

THE OFFICIAL NEWSLETTER OF THE

TI-99/4A

APRIL 1986 NO. 7

FOR THE RECORD

by Ed Bittner
Recording Secretary

The March meeting of the West Penn 99'ers was opened by the newly elected president Scott Coleman promptly at 7:00. The initial discussion centered around the possibility of locating the meeting at alternative places. No conclusion was reached but it was stressed that not only convenience was important but also cost of rental. At the last executive committee meeting (Feb 25), yours truly volunteered to seek out the source of a newspaper article offering DS/DD diskettes at a very low cost. Ken Farr did indeed receive diskettes from this company and vouched as to their reliability. Scott asked for a motion to proceed with the purchase of 200 of these diskettes for club use and resale (as well as purchase of 50 cassettes (C-60) in conjunction with the PUG). John Willforthingly so moved and it was seconded and a unanimous vote ensued to go ahead with the purchase with club funds. The club will now have DS/DD diskettes (which may also be used as SS/SD) for sale for the unbelievably low price of \$.60 each in lots of 5 or more to club members. No firm price was established for larger quantities. J. Willforth reported that no new information was yet available regarding liability insurance for the group and Jan Travers gave the treasurers report.

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

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

Following the official business, John Willforth did a nice demo of a RAM DISK (20150 in kit form) which allows for 90K storage when used in the SS/SD mode but can be upgraded to DS/SD easily (so John says!). With the RAM DISK access to programs is at a minimum, often loading programs faster by a factor of 10. It also is battery backed which brought a lot of ooze and ahz from the members.

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

Erroneously submitted,
Scoops Bittner

BASIC BASICS

by Charles Strink

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

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

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

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

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

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

OFFICERS

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

YES WE HAVE DISKS!!!!!!

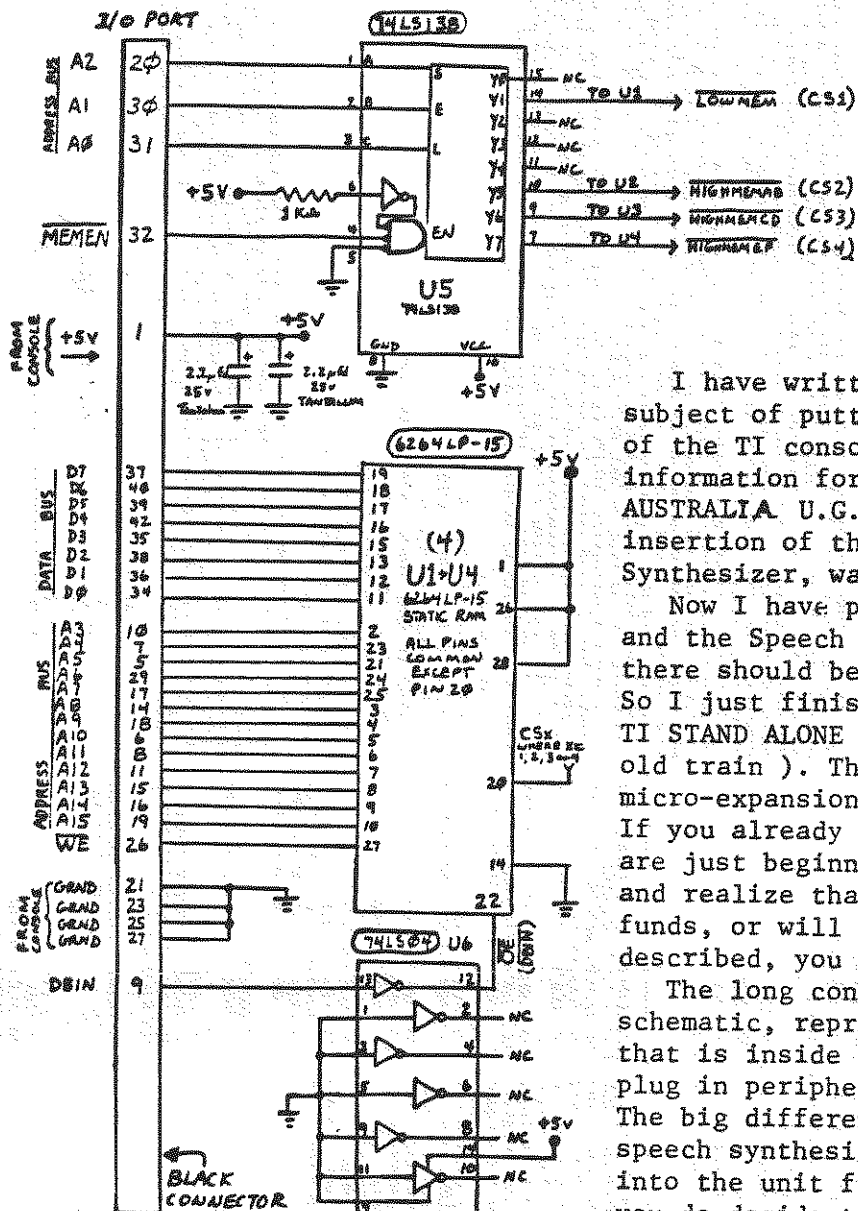
AS OF THIS WRITING, WE HAVE DISKS, AND WILL HAVE MORE FOR THE MEETING. THE DISKS ARE DOUBLE SIDED DOUBLE DENSITY (DSDD), AND CAN BE USED ON ALL DISKETTE DRIVES THAT CAN BE CONFIGURED ON THE TI-99 COMPUTER. IF YOU CAN'T MAKE THE MEETING, AND WOULD STILL LIKE TO GET SOME, THE PRICE IS \$8.00 FOR 10 DISKETTES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

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

BALANCE AFTER FEB. MEETING	6439.04
AMOUNT PAID OUT IN FEBRUARY	
CHECKS	\$ 4.60
POSTAGE	22.00
EQUIPMENT	126.59
(2 SUPER DISKS)	
(1- 4 A/TALK)	

BALANCE AS OF
MARCH 16, '86 8285.85

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



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

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

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

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

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

The long connector on the left of the schematic, represents the large 44-pin conn. that is inside the speech synth., or any other plug in peripheral ie: Stand-alone Disk Cont.. The big difference, however, is that ONLY the speech synthesizer carries pins 1,2,43, and 44 into the unit from the console. Therefore if you do decide to put memory into any other unit than the speech synthesizer, I would recommend that you wire across that unit, in other words

you should run a wire from pin 1 on the console connector to pin 1 on the output end of that unit, where the 2nd unit from the console might be plugged in, and do the same for pins 2, 43, and 44. This will enable you to put the very small speech synthesizer out on the end, instead of between the 2 much larger units (console and Disk Controller). There is only one lead that is involved here that is a must, and that is the pin 1, since I have stayed with using the +5 VDC from the console, rather than tapping it from the +5 Volt source in the unit where this is installed.

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

VCR TITLE SCREEN PROGRAM IN BASIC
by John Hedstrom

This is a simple program to title VCR tapes. It allows six lines of text (at rows 5,8,11,14,17,20) and 28 characters per line. To skip a line, press ENTER to print a quotation mark, type in 3 of them consecutively. The program will center each line horizontally and will then draw a border which looks like a filmstrip. The border uses the cursor which is ASCII code 830 and it remains black.) The computer can be hooked up to the VCR via the VHF antenna input (with modulator cable and VCR set to tuner) or via the Video In input (with monitor cable and VCR set to line) and the title screen can then be taped for any length of time.

```
100 REM VCR Title Screen
110 REM by John Hedstrom
120 REM December 3, 1984
130 CALL CLEAR
140 INPUT "Screen Color?";R
150 INPUT "foreground Color?";F
    IF
160 INPUT "Line 01";L1$
170 INPUT "Line 02";L2$
180 INPUT "Line 03";L3$
190 INPUT "Line 04";L4$
200 INPUT "Line 05";L5$
210 INPUT "Line 06";L6$
220 T1=(30-LEN(L1$))/2
230 T2=(30-LEN(L2$))/2
```

MSP 99 NEWSLETTER

DISPLAY/VARIABLE GO TO PROGRAM CONVERTER
created and written by an editor

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

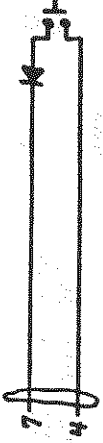
Now why, you might ask, would you want to do that? Good question! And now it's yours to tell me! If you use COMPUSEARCH there is a special interest Group (SIG) for TI home computers, with over 1000 members. These wonderful folks are constantly trying to out-do each other. By putting excellent Public Domain software into the Bees Load area of the Sig, think of it a Mail Box, and the files are NOT billed.

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

This also give you another way to edit existing files you may already have. Let's say you have a program, and you want to make a LOT of changes to it!

```
100 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
110 !! TRANSLATED FROM !!
120 !! DIS/VAR TO MERGE !!
130 !! !!
140 !! !!
150 !! !!
160 !! Use a "Full Screen" !!
170 !! Editor to create Ext- !!
180 !! ended Basic programs !!
190 !! !!
200 !! Create a file using TI- !!
210 !! Writer !! Make sure you !!
220 !! disable the word wrap !!
230 !! mode and limit the line !!
240 !! length to 80 characters !!
250 !! !!
260 !! !!
270 CALL CLEAR
First, you would simply load the program into memory, as usual, by entering "OLD DISK1.NAME". When loaded, you would then enter "LIST "DISK1.NAME"/". That would cause the program to be written to the disk in DIS/VAR 80 format. The reason for changing the name should be obvious to prevent over-writing the original file! Next, instead of filling through the entire program line by line, looking for variable #VZ you can use TI-WRITER or EDITOR/ASSEMBLER and do "Global" search and replace! Those features are described in the respective manuals. After you have modified the program as needed, you simply run the conversion program, which renames the file with a .MER name, and "MERGE "DISK1.PROGRAM"/", and proceed as before!
30000 SUB HOLD
30010 CALL KEY (S,K,S)
30020 IF S=0 THEN 30050
30030 CALL KEY (S,K,S)
30040 IF S=0 THEN 30090 ELSE 30030
30050 CALL KEY (I,K,S)
30060 IF S=0 THEN 30010
30070 CALL KEY (I,K,S)
30080 IF S<>0 THEN 30070
30090 SUBEND
The construction of the Magic Button was done with parts from a local "surplus" store. The button itself was an 8 pole double throw momentary contact switch with 3 poles cut off; the long base of the switch made an excellent handle. Almost any switch will do, and the remote switch on a cassette microphone will do very nicely. The diode is a general purpose low power type (1N914, 1N4001, etc.). The 50K plug is somewhat more difficult to find, but if you have a broken joystick you can use the plug from it. I used a 16 foot silver telephone cord because it is small and flexible. This is how the button is wired.
```

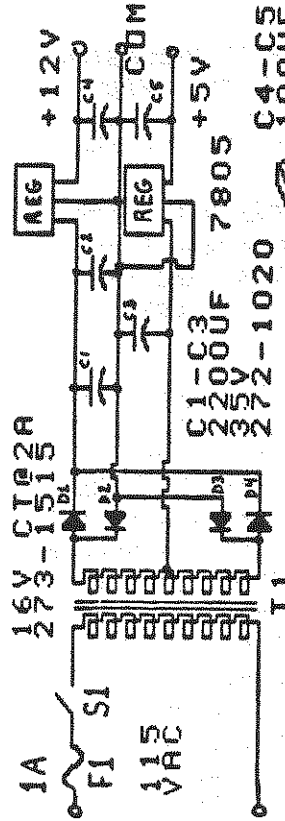
D8-9



I have enjoyed developing this technique for using the T199, and learning more about it. I hope others will continue to improve their T199 and find other uses for our "little plug" together we'll show Lubbock that what we ain't giving up yet!

Jim Hubbard, ASCUC

7812



ALL PART #'S ARE RADIO SHACK PART #'S I'M NOT SURE WHO AUTHORED THIS CIRCUIT.

- C4-C5 100UF
- 100UF
- 35V
- 272-1020
- D1-D4 COM
- 3A IN 5402 COM
- 276-1143 +5V

DISK DRIVE POWER SUPPLY

REMOTE CONTROL DISPLAY

As I began preparing for a music demonstration at an Atlanta User's Group meeting my program to think of coding using the T199 computer itself to help me present several lines of information stop for a moment. I then change any of the lines or clear the screen and present the information in a different color. Color changes could also be used to emphasize certain parts of the programs. I started out using a simple "press any key to continue" routine, and wrote my demonstration program to pause after each segment. This worked well, but it restricted me to the keyboard. So I created a "Magic Button" a simple pushbutton switch connected to the joystick port which provided me with room to move around. I could point to the display or execute an instruction with the press of a button, and then change the display or execute an instruction with the press of the "Magic Button". The button is electrically the same as the FIRE button of the joystick, and is read by CALL KEY(I,K,S).

I wrote a subroutine "MAGIC" (XBasic) to insert at end of each called program I used. Then I could insert the instruction "CALL MAGIC" anywhere in the program to pause, and discuss what was happening, or about to happen. The subroutine listed below will recognize any key, on the keyboard or the "Magic Button" (FIRE Button) and will wait until it is pressed. This prevents accidentally skipping short segments of program by holding the button too long.

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


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

BASIC

```

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

FORTH

```

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

ASSEMBLY (Assemble with the R option)

```

DEF START TYPE 'START' UNDER PROGRAM NAME
REF VSRW,VMBW,KSCAN,VWTR
STATUS EQU >B37C
KEYADR EQU >B374
KEYVAL EQU >B375
SPACE BYTE >20
TITLE TEXT " TYPEWRITER PROGRAM WRITTEN IN ASSEMBLY "
START MOV R11,R10 SAVE RETURN ADDRESS TO GO BACK TO E/A OR BASIC
LI R3,>F018 >F018 VDP REGISTER 1 VALUE
* WRITE TO VDP REGISTERS TO GET 40 COLUMN TEXT MODE
MOV R3,>B3D4 MOVE VALUE OF VDP REGISTER 1 TO >B3D4
LI R0,>01F0 VALUE TO BE PUT INTO REGISTER 1 >01 >F0 TEXT MODE VALUE
BLWP @VWTR WRITE THE VALUE TO REGISTER
LI R0,>07F0 >07 REGISTER 7 >F0 >F WHITE ON >0 BLACK
BLWP @VWTR WRITE THE VALUE
LI R0,>0401 SET CHARACTER TABLE SO EITHER EXB OR E/A CAN BE USED
BLWP @VWTR
* CLEAR SCREEN
CLR R0
LOOP MOV R0,SPACE,R1 LOAD R1 WITH SPACE CHARACTER
BLWP @VSBW WRITE THE SPACE
INC R0 ADD 1 TO ADDRESS
CI R0,960 IS THE ADDRESS 960? (ROW 24, COL 40)
JNE LOOP NO, JUMP BACK TO LOOP
* PUT TITLE ON TOP OF SCREEN
LI R0,0 COLUMN 1 (ADDRESS 0)
LI R1,TITLE GET ADDRESS OF FIRST LINE OF TEXT
LI R2,40 40 CHARACTERS TO WRITE
BLWP @VMBW WRITE IT
LI R7,B0 CURSOR POSITION ROW 3 COLUMN 1 (ADDRESS B0)
LI R1,0 KEYBOARD SCAN 0 (ENTIRE KEYBOARD)
MOVB R1,KEYADR ADDRESS USED TO HOLD KEYBOARD SCAN NUMBER
CLR R1
* MAIN KEY INPUT
AGAIN MOV R7,R0 LOAD REGISTER 0 WITH CURSOR POSITION
LI R1,>1E00 CURSOR CHARACTER
BLWP @VSBW WRITE THE CURSOR TO SCREEN
BLWP @SCAN GET KEY INPUT
MOVB @STATUS,@STATUS
JED AGAIN NO KEY PRESSED
MOVB @KEYVAL,R1 R1 KEY VALUE
CI R1,>0200 IS FCTN 4 PRESSED?
JED LEAVE YES, LEAVE PROGRAM
MOV R7,R0 CURSOR POSITION
BLWP @VSBW WRITE CHARACTER TO SCREEN
INC R7 INCREMENT CURSOR POSITION
CI R7,959 IS CURSOR AT BOTTOM OF SCREEN
JED LEAVE YES, LEAVE PROGRAM
JMP AGAIN NO, GO BACK AND CHECK FOR KEY PRESS
LI R0,>0400 RESET CHARACTER TABLE TO E/A OR BASIC
BLWP @VWTR WRITE TO REGISTER 4 WITH 0
MOV R10,R11 PUT RETURN ADDRESS BACK IN R11 TO RETURN
RT END RETURN TO BASIC OR E/A
    
```

PASCAL

```

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

The following program comes to us from Jim Peterson of Tigercub Software. Jim submitted the program in response to a challenge which appeared a few issue back. The challenge was to be able to list a program to the printer in condensed print while defeating the 80 character per line default. Well, Jim has done it again! The program requires that you have both extended basic and a disk drive. The program also requires that you 'list' the program to the disk, in order to create a DISKVAR 80 file. I know that the program will not work on a Prowriter, due to the variable line length in line 220. NOTE: The section of line 210 that reads 'PRINT CHR\$(15);CHR\$(27);CHR\$(70);CHR\$(3)' sets condensed print and skip over perforation at three lines from the bottom of the page. I wish the program would work with the Prowriter, and maybe someone can send me either correction or a new modified listing. The problem was using this program with a Prowriter, is that should the line exceed 136 characters (which is the maximum number of characters per line in condensed print on Prowriter) then you do not get a line feed, thus you get printing atop printing. However, the problem does not exist when using it with an Epson. I hope someone can send me corrected version. How about it Jim? Do you have an idea on how to make it work for the Prowriter?

```

100 ! 132 COLUMN LISTER
110 ! BY
120 ! Jim Peterson
130 ! of
140 ! TIGERCUB SOFTWARE
150 !
160 CALL CLEAR :: CALL SCREEN(5) :: FOR A=0 TO 14 ::
CALL COLOR(A,14,5) :: NEXT A :: DISPLAY AT(12,1) : "
CONDENSED PROGRAM LISTER"
170 DISPLAY AT(14,1) : " WRITTEN BY: Jim Peterson" : :
: " TIGERCUB SOFTWARE " : : " PRESS ANY KEY TO
BEGIN"
180 CALL KEY(0,0,C) :: IF C=0 THEN 180 ELSE CALL CLEAR
: : DISPLAY AT(2,1) : " CONDENSED PROGRAM LISTER" :
: " IN ORDER TO USE THIS PROGRAM"
190 DISPLAY AT(4,1) : " YOU SHOULD HAVE RESEQUENCED" :
: " YOUR PROGRAM (STARTING WITH " : " LINE 100 AND IN
INCREMENTS" : " OF 10) AND ALSO LISTED IT TO"
200 DISPLAY AT(14,1) : " THE DISK (WHICH WILL CREATE" : " A
DISKVAR 80 FILE). HAVE THE" : " TWO ABOVE
REQUIREMENTS BEEN" : " PERFORMED ? Y"
210 ACCEPT AT(20,13) : VALIDATE ("Y") : SIZE(-1) : A0 :: IF
A0="N" THEN CALL CLEAR :: END ELSE DISPLAY
AT(12,1) : ERASE ALL : " PLEASE ENTER THE FILENAME:"
220 DISPLAY AT(14,1) : " DSK1." :: ACCEPT
AT(14,6) : SIZE(10) : B0 :: OPEN #1 : " P10", VARIABLE 255
: : PRINT #1 : CHR$(15) ; CHR$(27) ; CHR$(70) ; CHR$(3)
230 OPEN #2 : " DSK1." : B00 :: D=100 :: LINPUT #2 : C0 ::
PRINT #1 : C0 ;
240 LINPUT #2 : C0 :: E=POS(C0, " ", 1) :: IF E=0 THEN PRINT
#1 : C0 ; : : GOTO 240 ELSE IF SED0(C0,1,E-1)
(C)STR$(D) THEN PRINT #1 : C0 ; : : GOTO 240
250 PRINT #1 : : PRINT #1 : C0 ; : : D=D+10
260 IF EOF(2) THEN CLOSE #1 :: CLOSE #2 :: END :: ELSE
GOTO 240
    
```

SPRITE ONE LINER
by Barron Bartlett

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

```

CALL CLEAR :: CALL SCREEN(5) :: CALL MAGNIFY(2) :: FOR I=1 TO 28 ::
CALL SPRITE(#1,64+I,14,80,80,3=I,8) :: NEXT I :: FOR J=1 TO 5000 ::
NEXT J
    
```

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

Whilst I was scrounging around at Radio Shack at Lindale Mall, I came across a very interesting item. It is a Coleco plug in power supply for the Adac computer, Radio Shack #277-1022. This power supply plugs right in the power socket on the wall, and has about a 4 foot connection cable. The end of the cable has the same mating connector that is presently on the back of our TI. Well, I had to see what this thing could do.

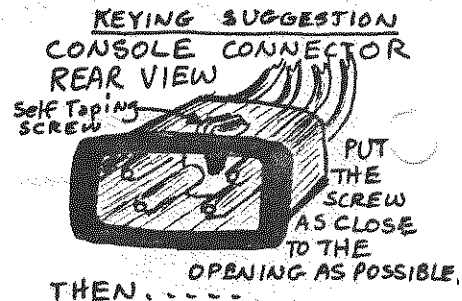
My thoughts are to completely remove the internal power supply from the console and use only this wall mounted unit. This would result in greatly reduced heating in the console, and more room for goodies to be installed. The only drawback is the off-on switch would be gone, and there isn't one on this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

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

EDITORS NOTE:

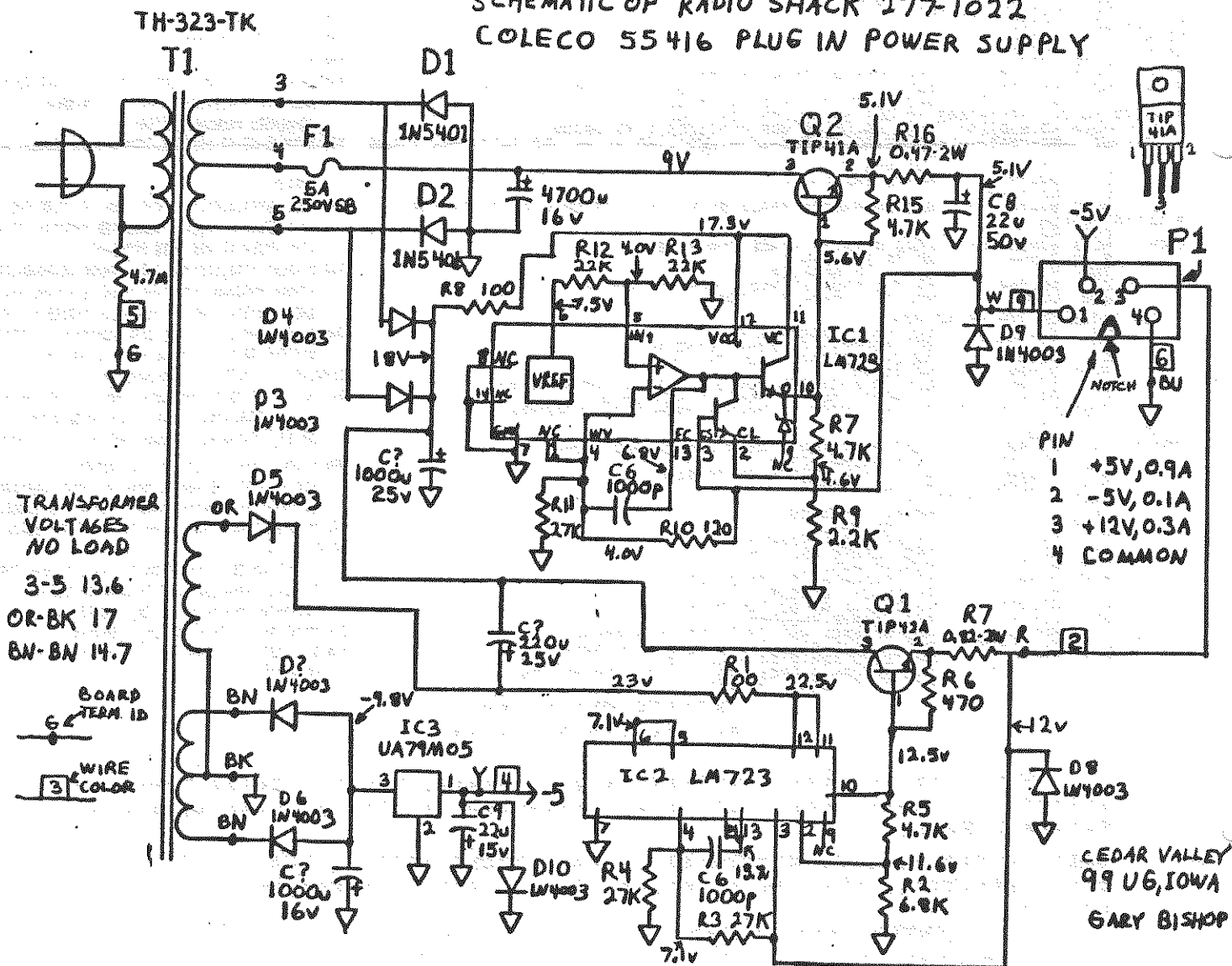
I have talked to Gary Bishop several times about this power supply since he actually put one into use. There are a couple things that should be noted, if you plan to go ahead with this project.

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



NOTCH THE PLUG FROM (P1) THE COLECO SUPPLY WITH A KNIFE SO THAT THE RUBBER CONNECTOR CAN SNUGGLY ENTER THE CONSOLE RECEPTACLE ABOVE. *PUT THE SCREW BETWEEN THE 2 WIDLY SPACED PINS* NOW. --- A standard T.I. Transformer power supply connector should NOT enter.

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



From LEHIGH 99'ER

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

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

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

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

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

Our friends at TI decided to put the whole thing in ROM, which has a few bad side-effects. First, it makes it hard to update and improve the software, which is located in the Disk Controller Card. Second, although the machine is a 64k machine, just like all the others, TI has set aside so much

memory for special purposes, that there is only 32k left to play with. They set aside 8k for cartridges, 4k for disk drives, 4k for RS232/P10 cards, 4k for the Operating System, (can't complain about that one), and 8k for various interfaces (speech, sound, VDP). Ok those are all good applications to have, but if you don't use them, you still can't use that memory for other things.

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

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

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

Wow, you say, 250k! That is about 25k bytes per second, right? How come I can not load a 25k pgm in one second then? Two reasons. First, as I said, the transfer of data is actually controlled by the ROM software in the TI99/4A. And to be as good as it is, it had to be a little bit slow. Not REAL slow (anyone ever use a 64k disk drive?), but not as fast as it

could be. The second reason also has to do with software, but it is a universal problem associated with single density storage.

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

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

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

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

```

10 REM *ZLOTTO-CHIG V4 #9
20 REM *(PRTR MANDATORY!!!)
30 REM *4/15/85 Rel 1.0
40 REM *AUTHORS:Jerry Novak
50 REM *and Jerry Tvrdik
60 REM *Set-up Menu
70 CACL CLEAR :: CALL SCREEN
(3)
80 DISPLAY AT(2,2):"Program
ZLOTTO:"
90 DISPLAY AT(3,9):" Generat
e 6 Random Numbers Per Gam
e Set-4/15/85 R1.0"
100 DISPLAY AT(7,5):"(Printe
r MANDATORY)"
110 DISPLAY AT(14,14):"By":T
AB(10):"Jerry Novak "
120 DISPLAY AT(16,10):"Jerry
Tvrdik"
121 DISPLAY AT(18,5):"CHICAG
O TI USERS GROUP"
130 DISPLAY AT(24,2)BEEP:"Pr
ess any Key to Continue"
140 CALL KEY(O,K,N):: IF N=O
THEN 140
145 CALL CLEAR
150 GOSUB 520
160 DEF R=INT(RND*44)+1
170 INPUT "-Number Games to
Play-?":A
180 REM *Your PRTR(PIO,ETC)
190 OPEN #1:"PIO"
200 PRINT #1:" -":A:"-":ZL
OTTO Cards"
210 PRINT #1:"-----
-----"
220 CLOSE #1
230 RANDOMIZE

```

```

240 DIM D(6)
250 FOR L=1 TO A
260 FOR M=1 TO 6
270 D(M)=R
280 FOR Z=1 TO M-1
290 IF D(Z)=D(M)THEN 270
300 NEXT Z
310 NEXT M
320 GOSUB 330 :: NEXT L :: E
ND
330 FOR I=1 TO 6
340 FOR J=1 TO I-1
350 IF D(I)>D(J)THEN 370
360 TEMP=D(I):: D(I)=D(J)::
D(J)=TEMP
370 NEXT J
380 NEXT I
390 FOR X=1 TO 6
400 PRINT D(X)
410 IF D(X)<10 THEN PRINT "
"
420 NEXT X
430 GOSUB 460
440 PRINT " " :: RETURN
450 REM *Your PRTR(PIO,ETC)
460 OPEN #1:"RS232:BA=1200"
470 FOR X=1 TO 6
480 PRINT #1:D(X)
490 IF D(X)<10 THEN PRINT #1
:" "
500 NEXT X :: CLOSE #1 :: RE
TURN
510 REM *Your PRTR(PIO,ETC)
520 OPEN #1:"RS232:BA=1200"
530 PRINT #1:"* Super ZLOTTO
Numbers *"
540 PRINT #1:"-----
-----"
550 CLOSE #1 :: RETURN

```

```

190 OPEN #1:"RS232:BA=1200"
460 OPEN #1:"RS232:BA=1200"
520 OPEN #1:"RS232:BA=1200"
CHANGE THE ABOVE 3 LINES
TO USE A PARELELL PRINTER
RUN THE ABOVE VERSION (RS232)
WITHOUT A PRINTER
AND THE NUMBERS
WILL PRINT ON THE SCREEN
BUT WILL SCROLL OFF
THE TOP OF THE SCREEN
IF YOU RUN MORE THAN 3 GAMES

```

SAMPLE PRINTOUT

* Super ZLOTTO Numbers *

- 6 -ZLOTTO Cards

5	17	30	31	37	38
2	9	16	19	23	44
15	22	23	25	33	43
7	10	21	22	33	41
4	10	13	14	36	43
1	10	14	15	34	43

File Transfers on IIBBS By John Pearce

File transfers is another way of saving upload/download and that is the subject of this article. There are three communications programs in general use in the TI-99 community: Terminal Emulator II cartridge, 4A/Talk, and Fast-Term. Each of these programs is capable of doing IZ2 protocol file transfers. The remainder of this article covers the essential points of doing file transfers with the IIBBS bulletin board using the IZ2 protocol.

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

Terminal Emulator II Cartridge

The first step is to select the communications configuration. For IIBBS, you may either choose the default option (menu choice 3) or you may select individual parameters (menu choice 2). If you take the latter choice, you may select either 110 or 300 baud, either even or odd parity, and select full duplex. To download a file, you will need either the file number from the download file list or a file name, such as TUTOR4. To initiate a download from IIBBS, you select D for download in the IIBBS File Transfer section (U for upload/download at IIBBS' main menu). IIBBS will ask for a file number. Enter the file number or file name and press ENTER. If you make a mistake while typing, use CTRL-H (backspace) to correct your error as FCIN-S will not generate a backspace character. IIBBS will then send information which initiates the transfer. After some preliminary information, the screen will clear and IZ2 will display the message HOST HAS STARTED FILE TRANSFER. After a short delay, the message PLEASE ENTER DEVICE NAME TO OUTPUT DATA ID is displayed. The response to this is DSK1, DSK2 or DSK3 (without a period) then press ENTER. The next message is PLEASE ENTER FILE NAME TO OUTPUT DATA ID. This is the file name (up to 10 characters) of the file you are downloading. Type in the file name and press ENTER. The advisory message FILE TRANSFER IN PROGRESS is displayed along with record count, block count, and retry count. When the transfer is finished, the message SUCCESSFUL FILE TRANSFER is displayed. The screen is cleared and IIBBS will display the File Transfer prompt. If the file transfer is aborted, IZ2 will display such a message.

To upload a file to IIBBS, select U for upload. IIBBS will request a file name for the file to be uploaded. Type in a file name up to 9 characters long and press ENTER. IIBBS will clear the screen and display a copyright notice. You must press CTRL-Y to start the transfer. IZ2 will display the message ENTER DEVICE NAME TO TRANSMIT DATA FROM. This should be DSK1, DSK2 or DSK3 (again without the period). Then the message PLEASE ENTER FILE NAME TO TRANSMIT DATA FROM is displayed. Type in the file name you wish to upload. The transfer will be initiated and the status information will be displayed as for the download. When the transfer is completed, the SUCCESSFUL FILE TRANSFER message is displayed. If it is necessary to abort either upload or download, use CTRL-3. After aborting a transfer, it may be necessary to press ENTER to force IIBBS to display the file transfer prompt. There are three general reasons why a transfer would be aborted: the caller pressed CTRL-3, there was a disk error, or the disk was full. The last two errors can occur at either the caller's machine or at IIBBS. IIBBS will usually not permit an upload to start if there is less than 15 sectors available on the upload disk drive.

4A/Talk

Configuring 4A/Talk is straight forward. You may select either 110, 300 or 1200 bits per second for baud rate (depending on your modem). The parity selection must be odd parity (PA-O) and data bits must be 7 (DA-7). An example might look like "R5232-PA-O-DA-7" for a 300 baud modem on port 1. Unlike IZ2, it is necessary to specify odd parity for communicating with IIBBS. The entries to and responses from IIBBS are the same for 4A/Talk and Fast-Term as for Terminal Emulator II. To initiate a download from IIBBS, you select D for download. Enter the file number or file name and press ENTER when requested. If you make a mistake while typing, you

may use CTRL-H (backspace) or CTRL-S to correct your errors. 4A/Talk generates a backspace character when FCIN-S is pressed. IIBBS will then send the command which initiates the transfer. After some preliminary information, the screen will clear and the IZ2 file transfer status display will be shown. The central portion of the screen looks like this:

Filename: DSK1.

Retries:

Record Number: 0

Sectors Remaining: 0

The cursor is positioned at DSK1 waiting for you to enter the name of the file. You may change the 1 to 2, 3 or 4 as needed. The Record Number counts from 1 to 5 while the Sectors Remaining starts with the sector size of the file and counts backward to zero. When the transfer is complete, a success message is displayed at the top of the screen. To upload a file to IIBBS, select U for upload. When requested, type in a file name up to 9 characters long and press ENTER. IIBBS will clear the screen and display a copyright notice. You press CTRL-1 to select the IZ2 transfer menu then select choice 1 - send a file. The IZ2 file transfer status display (as above) will be shown. Enter the file name to upload and press enter. The file transfer will be initiated. The Record Number will count from 1 to 5 as with the download and the Sectors Remaining will count down to zero. When the upload is completed, a success message will be displayed at the top of the screen. To abort either upload or download, press CTRL-X.

Fast-Term

The configuration of Fast-Term can be easier than either the IZ2 cartridge or 4A/Talk. You can create a separate configuration file for IIBBS and specify that file when the program starts. Again, you must specify odd parity. Fast-Term will automatically set 7 data bits. You must also enable the IZ2 protocol by pressing FCIN-Shift-I all at the same time. If done correctly, the message "IZ2 Protocol On" will be displayed on the screen. To download a file from IIBBS, you should first press FCIN-N and enter the name of the file. Then enter D for download to IIBBS. Enter the name of the file when IIBBS requests it and press ENTER. Fast-Term displays a small file transfer status block similar to that displayed by the IZ2 cartridge. The status block shows the record count, block count, and retry count. When the transfer completes successfully, you will hear the chimes and a completion message will be displayed. The reason for entering the file name with FCIN-N before entering the D for download to IIBBS is to avoid having IIBBS "time-out" and abort the transfer. To upload a file to IIBBS, select U for upload and enter the file name when requested by IIBBS. Then press FCIN-N and enter the name of the file to be uploaded and press ENTER. Then press FCIN-, (comma) to begin the transfer. The file transfer status block will be displayed showing the record, block, and retry counts. When the transfer completes successfully, you will hear the chimes and a success message will be displayed. The reason for entering the file name (FCIN-N) after giving IIBBS the file name is to avoid starting the file transfer (FCIN-) before IIBBS is ready. After receiving the file name, IIBBS does 3 to 5 seconds worth of housekeeping to get ready for the upload. If the file transfer initiation message arrives from Fast-Term while IIBBS is still doing housekeeping, the message will be lost. To recover, abort the transfer (FCIN-Y) and start again.

Summary

The IIBBS procedures and displays are the same regardless of which communications program you use. However, the procedures with the uploads and downloads. There is no great mystique to doing selecting odd parity and enabling IZ2 in Fast-Term. In both 4A/Talk and Fast-Term you must select odd parity. The parity selection in the IZ2 cartridge seemingly makes no difference. In Fast-Term you must enable the IZ2 protocol before initiating the transfer.

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

Fixing Blown disks
by Terry Atkinson

No. NJ 99'er User's Group Newsletter

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

Fixing the disk bit map-(AUG).

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

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

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

Another possibility is that 50 has been damaged, perhaps by magnetism or a scratch on the surface. In this case, you'll quickly find out when you try to read/write sector 0. You won't be able to. Now you have a problem, but not insurmountable. The only "fix" for this is to copy all sectors from the bad disk to a good disk, sector-by-sector. A tedious chore to be sure, but at least you can get all your programs back. It will still be necessary to proceed as above to get your programs back, as the bit map on the new disk will not be correct. Now, I am not sure how FORTH would behave under this circumstance. I know FORTH will "choke" when it tries to copy a damaged sector, but whether or not it will continue to copy the "good" sectors and put them into their proper places on the new disk, is beyond me. I wouldn't chance it. Better to be safe than sorry, and stick to tried and proven methods. Of course, you could experiment! If it works, let us all know, if some of you FORTH addicts out there could shed some light on the subject, your comments would be most welcome.

Fixing the Directory Link Map (DUI).

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

First, look at AUG. Read the bit map to determine which sectors between 2 and 33 inclusive (02-21) are flagged as used. Make a list of these sectors in a column. Now, load each of these sectors in turn, and examine the first 10 bytes of each. Copy the bytes down beside the relevant used sector. Determine the alphabetical order of these programs merely by reading the numerical values. The lower the number, the closer to the front of the alphabet it is. Now, produce a list of these sectors arranged alphabetically. Here's a short example:

Sector used	Hex Values in 1st 10 bytes ()	Program Name
2	4B 20 20 20 20 20 20 20 20 20	K
3	49 20 20 20 20 20 20 20 20 20	I
5	4C 20 20 20 20 20 20 20 20 20	L
6	41 20 20 20 20 20 20 20 20 20	A
9	41 42 20 20 20 20 20 20 20 20	AB

Re-arranging the above alphabetically by sector would produce 6,A,3,2,5 which are going to form the directory link map in WORD
Next, copy sector 1 from ANY freshly initialized disk and write it to the bad disk. This is the easiest way to "restore" S1 to all zero's. Now, use the (Alter) command, and change the first, and each successive word to produce the alphabetical pointers. For example:
0005 000A 0003 0002 0005 0000

Note the 0000 at the end. The directory link map must be terminated with this value. Now, write this sector to the bad disk, and you're in business.

Retrieving an accidentally 'deleted' program

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

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

Locate the sector containing the deleted file's directory (between 02-21). You can do this by using the "FIND STRING" command, or, if you disk fiker does not have this command, merely load them in one at a time and look for your "deleted" program's name in the first 10 bytes. Change the program name to "ZZZZZZZZ" (HEX code of course). Now write that sector back to it's proper spot. Load in sector 1 and locate the first word containing 0000 and replace it with the directory sector # of your deleted program. Ensure the next word is 0000. Now, exit the DF and load the subject program as per normal. Exit the disk-fiker, and load the program as normal and save it BACK to the same disk under the same program name (ZZZZZZZZZ). Why? Because this will automatically update the disk bit map (AUG). Now use DM2 to change the program name back to it's original name and the task is complete. You have recovered your "lost" program.

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

31996 I PROGRAM TO EXTRACT
LINES FROM ONE PROGRAM TO BE
INCLUDED IN ANOTHER PROG
RAM

31997 I ELIMINATES UNWANTED
LINES OR SEGREGATES PARTS OF
ONE PROGRAM TO REMOVE

31998 I ROUTINES OR TO SAVE
31999 I REKEYING ROUTINES
32000 CALL CLEAR : CALL INI
T : INPUT "Line numbers of
routine to be saved: F
first, last? " :L,M : G=2
56 : CALL PEEK(-31952,H,I,J
,K)

32001 C-INT(M/G): D=M-C+G :
K=(J-G)*G+K : FOR E=(H-G)
+G+1 TO F STEP 4 : CALL
PEEK(E,A,B): IF A=C AND B=
D THEN 32003

32002 NEXT E : PRINT "Line
" :0!"not found!" : STOP !0P

32003 H=INT(E/G): I=E-(G*XH)
: H=M+G : C=INT(L/G): D=L
-C+G : FOR E=E+4 TO F S
TEP 4 : CALL PEEK(E,A,B):
IF A=C AND B=D THEN 32005 :0
P-

32004 NEXT E : PRINT "Line
" :N!"not found!" : STOP !0P

32005 E=E+3 : J=INT(E/G):
K=(G*XJ): J=J+G : CALL LO
AD(-31952,H,I,J,K): STO
P !0P-
32006 !0P-

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

Some of you may not be aware that you can use your cassette recorder to do more than just load or save programs. Your cassette can also store data files which can be read into the console by a running program, modified by the user, and saved for later reference. By learning to use the basic commands, OPEN #, INPUT #, PRINT #, and CLOSE #, you can open up new horizons with your TI-99/4A by being able to save and recall data from cassette. One important point to get clear first is the concept of "BUFFERS". The word "BUFFER" is used to describe an area of computer memory (or hardware) that is used to temporarily store data that is written into and out of the computer.

Buffers are required whenever the computer must talk or listen to another device which does not operate at the same speed or in the same manner as the computer does. For example, information to the processor, the keyboard on your machine uses a buffer to pass computer speeds; consequently the computer must use a buffer to transfer information to the device. Briefly, a buffer is a block of memory of fixed size which is compatible with the output device. When the buffer is emptied, more data is written into it until the data transfer is complete. An important point to realize is that the transfer of data from the buffer to the external device is done automatically only if the buffer is full. If the buffer is only partially loaded when the application program ends, this data could be lost unless you instruct the system to close all open files (buffers). This will cause the system to finish dumping the buffer to cassette. The last data item is always an END OF FILE (EOF) marker.

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

The buffers have a numerical tag. In TI BASIC or EXTENDED BASIC, you can specify a tag from 1 to 255 with each buffer being distinguished from others by the tag number. Buffer number 0 is reserved for system use and is, in fact, the keyboard (and screen) buffer. You can use more than one buffer at a time for different purposes, however the number of buffers that are open at the same time is limited to a default of three (3). If you need more than 3 buffers open, use the CALL FILES(n) command, where (n) is any number from one to nine. NOTE: This will limit you to 9 open files or buffers at a time. The CALL FILES command must be used in the following manner:

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

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

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

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

OPEN #n - This command prepares the system to transfer data to an accessory device. The buffer number (n) (the TI manual calls buffers "FILES") is specified by you as well as the device name (such as CSI) to which the data is to be written to or read from. Additionally, you must specify the structure of the data file to be written on the cassette. Until you have become thoroughly familiar with the TI User's Reference Guide and you have gained some experience working with cassette files, always specify "CSI", INTERNAL, SEQUENTIAL, FIXED for your file structure. Furthermore, you must tell the system the size of the data string to be written (so it will know how to read the data back later) by placing 64, 128, or 192 after the FIXED notation. You must plan the maximum length of each data item to be stored. If you choose FIXED 64 in the OPEN # statement and then write a data statement 70 characters long, the last 6 characters would either be lost or would overflow into the next character string, producing an unwanted concatenation or a "trashed file". On the other hand, if your string was only 60 characters long, the system would automatically pad the string to 64 characters with blank characters, which are removed when the data is recalled.

PRINT #n - This causes the system to transfer (print) data FROM the computer TO the device identified by and in the format specified by the OPEN # statement whose buffer number corresponds.

CLOSE #n - This statement will cause the computer to empty the specified buffer number (n) of pending data by completing the transfer sequence. Treat OPEN # and CLOSE # statements like matched bookends. Do not place any statements between that would cause a transfer of the program control out of the program block defined by these two statements. If you experience a program error message during a file transfer sequence, DO NOT USE FCTN QUIT !!!!! This will cause all data in the buffers to be lost. Instead type BYE, RUN, OLD, SAVE, LIST or else EDIT a line number. Any of these actions will cause the buffer to close properly.

EOF

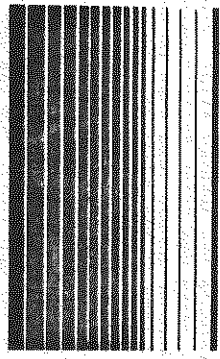
YE OLDE' TEST PATTERN
by H MacDonald

```

While programming the other day I
came to a part I wanted to print to
the screen. I've set up many screen
formats and am sick of blue or light
green. So I whipped out the QUICK
REFERENCE CARD and opted #14 -
Magenta. I finished the program and
ran it just as my wife came into the
room. "Oh what a pretty shade of
red!", she said. "No, that's not
red, that's Magenta!", I said. She
said something like "mumble mumble",
and left. So I looked Magenta up
and found out it is really a shade
of purple. (She told me later she
thought Magenta was red.) So the
following program prints a Test
Pattern like my Dear Ol' Dad used to
use in the TV Repair Shoppe when I
was just a child in 1955 BC (before
color).
I tried just putting it into a loop
but the colors are not lined up in
any way to compare them properly.
Also, if you want to use inbasic,
just start a new line every time you
see ::. Run the program and use the
test pattern to adjust the color
"HLJE" and "INTENSITY" controls on
your TV or monitor. You too can
have purple magenta or red. Orange
red is nice too, olive green magenta
is not so nice.
100 CALL SCREEN(16)
110 CALL CLEAR
120 A=1
130 CALL COLOR(0,4,4)::CALL
VCHAR(1,A+1,31,48)
140 CALL COLOR(1,3,3)::CALL
VCHAR(1,A+3,39,48)
150 CALL COLOR(2,13,13)::CALL
VCHAR(1,A+5,47,48)
160 CALL COLOR(3,10,10)::CALL
VCHAR(1,A+7,55,48)
170 CALL COLOR(4,9,9)::CALL
VCHAR(1,A+9,63,48)
180 CALL COLOR(5,7,7)::CALL
VCHAR(1,A+11,71,48)
190 CALL COLOR(6,8,8)::CALL
VCHAR(1,A+13,79,48)
200 CALL COLOR(7,6,6)::CALL
VCHAR(1,A+15,87,48)
210 CALL COLOR(8,5,5)::CALL
VCHAR(1,A+17,95,48)
220 CALL COLOR(9,2,2)::CALL
VCHAR(1,A+19,103,48)
230 CALL COLOR(10,15,15)::CALL
VCHAR(1,A+21,111,48)
240 CALL COLOR(11,16,16)::CALL
VCHAR(1,A+23,119,48)
250 CALL COLOR(12,14,14)::CALL
VCHAR(1,A+25,127,48)
260 CALL COLOR(13,12,12)::CALL
VCHAR(1,A+27,135,48)
270 CALL COLOR(14,11,11)::CALL
VCHAR(1,A+29,143,48)
280 GOTO 280

```

Mac



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

```

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

```

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

***** (from MID ILLINOIS') *****



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



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

I-BASIC	E-EDITOR/ASM	DESCRIPTION	M-MINIMASY	CARRIDGE	T-TI-WRITER	PRICE	CHECK
1)	PUB WRITER	TI-WRITER LOADER & FILES	KB			\$10	
2)	UPDATES	TI-WRITER AND MULTIPLAN LANGUAGE AND MANUAL	T,MP			\$5	
3)	TI-FORTH	SEVERAL GOOD FORTH PROGRAMS	E,M			\$15	
4)	FORTH SHAPES	ARCADIA GAME WITH 50 SCREENS	E & FORTH			\$10	
5)	TI-RUNNER	COMPILES I-BASIC SUBROUTINES	E,M			\$10	
6)	S-BASIC COMPILER	ASSEMBLY "BREAKOUT GAME"	X			\$10	
7)	TILE BREAKER	ASMB BOXING GAME(2 PLAYERS)	X,E,M			\$10	
8)	BOXER	ASSEMBLY SPACE BATTLE GAME	X,E,M			\$10	
9)	D-STATION II	CONTINUATION OF D-STATION	X,E,M			\$10	
10)	STARBAZER I	MAPS COMTELLATIONS	X,E,M			\$10	
11)	SCRABBLE	ASMB COLOR + 20,000 WORDS	E,T			\$10	
12)	MASS COPY 1 & 2	DISK COPIERS	X,E,M			\$10	
13)	TEXT	TERMINAL EMULATOR	E			\$10	
	FAST TERM	BEST TERMINAL EMULATOR	E			\$10	
	DM1000V2.0	BEST DIK MANAGER	E			\$10	
	DM1000V1.0	DIK MANAGER	E			\$10	
14)	SUPER DEBUGGER	TERMINAL EMULATOR	X			\$10	
	TI-DIG-ASMB	ADVANCED DISASSEMBLER	E,T			\$10	
	MARTY'S CATLDER	TI-A DISASSEMBLER	E,M			\$10	
	MARTY'S DISBRN	SEE OCT NEWSLETTER	E,M			\$10	
	TICK	BY MARTY KRULL JR	E			\$10	
15)	MASS TRANSFER	INTERLUPT DRIVEN CLOCK	X			\$10	
	DISK02	ASMB FILE TRANSFER UTILITY	X			\$10	
	COMPACTOR	DISK SECTOR EDITOR	X			\$10	
	UNCOMPACTOR	COMPACTS DIS/FIX 90 FILES	E			\$10	
	DISASSEMBLER	YET ANOTHER DISASSEMBLER	E			\$10	
16)	The disk manager	Good DM program	E			\$10	
	DM1000V3	Updated version	E			\$10	
	Fast-Term	Terminal Emulator	X,E			\$10	
	Fast-Docs	Checks for fast-term	E			\$10	
17)	Diagnosical	Diagnoses P-cards	M			\$10	
	Diagnosical2	Checks console	X			\$10	
	TI-runner edit	edit ti-run screens	X			\$10	
	Breakout	assembly game	X			\$10	
	File Transfer	Transfer Adventures	E			\$10	
18)	Clydes Loader	Load assembly in Ibasic	X			\$10	
	Spotshot	Assembly Game	E,X			\$10	
	Cubit	Cubit Assembly game	X			\$10	
19)	Funnel	Menu driven program that has a bunch of utilities like copiers,des and others. Must have!	X			\$10	
20)	Printer disk	PRINTS PICTURES(3 DISK SET)	X & PRINTER			\$10	
21)	OLD DARK CAVES & MYSTERY DISK #3	Printed Manual	X			\$10	
22)	Road Race	all assembly	E			\$10	
23)	Burger Builder	Assembly Pole Position	X			\$10	
24)	MYSTERY DIK	Assembly Burgerline clone	X			\$10	
25)	Midnight Mason	COMBO OF FORTH AND ASMB DBDD Assembly Game	X & E			\$10	

26)	Micro Pinball						
27)	Monopoly						
28)	Atlreids						
29)	Arthropod						
30)	Beyond Space						
	Face Chase						
31)	Great Word Race						
32)	Star Trap in 3D						
33)	C-Language						
34)	TI-pilot						
35)	Multiplan Tips						
36)	Martys Disasmb						
37)	Martys Cataloger						
38)	Clydes Loader						
39)	Super Bug II						
40)	Screen Dump						
41)	Neatlist						
42)	Pr-Shes						
43)	Tackle 888						
	TI-Writer Loaders						
44)	DOM Jan 1986						
45)	DOM FEB 1986						
46)	Dom March 1986						
47)	Spies Adventure						
48)	Salactic Battle						
49)	TI-Tod						
50)	Jet Graphics						
51)	Universal Disassembler						
52)	Enhanced FDRch						
53)	Mystery Disk #2						
54)	DOM 4/86						
55)	New Membership						
56)	Newsletter Subscription only						
	subtotal of order items						
	Amount enclosed						
	FREE MYSTERY DISK #4 with every \$35 purchase. (Assembly)						
	Please print neatly.						

Name _____ Date / /86
 Address _____ PH() _____
 CITY _____ State _____ Zip _____
 Orders processed as soon as possible, usually within 3 days. If you get a bad copy OF ANY PROGRAM, CALL ME!!! DON'T POUT, CALL CALL!
 Mail to Sapphire Software
 P.O. Box 18124
 Pgh PA 15236
 ALL ITEMS EXCEPT THOSE WITH AN ** AFTER THEM MAY BE ORDER BY ANYBODY, ANYWHERE.
 Thank you for your order!

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

DISK DRIVES

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


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

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

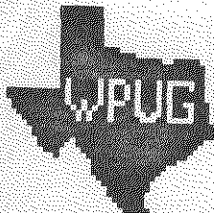
THERE ARE MORE, THAN THESE OF COURSE, BUT IN GENERAL THOSE THAT ARE COMPATIBLE WITH APPLE, COMODORE AND ATARI, WILL NOT WORK ON THE TI-99/4A. THOSE THAT WILL WORK ON THE P.C.'S, ARE GOOD. (EXCEPTION WOULD BE THOSE THAT ARE 96 T.P.I., OR SAY 80 TRACK DRIVES).
6000 HUNTING!!!!

REWRITTEN AND ADDED TO BY JOHN WILLFORTH WPUG.

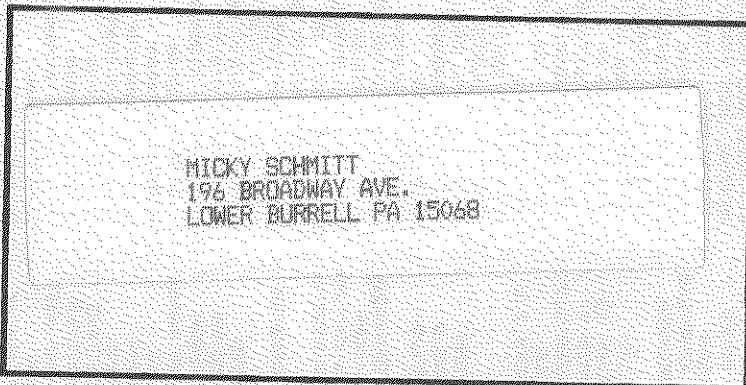
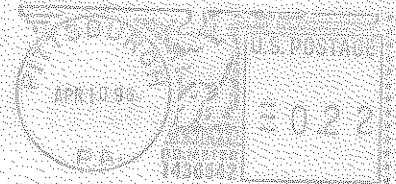
Your
BEST FRIEND KNOWS...



THAT THE WEST PENN 99'ers IS THE BEST THING THAT CAN HAPPEN TO YOU AND YOUR TI-99/4A. SO IF YOU NEED A GOOD FRIEND, CALL 412 527-6656 TO INQUIRE OR JOIN. ASK FOR JOHN WILLFORTH.



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



WEST PENN 99ERS

