F	____
~) Grtr Than.	192	>C0	FCTN J
~+.....	193	>C1	FCTN K
~-.....	194	>C2	FCTN L
~*.....	195	>C3	FCTN M
~/.....	196	>C4	FCTN N
~^.....	197	>C5	FCTN Q
~ undefined	198	>C6	FCTN Y
Flag Quoted$	199	>C7	____
Fl Unquoted$	200	>C8	____

COMMAND	ASCII	Hex	Key
DEF.....	137	>89	CTRL I
DIM.....	138	>8A	CTRL J
END.....	139	>8B	CTRL K
FOR.....	140	>8C	CTRL L
LET.....	141	>8D	CTRL M
BREAK.....	142	>8E	CTRL N
UNBREAK.....	143	>8F	CTRL O
TRACE.....	144	>90	CTRL P
UNTRACE.....	145	>91	CTRL Q
INPUT.....	146	>92	CTRL R
DATA.....	147	>93	CTRL S
RESTORE.....	148	>94	CTRL T
RANDOMIZE...	149	>95	CTRL U
NEXT.....	150	>96	CTRL V
READ.....	151	>97	CTRL W
STOP.....	152	>98	CTRL X
DELETE.....	153	>99	CTRL Y
REM.....	154	>9A	CTRL Z
ON.....	155	>9B	CTRL .
PRINT.....	156	>9C	CTRL ;
CALL.....	157	>9D	CTRL =
OPTION.....	158	>9E	CTRL 8
OPEN.....	159	>9F	CTRL 9
CLOSE.....	160	>A0	____
SUB.....	161	>A1	____
DISPLAY.....	162	>A2	____
%IMAGE.....	163	>A3	____
%ACCEPT.....	164	>A4	____
%ERROR.....	165	>A5	____
%WARNING.....	166	>A6	____
%SUBEXIT.....	167	>A7	____
%SUBEND.....	168	>A8	____
%RUN.....	169	>A9	____
%LINPUT.....	170	>AA	____
	171	>AB	____
	172	>AC	____
	173	>AD	____
	174	>AE	____
	175	>AF	____
~THEN.....	176	>B0	CTRL 0
~TO.....	177	>B1	CTRL 1
~STEP.....	178	>B2	CTRL 2 ?
~Comma (,)	179	>B3	CTRL 3
~Semi-cln(;)	180	>B4	CTRL 4
~Colon (:)	181	>B5	CTRL 5
~Rt. Paren.)	182	>B6	CTRL 6
~Lt. Paren. (	183	>B7	CTRL 7
~Ampersand &	184	>B8	FCTN ,
	185	>B9	____
~%OR.....	186	>BA	FCTN /
~%AND.....	187	>BB	CTRL / ?
~%XOR.....	188	>BC	FCTN 0
~%NOT.....	189	>BD	FCTN ;
=.....	190	>BE	FCTN 8
< Less Than.	191	>BF	FCTN H

COMMAND	ASCII	Hex	Key
Flag Line No	201	>C9	____
EOF.....	202	>CA	____
ABS.....	203	>CB	____
ATN.....	204	>CC	____
COS.....	205	>CD	____
EXP.....	206	>CE	____
INT.....	207	>CF	____
LOG.....	208	>D0	____
SGN.....	209	>D1	____
SIN.....	210	>D2	____
SQR.....	211	>D3	____
TAN.....	212	>D4	____
LEN.....	213	>D5	____
CHR$.....	214	>D6	____
RND.....	215	>D7	____
SEG$.....	216	>D8	____
POS.....	217	>D9	____
VAL.....	218	>DA	____
STR$.....	219	>DB	____
ASC.....	220	>DC	____
%PI.....	221	>DD	____
REC.....	222	>DE	____
%MAX.....	223	>DF	____
%MIN.....	224	>E0	____
%RPT$.....	225	>E1	____
	226	>E2	____
	227	>E3	____
	228	>E4	____
	229	>E5	____
	230	>E6	____
	231	>E7	____
%NUMERIC.....	232	>E8	____
%DIGIT.....	233	>E9	____
%UALPHA.....	234	>EA	____
%SIZE.....	235	>EB	____
%ALL.....	236	>EC	____
%USING.....	237	>ED	____
%BEEP.....	238	>EE	____
%ERASE.....	239	>EF	____
%AT.....	240	>F0	____
BASE.....	241	>F1	____
TEMPORARY...	242	>F2	____
VARIBLE.....	243	>F3	____
RELATIVE....	244	>F4	____
INTERNAL....	245	>F5	____
SEQUENTIAL..	246	>F6	____
OUTPUT.....	247	>F7	____
UPDATE.....	248	>F8	____
APPEND.....	249	>F9	____
FIXED.....	250	>FA	____
PERMANENT...	251	>FB	____
TAB.....	252	>FC	____
# (FILE NUM)	253	>FD	____
%VALIDATE	254	>FE	____
2 MARK EOF	255	>FF	____

* Used in Pascal. Useful information found in the Users Reference Guide page III-2.

% Used in Extended BASIC.

? Little used parameters related to disk files. See the TI Users Reference Guide, page II-121.

~ Compiled from MicroCompendium. Other information compiled from past issues of 99'er Magazine.

We wish to give credit to the PUGET SOUND 99'ERS who printed this All Purpose Handy Dandy Reference sheet in their APRIL 85 Newsletter.

COLOR CODES

COLOR	VALUE
TRANSPARENT	1
BLACK	2
MED. GREEN	3
LT. GREEN	4
DARK BLUE	5
LT. BLUE	6
DK. RED	7
CYAN	8
MED. RED	9
LT. RED	10
DK. YELLOW	11
LT. YELLOW	12
DK. GREEN	13
MAGENTA	14
GRAY	15
WHITE	16

PATTERN IDENTIFIER CONVERSION TABLE

0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	A
1	0	1	1	B
1	1	0	0	C
1	1	0	1	D
1	1	1	0	E
1	1	1	1	F

ERROR CODES

FIRST	COMMAND OR STATEMENT
0	OPEN
1	CLOSE
2	INPUT
3	PRINT
4	RESTORE
5	OLD
6	SAVE
7	DELETE
9	EOF

SECOND	TYPE OF ERROR
0	DRIVE NOT FOUND
1	DEVICE or FILE WRITE PROTECTED
2	BAD OPEN ATTRIBUTE
3	ILLEGAL OPERATION
4	OUT OF SPACE
5	ATTEMPT TO READ PAST END OF FILE
6	DEVICE ERROR or HARDWARE ERROR
7	FILE ERROR - File or disk does not exist

ASCII CODES

CODE		CODE		CODE		CODE		CODE		CODE		CODE		CODE	
30		48	0	65	A	82	R	99	c	116	t	133	GO	150	NEXT
31		49	1	66	B	83	S	100	d	117	u	134	GOTO	151	READ
32		50	2	67	C	84	T	101	e	118	v	135	GOSUB	152	STOP
33	!	51	3	68	D	85	U	102	f	119	w	136	RETURN	153	DELETE
34	"	52	4	69	E	86	V	103	g	120	x	137	DEF	154	REM
35	#	53	5	70	F	87	W	104	h	121	y	138	DIM	155	ON
36	$	54	6	71	G	88	X	105	i	122	z	139	END	156	PRINT
37	%	55	7	72	H	89	Y	106	j	123	{	140	FOR	157	CALL
38	&	56	8	73	I	90	Z	107	k	124	|	141	LET	158	OPTION
39	'	57	9	74	J	91	[	108	l	125	}	142	BREAK	159	OPEN
40	(	58	:	75	K	92	\	109	m	126	~	143	UNBREAK	160	CLOSE
41	)	59	;	76	L	93	]	110	n	127	DEL	144	TRACE	161	SUB
42	*	60	<	77	M	94	^	111	o	126		145	UNTRACE	162	DISPLA
43	+	61	=	78	N	95	_	112	p	129	ELSE	146	INPUT	163	IMAGE
44	,	62	>	79	O	96	`	113	q	130	::	147	DATA	164	ACCEPT
45	-	63	?	80	P	97	a	114	r	131	!	148	RESTORE	165	ERROR
46	.	64	@	81	Q	98	b	115	s	132	IF	149	RANDOMIZE		
47	/														

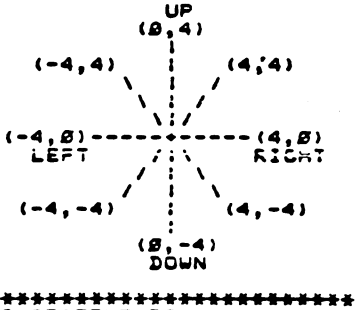
CHARACTER SETS

SET	ASCII CODES
0	30-31
1	32-39
2	40-47
3	48-55
4	56-63
5	64-71
6	72-79
7	80-87
8	88-95
9	96-103
10	104-111
11	112-119
12	120-127
13	128-131
14	136-143
15	144-151\BASIC
16	152-159/ONLY

CALL KEY VALUE OF KEYSTROKES

CODE	KEYSTROKE
1	FCTN 7
2	FCTN 4
3	FCTN 1
4	FCTN 2
5	FCTN =
6	FCTN 6
7	FCTN 3
8	FCTN 5
9	FCTN D
10	FCTN X
11	FCTN E
12	FCTN 6
13	ENTER
14	FCTN 5
15	FCTN 9

JOYSTICK RETURN VALUES



CONTROL CODES

ASC	PRESS	COMMENTS
1	CTRL A	START OF HEADING
2	CTRL B	START OF TEXT
3	CTRL C	END OF TEXT
4	CTRL D	END OF TRANS.
5	CTRL E	INQUIRY
6	CTRL F	ACKNOWLEDGE
7	CTRL G	BELL
8	CTRL H	BACKSPACE
9	CTRL I	HORIZ. TAB
10	SHFT/ENTR	LINE FEED
11	CTRL K	VERTICAL TAB
12	CTRL L	FORM FEED
13	ENTER	CARRIAGE RETURN
14	CTRL N	SHIFT OUT
15	CTRL O	SHIFT IN
16	CTRL P	DATA LINK ESCAPE
17	CTRL Q	DEVICE CONTROL 1
18	CTRL R	DEVICE CONTROL 2
19	CTRL S	DEVICE CONTROL 3
20	CTRL T	DEVICE CONTROL 4
21	CTRL U	NEG. ACKNOWLEDGE
22	CTRL V	SYNCHRONOUS IDLE
23	CTRL W	END OF TRANSMIS.
24	CTRL X	CANCEL
25	CTRL Y	END OF MEDIUM
26	CTRL Z	SUBSTITUTE
27	CTRL .	ESCAPE
28	CTRL M	FILE SEPARATOR
29	CTRL J	GROUP SEPARATOR
30	CTRL 8	RECORD SEPARATOR
31	CTRL 9	UNIT SEPARATOR
127	SHIFT V	DELETE CHARACTER

EXTENDED BASIC STATEMENTS

ACCEPT	DISTANCE	KEY	ON WARNING	SCREEN
CHAR	END	LET	OPEN	SOUND
CHARPAT	ERR	LINK	OPTION BASE	SPSET
CHARSET	FOR	LINPUT	PATTERN	SPRITE
CLEAR	GCHAR	LOAD	PEEK	STOP
CLOSE	GOSUB	LOCATE	POSITION	SUB
COINC	GOTO	MAGNIFY	PRINT	
COLOR	HCHAR	MOTION	RANDOMIZE	
DATA	IF THEN	NEXT	READ	
DEF	IMAGE	ON BREAK	REM	
DELSPRITE	INIT	ON ERROR	RESTORE	
DIM	INPUT	ON GOSUB	RETURN	
DISPLAY	JOYST	ON GOTO	SAY	

CALL LOADS

ADDRESS	PARAMETERS	DESCRIPTION
-32768	POKE 0-255	RATE of FLASH of CURSOR
-31786	POKE 192	DISABLE SPRITE action
-31804	PEEK X,Y	Returns to Title Screen
	POKE X,Y	
-31806	POKE 16	DISABLE QUIT KEY(Fctn =)
	POKE 32	DISABLES SOUND
	POKE 64	DISABLES AUTOSPRITE MOTION
-31860	POKE 4	Goes to Console BASIC after 'NEW' is typed
-31860	POKE 0	Runs DSK1.LOAD
-31876	POKE 0-26	Highest # SPRITE in motion
-31866	POKE 63,255	Disables Disks,NEW free.mem
-31931	POKE 0/128	Unprotects/Protects prgrm
-31962	POKE -2	Returns to TITLE SCREEN
-31962	POKE -55	Runs DSK1.LOAD
-31952	PEEK A,B,C,D	Recovers program with LOAD
-28672	PEEK A	96=Speech Syn. 0=No Speec

UTILIZING MINI-MEMORY CAPABILITIES

Part 1

Two new low-cost cassette-based programs using the Mini-Memory cartridge will be available October. . .

TI-Miniwriter, a new low-cost word processing system for the TI-99/4A, offers users full screen text editing, a 24 X 40 character window on an 80 scrolling character display, upper and lower case characters, movable copy, and add and delete functions for characters or lines. Up to 9500 characters may be stored per file.

The TI Mini-Memory cartridge and a TI Program Recorder are the only requirements necessary to bring word processing capabilities like these to your Home Computer. An RS232 Interface and printer are required to obtain printed copies of documents.

Other features of the new TI-Miniwriter are a search function; up, down, left, or right scrolling; text buffer purging, and cancel commands. TI-Miniwriter will be available at a suggested retail price of $19.95. The TI-Miniwriter program cassette was developed for Texas Instruments by Model Masters, Inc.

Another new TI-99/4A cassette-based program which utilizes the TI Mini-Memory cartridge is Entrapment. Entrapment is a game of skill and speed where players are in command of a spaceship that patrols the Earth's atmosphere. Players must defend the Earth against an invading legion of hostile larvae entering the atmosphere.

Entrapment requires a TI Mini-Memory cartridge and TI Program Recorder. Joysticks are recommended. This program will be available at a suggested retail price of $19.95. Entrapment is manufactured by TI under license from American Software Design and Distribution Company.

UTILIZING MINI-MEMORY CAPABILITIES

Part 2

For the computer enthusiast looking for a versatile and economical way to customize his or her basic Home Computer system, the TI Mini-Memory cartridge offers unique expansion capabilities.

This expansion cartridge increases the memory capacity of your Home Computer by adding a total of 14K bytes of memory to your system. This includes 4K RAM (Random Access Memory), 6K GROM (Graphics Read Only Memory), and 4K ROM (Read Only Memory.)

The Mini-Memory's built in battery enables a user to store a 4K BASIC program or data file in the cartridge even after the cartridge has been removed from the console. (4K memory will accommodate approximately 250 lines of BASIC).

With the use of a cassette-based Line-By-Line Symbolic Assembler, a user can develop Assembly Language programs to add new instructions to the computer's BASIC Language which are not already available. (The Line-By-Line Assembler is packaged with the Mini-Memory cartridge). The Assembler cassette loads directly into the Mini-Memory cartridge from a cassette recorder. This cassette enables users to write assembly language programs which can then be saved on additional cassette tapes.

Users can link BASIC programs to Assembly Language subprograms by utilizing the TI BASIC subprograms and utility routines available in the Mini-Memory cartridge. This allows a user to address the internal components of the Home Computer directly.

With access to the machine resources of the computer, a user can build a customized system using a minimum computer configuration of console, Mini-Memory, and monitor. With a TI Program Recorder, procedures can be saved for future use.

TI Mini-Memory is available at a suggested retail price of $99.95. The TI Program Recor is recommended for saving programs developed with the Mini-Memory cartridge. Mini-Memory also may be used with all other Home Computer peripherals.

Conclusion of a two part article on the TI Mini-Memory. Taken in part from the September, 1983 TI Newsletter.

More On TI Memory

From COMPUTE!, August 1983, page 14; By Howard Patlik

Many owners of the TI-99-4/A would be interested in determining the exact amount of available memory (in bytes). This two-line program is very simple and can save a lot of hair pulling when you write programs which fill the memory. Here is the Program:

Step 1
Enter the following:

```
1 A=A+8
2 GOSUB 1
```

Do not use a variable that has already appeared in the program. For example, if you have used the variable "A" within the program, choose another. Second, the program must work correctly before using this mini program.

Step 2
Once this is entered into the memory, enter the RUN command. The process will take between 15 and 30 seconds to execute, depending upon the length of your program. After execution, MEMORY FULL IN 1 will appear. Now enter PRINT A (no line number) and a value will appear on the screen. This value is the number of bytes remaining in the computer's memory.

To determine the total amount of free memory available. Clear the memory (store your program first) and repeat Steps 1 and 2. The value displayed will be 14536. There are 14536 free bytes available (the mini-program itself uses 40 bytes, so add 40 to the 14536) . The computer is advertised as having 16K bytes. 1424 are used for screen display, etc. So when a program is stored in the memory and you want to determine how many bytes the program used, enter the following:

```
Print 14576-A
```

DON'T REPRESS THE URGE TO SUPRESS A SURGE!

I have heard from many of you who have power problems that are causing you great concern when you are using your computer. Perhaps you could avoid most of these problems by taking a few easy and inexpensive safety precautions. For instance, the December, 1983 issue of BYTE magazine features an article dealing with these problems. Read it! This guy was operating his computer during an electrical storm, an absolute no no!

You should be using a line voltage supressor with a maximum voltage tolerance of about 130 volts A.C. Some of these supressors don't knock the voltage down until it hits 500 volts! Needless to say, when it gets that high there isn't any point in worring about your equipment!

Try to keep your equipment away from flourescent lights, sas they can cause all sorts of quirks in your equipment. Especially a faulty starter in the older models. You can also have difficulties from a power drain or votage spike. These can be caused by an electrical appliance engaged with a high amperage drain on the same circuit as your computer; such as a dish washer, toaster, iron or refrigerator. This can cause a momentary change in power and it can affect, and sometimes, damage, your equipment. Avoid static electricity, as it is an instant killer for disks, tapes, and all electronic hardware! Use anti static spray on your carpet if it is a problem. With a little common sense you can avoid the problems, so go visit Radio Shack and end a few dollars for some peace of mind and computer security!

Editor's Note: This article was taken, in full, from the 99/4 Users of America Newsletter in our ongoing Newsletter Exchange Program.

BACKGAMMON INSTRUCTIONS

by Dennis Sherfy

Backgammon is a game for two players, or for one player against the computer. For the two-player version, enter the initials of each player in response to the prompt. To play against the computer, enter 99 as the second player's initials.

The game-board in the computer game can be confusing if you are not already familiar with the game. The points are not displayed on the board. They are represented by numbers from 01 to 24. It would be beneficial for you to obtain a Backgammon board and become familiar with it before trying to play the computer version.

We will begin with a few definitions.

POINTS: These are the spaces on the playing board. On a regular board, they are shaped like narrow triangles. Players land on points as they move around the board.

STONES: The playing pieces.

BAR: The place where pieces are held after they are hit.

HIT: Landing on a space which is occupied by a single opponent's stone. This sends your opponent's stone to the bar. Your opponent must re-enter that stone at his/her first point. In other words, that stone must start over again and you have lost the progress that stone achieved during the game.

When a player has a piece on the bar, it must be re-entered before that player may move any other stone. If the player cannot re-enter the stone on his/her next turn, that turn is forfeited.

CLOSED POINT: This occurs when a player has two or more stones on the same point. This is also called MAKING A POINT. When a player makes a point, an opponent may not land on that point, and the player's stones are safe. They cannot be sent to the bar. One strategy is to keep as many of your stones as possible on safe points. Thus, they cannot be hit.

BEAR OFF: This means to remove your stones from the board. The first player to bear off all his/her stones wins the game.

The object of the game is to move all your stones (pieces) into your inner table, and then off the board. White moves in a horseshoe manner from space 24 to 01. Black moves from 01 to 24. As the board is displayed, points 01 to 06 make up white's inner table, and points 07 to 12 make up white's outer table. Similarly, points 24 to 19 make up black's inner table, and points 18 to 13 make up black's outer table.

THE PLAY

The computer chooses one player to go first and rolls the dice. The first person whose initials are entered is white and moves first.

Each die is played separately. If you roll a 2-5, you must move one stone 2 spaces, and another 5 spaces, or you may move one stone 2 spaces, and the same stone 5 more spaces. If you roll doubles you move the value of each die twice. For example if you roll double 5, you move 5, four times.

If you make an error when entering a move, press enter before completing the move. This will generate an error message and allow you to enter your move again. If you have keyed in your complete move, and want to change your mind, do not press enter. Instead, make an intentional error, such as entering a letter. When you press enter this will create an error and you can enter a new move. You must always enter a move with two digits. a move from space 9 to space 11 must be entered as 09 to 11.

If a player has two or more stones on one point, it is called a "closed" ""point"". A player cannot land on a closed point occupied by the other player. He may, however, jump over a closed point.

If there is a single stone on a point, it is called a ""blot"". If a player lands on an opponent's blot, the opposing stone must go to the ""bar"". This means that the stone must start at the beginning and be re-entered on the board.

Black must re-enter a stone to a point between 01 and 06 which is not blocked by white, (a closed point). White must re-enter to a point between 24 and 19. Again, the point cannot be closed by black. If either player has a

BACKGAMMON

LIBRARY BITS, cont'd.

stone on the bar, and cannot re-enter the stone because the points are closed by the opponent, that player forfeits his move. If either player has a stone on the bar, and cannot re-enter the stone because the points are closed by the opponent, that player forfeits his move.

A player must use as much of the dice roll as possible even if it means leaving a blot on a point. That is the luck of the game. This means if you roll a 2 and a 4, you must move a 2 and a 4. If you are unable to use the full roll, you must use either the 2 or the 4. (You would not split up the 4 and move a stone 3 points). This could only happen when your opponent has every possible point blocked to which you could move. The computer will keep track of this automatically. If you cannot move, or if you cannot re-enter a stone from the bar, the computer will pass the turn to the other player.

The strategy of the game is to hit as many of your opponent's blots as possible while keeping your stones paired on points, and safe.

This game is a game of chance in more ways than one One gamble is to leave a lot on your back two points (points 01 02 or 24 23), tempting your opponent to ""hit"" the blot. If your stone is sent to the bar, you have not lost much, and your opponent may have left a blot for you to hit in return.

The computer game does not utilize the doubling cube. This is a way to double and re-double the stakes of the game.

The player who removes all his stones from the board first, wins the game.

If a player removes all his stones from the board before his opponent removes his first stone, it is called a Gammon, and is the equivalent to winning two games.

If, in winning, a player removes all his stones from the board while his opponent still has one or more stones in the winner's inner table it is a Backammon and is equivalent to three games.

THE BACKGAMMON BOOK by Jacoby and Crawford is a reasonably priced book in the softbound version. It is a good reference. The game is easy to learn and can be played in a short time. It represents a good mix of skill and luck. There are strategies to the game which are too extensive for this presentation. Play the game and buy the book if you are interested. I think that you will enjoy it.

TRIALS AND TRIBULATIONS

OF A FLOPPY NATURE

(or how to make a backup)
(copy of a cleaning disk)

by Jerry Trinkl

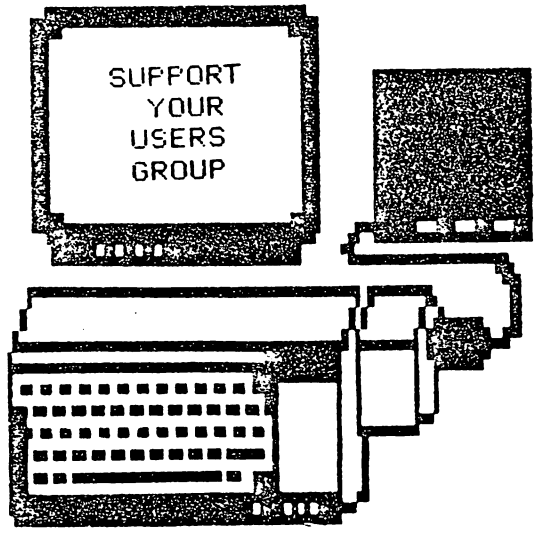
The other day I had the unfortunate experience of losing a disk full of programs. It didn't happen all at once, I occasionally couldn't load a program here and there and sometimes got the error message, "06", the error for no disk drive. In the end, all I got was a no disk drive error. Looking at the disk I found two circular gouges, one on the inside track closest to the hub and one towards the outer most track. I tried to move the disk inside it's jacket and ah ha! The problem was that of a cheap disk that jammed up. It could not even be turned by hand.

Well I had some very good programs on that disk and wasn't about to throw it out just yet. My first thought was to try Disk Fixer, after all that's what it was designed for, right? But if the disk would not turn then Fix could not even read a sector. OK so now what?

I carefully cut out the bad disk from it's jacket and inserted it into an em Verbatum cleaning jacket. And Voila! it worked! I quickly backed up my makeshift cleaning disk on an good disk and was tempted to turn el-cheapo into a frisbee. But decided to save it anyway as a backup and reminder of a great idea that actually worked.

So dig out that old frisbee and give it a try. Your trials will not turn into tribulations and you will occasionally find use for your old useless cleaning jackets.

This article was taken from HOCUS Newsletter of the Milwaukee Area 99/4 User Group, Wauwatosa, Wisconsin.



YOUR QUESTIONS ANSWERED ABOUT "AXIOM" DIRECT-CONNECT PRINTERS!!

by Marc Blaydoe

At the last HUGgers meeting much interest was expressed about the Axiom Direct-Connect Printers. Mr. James Franks, Applications Engineer for Axiom Corporation has provided some information in addition to the brochures.

These printers connect to the TI 99/4A without the need for a P-Box or RS-232 interface. The biggest question at the last meeting was "where does the printer plug into the console?" According to Mr. Franks, the printer plugs into the same outlet that the P-Box and Speech Synthesizer use. However! Mr. Franks happily points out that we can still keep our P-Boxes plugged in!

The printer plug is a "t" connector with an extra port at a right angle to the connector to plug in other peripherals. The secret to the direct connect is a Centronics parallel to parallel interface. While the interface is not an RS-232 it has been designed to answer to the RS-232 port command in your software. By the way, the "bottom line" printer, the GP-100, has true descenders! (i .e., the lower case g's and y's look like g's and y's).

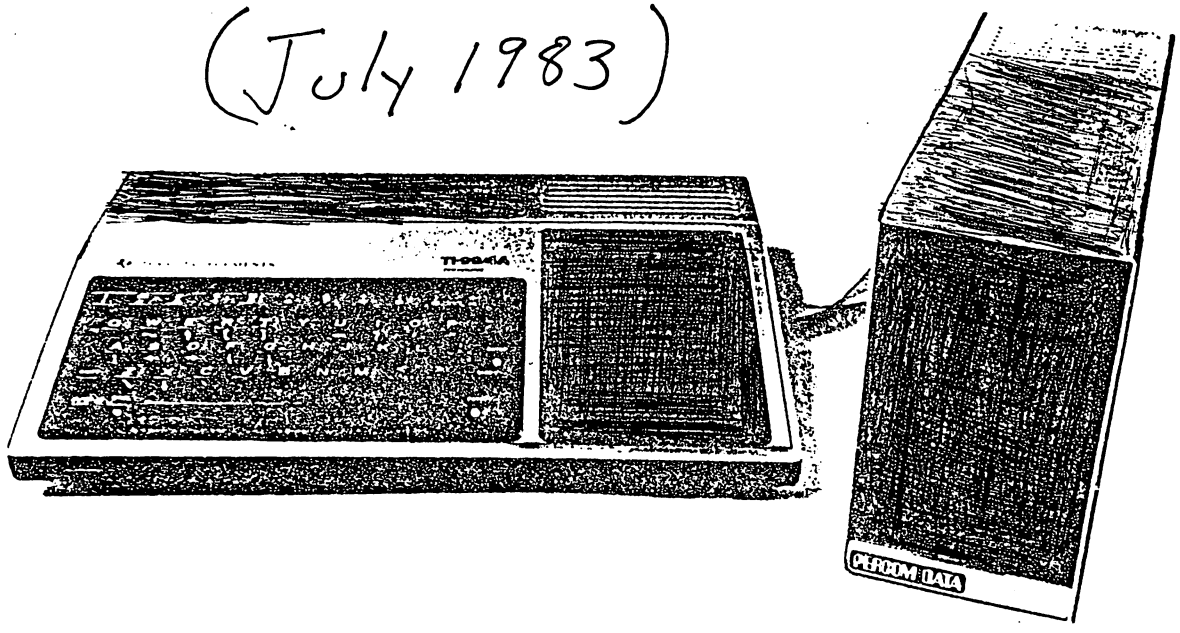
List prices for the Axiom printers are:

GP-100 $309
GP-550 $409
GP-700 $609 (A full color printer!)

Mr. Franks has promised to send us more info soon. If it gets here in time, we will give you an update at the next HUGgers meeting.

Now you can buy a disk drive for your TI-99/4A for $400 LESS than you thought you could!

(July 1983)



Putting a disk drive on your TI-99/4A can be an expensive proposition...unless you add a Percom Data TX-99 for a mere $499. A similar TI-supplied disk configuration requires that you purchase a Peripheral Expansion Box, a Controller Card and, of course, the Drive itself. The tab on that little package is a hefty $899!

MICROS in ACTION

by Wm. H. Cagle V.P.

I just came back from a service school on an X-ray system that my company sells. This machine is used in hospitals to visualize body organs and bone structure. What makes this machine of interest to us, is the fact that this system is under control of a microprocessor. The CPU in use in this system is smaller that the one in our computer, but it is none the less a CPU with memory and data-busses and all the rest of the hardware that is currently in use in any other system.

On power-up the CPU goes to a ROM (read-only-memory) and gets a program that starts a test to see that every assembly that will be used, is in good working condition. This is accomplished by writing a preselected code to each sub-assembly and reading back the "echo" response from these devices. If the response is there, and if it matches what the program said that it should, then the CPU goes to the next test. When all of the tests have been passed, the CPU turns over the system to the operator for setting the conditions for the first X-ray procedure.

When the procedure is selected from the key-board, the CPU reads the keys pressed and stores the operator input and it then checks to see that the input is indeed a legal input. On some systems, the operator can destroy a five thousand dollar X-ray tube by using to much power before the tube is warmed up and they can also ruin a tube by getting it to hot. This system protects the patient and the equipment from such operator error.

When the actual x-ray is in progress, the CPU watches the voltage impressed on the tube, the current through it, the amount of time it has been on, the current through the filament and a host of other things. In fact the only thing the operator has to do, is set the parameters, push and hold the go button and the CPU watches everything else. With this level of control over the operating system, the consistency of the pictures are very good.

For the technical people in our club, imagine a CPU system controling a machine that produces 100,000 volts and delivers 1000 miliamps. Also imagine how much damage occurs when the tube arcs over inside. Some of the servo systems have multiple loops nested inside each other. The amount of care that the designers took to protect the patient is awesome. Every thing is set up so that it is "fail-safe". This means that any time the CPU can't move some value, it will shut down the entire thing and not allow another shot to be processed until the fault is rectified. It also has a trouble code reporting system. There are about sixteen pages of trouble codes in the back of operators manual. This is a great trouble shooting help, when the Tech. calls you and says that " I have an FEC error."

While this system is very expensive, we are selling a lot to hospitals that do the type of procedures where this level of operation is needed. In some of the X-ray procedures used today, it is necessary that each shot in a high-speed run is very consistant on a picture by picture basis and this equipment will do this type of procedure very nicely.

CONSOLE COMMENTS

by Bill Jones

I purchased a copy of THINKING FORTH by Brodie and read it cover to cover last night. The book is definitely advanced, but, as the author says, is not just for FORTH programmers. Brodie defines the discipline of programming, as professionals do it. Most of us amateurs learned to program thru the school of hard knocks, and that school teaches some bad habits. I wish Brodie had written this book when I was just starting out.

THINKING FORTH clearly explains the best ways to bring a project from start to finish, producing code that is more than functional-it is elegant. 90 percent of this books contents applies to any language, not just FORTH, because it describes a programmers philosophy that caused FORTH to be created.

If you do any serious programming, READ THIS BOOK!

SPEAK UP!

Editors Note: The following article was copied from the October, 1984 issue of MicroCompendium.

Sometimes solutions to problems can be so obvious that one never sees them. Anyone who has ever used a cassette player to load data knows how long it can take to locate a particular program or file. Of course, we're supposed to use tape counters and write down the location of each program. But, nobody's perfect.

Here's a tip from the Nine T Nine Users Group of Toronto, Ontario, that may be of use to those who neglect to put things in writing: Put it in speech. That is, before you start recording a file or program onto cassette, record its name using your voice. Doing so, you can hear your voice through the television or monitor speaker when you're searching the tape for the desired program. (Oh, yes, remove the microphone jack from the recorder to record your voice and replace the jack before recording or loading your program.)

Now, wasn't that just too obvious to mention?

BEWARE THE WHITE ELEPHANT!

The comfortable days of our black and silver TI 99/4A's are gone and the new cheaper to produce-and much cheaper looking-white TI 99/4A console. Unfortunately, the third party modules won't work on most of them! There is no way to identify which ones will work by their product numbers, it must be operating to see if you bought a white elephant or a useable console Observe the bottom right hand edge of the screen and look to see if it says c Texas Instruments 1981 or c Texas Instruments V2.2.

If you have version 2.2, then you have the white elephant! The problem is a little thing known as an auto-incrementing memory. Simply put, if the software module does not have the auto-incrementing memory chip, it is rejected by the console and the title screen reappears. Thus, the console will not run modules produced by Funware, Romox, Atari, or Parker Bros. Unfortunate, being as TI isn't going to produce modules either!

Editors Note: The preceding article was taken from the 99/4 Users Of America Newsletter in our continuing Newsletter Exchange Program. For more information on the auto-incrementing memory chip, see the TI Care Package article elsewhere in this Newsletter.

A SPECIAL APPLICATION FOR THE TI

By Dana Walker

Editor's Note: The following article was taken, in full, from "Bug News", the Newsletter of TIBUG, Montgomery, Alabama.

Dr. David Walker is a part time TI rep in Montgomery, Alabama. He teaches at Auburn University in Montgomery. His wife, Dana, teaches at the same University. Dana contributed the llowing article to our newsletter. She is blind.

Having a talking computer used to be only a pipe dream for blind programmers, except rich ones. Now, with the TI 99/4A, the Terminal Emulator module and the speech synthesizer, you can have a talking computer for a few hundred dollars instead of a few thousand. As a result, there is now a loose network of blind TI users all over the country who swap programs and other information.

The first thing I did, after setting up the computer, was to label a few keys in braille, to help me get oriented to the keyboard. A friend had already had the reference manual and the TI BASIC book read onto tape, so I got a copy and began studying.

The first big problem was that the computer is rather uncommunicative about errors. It would just beep and sit silent and uncooperative until my husband came in to read the error message to me. The solution was an error-read program. I just load it before I start my own programming. Then when the computer beeps, I enter "RUN 30000",. The computer then will tell me just what I've been doing wrong.

I've gotten several good programs that are "speeched up". One keeps tract of my talking-book request list; another is a Morse code practice program; another is an address file with an alphabetic sort routine. I am looking for word games like Hangman that can be speeched. I also want to work on a word-processing program to use in writing at the computer; I think best at a keyboard, and the TI 99/4A could tell me what I've written. A friend is working on a program to make an impact printer produce braille hard copy, which will be a big advance.

The only drawback to the 99/4A is that with the Terminal Emulator in use, you can't use any the other program modules. This means that you can't use the memory expansion card. Being able to run two modules at the same time would expand the possible uses of the TI 99/4A for the blind programmer. TI says, by the way, that in about a year they're coming out with te speech on a disc. This will be a great advantage. Meantime, I keep thinking of more things to dow with the computer - like the poetry-writing program. I wonder if it could be modified so that the computer writes limericks.

MICROS IN ACTION

by Bill Cagle

While on a plane, I read an article in a newspaper about computers. This article was based on the time spent on home computers and where this time came from. This data was obtained by a poll sent out by a computer magazine and of the fifteen hundred people responding, fifty-four percent said that they spent less time watching television. They also stated that most of the computer time was spent in playing computer-games with other members of the family or by themselves. This sort of information makes me feel good about the future of computers in our country. I am also glad that computers are beginning to be considered by the lay public as an entertainment device. Just think, if we could also see a swing to programming as a national pastime, imagine what an enormous amounts of software a million people could write.

I also noticed an article in this same paper, about kids and computers. This article was about an undercurrent in our industry, that holds to the notion that "If a child doesn't have a computer by age five, it will be unemployed by twenty-five". That just isn't true, because computer operation can be learned just like anything else and as software becomes more user friendly, the task of operating computer systems will require less and less training. As hardware designs mature more and more, we might see the day of computers without keyboards. This makes the future look very bright indeed.

Recently, my son talked me into seeing a movie named "The Terminator". This film is based on the premise that at some point in the future, the world leader's will "put the security of the world, into the hands of a global computer that has the ability to learn and when this computer learned enough to get tired of the frailties of human nature, it decided to be done with all humans. To accomplish this, it created a global nuclear holocost. The first thing wrong with this concept, is that computers can not reason or learn. The very best they can do, is to compare numbers and detect the bigger or the smaller and sometimes look for a comparison. While some software is so sophisticated, it creates an illusion that the computer is "thinking", while all it is really doing, is just going through a series of, "if then's" and "goto's", etc. There is some research going on at present in the field of artificial intelligence and maybe I will do an article in the near future, on that subject. One example of software that causes the computer to appear to think, is the program called Eliza. This program will be available on our Bulletin Board Service as soon as Bill Jones gets it done.

There is another disturbing trend beginning to move into the white-collar industry. The bigger and better "work station" on your desk, the more powerful you are and the more prestige you have. You will notice that in that last sentence, there was no mention of being able to use the power (of the work station). To make things even worse, the computer industry has responded to this trend by advertising the "up" features of their product. This will, in time, force these user's to buy computers with features that have no bearing on the job that they will be performing. This will tend to produce a sellers market and will probably inflate the price of the "top of the line" business products.

Another year has seen it's last and in retrospect it has been a very full one for the HUGgers. We have seen the BBS be installed and then we have watched with great pleasure as Steve Sims and Bill Jones reworked it till the orginal identity is almost completely gone. For the next year we are going to undertake a massive membership drive and this will give us some new blood. We are also going to continue with the multiple workshop concept for our meetings. There is going to be some effort expended to install more assembly and Forth programs into our club libary. The future looks better and better for our club, doesn't it?

Go FORTH young man and program the world to your likeing.

HAM RADIO AND COMPUTERS

by Bob Summers

Recent rule changes by the FCC have allowed a greater use of the computer by hams. Previously about the only use was for machine generated CW or continous wave telegraphy (morse code.)

Now hams may transmit ASCII and digital information as well. This has allowed 'nets' to spring up as well as bulletin boards. Program exchanges between hams (machine to machine) are now quite common, for example a station here may exchange with one in Japan and Europe at the same time and no long distance phone bill either!!

Another new form of very rapid communication is the advent of packet radio. When transmitted, it sounds like a very short burst of static noise. It can either accept for use at the resident station, or if so coded, relay onto a more distant destination. It operates on a 'handshake basis' - verifying received data against data sent. The most amazing feature is it's ability to use channels that are in use by other modes without causing interference with those modes. It waits until the channel is free and quickly "does it's thing" and then waits until another free period occurs to continue the process.

Plans are now being laid for a "packet satellite" that will give nearly world-wide rapid error free communications to hams. Incidentally, the first non-military, non-commercial satellite ever orbited was OSCAR-1 (Orbiting Satellite Carrying Amateur Radio) back in the mid 70's.

Locally we have a packet network on a 'Simplex 2' meter Frequency. The RCA Radio Club maintains a bulletin board on a 2 meter repeater. There is also a fast-scan T.V. repeater with a weather radar feed from the National Weather Service that is utilized during severe weather alerts. Those of you living on the east side of Indy and having a T.V. with continous tuning slugs in the UHF tuner may be able to tune Channel 14 down and receive this machine without any further antennas or modifications. It's frequency is 425.250 megahertz.

The computer is also utilized by hams in Teletype or RTTY. It's use requires some hardware interfacing and the appropriate software to accomplish. It's primary usage is on the 'low bands' where ranges are world wide. The hardware and software are still available for the TI at about $100 or slightly less. The hardware for CW can be built by the user and I have the program available for anyone interested. All parts are Radio Shack-available and will cost under $20.

Enough for now. But the rules changes have given a more wide-spread use of the computer for communications and pleasure usage of the ham.

73

FROM THE HUGbbs.....

.....AN EDITORIAL.....

.....BY TI TIPS

Our users group seems to be going off into almost as many different directions as there are members. There are people in our organization that are exploring nearly every aspect of the little TI computer.

We try to present things of common interest at monthly meetings but I think the glue that really binds us is the little tidbits of information that we collect and share with one another. On this board I see evidence of an undercurrent of activity amoung very different kinds of people. One gets information on Pascal from another who is working with Forth. Another is picking up valuable information from an assembly programmer that he uses in Forth. At the swap table at the last meeting, there was a flurry of activity among many of the members. Because of our newsletter, we have a reputation around the country as one of the top 5 groups in the U.S.

Unfortunately, more and more people are abandoning their TI's as the date of last production fades farther into the past. One group published an estimate stating that 1000 TI's were being thrown in the closet each week. That only means that we will all have to work that much harder.

Not to keep the TI alive, it will die in time dispite our best efforts, but to use it for what we bought it for. LEARN, that's what you got a computer for. And share it with others. You joined a group for just those reasons. Participate !!

The folling TI Tip was taken from our HUGbbs!

This little tip comes from the Piedmont Computer group in Greenville, SC

There are a lot of times when programs we write need to test for single key input from the keyboard. Usually, the line CALL KEY(0,K,S) or something similar is used. The problem here is that if we are looking for an upper case key and the user puts in a lower case when the alpha lock is up, he misses the intended input.

A solution to the dilema is to change the statement to CALL KEY(3,K,S). Scanning this way ignores the alpha lock key and returns all keystrokes as upper case only.

When you're finished, add the statement: CALL KEY(5,K,S) to set the scan back to normal or it will cause some strange results.

The author says that it's in the XBASIC manual, but hard to find.

DOES IT PAY TO LEARN PROGRAMMING?

This is an article which appeared in the May/June issue of The Electron, and submitted for our Newsletter by Dennis Sherfy.

It is popular these days to assure hesitant computer buyers that they need not learn to program in order to use a computer. That statement is true, but knowing how to program can certainly be an advantage. It can also be quite lucrative. It is estimated that authors Dan Bricklin and Rob Franston have earned a minimum of $10 million in royalties for their hot-selling business program, Visicalc. Mitch Kapor, once an aid in a mental hospital, bought an Apple II on impulse without knowing a thing about programming. Within a year he had written a statistics package that became the heart of Visitrend and Visiplot. His efforts made him $1,700,000 richer.

Paul Lutus lives on a mountain top in the Cascade Range of Oregon. When he saw the first ad for an Apple Computer he bought one, even though he had no electricity. He ran a 1300 ft. extension cord down the mountain side to the nearest power line. Because no word processing equipment was marketed at the time, he wrote a program for his own needs. He sold the program, Apple Writer, outright for $7500, not knowing that royalties would have brought him 20 times that amount. He has wised up since then, and is now in the chips. When he as to go see his publisher, he coasts down the mountain on his bicycle to his private plane, loads the bike aboard, and completes the trip from the publisher's nearest landing strip on his bike.

You don't have to be very old to get into big bucks in programming. Imagine Software of England hired a 16 year old boy to write games at $54,000 a year before he even had his school leaving certificate. Although he may be qualified to make this kind of money, British law doesn't think he is qualified o spend it. At 16 he cannot apply for a credit card and he cannot open a bank account to put his money in!

Another 16 year old English school boy, Peter Simons, has come up with Simons' BASIC, a fantastic software package that adds no less than 114 additional commands to the basic set of 72 commands on the Commodore 64 computer. Some reviewers think Simons BASIC will eventually be built into the Commodore 64 as a standard part of the computer.

There are at least 20 persons in America under the age of 25 who are earning over $100,000 a year developing programs for personal computers. Homemaker Robeta Williams was 26 when she played her first computer game. She decided immediately that she could design a better game, coming up with Mystery House. Her husband programmed it, hoping that it would bring them a little extra money. It sold like gangbusters, earning them the money to start their own software company, Sierra On-Line. Now, three years later, Roberta has five popular games to her credit, including the longest adventure ever published, Time Zone. It takes the average player six months to complete the game playing 10 hours a week.

A QUICK FIX FOR S-BUG BUG

by Tom Knight
Jacksonville, Florida

Editor's Note: The following article is printed in the HUGger Newsletter via the August, 1985 issue of Penn Ohio Users Group Newsletter, Youngstown, Ohio.

When TI finally released Super Bugger it had a "bug" in that it is supposed to be able to disassemble or dump memory to a disk and will not properly do this. (In my opinion this was a TI-induced bug.)

I have been working on this problem and have found a solution that, so far, seems to work fine.

With no other program in memory, "S-Bug" loads from >A000 to >B96A and I will be referencing memory with this assumption.

Memory Location	Contains	Change To
A15A	3F20	101F
B20E	7F00	0FFF
B2F2	3F09	1009
B32A	7F20	101F
B342	7F05	1005
B356	7F00	0FFF
B366	3F09	1009
B37A	7F00	0FFF
B382	3F09	1009

These locations are all references to either the PAB or the data buffer which is used by DSRLINK which, by the way, is included in Super Bugger as are the other utilities used by the program. It is completely stand-alone. All of the utilities are very similar to the ones that come with the Editor/Assembler Cartridge.

There are three ways to make these changes:

1. Each time you load the program you can make the changes while the program is running.

2. The regular version (uncompressed) can be changed using the "Editor" or with TI-Writer. Be sure that on each line that you change you also change the "checksum" flag to an 8 (it is normally 7.)

3. To change the compressed version you need Disk Fixer or something similar. You actually change the disk information. If you are familiar with the use of Disk Fixer you should have no problem, otherwise it could get very hairy.

MICROS IN ACTION
by Bill Cagle

Perceptions:

Recently, while talking to a computer hobbyist, I remarked about what an excellent disk manager TI had installed in the 99/4A. He seemed interested so I pressed on and started to recount the many blessing's of not having to reserve disk space prior to writing files or saving programs. I also told him that when we read in a program, enlarge it and then re-save the program, TI's "Disk Manager" would fill the old hole, fractionate the remainder and continue storing the remainder in the next available space. This person owns an IBM and when he heard this, he said "What price do you pay for this in time?" I told him that it was so quick that you couldn't tell the difference. He flushed, turned and walked away from me and I knew that while I had won a battle, I had hurt him badly. The previously mentioned encounter points up a danger in our hobby. When we talk to other people in our hobby, we should be aware that other people love their "hobby toys" as much as we love our's.

Each of us perceives our computer in a different light. The very existence of this club stems from the desire to learn and use the full potential of our computer to a higher degree, than we could do otherwise. Most people develop a fondness for their "toys" that far outreachs the utility value to them. This might be in part due to the enormous amount of time and money we must invest in our systems to bring them to their full potential.

I have the perception of my computer as a complete set of tools in a magic package; push a button and you have a hammer, push it again and you have a drill, push it again and you have a saw, push it still once more and you have a sander. This perception is more real than you might think, as your computer can change its identity by merely pushing a button (loading different software). This means that your magic tool needs software to change its identity, so keep up your efforts to write software (as well as purchasing it). To illustrate this, I recently purchased some software called "Multiplan" and when I tried to use it on my expenses I was dismayed to see how slow it was. This shows that a general purpose software is never as good as some written to do a specific job.

For all of you who are interested in Forth, I have found that the TI Doom series is written in Forth. If you have this game, you might want to look at them with the Forth Editor and study the programing style and methods. If you can figure out what the author is doing, you can increase your own programming knowledge.

THREE SLOT EXPANSION KIT
by William M. Lucid

This kit gives the TI user a mini-expansion system that connects directly to the right side of the console, there is NO FAN required on this expansion kit. Source of the kit:

Captain's Wheel
17295 Chippendale
Farmington, MN 55024
(612) 460-6348

Cost of the kit is $35.00, plus $5.00 shipping and handling. Disk power supply option is an additional $10.00. The kit comes with all edge connectors (gold-plated) soldered in place, this saves the kit builder 180 plus delicate soldering connections. All parts needed to complete the kit are included, except an enclosure. Dimmensions are given in the documentation for constructing your own enclosure. Captain's Wheel no longer sells three slot kit enclosures. Balance of the assembly consists of a few resistors, a few capacitors, two intergrated circuits and 30 jumpers, (another some 60 solder points). Wire provided with the kit, is rainbow, muticonductor ribbion cable, that is stripped for jumpers, (insulation on this wire has a very low melting point). Solder traces on the one sided pc board a very fine line and closely spaced.

Documentation is complete with good illustrations. During assembly you proceed by checking off each step on the instructions. Verification check out explained in the documentation, this consists of using a voltmeter to verify voltages called out at points on the pc board. This is IMPORTANT to assure proper voltages are present, so NO DAMAGE results when hardware cards are installed into the system.

Captain's Wheel allows you to return the assembled kit for check out verification and installation of cards at no charge; however, you must send $5.00 to cover the return cost of shipping and handling.

MICRO'S IN ACTION

By Bill Cagle

Millions of light years from Earth, an electronic frontier was constructed across the universe to protect the Star League of Planets from its enemy, the Ko-Dan. Now the Ko-Dan's gigantic mother ship is pictured gliding into position to blast its way through the frontier as numerous smaller spacecraft take of and land on its multilevel deck. Only the skill and daring of a small group of starfighter pilots, lead by an 18-year-old videogame champion shanghaied from Earth, can save the Star League from destruction.

That scene was from the movie "The Last Starfighter" and it only existed in the memory of a Cray X-MP computer. This movie was the handy work of a company named Digital Productions and they have a copyrighted process that generates extremely complex and realistic pictures of places and things that never existed except in a computer memory. The name of this process is Digital Scene Simulator.

The art of computer imaging and animation have been with us for several years and has even been used by the auto industry and others as well. Only the military and a few of the largest scientific laboratories are using as large a computer as Digital Productions. (The Cray X-MP can do about one billion calculations per second. (The TI99/4A can do about six thousand calculations per second by comparison.)

Another measure of the power required to generate high detail scenes, is the number of polygons of information is required for each frame. The Starfighter picture used from 350,000 to 4 million polygons of information for each scene. To give you an idea of how many this is, the Disney picture "Tron" used about 7 thousand polygons of information per frame.

There is some thought geing given for the production of human form and speech. The amount of processing power to recreate the human form with all it's nuances of movement will be ready in about seven to eight years. Just think, you will have movies with The Duke and Clark Gable at our command. Producers, writers and directors are beginning to realize that traditional special effects cannot produce, in a cost effective way, the motion picture sequences they have in their imaginations.

If Sam Goldwin were alive today, he would probably have a heart attack with the excitement this tool will eventually bring to the entertainment industry.

/4A DISK FORMAT

Editor's Note: The following article was copied from "The Paper Peripheral", Newsletter of the Central Texas 99/4A Users Group P.O. Box 3026; Austin, Texas.

We are indebted to Wayne Talbot, who found this information in a file on CompuServe. The information was contributed t CompuServe by Earl Hall, and we are grateful to him for the information. Wayne was looking for this information in order to get good look at the disk for Tunnels of Doom, in order to figure out what is on that disk, to further his effort to write a new TO adventure. (If anyone out there has any information that would help Wayne in this endeavor, please contact him through th Central Texas Users Group. All TOD fans will appreciate it!)

The following is a complete and, to the best of my knowledge, accurate description of the disk directory format and file storag allocation used by the 99/4A computer.

Sector 0 contains the Volume Information Block

Address	Contents
0000-0009	Disk name--up to 10 characters
000A-000B	Total number of sectors on disk ()0168=360,)02D0=720,)05A0=1440)
000C	)09 (# of sectors/trk)
000D-000F	'DSK' ()44534B)
0010	)50=Disk backup protected,)20=not protected
0011	# of tracks per side ()28=40,)23=35)
0012-0013	# of sides/density ()0101=SS/SD,)0201=DS/SD,)0202=DS/DD)
0038-end	Sector allocation bit map. See note below.

Note on)0038-end: This is a sector-by-sector bit map of sector use; 1=sector used, 0=sector available. The first byte is fo sectors 0 through 7, the second for sectors 8 through 15, and so on. Within each byte, the bits is for sectors 0 through 7, th second for sectors 8 through 15, and so on. Within each byte, the bits correspond to the sectors from right to left. Fo example, if byte)0038 contained)CF00 then the first byte equals 1100 1111. This means that sectors 0 through 3 are used sectors 4 and 5 unused and sectors 6 and 7 are used. Information for the 2nd side of a DS/DD disk starts at byte)0065 and end at byte)0091.

Sector 1 contains the Director Link

Each 16-bit word lists the sector number of the File Descriptor Record for an allocated file, in alphabetical order of the fil names. The list is terminated by a word containing)0000; therefore, the maximum number of files per disk is 127 [(256/2)-1]. I the alphabetical order is corrupted (by a system crash during name change, for instance), the binary search method used to locat files will be effected and files may become unavailable.

Sectors 2 to 21 contain the File Descriptor Records

Address	Contents
0000-0009	File name--up to 10 characters
000C	Filetype:)00=DIS/FIX,)01=Program (memory-image),)02=INT/FIX,)80=DIS/VAR,)82=INT/VAR File deletion protection invoked by Disk Manager 2 will be shown by)08 added to the above.
000D	# of (MAXRECSIZE) record/sector.
000E-000F	# of sectors allocated to this file. (Disk Manager 2 will list one more than this number, thereby including thi sector in the sector count.)
0010	For memory-image program files and variable-length data files, this contains the number of bytes used in the las disk sector. This is used to determine end-of-file.
0011	MAXRECSIZE of data file.
0012-0013	file record count, but with the second byte being the high-order byte of the value.
001C-end	Block Link. See note below.

Note on file storage: Files are placed on the disk in first-come/first-served manner. The first file written will start at sectc 0022, and each subsequent file will be placed after it. If the first file is deleted, a newer file will be written in the spac it occupied. If this space isn't big enough, the file will be 'fractured', and the remainder will be placed in the next availabl block of sectors. The block link map keeps track of this fracturing. Each block link is 3 bytes long. The value of the 2r digit of the second byte followed by the 2 digits of the first byte is the address of the first sector of the extent. The valu of the 3rd byte followed by the first digit of the 2nd byte is the number of additional sectors within this extent. Sectors through 21 are reserved for File Descriptor Records and are allocated for file data only if no other available sectors exist. I more than 32 files are stored on a disk, additional File Descriptor Records will be allocated as needed, one sector at a time fr the general available sector pool.

ALGORITHMS.....A NEW FEATURE FOR OUR LIBRARY

By Bill Jones

Most of us purchased a computer (or one was purchased for us), so that we could learn to program it. And one feature of our user's group is the education we can provide for newcomers. Many of the members have taken advantage of classes in BASIC and Extended BASIC and have learned a great deal. But that's only the beginning of learning to program computers.

If I gave you a set of the finest Swiss watchmaker's tools and spent many hours teaching you what each tool could do and how to use it, I still would not have trained you to make a watch. You may develop the skills you need to make a watch, but you wouldn't know HOW to make a watch even though you knew how to use the tools. It illustrates the difference between learning a computer language and learning to program.

When we learned the language, we looked at the programming style that others used and we learned method too. Over the few years that we've been together as a group, we've collected a very complete selection of the programming efforts of our member's and others from around the country and until now, the library has been kept mostly on disks. There will be a new part to our library, a part that teaches method not language. It will be kept, not on disks, but on 4 X 6 file cards and you are about to see how to use it.

Apart from any language, there is an underlying method or formula for getting a job done. We will be assembling "receipe cards" on how to do small, individual programming tasks. The formula is known as an ALGORITHM and it is a precise statement of events necessary to make a process happen. For example, some programs I write need to remove a character from an input line. (say a comma). In extended BASIC, the code to do that job might look like this:

```
100 LINPUT A$
110 A=POS(A$,",",1)
120 IF A THEN 130 ELSE 200
130 IF A=1 THEN 180
140 B$=SEG$(A$,1,A-1)
150 C$=SEG$(A$,A+1,LEN(A$))
160 A$=B$&C$
170 GOTO 110
180 A$=SEG$(A$,2,LEN(A$))
190 GOTO 110
200 PRINT A$
210 END
```

If this is the task as it is programmed, then here is its algorithm:

1. Get a string of characters.
2. Is there at least one comma in the string?
   if not, we're ready to print it and quit.
3. Is a comma in the first position?
   if so, get the rest of the line, leaving the comma behind, and go back to check again.
4. Otherwise get all the string before the comma and all the string after the comma and leave the comma behind.
5. Go back to step 2 and check for another comma.

An algorithm is not specific to any one language, it can be used in most any language and it will do the same task in each. It only gives you the method of solving a task not the program. And an algorithm is not a description of an entire program, just a single focused task within a program. When you solve a problem, write down its algorithm and put it in the library. In time, we will have a solution library that can be used by programmers in BASIC, FORTH, P-code and assembly. To put it in the library, begin by giving it a subject at the top of the card. Mine is the "ONE-CHARACTER REMOVER". Then fill out, step by step, how the process works. You might also include the name of the program it came from if it's in the library too. If it's long, use more than one card, but put the subject on each card. Don't write the program, just its algorithm. Then drop them in the section marked "NEW" so the librarians can keep the card catalog current each month. If you need to know how to do some special task in one of your programs, check the catalog, may be someone has already figured out how. From time to time, the best ones will be published here in the newsletter.

THE COMPUTER OPERATOR

Fleet winging, heart singing, he trots through the door, so happy to be amidst the clatter and roar. Computer and printer, the job is a whole, is heaven to him, provides food for his soul.

No other, his mother, his kids nor his wife, receives such devotion, gives meaning to life. To enter the center is life's greatest joy, providing a pleasure that surely won't cloy.

Pulsating, awaiting his gentle commands, the rig seems to recognize capable hands. Confidant, competant, he flits here and there, getting things ready to go on the air.

Drives counted, tapes mounted already to go, he pauses a moment, his features aglow, serenely, routinely, he pushes the start, and its just about then things fly apart.

One tape, then another, gives out whistles and screams, the printer goes mad, spewing paper in reams, the lights on the console give a fireworks display, and in a momentary panic his feet turn to clay.

His heart begins pounding and surely must burst, as the whole crazy rig acts like something accursed for what seems an eternity but is only a flash, his feet bogged down in a glutinous mass, he's unable to move and unable to speak as the computer goes dead with a pitiful squeak.

Head ringing, eyes stinging, he goes for the switch, knocking down the power on his beautiful witch, benumbed feeling stunned, not yet able to guess the calamitous cause of this horrible mess.

Traumatic, dramatic, the shock is profound for fully a minute, he utters no sound, then walking, hands shaking, his temper gives way and the curses start flying, I'm sorry to say.

He curses the mainframe, the tape drives as well, he curses the card reader, consigns it to hell, he curses the printer, he curses the punch, he curses the console, and then on a hunch, he curses the program and while still untiring, he curses the diodes, transistors and wiring, he curses the present, he curses the future, he curses the day he saw a computer. At last, quite exhausted, he falls to the floor unable to utter one little curse more.

Bedeviled, disheveled, his face chalky white, eyes bloodshot, tongue lolling, a pitiful sight. It's over, all over, the battle is done twixt man and machine, the computer has won.

Muttering, stuttering, completely insane, he mumbles this warning again and again, IDIOTS, IDIOTS, CAN'T ANYONE SEE, THAT ANYTIME NOW YOU MAY END UP LIKE ME.....ANONYMOUS

****

This poem was copied from the 99'ers Users Group Association June, 1984 Newsletter. The poem was originally printed in the February, 1984 issue of the "Melbourne Times", Newsletter of the Melbourne, Australia TI-994A Users Group and reprinted in the United States by the Cin-Day Users Group in their May, 1984 Newsletter.

TI TERMINAL EMULATOR II ALTERNATIVES

by William M. Lucid

This article will cover two terminal emulator programs for the TI 99/4A system. I use the word system because these programs require: 1. RS232, 2. 32K memory expansion, 3. Disk system, and 4. Editor/Assembler module or CorComp disk controller. Programs covered in this article are TE-1200, and PTERM-99.

The purpose of a terminal emulator is to enable transfer of data between computers, allowing sending and receiving systems to "talk" in a recognized method of handling data, even when sending computer is different than receiving computer. The terminal emulator program is used to set communcation parameters for using a MODEM with a RS232 interface. Some parameters encountered in terminal emulators are BAUD rate, RS232 (1-4), Parity, Stop bits, Echo, and Data bits. Another use of a terminal emulator program is to software interface radio amateur equipment. I have been able to use these programs to interface with KANTRONICS UTU for receiving RTTY as well as other modes of communications without having to layout additional money for KANTRONICS, HAMSOFT software for the TI 99 4/A.

TE-1200 by E. Earle Thompson was the first alternative for the TI TERMINAL EMULATOR II, that had capabilities of "auto-logging". Auto-logging allows you to use 32K memory expansion to "hold" incoming data. When the 12.5 k of ram buffer is full, program automatically dumps the 12.5 k bytes, (approximately 48 sectors) of data to a disk file that has been set-up pressing "Control" and "4" keys when parameters were inputed. The outputted file is a display, variable 128. Outputting of the file is done while ON-LINE, this increases your on-line charges, possibly long distance such as SOURCE and COMPUSERVE. TE-1200 communcations parameters can be re-entered anytime by pressing "control" and "1" keys. Also to recall an Auto-logged file to use with a editor program such as TI-WRITER or EDITOR/ASSEMBLER the display, variable 128 file must be conveted to a display, variable 80 file. TE-1200 allows the user to select baud rates 0f 110 upto 9600, baud rate depentent on modem baud rate, capability of RS232, and some other variables such as line noise, tolerance for error. The last item is important in cases where no "missing" data can be tolerated, such as transmitting an assembly file. The faster the baud rate, the greater the risk of garbled or lost data when using non-deciated telephone lines. TE-1200 supports same file transfers as TI TERMINAL EMULATOR II, this feature allow you to use downloading feature of T.I.B.B.S. bulletin boards which require either TI TERMINAL EMULATOR II or TE-1200.

Another excellent terminal emulator program is P-TERM-99 by C. Richard Bryant. This program lives up to its claim of, "the ultimate terminal program. 300/1200 baud, 24K download buffer, 20K upload buffer, 256 color combinations and many more options." One of those other options is the ability to toggle your printer on and off by pressing "control" and "1" keys, with this feature you can simutaneously display and print at the same time or toggle printer off to only display data. Default values are for 300 baud, RS232 port 1, even parity, 1 stop bit, and 7 bits data. Default values can be selected at first prompt by pressing enter twice. PTERM-99 will load from Extended Basic, Editor/Assembler or Mini-Memory modules. This program allows for resetting communication parameters at any time by pressing "control" and "7" keys. Pressing "function" and "7" allows you to select foreground and background colors of your perference, choices are presented on screen to aid in making selection. Outputting of the download buffer when full will occupy about 110 sectors. When the download buffer is within 1K of being full the screen will turn red, this feature works very well. When screen turns red you must dump the download buffer or the download buffer will be over written. Download buffer file output, if a disk file is in the display variable 80 format. Upload text files need to be in display variable 80 format. I have found this works best by removing control characters. T.I.B.B.S. bulletin boards will not "recognize" PTERM-99 the first time you attempt to log on. Another disadvantage is true TE II "File transfer protocal" is not a feature of this well planned, easy to use, and ecomonical program. PTERM-99 is a reliable, dependable, and proven program, well worth the $17.50!

(TE-1200 is published by Softmail, PO Box 745, Rockwall, TX 75087. TE-1200 is listed in UNISOURCE catalog. UNISOURCE has a toll free telephone number 1-800-858-4580, there address is Box 64240, Lubbock, TX 79464 the last price update I received shows TE-1200 costing $39.95. PTERM-99 is being sold by TEXAMENTS, 53 Center Street, Patchogue, NY 11772, program costs $17.50 and includes shipping.)

TERMINAL EMULATOR II ALTERNATIVES by William M. Lucid

This article discusses Joe Freeman's TE3C, Version 3.1 program. TE3C is public domain, a copy of this program along with source code is available in the User Group Library. TE3C disk in the library includes source code, true lowwer case letters, two configurartion files, as well as documentation on disk in display, variable 80 format.

Features of TE3C include the following: 1. Choice of 40 column screen or 80 column screen. 2. 24 K byte download caputure buffer (this is about 100 sectors). 3. 20 K byte upload buffer (this allows composing text off line using program capable of creating display, variable 80 files, i.e. TI-WRITER editor or EDITOR/ASSEMBER's editor). 4. Program has an auto-dial feature, which is also documented on the library disk. 5. Loads TI-WRITER's lowwer case characters. 6. On-screen help is available while program is running by pressing the function key and 7 at the same time. 7. Supports ADM3A protocal for UNIX systems or SERIES 1's running YALEIUP. 8. Upload and download files are done in the display, variable 80 format (program does not suport TE II file upload and download protocal). 9. Program supports eleven function keys and eight control keys. 10. Baud rates supportted are 300 and 1200.

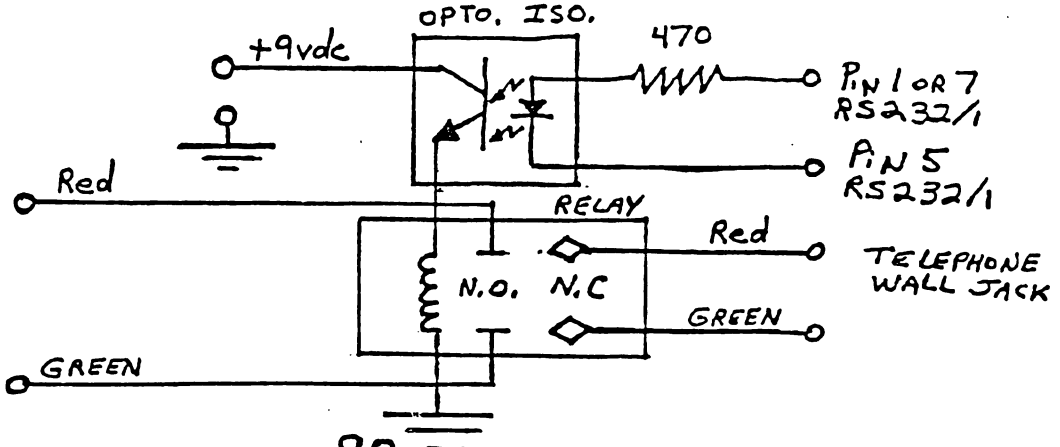
Loading program for use requires a disk system, 32 K memory expansion and EDITOR/ASSEMBLER module or CORCOMP's disk controller. Using EDITOR/ASSEMBLER select option 3, load and run option, filename to enter is TE3C. Program will run automatically. When prompted for configure filename, enter UFENG or CONFIG. An alternative to enter a configuation filename is to press enter key and select parameter as prompted. Documentation tells how to create your own configuration file.

Joe Freeman is to be commended for making TE3C, Version 3.1 public domain. Source coding for TE3C is well documented.

I have had my disk controller fail and have had no alternative except TI's TERMINAL EMULATOR II, it really made me feel handicapped after have used one of the other alternative terminal emulators; however, with TE II, you do not need a disk system. This comment is included here for those that might consider selling or trading TERMINAL EMULATOR II after obtaining an alternative terminal emulator.

Here is Joe Freeman's schematic for implementing auto-dial function. There are no specific part numbers listed, use of this schematic is at ONE's OWN RISK!

TE3C
AUTO-DIALER



TERMINAL EMULATOR II ALTERNATIVE by William M. Lucid

This article discusses Paul Charlton's FASTERM version 1.16. This is a FREEWARE program the asking donation is $15.00. Paul Charlton's address is 1110 Pinehurst Ct, Charlottesville, Virginia 22901.

Version 1.16 consists of two "program image" files (FASTERM and FASTERN), DEFAULT (a program to setup a default parameter file), and COMMSUM (a text file with commands summarized). FASTERM loads from EDITOR/ASSEMBLER, option 5, TI-WRITER, option 3. TI-EXTENDED basic program VOLKS12 will also load FASTERM, a word of caution using VOLKS12 to load FASTERM, some bulletin boards do not allow enough time to load program after connecting and will hang up.

Creating a default file is accomplished by using console basic with the EDITOR/ASSEMBLER inserted, and running the basic program DEFAULT. Follow the directions on screen for setting up default file. When you load FASTERM, you will be given a prompt for default filename. Enter the default filename you used when you created default file using DEFAULT program.

File transfers with FASTERM is done with one of three protocals, which are 1. ASCII text files, 2. Terminal Emulator II protocal, and 3. XMODEM protocal.

Receiving ASCII files is done by creating a logging file by pressing (FCTN), (B) at the same time. This will close any previous logging file and ask for a new logging filename. Pressing (FCTN), (.) will stop data from going to the logging file. Press (FCTN), (B) again, this will write data to logging file and close logging file. A proper setup of FASTERM using the DEFAULT program will keep you from losing data. Data in the memory buffer can be cleared at anytime by pressing (FCTN), (Y) at same time.

Sending ASCII files is done by pressing (FCTN), (N) at the same time and giving filename of the file you want to send. Press (FCTN), (,) at the same time to start sending the file. You be will asked if you want to send the file line-by-line. If file is send line-by-line you must press the (SPACEBAR) everytime you send another line. Press (FCTN), (4) at the same time to stop sending from the file. If the file is not sent line-by-line, then the file is sent as fast as the other system can handle it (XOFF/XON handshaking). When file is finished press (FCTN), (4). ASCII files sent by either of the methods may be DISPLAY,FIXED 80 or DISPLAY, VARIABLE 80. Linefeeds may be added after every carriage return by pressing (FCTN), (J). Pressing (FCTN), (J) again will disable linefeeds from being added to carriage return.

Uploading with XMODEM is done by pressing (FCTN), (N) give the filename you want to send, if you do not do this FASTERM will attemp to send DSK1.SENDFILE file. Tell the other system you want to upload. When the other system is ready for upload, press (FCTN), (SHIFT), (X) at the same time, enter "S" to send file. If asked, if you want CRC, answer that you do. Press (FCTN), (4) at anytime to abort file transfer.

Here are some "special functions" of FASTERM. Pressing (FCTN), (K) at the same time will enable an elapsed timer (timer is stopped with any disk operation also, the elapsed timer is disabled at baud rates over 1200). A "short catalog" can be obtained by pressing (FCTN), (9), this "short catalog" only gives filenames. FASTERM allows you to suspend incoming data by pressing (FCTN), (O). Pressing the (SPACEBAR) will scroll back through the memory buffer. While in this mode it is possible to get a screen dump by pressing (FCTN), (SHIFT), (P). Press (FCTN), (O) to return to incoming data, several pages of data may scroll off the screen, as the buffer catches up with the screen display.

FASTERM on HUGbbs produces reverse video, to correct this, press (FCTN), (SHIFT), (D) all at the same time, this will put you into half duplex. Now press (CTRL), (N) at the same time to produce normal video. Press (FCTN), (SHIFT), (D) again to return to full duplex.

Sending files using TERMINAL EMULATOR II protocal requires pressing three keys at the same time, press (FCTN), (SHIFT), (T). Press (FCTN), (N) give the filename you want to send. Tell the other system that you want to upload. Pressing (FCTN), (,) to send the file. You may abort the transfer anytime by pressing (FCTN), (4).

Receiving files using TERMINAL EMULATOR II protocal requires pressing three keys at the same time, press (FCTN), (SHIFT), (T). Press (FCTN), (N) and give filename you want data to go into on your system. Tell the other system you want to download. Press (FCTN), (4) to abort transfer at anytime.

REMEMBER the default, that is if you do not enter a filename by using (FCTN), (N) FASTERM will write to a file on your system with the filename DSK1.SENDFILE. There is a risk of over writing an earlier file that was downloaded as the default name DSK1.SENDFILE be forewarned!

Downloading with XMODEM is done by pressing (FCTN), (N) give filename you want data to go into on your system. Enter the command to tell the other system you want to download. Wait until the other system tells you that the download has started. Press (FCTN), (SHIFT), (X) to select XMODEM file transfer protocal, enter "R" to receive file. If asked, if you want CRC, answer that you do. You may press (FCTN), (4) at anytime to abort transfer.

TI STATISTICS

By Roger B. Crampton

From Compute!, December, 1983; page 196.

In many professions there is a need to analyze something statistically. Engineers, medical researchers, psychologists, and social scientists often must generalize from data samples and make predictions concerning the probability of events. Not many years ago this data analysis was a tedious and expensive task, using calculators and many clerical assistants to perform manual computations.

In addition, because the mathematics of statistics appear so formidable, professionals often hesitate to try to explain the implications of their data.

Texas Instruments has helped remove some of this anxiety with its Statistics Command Module, a series of programs that perform dozens of the most commonly needed statistical techniques.

The module leads the researcher through the procedures of statistical analysis in a friendly and efficient way. The only hardware requirements for running complicated statistics programs are the TI-99/4 or 4A console, a monitor, and the module. While not essential, a printer and a cassette or disk drive will eliminate having to reenter the data set and file structure if you want a second look at your findings.

Learn The Basics First

Before plugging in the module, it is important that you thoroughly read the 48-page instruction manual at least twice. The time spent will be rewarded with a clear understanding of the module's capabilities and a basic understanding of statistics itself.

When the module is inserted into the console, a title screen is displayed, followed in a few seconds by the first of several menus (see Figure 1).

Figure 1: Program Options

```
Press
   1 TO CREATE A NEW FILE
   2 LOAD AN EXISTING FILE
   3 USE SIGNIFICANCE LEVEL CALCULATOR
   4 QUIT
```

Typing 1 allows you to set up your file structure. You name each variable, determine its type (alphanumeric, integer, decimal, or scientific notation), and enter the maximum number of digits of each variable. The number of variables allowed depends on the width of each entry and the number of observations.

Conversely, the number of observations that you will be able to enter depends on the number and specifications of the variables you have selected. It is important to carefully define the parameters of the problem so that you will be able to use all of your observations without getting a MEMORY FULL message.

Another reason for care when you specify the initial file structure is that there is no provisions for editing file specifications once they have been set up.

When the file structure has been established, the next menu will be displayed (see Figure 2).

Figure 2: Basic File Structure

```
                Main Index
PRESS
   1 TO SEE FILE DEFINITION
   2 ENTER OBSERVATIONS
   3 CHANGE OBSERVATIONS
   4 ANALYZE DATA FILE
   5 SAVE DATA FILE
   6 QUIT
```

At any time, you can return to the main index, select option 1 and review the specifications of the file. But remember, you don't have a chance to change anything, unless you're willing to reenter the entire file definition.

Entering Data

When you are certain that your file is arranged exactly as you want it, it's time to select option 2 and begin entering data. The module will prompt you with the names of the variables as each is typed in.

Data entry is slow. A fast typist must slow down to about half speed because the module will not accept entries at usual typing speed. The first variable value will be accepted, but the initial digit of the second or succeeding variables often gets lost. An entry of 84 becomes 4, an entry of 1.3794 will become .3794.

After all your data has been entered, you can verify its accuracy by selecting option 3 from the menu and single-stepping through your data set, making any changes that are necessary. There is no provision for LISTing your data to printer to check each observation for accuracy. This would be desirable, especially to see that decimal data is properly entered.

Analyzing The Data

At last the preliminaries are completed, and you're ready to get down to the real purpose of the program: looking at your data from a statistical point of view. By pressing option 4 of the main index you are given a new menu (see Figure 3).

Figure 3: Analysis Options

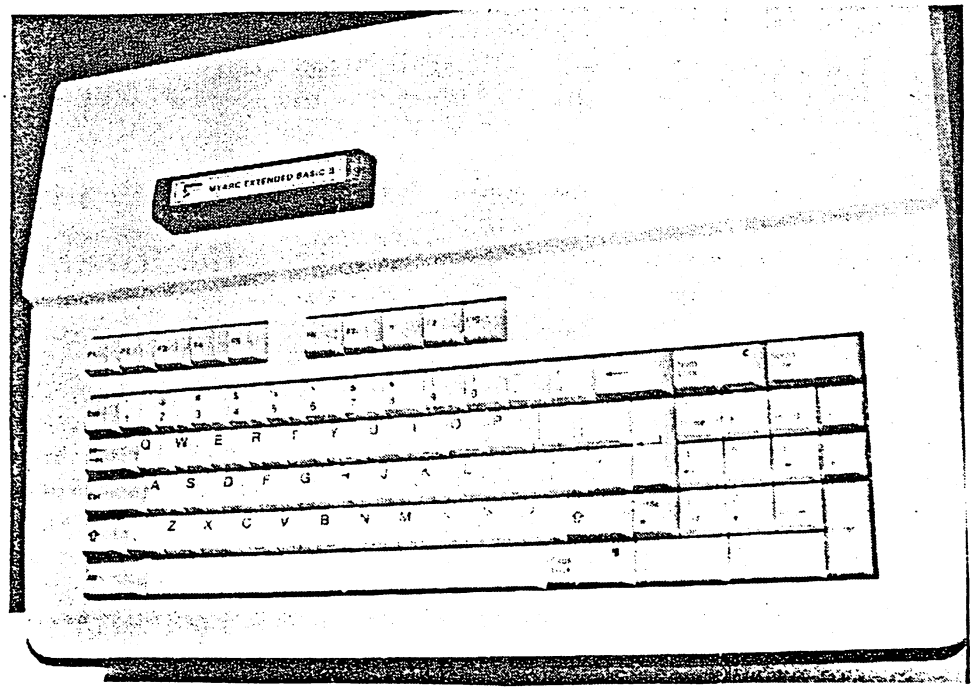
```
    ANALYZE DATA FILE
PRESS
   1 FOR DESCRIPTIVE STATISTICS
   2 CORRELATION
   3 LINEAR REGRESSION ANALYSIS
   4 INFERENTIAL STATISTICS
   5 TO EXIT THIS SECTION
```

Each of the four options is thoroughly described in the user's manual. Few researchers will need all of the procedures available. In fact, it may be best to learn to use one technique at a time. The enormous amount of information from the analysis of even a simple data set can be overwhelming.

Although the Statistics Command Module can provide volumes of information about a data set, it does have limitations. Evaluation of a great deal of information can be hampered by memory problems if the module is used without memory expansion. In addition, no provision exists in the program to screen out data entry errors by specifying acceptable ranges for each variable.

For nonprogrammers who need a means of analyzing fairly simple data sets, however, the module can be a useful tool. And for anyone seeking a relatively painless introduction to statistics, it is superb.

MYARC'S NEW COMPUTER!



TI-99/8 PC?
CYPRESS PC!
TI-99/8 PC?

Today, Charles Ehninger of Ehninger Associates, Inc., and the software programmer responsible for the FUTURA business packages developed for TI-99/4a confirmed reports that his company has been retained by Myarc, Inc., to develop software for the TI-99/8 personal computer. While there remains a bit of confusion over what the market name will be, reliable sources have confirmed the internal name to be CYPRESS. The internally named Cypress is expected to debut sometime late this summer.

99/4A COMPATIBILITY

For those of you like myself, have "kept the faith" and expanded your system with such peripherals as the PE BOX, RS232 CARD, 32K MEM EXP. CARD, DISK DRIVE(S), DISK DRIVE CONTROLLER CARD, MODEM, and recently, the 128K Card, you can relax!! The Cypress personal computer will be 100% compatible with the PE Box and its host of cards and software with more lace on the way.

A NEW LOOK!

The ? computer, internally called the Cypress, will have a new look! "Its keyboard is styled like that of the IBM PC," Mr. Ehninger said. This desirable modification will certainly be appreciated. With the new styled keyboard, comes a standard 80 column text display for you monitor users. The price tag for this gem is speculated to be around $400.00. This feature alone should shelf any plans to purchase an 80 column card for the 99/4a at least until this summer.

Move over PC! Watch out Apple! Byte the dust Commodore!!!!!!!

With the caliber of committment to quality Ehninger associates has evidenced by the Futura intergrated business package for small businesses, we can anticipate that the consumer will be the real beneficiary of any contractural arrangement made between Ehninger Associates and Myarc.

CYPRESS PC?
TI-99/8 PC?
CYPRESS PC?

The price tag for the Cypress is speculated to be in the neighborhood of $400.00.

Attempts to get more detailed information on the internally named Cypress were fuddled as Myarc's president, Lou Philips, is taking no chances of releasing any information about the Cypress prematurely. However, reliable sources have confirmed Myarc is presently searching for venture capital to mass produce the already designed Cypress. Further, the Cypress will be "twice as powerful as the Texas Instruments' TI-99/8" which TI had designed and produced operational prototypes but never marketed. Another unnamed source said, "Myarc does not wish to degrade the performance of their computer by naming it the TI-99/8."

Once the venture capital required is obtained, you can look forward to a magazine dedicated to the performance of this computer that will be done through a publisher "larger than McGraw Hill", my source said.

With new memberships on the decline in many User Groups, It will be interesting to see how User Groups, their leadership, and the members will take advantage of this opportunity and prepare for the new wave of computer users. Someone wisely noted, "Opportunity seldom knocks twice." The Cypress may--just may be the second knock a lot of User Groups need!

Whatever the outcome, we have you in our prayers Myarc, for the birth of the Cypress, or whatever the market name will be, will come at a time when it is sorely needed for many User Groups.

Bill Barnes
GO99ER

This article comes from the May, 1985 issue of The 99'er, Newsletter of the Greater Orlando Users Group, of Maitland, FL.

TI-ARTIST REVIEWED

Console, monitor or TV, disk system, expansion memory, one of the following: (Extended Basic, Mini-Memory, Editor/Assembler, TI-Writer, Corcomp or Myarc disk controller card (in short an assembly object code loader)), joystick (optional), Super Sketch (optional), mouse (optional), track ball (optional).

PERFORMANCE

TI-ARTIST has to be the best all around graphics program on the market. Even though GRAPHX, Master Painter 99, Draw and Plot, and Super Sketch each offer certain unique features, TI-Artist offers the best features in a single package.

It is an extremely easy program to use and is completely menu driven from a picture title screen. Simply move your cursor to the picture of the desired action or press the corresponding letter on the keyboard and presto you are in that mode. Some of the unique features are: Continous Lines from point-to-point, Rays, Fills with user selected patterns, Frame, Box, Circle, Disc, perfectly horizontal or vertical lines, alphanumeric lettering in 81 possiable sizes determined by user, Store or load from disk, zoom the picture size 4X for detail drawing, Mirror your drawing in quadrants, Input your drawings using keyboard, joystick, Super Sketch, trackball, or a mouse, Multiple brush sizes/styles, index pictures previously saved on a disk, optional density, magnification, and line spacing printouts on EPSON compatiable (most printers), Okidata, or Seikosha GP-100 printers. One option which I particularly like is the ability to load drawings previously drawn with Draw A Bit, Draw A Bit II, Draw and Plot, and GRAPHX and convert them to drawings which may be used with TI-Artist.

There are several features which could be easily added which I feel would enhance the program. First, on printing out the picture the user should be able to permanently re-define his printer specifications instead of entering them each time, abort the print out, and define shades of print densities corresponding to screen colors. Second, a provision for pre-drawn ICONS for the inclusion in drawings would have been nice.

EASE OF USE

From the moment I loaded the program I was able to begin drawing with minimal reference to the instructions. Even the inexperienced user should be able to quickly master all facets of the program. This program is also well suited for children as an introduction to drawing on the computer.

VALUE

TI-Artist at $19.95 has to be the best buy on the market. When one considers the effort required to write this tremedous program plus the cost of materials, postage, and handling it would be impossiable to find a better deal.

FINAL GRADE

TI-Artist certainly deserves the top grade. I found the program a joy to use and I am certain it will not become another useless program sitting on the shelf. The author of the program is Chris Faherty of INSCEBOT INC. Software.

(Comment: This review by Mack McCormick was printed in the Penn Ohio 9/4A Home Computer Users Group Newsletter, May 1986. Version reviewed was not stated, The TI-Artist version I have is 2.0. Additional software packages for TI-ARTIST are: Artist Companion #1, Artist Companion #2, Display Master, and Artist Extras. Use of Super Sketch pad and use of a mouse require special DSR routine on the Artist Extras disk. - Wm. M. Lucid)

!! NEW GRAPHICS CAPABILITIES AVAILABLE !!

Thanks to two Compuserve/TI Forum users, Travis Watford and Paul Gray, the TI community is now able to use RLE (Run Length Encoded) Graphics. This means that many high resolution pictures created for other computers are now available to 99/4A users. The pictures on this page are examples of RLE images available in various "libraries" on Compuserve. They were downloaded just like any other file and then run through a program written by Travis Watford to make them "usable" on the 99/4A.

"Run Length Encoding" is a standardized method of storing graphic images. Other computers have been able to use this type of files for quite some time, but until recently TI users had been left out. The files are called "Run Length Encoded" because of the manner in which the graphic images are converted into a "standardized code". A "RLE" graphic screen has 256 columns and 192 rows (luckily, that is just the "dimensions" of a 99/4A bit-map mode screen). To convert a high-res image into an RLE file the encoding program begins at the upper left corner of the screen and determines if the first pixel is "on" or "off". For the sake of this "explanation", let's assume the first pixel is "on". Then, the program would count across the top row of the screen until it reached the first pixel that was "off", and then it would begin counting how many pixels in the row were "off" until it came to the next pixel that was "on". This sequence is repeated until the entire image has been "encoded". (Each of the "numbers" obtained while the encoding takes place has 32 added to it to convert it into the hexidecimal code for a "printable" ASCII character, and the final RLE file is simply a collection of these ASCII characters.) To reconstruct the image, the "decoding" program follows the reverse procedure. It begins at the upper left corner of the screen and then row after row turns "on" or "off" groups of pixels depending on the values in the RLE file.

The program that allows 99/4A users to use RLE pictures is called "Max-RLE", and has been placed into the public domain by the author. With it, we can load RLE files that have been previously downloaded (as either DV/80, "ASCII" files or as DF/128, "Xmodem" files) into the computer and then "manipulate" them in several ways. We can save them back to disk in a form that can be used by "Graphx" or "TI Artist", which allows us to "color" or change the pictures however we desire. OR we can print the image out on a printer.

If all that could be done was "decode" RLE files, this would be a great program, but "Max-RLE" also allows the user to "encode" pictures drawn with either "Graphx" or "TI Artist" into RLE files. This means that pictures drawn by 99/4A users can be seen and used by owners of other kinds of computers that support RLE graphics. (We may be orphans, but we're still an "active" part of the home computer community!!!)

Currently there are over 300 such pictures in various libraries on Compuserve. But those of you who don't subscribe to that service may get a chance to "experience" RLE graphics anyway. Now that we have the "Max-RLE" program, we can keep this kind of pictures on smaller bulletin boards also. Before long we may be sending "art" with our text all over the country. (Now, if someone would just come up with a "cheap" way for 99/4A users to "digitize" photographs.....)

I ran across an article that compares MS-DOS to 4A DOS, using a CorComp Disk Controller. The comparison shows that the TI with all of its drawbacks has a pretty fair DOS. I know from my own experience that it is much more stable than the CoCo II operating system. The comparison goes like this:

System	#-tracks	#-sec/tk	#-Bytes/sec.
CC	40	18	256
MS-DOS	40	9	512

#-sides	Total Storage
2	368,640
2	368,640

The MS uses 512 bytes/sector compared to 256 for TI. For a closer look at just what this means, consider you save a program of 700 bytes on both systems.

System	sec/used/file	bytes/available
CC	3	768 (3x256)
MS-DOS	2	1024 (2x512)

bytes/used/file	bytes wasted
700	68
700	324

So now, what do you think of our little orphan? This also explains why the IBM and compatibles can not read 4A disks and vice versa.

PRBASE VERSION 2.0 by Joe Nuvolini

(Front Ranger Aug. 86 : Front Range 99er Computer Club Colorado Springs, P.O.Box 9572,CO 80932)

(Keyed In by Roger Quintanilla 10-86)

Version 2.0 of Willaim Warren's PRBASE is finally ready. The new version supports double sided operations and will hold 710 records. Since the major changes are in the Create portion of the program, let's talk about that first.

In my original review I was a bit critical of this portion of the program, but Bill has made the use of it much simpler. Upon selection of this area you encouter a menu with eight selections.

Option one is Select Data Drive. With this option you can make drive 1-5 your data drive.

Option two allows you to format a data disk as either single or double sided.

Option three is your Design Data Screen. The procedures to set up this screen are about the same. When done, you can print out the data screen which is helpful when it comes time to design your labels and reports. At the end of this option you input the data disk name and you output device, for me PIO/1. make sure that you enter FCTN 3 [ERASE] before entering your printer device name or enter spaces after your printer name to the end of the printer device input field or you will encounter an output device error when accessing you printer.

The next option is design Tabular Reports. Here's where the improvements are REALLY noticeable. You can design five reports and the good news is that if one doesn't come out right on the first try you can go back and fix it witout redoing the entire report. After you select the report number you want to design, you can select 80 or 132 column format, the number of lines in the report, and the report title. Next you enter in ASCII the control codes you want, up to 6. I used 15 27 78 10 to get the condensed print for my 132 column report and skip over perfs so I can print the whole report at once and not have to print sections of it, so it doesn't print over the perfs. After the control codes you reach a screen titled Report Format Design. Here you can see the location and size of each field in the data screen. At the bottom you enter the Log Device, again PIO/1 for me, and can print this screen if you wish. The next screen, titled Design Tabular Report, is where you actually design the report. The first 16 fields are automatically here when you arrive and you must move them around, delete some and add others. You must enter the screen location, number of characters, report line, and column position for each item in the report. When done you can also print this screen. When you are satisfied with the layout you press FCTN 6, [PROCEED], and the data fields are initialized. When this is done you will see the number of lines used and the number of lines desired. Press enter and you reach a screen titled Enter Column Header. It shows the starting position of each field in the report with a caret [^] so you know where to place your headings. A caution here, as you only have 84 characters [12 sets of 7] available. Use abbreviations when necessary, to conserve characters. When you finish labeling your headings press enter and your report format is saved on your data disk. To change it just go through the process again and make your desired changes on the design screen. NOTE- if you change any of the heading titles you must replace ALL the carets which reappear with the letter that belongs there. So much for option 4.

Number 5 allows you to design your mailing labels. It is quite similar to the report option but shorter. Here you choose the number of lines and set the locations for the data. Then the data fields are initialized and the format is saved. Again you can go back and change it later if you like.

Option 6 is used to set printer control codes. You can set five sets of control codes for your printer up to six ASCII characters long. You select a number between 1 and 5, enter the text for the code, e.g., condensed print, and then enter the code, for the 15. You then have the option of saving it to disk. These are accessed through the C command in the data management part of the program.

Option 7 is Setup Options. Here you can set the data disk name, printer name, single or double sided disk, and set the left and right tabs for two-across labels. A zero for the right tab will print single labels. The same caution about entering your printer name applies here that was mentioned in option 3.

Finally, option 8 is exit and that should need no further explanation.

There are only a few changes on the management side of the program. The first is that if you can't remember the name of your data disk, you can enter DSKx?, x being the drive your data disk is in, and it will read the data regardless of the disk name. If you enter N from the menu for record number, the highest record number will apear as a default value. When Editing you no longer have to keep pressing enter to get the cursor through the entire screen. When you are done editing just enter FCTN 6 [PROCEED], and the edited record will be saved to disk. You now can print reports in 80 or 132 columns. Mailing labels can be printed one across or two across, and C on the menu now lets you select the control codes for your printer. So, if you want to print a report in condensed print and it normally does not, you can do it by selecting condensed print here. You have the five selections you set up earlier or you can enter one manually here.

There is also a selective search now, which you can select by entering Y at the prompt in the Options area where you can index by name, for example, and a string in another field, let's say 80917 under a zip code field. After indexing and sorting you can print out alphabetically all the people who have zip code 80917. The rest of the management portion is about the same as in the original version.

I might mention that there is now a set of PRBASE UTILITIES. They were written by John A. Johnson, and are excellent. They contain a menu with the following options: Copy Database Header (sectors 1-9), Copy a Group of Records, Copy a Single Record, Search and Select Records, Sort and Rewrite to Copy, Configure Drives, and Exit Program. Most of these are self-explanatory and all are covered in the DOC's that come with the program. These utilities and their DOCS are now included on the disk with PRBASE.

Bill has really outdone himself this time. I would have to upgrade my report card mark for ease of operation to an A, and my final grade to A+. Bill has sent out postcards announcing the availability of version 2.0 to all registered owners. You must send him 720 sectors of initialized disk space, and a stamped return mailer. Bill will supply the mailer and postage for $1.00 or he will supply everything for $5.00. If you are not a registered owner then send the disks and $10.00. Bill's address is 2373 Ironton St, Surora, CO 80010. When you recieve his card announcing the availability of the new update, be the first on you block to get one. It's the BEST!

EXTENDED BASIC QUIRKS: by Ed York; Cin-Day 99'er User Group of Cincinatti, Ohio

Many of you who have Extended BASIC may not know that you can load a program from cassette and then execute or RUN it automatically by typing in "RUN CS1". This is known as an undocumented command. Now, doesn't that kill two birds with one stone? Also many of you have been trying to make the computer speak phrases using Extended BASIC and the Speech Synthesizer but are disappointed with the results! Well again it is not documented that you need the # symbol before and after phrases that are listed (see the List Of Speech Words located in Appendix L of the Extended BASIC Manual). Phrases such as "WHAT WAS THAT", "READY TO START" and "THAT IS RIGHT" must be entered as CALL SAY("#WHAT WAS THAT#"), CALL SAY("#READY TO START#") and CALL SAY("#THAT IS RIGHT#") in order to hear the computer speak them correctly.

WORDSEARCH # 6

This WORDSEARCH contains the names of 20 99/4A games.

WORD LIST

PARSEC	SUPERDEMONATTACK
HOPPER	BURGERTIME
YAHTZEE	VIDEOCHESS
MUNCHMAN	CHISOLMTRAIL
SLYMOIDS	TOMBSTONECITY
STARTREK	TUNNELSOFDOOM
SNEGGITT	ADVENTURE
MOONMINE	ALPINER
CARWARS	HUSTLE
OTHELLO	TIINVADERS

```
MYYUBGVMKCSLOHYGIYAJNAYAUMEXIDNQ
GMMXIEYIJBXZJPSTSTYMYBKAMFAGQOJW
LOYORHXBONSOTHAZNIKHNAVPNUPZCDIC
GOKPFDSLYMOIDSNRIIOJMKXKMZHGRFUF
YNVCVUOGWFBOCVGKWNGFQABSOSLRJHBW
JMUPLKWQGUOIRLUEDVZZOSGTWXDNXRBJ
JIMAWSRBRHIUGVPKAAFCGWZNHZNWVJAE
NNMDVGQGQJTATPGKHDKHMOJUWUFNGUMB
PEDUOMERWMDIOQBGSEDIVUCJGCSROBOV
YTVNTRGCNVJBYEXUHROSMTNDFDNTHGQF
LMIDTGIUEGJAUPPTGSSOCZNCYYIQLAWO
TIHIJDQNOWHEDEGXQTPLDASVHTWANEZP
USMAHITUATRURKYSKGSMOYRQSMSHBCFY
NERFWULOZTXDDCMIJHHTFJSWWTAOTETO
NJEURPAEGYETHBJSSNGRESXNAQMNWUKI
EEFEDUEYXMQYLFMNDZYAHLERERLQMFGA
LLKRAACGOBTVXDMBVWQIMOTLQGSBATJK
SKNFOKONKHICNBUKEBPLGRPPROGALHUC
OVUBYEABUQDIOQGZADPXEFYPKEPIWTEX
FOWDDTUSSEHCOEDIVCKKTQFEEOEUTAAG
DAVWTAKASLFPKEEQCTXPQVWFERWNYTKQ
OFBARTHUWDHGOZPYABTYNSKBOHIZKRZP
ONCOLLEHTOPARSECTOMBSTONECITYKYD
MKYXHTQLAFQPBFYRTOZADLTRAZGNERYR
FPIYENALPINERTVIGCMXZAFIMZGLWFYJ
```

AUXILLARY POWER SUPPLY FOR DISK DRIVES by William M. Lucid

This is a presentation of a power supply, that I have been using over a year with no problem, and it is still on-line. Before I begin, this presentation deals with modifications to the TEXAS INSTRUMENTS EXPANSION SYSTEM, also known as the P-Box, and could void warranty and/or exchange rights.

Developing an aux. power supply I had the following goals: (1.) Provide power for up to three additional drives, (2.) Eliminate turning on an additional power switch, and (3.) Mount the aux. power supply inside the P-Box.

Fairchild voltage regulators 78H05 and 78H12 were selected because they are capable of delivering five amps and offer excellent ripple rejection. Before begining my project I studied the inside, physical layout of the P-Box side that contains the main P-Box power supply. Anything mounted in this area needs to avoid physical contact with: (1.) P-Box power supply printed circuit board, (2.) A.C. line filter, (3.) P-Box transformer, (4.) Ventalation fan, and (5.) P-Box power switch and its mounting bracket. Anything mounted inside the P-Box must allow the reassembly of the P-Box "chassis cover" to the P-Box chassis.

Using an A.C. voltmeter I located a "switched source" of 110 A.C. volts that allows the aux. power supply to be off when the P-Box is off, and on when the P-Box is on. This source for me turned out to be the top two "quick connectors" on back of the P-Box power switch.

Mounting of the aux. power supply was accomplished by designing a printed circuit board. Voltage regulators, heatsinks, full-wave bridge rectifier, filtering capacitors and a few other components are mounted on the printed circuit board. Space inside the P-Box is very limited! I found this to be a limiting factor as to the physical size of the aux. power supply transformer. The transformer I used was rated at eighteen volts, three amps. Aux. power supply printed circuit board end with the heat sinks is mounted about one inch from the ventilation fan and is mounted flush against the upper left side of the "chassis cover" using one inch spacers between aux. printed circuit board and "chassis cover". This mounting places the heat sinks directly in the air flow.

All voltages were verified with and without loads using a voltmeter and a oscilliscope, before hooking the aux. power supply up to a disk drive. This aux. power supply should work with other disk drives requiring plus five volts and plus twelve volts. I am using a pair of QUME 142 disk drives mounted internally in the P-Box.

Looking back, I feel if I was just starting this project I would build the aux. power supply in its own enclosure. A "Cinch" connector could be used on the back of the P-Box, then come off the "Cinch" connector inside with the standard disk drive power connector.

ARTS LIST FOR AUXILLIARY PLUS FIVE AND PLUS TWELVE POWER SUPPLY
by William M. Lucid

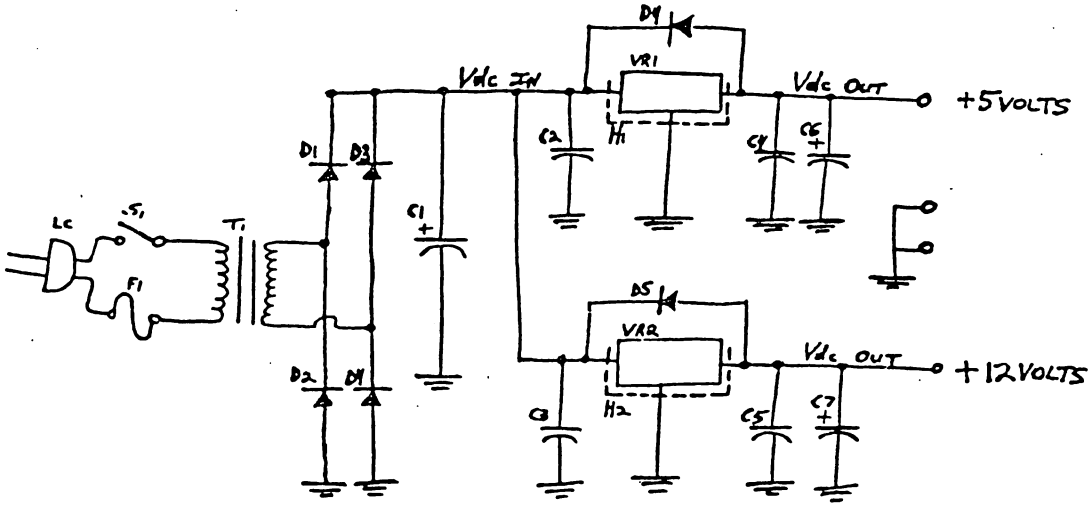
VR1		FAIRCHID 78H05 Voltage Regulator
VR2		FAIRCHILD 78H12 Voltage Regulator
D1-D4	*	FULL-WAVE BRIDGE (Rectifer diodes 400 PIV, 3 Amps)
D5,D6		IN4002 Diode
C1		4,700 UF, 50vwdc, Radial lead, Electrolytic capacitor
C2,C3		.33 UF, 50 vwdc, capacitor
C4,C5		.1 UF, 50vwdc, Tantulum, capacitor
C6,C7		470 UF, 50vwdc, Radial lead, Electrolytic capacitor
T1	*	Transformer 120 volt primary, 18 volt secondary, rated at 3 Amps
F1	*	Fuseholder and fuse, fuse rated 3 Amps
H1,H2	*	Heatsinks for single TO-3 case with one inch fins
LC		Linecord with plug
S1		Switch, single pole, single throw

Comments: Vdc IN needs to be 15.5 to 25 volts at 78H05 and 78H12
* Items need to be changed to get full five amps if needed.
Mount C2, C4, D5 close to VR1
Mount C3, C5, D6 close to VR2

Printed circuit board mentioned in this presentation was produced using the photo negative method, I produced the orginial artwork, exposed the board, etched, and assembled the whole assembly.

There are POTENTIALLY dangerous voltages involved, along with HEAVY CURRENT, CAUTION must be observed. This is not intended as a beginers do-it-your-self type project.

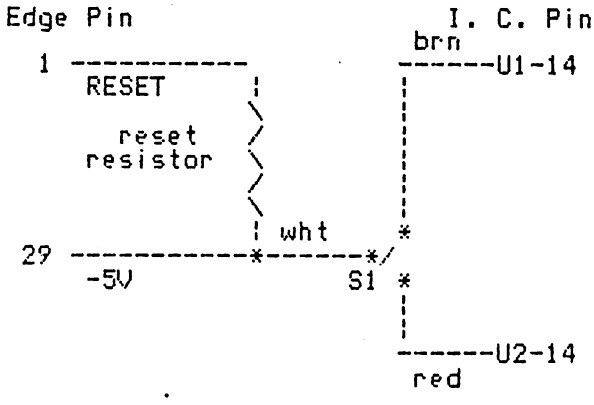
Printed on a PROWRITER 8510A set to bold command, text prepared using TI-Writer/Word Processor.



MORE FOR YOUR MONEY

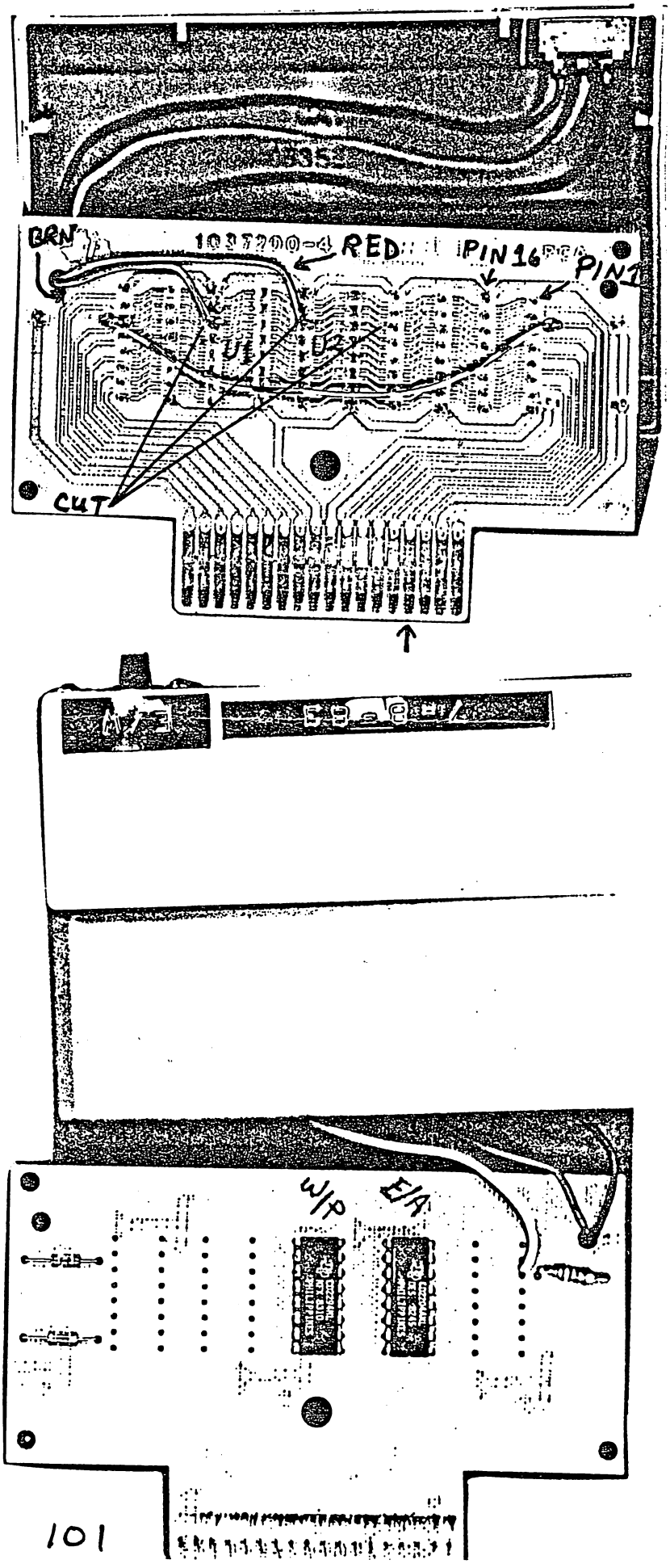
by Jim Ellis

First, let me say that this project is not for the inexperienced. If you don't have the tools or experience or just not sure you want to tackle it, I'm sure you can find someone with the skill to do it for you at a nominal fee. You can have your own cartridge expander or how to make some modules serve double purpose. I recently got tired of swapping my E/A and TI-writer modules. Even tho' I had an expander, I needed to use one more cartridge than it allowed, so I came up with this little modification. I moved my TI-writer GROM into the Editor Assembler module, cut a few lands, added a s.p.s.t. switch and eureka I had it. These boards are laid out with all pins connected together, i. e. pin 1 to pin 1, etc. So you have to cut open the chip select line (pin 14), solder a wire to each pin 14 that you are using, (in my case two), run the wires to a switch, run a wire from the common terminal of the switch to the reset resistor on the card and its done. Pin 14 originally goes to the reset resistor. Also, you must add a jumper from the reset resistor to pin 29 (-5V), see figure. You need to make (n+1) cuts on the line connecting pins 14 together. E.g. two chips require 3 cuts. Oh, yes, the switch is Radio Shack 275-407, which comes two in a pkg., priced @ 79 cents. If you have any questions, leave mail in box 12 on the HUGbbs or call me at 831-5791. If you read diagrams, I included a simple schematic.

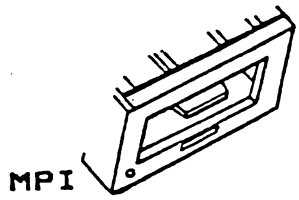


With the addition of two more switches it could be made to switch four different programs. Since there are five openings on the board, you could have two-one GROM and one-three GROM programs all in one module. How about that? TI fans!! Later.....

Radio Shack 275-407

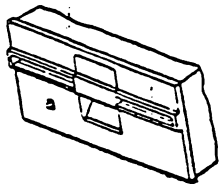


This article was reprinted from the August, 1985 issue the New Hampshire 99'ers Users Group Newsletter of Manchester, New Hampshire.



ADDING A SECOND TI DRIVE...
(a.k.a. IT WORKS FOR US!)

Helene M. LaBonville & Ellen J. Rule
121 Camelot Drive - Bedford, NH 03102
NH 99'ERS USER GROUP



SEARS (Manchester, NH) is currently closing out on their TI99/4A inventory. At this writing they had over 20 internal disk drives (PHP 1250) remaining at $50 each. They have both the SHUGART 400L's and MPI 51 S' in stock. We wanted a second drive BADLY! Since the cheapest, new, TI external drives were $135, and our funds were limited, we decided to "fashion" our own.

Armed with copies of five different articles on the subject, we abstracted "the best" and "the easiest" advise.

You will need the following parts:

The bare disk drive.....$50.00
An enclosure (vertical mounting) & power supply..... ~ $42.00
(One of our members sells power supplies for $15,
and we found an enclosure at a flea market for next to nothing.)
A 15" drive cable with two 34-pin edge card connectors....$6.00 (we had one)
A 4 position DIP SWITCH (RS# 275-1304A)....$1.60
A replacement resistor pack*....FREE from TI

TWO DRIVE CONFIGURATION	INTERNAL	EXTERNAL
#1.*	SHUGART	SHUGART
#2.	MPI	SHUGART
#3.	MPI	MPI

*On multiple drive set-ups, you must replace the termination resistor pack on all SHUGART drives except the last drive in the series. If the SHUGART is to be used with other TI drives (MPI's), install the SHUGART as the last drive in the series. A call to TI-CARES will get you a FREE replacement resistor pack (a 270 ohm resistor soldered onto pins 5 and 10 of a 14-pin DIP header). You may choose to just add a 270 ohm resistor between those two pins in the DIP socket itself. Our drives cost US less than $75 to assemble.

Only configuation #1 requires the replacement resistor pack. The INTERNAL drives in configurations #2 and #3 DO NOT require modification. Since we opted for configuration #1 (for aesthetic reasons), we had to replace the termination resistor pack on the internal drive. Because drive 1 gets most of the workout we also decided to use our newest drive as drive #1. In any event, if you opt for dual drive configuration #1, you must replace the termination resistor pack with a 270 ohm resistor pack:

>Turn off the Peripheral Expansion System and disconnect the power cord from the back of the unit. CAUTION - WAIT TWO MINUTES AFTER TURNING OFF THE UNIT FOR POWER TO DISCHARGE BEFORE PROCEEDING.

>Remove the top of the PES by depressing the latches on the back edge of the top and pulling up.

>Remove the four screws holding the internal drive in place and gently slide the unit out of the compartment. You need not disconnect the cables that attach the unit to the PES and controller card unless you decide to replace the drive with the newer one.

>Locate the termination resistor pack, noting the position of pin 1 in relation to the drive. Using a very small pocketknife blade or a small screwdriver tip, CAREFULLY pry the resistor up and out of the DIP socket. Take care not to bend the pins in case you need to use the resistor pack later.

>Insert the new resistor pack with pin 1 (the number "1" is either molded in or represented by a notch) occupying the same position that pin 1 of the termination pack occupied.

>Reinstall the disk drive, expansion system top, and power cord in the reverse order of the procedures above.

Because many of you may not have received the special cables or an adaptor board with your TI disk controller card, lost them, or have a CorComp Card, we have adopted John Hamilton & Ron Rutledge's (Central Iowa UG) method for adding a second drive... replacing the shunt pack with a DIP SWITCH, thus dispensing with the need for specal cabling and adaptor boards. The shunt pack is a workhorse. It controls the following:

Pins 1-14 ... Head Select - head is loaded by drive select signal
Pins 2-13 ... Drive Select 1 - selects drive as #1
Pins 3-12 ... Drive Select 2 - selects drive as #2
Pins 4-11 ... Drive Select 3 - selects drive as #3
Pins 5-10 ... Multiplex - means multiple drive configuration
Pins 6- 9 ... Door Lock - means drive in use when door is closed
Pins 7- 8 ... Head Motor - head is loaded by motor signal

The TI drives all come as Drive #1. The adaptor boards and "special cables" are "configured" to side-step the shunt pack and select the proper drive in multi-drive set-ups. Many of our members have encountered problems with the TI method. The Hamilton-Rutledge method has worked for us since day one (over two months). I even used this method to configure a TI SHUGART Drive to my RS Color Computer, and it is working great!

For the external drive then, we recommend that you first remove the thin aluminum housing that covers the drive. This allows for more airflow to occur, next:

>Fasten the disk drive to the enclosure floor.

>Attach the 4-pin polarized plug from the power supply to the 4 pins on the drive's board, then attach the grounding wire from the ps to the rear of the drive. CAREFULLY replace the shunt pack with the DIP SWITCH. As this is your second drive, position #1 should be ON, #2 should be OFF, #3 ON, and #4 should be OFF.

>Attach one end of the drive cable to the edge card on the drive (note that there is a "key" in the connector which mates with the "hole" in the edge card). Connect the other end DIRECTLY to the disk controller card.

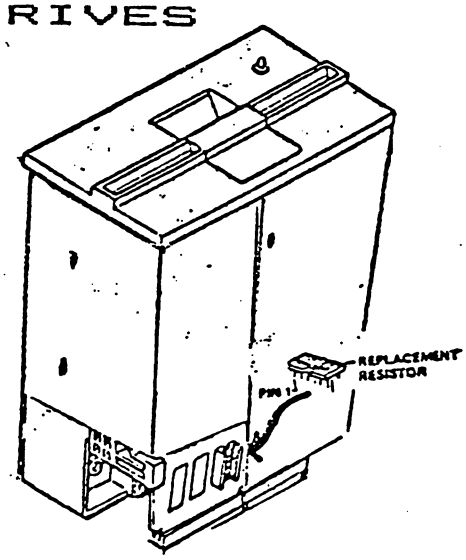
>Now secure the enclosure cover in place.

>Test the drives by cataloging a disk on each of them.

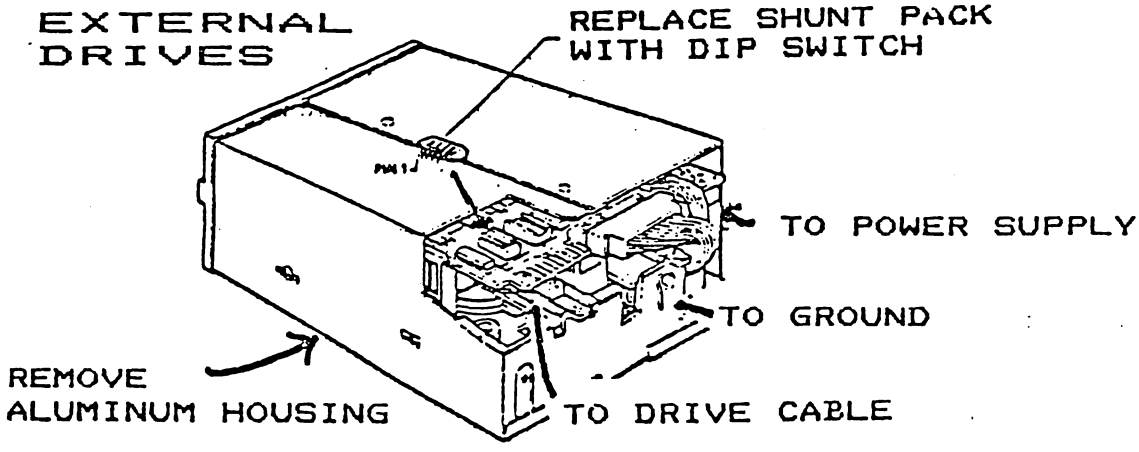
We must acknowledge the following people who have provided us with the basis for this article: Ed Lawson and Richard Bailey of the N Hamp. 99'ER UG, John Hamilton and Ron Rutledge of the Central Iowa UG, and John Worthington of MANNERS.

INTERNAL DRIVES

SHUGART INTERNAL DRIVES REQUIRE PEPLACEMENT RESISTOR



EXTERNAL DRIVES



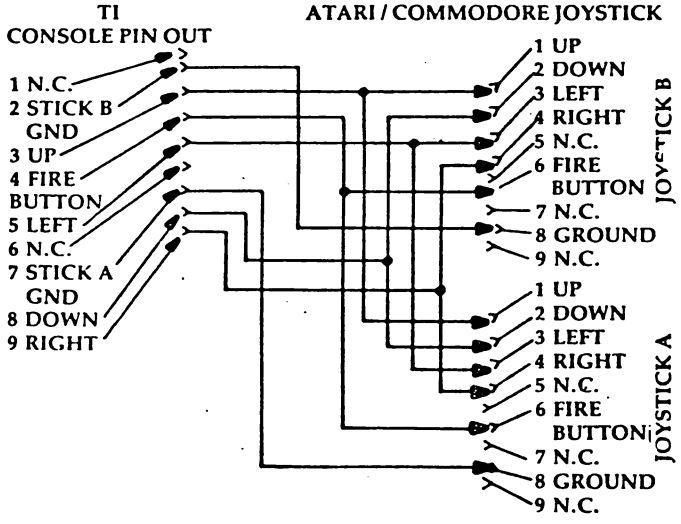
HOW TO BUILD YOUR OWN.....
.....TI-99/4A JOYSTICK ADAPTER

by Gary Cook

From Compute!, August 1983

The following article was included in the September, 1983 HUGger Newsletter.

As an owner of a TI-99/4A, I decided I wanted a joystick to go with it. To save time and money, I got the Atari pin configuration from a friend and TI's configuration from the TI toll-free information line. After that it was a simple matter of buying three nine-pin "D" connectors (two male and one female), a small box, and some wire. Following this wiring diagram, you can make this adapter in about an hour and be able to select any joystick from the wide variety of Atari-compatible joysticks sold.



KEYBOARD KLINIC

by Bill Jones (of Indy)

A few nights ago I was programming away and discovered I had lost the '=' key.

Step one was to pull the key cap and clean the contact. No luck. I pulled off the bottom of the computer and found the problem.

Most of the keyboards I've seen on TI's are made on a paper phenolic PC board that absorbs humidity from the air. That makes it swell and shrink, eventually, the solder joints crack and the key stop working. I checked my other computer and found it also had some cracks starting. Both of them are over two years old. One I use daily, and the other I keep as a spare. This problem is caused by time not by use.

ANOTHER USE OF THE LOAD INTERRUPT SWITCH

by Gary Brown
in the J*U*G*S Newsletter

Assemble this source code

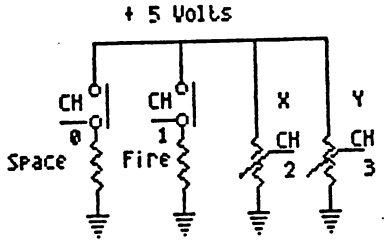
```
AORG >FFFC
DATA >A000,B0BE END
```

Load SBUG
Load the interrupt program you have assembled above.
Press Function =(Quit)
Do not turn console or P-Box off
Insert a cartridge and start it running.
Press the interrupt switch and you will have the SBUG title screen

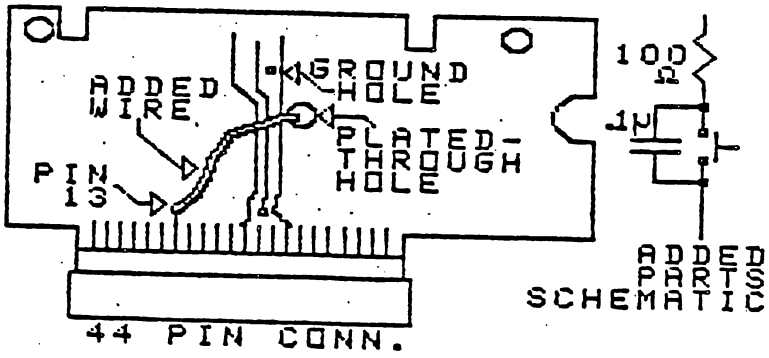
Follow the instructions and go into addresses 6000 thru 8000 find the program in these addresses. Then you can disasemble these addresses.

Remember no one assumes any liability for your computer should you try any console modifications.

MBP Analog-Joy Stick



All resistances greater than 10,000 ohms.



BOTTOM VIEW OF SPEECH SYNTHESIZER BOARD

ADDING A LOAD INTERRUPT SWITCH TO THE SPEECH SYNTHESIZER

Richard J. Bailey
68A Church Street
Gonic, N. H. 03867
NH99'ERS USER GROUP

A number of people have asked me about the load interrupt switch I had added to my speech synthesizer to allow dumping screens from the various cartridges using the excellent screendump program that was written by Danny Michaels. So here are instructions to allow you to modify you own synthesizer to accomplish this.

Keep in mind that you have to know enough about electronics to add the parts needed for the modification without messing up your synthesizer. I have made the modification to my own synthesizer so I know that it works, but if you mess up, then you're out a synthesizer. You could add the same parts inside the console and have a small switch sticking out the back if you want the modification self-contained or don't have a speech synthesizer.

The only part really needed is a miniature pushbutton switch with normally open contacts but if you add a 100-500 ohm resistor in series with the switch and a .01-.1 MFD capacitor across the switch, there will be less chance for contact bounce (If you really want bounce-free contact closure, use cross-coupled gates as an R-S flip-flop). The added parts schematic and location diagram of the speech synthesizer board is shown above. These were drawn with GRAPHX.

To modify your unit, do the following:

1) buy the parts. The switch must not stick >1/4in. beyond threads.
2) dismantle synthesizer. note how shield slides together.
3) clear large plated-through hole of solder.
4) solder 2 1/2in. piece of wire to pin 13 of 44 pin connector. (all other parts go on top side of circuit board)
5) solder one end of 100 ohm resistor in ground hole.
6) solder 1 1/2in. piece of wire to other end of resistor.
7) solder wires to switch and .1 MFD capacitor.
8) drill hole in middle top of shield for switch.
9) mount switch, making sure everything fits.
10) reassemble unit, making sure nothing shorts.

You can now follow the instructions for the screendump program to check the operation of the switch. You may find other interesting uses of the switch. If you do, please pass them on to the newsletter.

ELIMINATING BACKGROUND NOISE WITH THE R.F. MODULATOR

(Editor's Note: The following article is printed in the HUGgers Newsletter through the courtesy of the Indianapolis TI Exchange Center.

When experiencing background noise, such as humming or buzzing, wht the R. F. Modulator, internal adjustment in the Modulator will usually alleviate the problem. This can be accomplished by the user by following the steps below and referencing the illustration below. This procedure is to be done while all equipment is on and operating. If you have the old version of the TI900 Video Modulator, this procedure does not apply.

(Materials required: one small, flat, thin-bladed screwdriver)

To correct the noise difficulty:

1) Turn the volume of the television all the way down, but do NOT turn it off

2) Select the Master Title Screen on the computer (FCTN =, if necessary)

3) Using the title screen color grid, fine tune the television to the best color picture

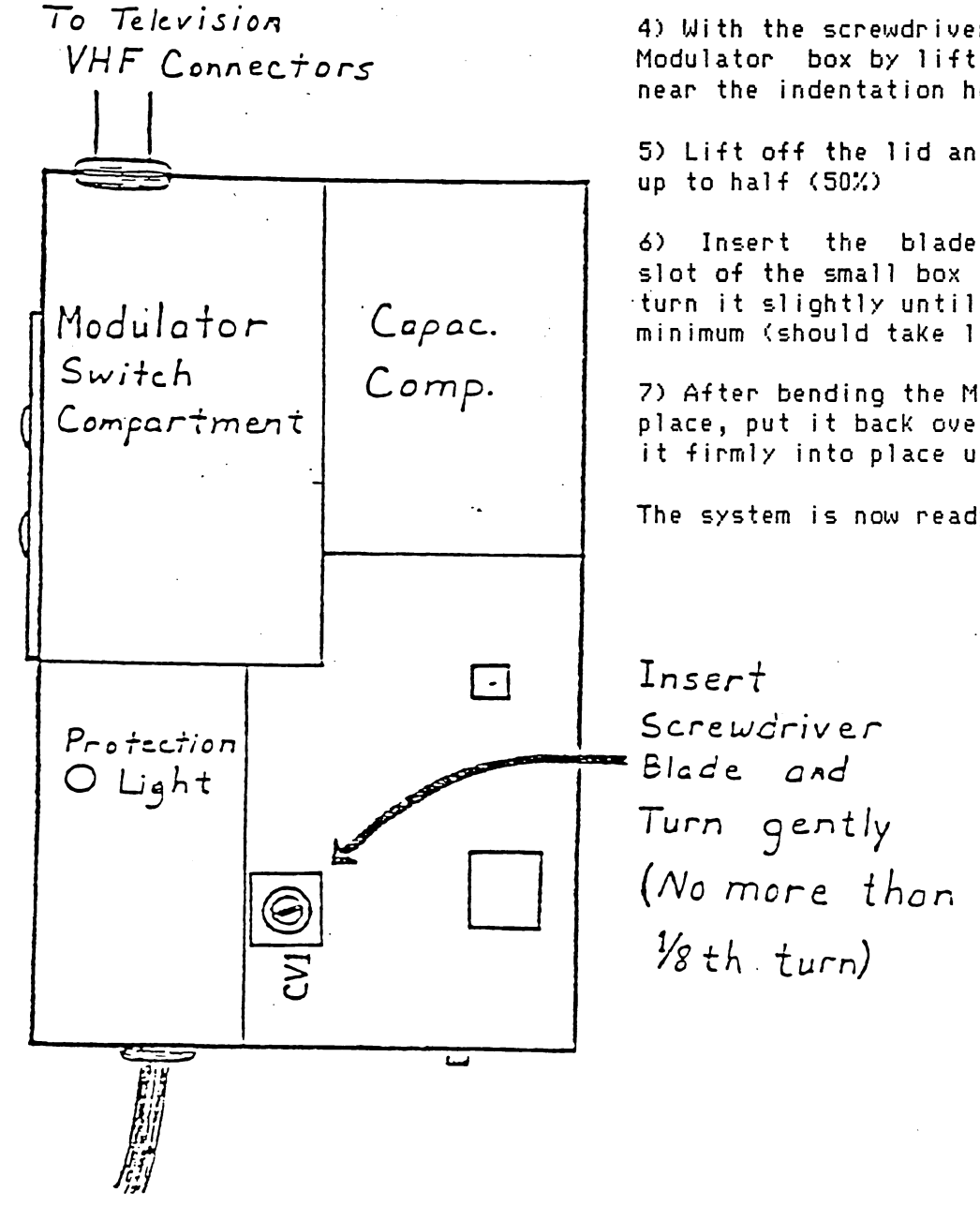
4) With the screwdriver, pry off the lid of the Modulator box by lifting under one edge of the lid near the indentation holding it on

5) Lift off the lid and turn the television volume up to half (50%)

6) Insert the blade of the screwdriver into the slot of the small box labelled CV1 (see fig.) and turn it slightly until the background noise is at a minimum (should take less than 1/8th of a turn)

7) After bending the Modulator lid edge back into place, put it back over the Modulator box and press it firmly into place until it snaps.

The system is now ready for optimum usage.



Does your TV Set/Monitor have a terminal case of wavy lines moving across the screen? Quite by accident, the Set Up Committee of the Hoosier Users Group has found a cure!

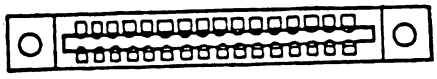
Due to large meeting and class attendance, we have to set up several monitors so everybody could see what was being displayed on the screen. A Radio Shack 4-Way Distribution Amplifier was the answer. This Amplifier allowed us to set up four TV's to the group's TI system, fine. But what we didn't realize, it also eliminated the wavy lines on ALL monitors that happened to be set up!.

For those who want to hook this type of Amplifier to your TI, you would need the following parts:

Distribution Amplifier (Cat. # 15-1119).....$12.95
VCR/TV Matching Transformer (Cat. # 15-1253)..... 2.99
75 Ohm Coaxial Cable with Male "F"
Connectors at each end. Priced from $2.49 to 5.99
TV-Matching Transformer (a Non-Radio Shack part)

And you said you can't get rid of your wavy lines? ..... Now, for under $25.00 and less than 5 minutes set up time, you can too!!!!!

I/O PORT PIN ASSIGNMENTS
GROM PORT

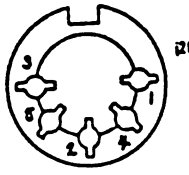


PIN	DESCRIPTION	PIN	DESCRIPTION
1	RESET	2	GND (SYSTEM)
3	D7	4	CRU CLK
5	D6	6	CRU IN
7	D5	8	A15/CRU OUT
9	D4	10	A13
11	D3	12	A12
13	D2	14	A11
15	D1	16	A10
17	D0	18	A9
19	+5 VOLT	20	A8
21	GS (GROM SELECT)	22	A7
23	MO/A14	24	A3
25	MI (DBIN)	26	A6
27	GROM CLOCK	28	A5
29	-5 VOLT	30	A4
31	GR (GROM READY)	32	WE
33	GND (GROM)	34	ROM G
35	GND (SYSTEM)	36	GND (SYSTEM)

I/O PORT PIN ASSIGNMENT
PERIPHERAL I/O port

PIN	DESCRIPTION	PIN	DESCRIPTION
1	+5 VOLT	2	SBE (SPEECH SELECT)
3	RESET	4	EXT INT
5	A5	6	A10
7	A4	8	A11
9	DBIN	10	A3
11	A12	12	READY/HOLD
13	LOAD	14	A8
15	A13	16	A14
17	A7	18	A9
19	A15	20	A2
21	GND	22	CRU CLK
23	GND	24	B 3
25	GND	26	WE
27	GND	28	MBE
29	A6	30	A1
31	A0	32	MEMEN
33	CRU IN	34	D7
35	D4	36	D6
37	D0	38	D5
39	D2	40	D1
41	HOLD/IAQ	42	D3
43	-5 VOLT	44	SPEECH

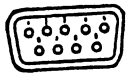
I/O PORT PIN ASSIGNMENTS
VIDEO JACK



PIN	DESCRIPTION
1	+12 VOLT SUPPLY FOR EXTERNAL UNITS SUCH AS MODULATOR
2	SHIELDING CONNECTION
3	SOUND OUTPUT
4	COMPOSITE VIDEO OUTPUT
5	GROUND CONNECTION

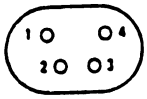
I/O PORT PIN ASSIGNMENTS
REMOTE WIRED HANDHOLD CONTROLS I/O PORT

MALE PLUG FRONT VIEW



PIN	DESCRIPTION
1	NOT CONNECTED
2	JOYSTICK B
3	KEY(0 (UP)
4	KEY 4 (PUSH BUTTON)
5	KEY 3 (LEFT)
6	NOT CONNECTED
7	JOYSTICK A
8	KEY 1 (DOWN)
9	KEY 2 (RIGHT)

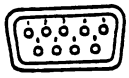
I/O PIN ASSIGNMENT
POWER RECEPTACLE (USA)



FRONT VIEW
MALE PLUG CONSOLE

PIN	DESCRIPTION
1	NOT USED
2	18 VOLT AC
3	COMMON
4	8 VOLT AC

I/O PORT PIN ASSIGNMENTS
CASSETTE I/O PORT



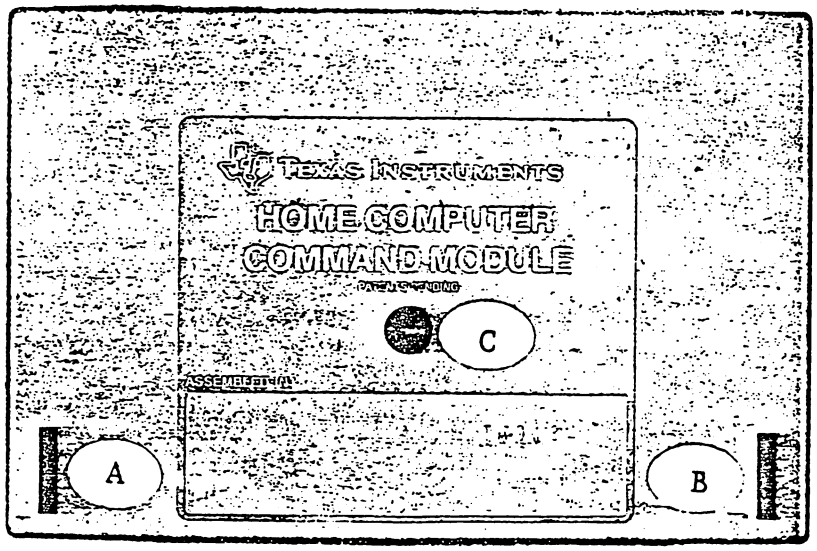
MALE PLUG
FRONT VIEW

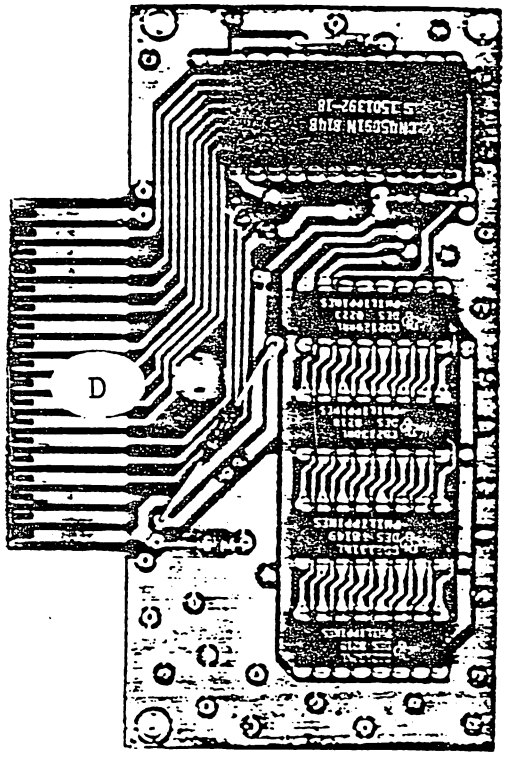
PIN	DESCRIPTION
1	CS1 MOTOR CONTROL (POS)
2	CS1 MOTOR CONTROL (NEG)
3	GND (SYSTEM)
4	SOUND OUT
5	RECORD OUTPUT
6	CS2 MOTOR CONTROL (POS)
7	CS2 MOTOR CONTROL (NEG)
8	AUDIO IN
9	AUDIO GROUND

Dirty contcts can screw-up any electrical device and the 4A is not an exception. The only place you are fairly likely to run into this problem is in using command modules. Both the module contacts and the port itself can become dirty but cleaning the port itself is a big job as you have to dis-assemble the console. The good news is that cleaning the cartridge will almost always suffice and can be done quickly without any special tools or cleaners. All you need is a regular screwdriver, some sort of rag, a standard pencil eraser, and in some cases a medium phillips screwdriver.

Remove the screw from "C" if there is one. Then pry the clips in slots "A" and "B" outward to pop open the cartridge. If there is a clip in "C" pry it back after "A" and "B" are loose. If it should bend off don't worry, it won't affect the performance of your module.

The module board can now be removed. Do this carefully and note how the spring-loaded "door" is assembled if there is one so that you can put it back together if it pops out. Once you have the board removed take your rag (a kleenex will work but something cloth is much better) and rub off any residue from the contacts, shown as "D". Remember to do the contacts on both sides if that particular module has them. Once the worst is removed take any soft rubber eraser and "erase" the contacts until they become dry, clean and shiny. You need to do only about the outer half of the contacts as that is more than ever gets used (you can see the scratch marks in the picture below). Once this is done simply put the cartridge back together and go. Some symptoms of dirty contacts are the console locking-up, strange errors where no occured before, etc (my XB cartridge giving me a syntax error when there was non for example). Don't jump to clean a cartridge on your first error, it could be alot of things like static, not having the module in tight, or a number of other things. But if you find you have a continuing problem cleaning the contacts is quick and free and may correct what was wrong.





WORDSEARCH ANSWERS

#1

```
COMPUTER....CLOSE........D.
........G.........C......I.
.......O.......TUPNI.....S.
......S....P........S....K.
.T...U.....R.........A...D.
..I.B......I..........B..R.
...G.......N.EMULATOR....I.
....E......T............V.
R....R...................E.
A.....C........TUPTUO......
N......U...................
D...S...B.............TXEN.
O..A....Y........R.....E....
M.V.....T.......COMPUTER....
IE......E........F.....M....
Z.......S...E.........I.....
E............T....M...N.....
..............A....U..A.....
...ASSEMBLY....D....N.L.....
................P...........
..INTERNAL.......U..........
................CLEAR.......
...........................
.....ROTCES..APPEND...CAGLE
```

#2

```
...........POOL.............
...........................
...........................
.......YRALUBACOV..........
...........................
...........M...............
.............A.............F....
.............G...........O.....
...............N.......D.R.....
.........P......AI.....I.T....E..
.......A........BF...G.H.....X..
......U..........OY.IC.......E..
.EDITS.............RT..O......C..
....E....EGDUMS....T...M.....U..
B.......................P....T..
U........D.......C........I...E..
F.........E......R..ERROR.L.....
F..........C.....E....B....E....
E............I....A.....O.......
RNEERCS......M...T......O.......
..............A..E.......T......
........UPDATE.L.............
...........................
...................EZIMODNAR....
.............SPRITE...........
```

#3

```
HUGBBS............ECAPSKCAB
.....C...T................E
G....R...SRORRE..M........L
O....E...I.....PREVIEW....L
O....ESRELLACTSILS.......G.
D..R.N...........S......N..
B..O.............A.....I..L
Y..T..E..........G....T...I
E..A...LOGOFF....E...N....N
...L..C.U........M..I...N.E
...U..H..D...S...O.R....O.F
...M..A...O..C...DP....TIDE
S..E..R....M.R...U......T.E
Y..L..A.....EO...L......P.D
S..A..CHAT...L...E....Y.O.S
O.U..T......LI.S.....R.T..
P..N..ED.....I.F.A....T.L..
...I..R.s....N....V...N.U..
...M..B..W...G.D...I..E.A..
...R..U...N.....A...N...F..
.PLEH.F....L.....O...G..E..
...T..F.....O.....L.....D..
......E......A.....P.......
.....SREDAEH..D.....U......
```

4

```
.........CALLHCHAR.........
.C.....................N.....
C.A...TAPRAHCLLAC...E......
A..I..............E........
L.K.L.............R.......C
LCNIOCLLAC.C..Y..C.......AA
V.I...O.....A.A.SA......L.L
C.L....L.....LSL.L.....L..L
H.L.....O.....L..L....C...C
A.L...C..R...ALJ.K...H....H
R.A...A.....C.A.OE..A.....A
..CCALLGCHAR..C..Y.R......R
...T..L...........S.......R
...E..L..........E.T.....AR
E..G..O..CALLINIT.......E.E
T..P..A.........K......L..L
A..S..D...NOISREVLLAC.C...L
C..L..........E......L....A
O..A.........P......L.....C
L..C.........L......A.......
L..........L.......C........
L.........A.DNUOSLLAC......
A........C.................
C..............ETIRPSLLACC
```

WORDSEARCH ANSWERS

#5

```
.....AHPLAU...
.....C.....C.....
.....A...A....ACCEPTAT.....
.....L.L.....O...
.....L.....N....
.....C.J.....D.....W.....
.....H...O.....IO...A.....C
.....A....DY.....NS..R.....A
.....R....I..S.....E..PN.....L
....S..C.S....T...R...IL.....L
...E...AP.O.....R...N..A.....P
..T....L..CN....OL..G....Y.....E
.....AL...AB..R..I.....U....E
.....Y.I....LR.....N.....S...K
....A..N.....LE.....P.....I...
...T...I.....LA.....U.....N..
.....TE.....IK.....T.I.....G.
.....R.....N.....M.....
.....A.....K.....A.....
.....S..S...GNISUTNIRP..G.....C
....I...E.....E.....
...Z....A.....NOISREVLLAC...
..E.....L..ETIRPSLLAC.....
.....L.....
....DAOLLLAC.....
```

#6

```
.....
.M.....T.....
.O.....I.....
.O....SLYMOIDS...I.....
.N.....B.....N.....
.M.....U.....V.....
.I.....R.....A.C....H.....
.N.....G...A.....D.HM....U.....
.E....E...D.....SE.I.U....S.....
.....R...V..Y..U.R.S..N....T....
....T...E..A..P..S.OO..C....L...
T..I...N..H..E.....L.A..H....E..
U.M...T..T..R.....M..R..MS.....
NE...U..Z..D.....T..SW.TA.....
N...R..E..E.....R...NA..N....
E..E..E..M.....AH..RER.....
L.....O.....I.OT..GS.....
S.....N.....L.RP...G.....
O....A.....E..P...I....
F....T.SSEHCOEDIV..K....E...T...
D...T.....R...T..
O..A.....
O.COLLEHTOPARSECTOMBSTONECITY...
MK.....
.....ALPINER.....
```

I'm sorry, but it seems no one actually worked this wordsearch.

WORDSEARCH # 7

COMPUTER	MONITOR
RUN	JOYSTICK
PRINTER	SAVE
MODULE	PROGRAM
ERROR	BASIC
SPRITE	MEMORY
SPEECH	COMMAND
CHARACTERS	CASSETTE
SUBPROGRAM	STRINGS
DISK	LOAD

HOOSIER USERS GROUP
P.O. Box 2222
Indianapolis, IN 46206-2222

Forwarding and Address
Correction Requested

#7

MEHZIIJCYIHRVEFUYQPCIXUODFSKCSRM
YSQBRJCCUOLUVIABDZDHCZWDEYQQXWRD
VSPRITEWLKKSGEGUBDDEHWFGQUTPTRKF
FQVVNMWUMBPWCGSCZLTNAQOHKGMQERUN
VVYCJTYSPVWGVCDPTTTSRMQEZBQTHOYE
IOLDFZYTIMXXNCZUEWHYAMHEJYUYFWPN
NTAQEZKLYROHZSGSNEBOCHOKVPKYULWR
QYESXLNTXZFBGISAFCCOTZFRMDEYRKEO
WXZRWCWEELFNUAPRDAOHEUQOUSAKLMXW
NCEJRETAVGIYCLQHPQCMRTCPZYVJSBET
ICFHCOMMYRXKGVEDUGRNSTIZBZTRNVHU
UZCSNKRWTOPBRRRVSFPNYZRSGCREAORK
IJRIQLSSKTHVUYGUQWKRASMFXOXSFTIX
GODKSUDUKQPTEAWWASOQNRFUNNBWGQAO
PYGLGAJKHONVCKSVIMSPDSGMVMYWDDTS
RSRXFVBAHIPXDRNDEVYAIFTOAZDNHHGB
OTTVLQRARYKFEKCMUPOZROIPRNQEPKPP
GIIVHPYPJQKDGODXLLQZNHZSAPERNYJF
RCNRWXTRZYYYLHSKQZKHGERMPWBQIJHV
AKRLQFTYMBWEWGSFPPPHGXMBKOGUBMEE
MRXALGGKDUKAXJVTPEWZKOPVWPYDSAFI
CNOBIPFGHSREUHJXYWKGCJLBLXAXGXEL
NYVMBJGTVONORYXCQYRUNFJVADEXDJYB
YLWGZWANRTNJVBDMHWHJNZLMONITORQM
EKMEOCZJRUJALRXZEAPFTCUXVFSYNQVC

WORDSEARCH # 7

This WORDSEARCH contains 20
COMPUTER RELATED termss.

WORD LIST

COMPUTER	MONITOR
RUN	JOYSTICK
PRINTER	SAVE
MODULE	PROGRAM
ERROR	BASIC
SPRITE	MEMORY
SPEECH	COMMAND
CHARACTERS	CASSETTE
SUBPROGRAM	STRINGS
DISK	LOAD

POCKET PROGRAM

Here is a short Basic Program to help you calculate payments for simple interest loans:

```
100 REM PROJECTED MONTHLY PAYMENT
110 CALL CLEAR
120 INPUT "AMOUNT TO BE BORROWED  ":A
130 INPUT "NUMBER OF MONTHLY PAYMENTS  ":N
140 INPUT "INTEREST RATE PER YEAR ##.##%  ":I
150 I=I/1200
160 P=A*(I/(1-(1+I)^-N))
170 P=INT(P*100+.5)/100
180 PRINT P
```

```
100 ! THIS PROGRAM WILL PRINT A 131 COLUMN LINE WITH THE PROWRITER.
110 OPEN #1:"PIO.CR" ! STOPS THE 99/4A FROM GIVING A CARRAGE RETURN AFTER 80 CHA
RACTERS
120 OPEN #2:"PIO" ! OPENS A  PRINTER CHANEL WITH CARRAGE RETURN AND LINE FEED
130 PRINT #1:CHR$(27);CHR$(81)! SETS PRINTER IN COMPRESSED TYPESTYLE (17 CPI INC
H)
140 FOR I=1 TO 3 ! LOOPS 3 TIMES THROUGH PRINT COMMAND
150 A$="00000000001111111111222222222233333333334444444444555555555566666666667
77777777888888888899999999990000000000111111111122222222223"
160 PRINT #1:A$
170 PRINT #2:! PRINTS A CARRAGE RETURN AND LINE FEED ONLY
180 NEXT I
190 REM by bill  FYI
```

00000000001111111111222222222233333333334444444444555555555566666666667777777777888888888899999999990000000000111111111122222222223
00000000001111111111222222222233333333334444444444555555555566666666667777777777888888888899999999990000000000111111111122222222223
00000000001111111111222222222233333333334444444444555555555566666666667777777777888888888899999999990000000000111111111122222222223

APPLICATION FOR MEMBERSHIP

Below you will find an application for membership to the Hoosier Users Group. Active membership entitles you to the Newsletter, up and download on the HUGbbs, attendance and voting rights at regular club meetings, access to the HUGger Library of Programs, special club activities and special guest speakers for one year. Subscribing members will receive the **NEWSLETTER** only.

Make check or money order payable to **Hoosier Users Group.** Send completed application to:

HOOSIER USERS GROUP
P.O. Box 2222
Indianapolis, IN 46206–2222

(Cut on dotted line)

Check One:

Active Member

New: $20 _____
Renewal: 15 _____

Subscribing Member

New: $10 _____
Renewal: 7.50 ____

Amount Enclosed: $ _____

_____ D _____
S _____

Name: _____ Today's Date: _____

Address: _____ Apt. # _____

City: _____ State: _____ Zip: _____

Phone: (_____) _____-_____

Interests/Comments: _____

_____

APPLICATION FOR MEMBERSHIP

Below you will find an application for membership to the Hoosier Users Group. Active membership entitles you to the Newsletter, up and download on the HUGbbs, attendance and voting rights at regular club meetings, access to the HUGger Library of Programs, special club activities and special guest speakers for one year. Subscribing members will receive the **NEWSLETTER** only.

Make check or money order payable to **Hoosier Users Group.** Send completed application to:

HOOSIER USERS GROUP
P.O. Box 2222
Indianapolis, IN 46206–2222

(Cut on dotted line)

Check One:

Active Member

New: $20 _____
Renewal: 15 _____

Subscribing Member

New: $10 _____
Renewal: 7.50 ____

Amount Enclosed: $ _____

_____ D _____
S _____

Name: _____ Today's Date: _____

Address: _____ Apt. # _____

City: _____ State: _____ Zip: _____

Phone: (_____) _____-_____

Interests/Comments: _____

_____