VOLUME 1 ISSUE 6 $1.50 FEBRUARY 1985

SUPER 99 MONTHLY

HARDWARE

We are very pleased to offer the article that follows. Dr. Szymkiewicz is very skilled in computer hardware and writing. If this is the type of information you are interested in, please be sure to let us know, as we have been receiving a number of submissions for hardware construction projects.

WARNING: All hardware construction projects are offered on the premise that the reader who wishes to undertake such projects is competent in evaluating the instructions offered and in building electronic hardware, including being aware of the potential risks involved, such as voiding of warranties, potentially damaging existing components, and the routine dangers of working with electrical gear. We offer no assurances of the accuracy of any article or the usability of the hardware described.

Adding a Numerical Keypad to the 99/4A

by Steven Szymkiewicz, MD
Clairton, PA

STANDARD: 1A

One of the deficiencies in the TI-99/4A console, when compared to more expensive computers, is the lack of a separate special function and numerical keypad for quick data input. This lack can easily be corrected without special software by connecting a keypad in tandem with the keyboard. The cost can be kept within the $10 to $15 range, and the project entails a minimum of soldering.

The materials used in the project and their costs are in Table 1. These were obtained from Jameco Electronics, and Radio Shack also carries them except for the keypads. Any SPST mechanical switch keyboard can be used for the keypad in this project.

The first step in connecting an additional keypad is to bring the keyboard lines outside of the console. Doing so, of course, will void the computer's warranty (if yours is still valid). If that is no obstacle, then turn the computer off, unplug all electrical connections, and flip it over. Remove the seven screws in the console bottom and lift the bottom off (in older versions, the on-off switch will have to be pulled out first). Inside, the computer is divided into a keyboard, a power supply and a main logic board. The easiest place to -->

connect lines for a tandem keypad is where the keyboard lines plug into the main logic board. For better access, remove the logic board by unscrewing the retaining screws near the I/O port, the joystick port, and the center of the back part. Unplug the keyboard and the power supply before lifting it out.

Fifteen posts rise from the logic board where the keyboard connects to it. Note that the posts are numbered. Strip a small amount of insulation from each wire of the 15 wire ribbon and wrap a wire tightly to each post. Carefully solder these attachments. Remember the console keyboard must plug back in, so keep the placement as low on the posts as possible. Trim off any excess solder and/or wire strands to avoid the possibility of an unwanted line to line contact. Next, connect the power supply and keyboard, and replace and secure the logic board inside the console.

The wire ribbon can be brought directly out through the ventilation area on the bottom. Since stress will then occur on the logic board if the ribbon is tugged, a better solution is to attach the ribbon wire to the console. Remove a portion of a bar in the ventilation area of the console's bottom and file it large enough to accomodate one of the 16 pin DIP sockets. Then glue a scrap piece of perfboard over the gap and install the DIP socket. The socket should be just about flush with the console. Solder the ribbon wires to the socket pins. Using the numbering from the posts of the logic board, keep track of which wire is connected to which socket pin (write it down!). The console can be reassembled as no further modification is needed. Now if the keypad is dropped, the stress will be on the socket and not the logic board.

The ALPS keyboard needs modification before use, as only the center portion is needed (a smaller keypad is less clumsy). Desolder and remove all the keyswitches, then desolder and remove the sliding switches. Using a hacksaw, cut along the holes to the left of where the large zero key was and to the right of where the large plus key was. Cut both the metal frame and the etched board. The alternative keypad listed in Table 1 will not need the above changes, but it is more expensive.

At this point, the enclosure should be prepared. The back of the enclosure will need an opening for the second DIP socket and a hole on each side for a nut and bolt to hold the second piece of perfboard to it. The top of the enclosure needs an opening for the keys to come through and two holes on each side to attach the keypad. For the ALPS board, cut a 2 7/8" x 4" opening for the keys and drill the holes about 1/4" from the opening. When drilling the holes, drill through the metal frame at the same time to ensure that they will line up. Then cut out the socket opening and drill holes for the bolts. Next, attach the second DIP socket to the second piece of perfboard and bolt the perfboard to the back of the enclosure. Then bolt the metal frame to the enclosure top, replace the keys, and resolder the keys to the pads on the etched board.

The TI uses a 7 x 8 matrix for keyboard encoding. The connection of a row to a column line allows the computer to determine which key was pressed. The matrix is shown in Table 2. The line numbers in the table match the line numbers within the computer. The ALPS keyboard does not follow TI's convention, so the etched lines must be cut from the keyswitch pads. Make new connections following TI's convention using bell wire, and attach the appropriate lines to the second DIP socket pins. Make sure the line numbers from both DIP sockets match or the keypad will not function as expected. Table 3 has a useful wiring arrangement, although others can be used.

Essentially, that is it. Use the DIP jumper cable to connect it to the console and try it out. If characters appear when no key is pressed, there is a short somewhere. If the wrong

-->

characters appear, the lines are simply not connected correctly or the DIP jumper is connected upside down. When it works properly, screw the enclosure together and use the finished product.

Table 1: Materials

Jameco products:		
15 wire ribbon (6")	$ .54	171-15
16 pin DIP sockets (2)	.38	16 pin LP
16 pin DIP jumper (24")	3.29	DJ16-2-16
ALPS keyboard	2.49	KB297040
alternative keyboard:		
Hexadecimal keyboard	$14.95	K19
Radio Shack product:		
Enclosure	$3.95	270-282

Perfboard scraps 2" x 2" and 1" x 2"
Bell wire
Small nuts and bolts (6 pair)

Table 2: Key Matrix (TI)

Line on outside, Key on inside

Line	6	8	9	12	13	14	15
1		P	Y	ENTER	0	I	U
2		0	6		9	8	7
3		A	G	Shift	S	D	F
4		;	H	Space	L	K	J
5		/	N	=	.	,	M
7	Alpha Lock	1	5	Funct	2	3	4
10		Q	T	Cntrl	W	E	R
11		Z	B		X	C	V

Table 3: Wiring and Key Arrangement

Line numbers are in the upper half
Key characters are in the lower half

10 14	2 15	2 14	2 13		
E	7	8	9		
11 13	7 15	7 9	2 9		
X	4	5	6	1 12	
2 8	7 8	7 13	7 14		
0	1	2	3	ENTER	
7	12	3 13	3 14		
FUNCTION	S	D			

E,X,S,D are for cursor control

Expansion Decisions

STANDARD: 1A 5A

Here is a question we recently received from LTC Chris C. Agrafiotis, of Concord, NH:

"...there seems to be a great deal of controversy surrounding the new Foundation Z80A Card and the accompanying 80 Column Card. The big question in my mind is ... of what practical use is it to me at this time to invest $500 more in my TI? What can I do with these cards alone without spending large sums of money in new software? Is it necessary to have a "true" monitor or can a "composite" do the job? and on, and on ...."

These are certainly some good questions that are on the minds of many TI-99/4A owners. We'll answer in two parts. We'll cover the fundamental issues this month and the details next month.

One major consideration is how long you anticipate using the product you are considering purchasing.
-->

Consider using a business approach, dividing the cost by the anticipated life to yield an amortized monthly cost, then ask yourself whether that figure is reasonable considering the benefits to be derived from the product and your current financial position. While we cannot anticipate how long you will be using your TI-99/4A, we can point out some factors that might cause you to over- or under-estimate your continued use of your TI-99/4A.

As you know by now, your TI-99/4A (along with all accompanying products) will still power up every day even though Texas Instruments is no longer active in the Home Computer market. However, for expanded capabilities, non-programmers obviously must depend on the continuation of the current large base of users that is keeping software (and hardware) producers in business (while a number of users abandoned the TI-99/4A shortly after TI departed the Home Computer field, our sources consistently report a very solid base of users at this time). Non-programmers must therefore be cautious in their purchases, while programmers can take a more liberal approach. This is one of the major reasons why we place such a strong emphasis on programming skills.

You should also consider your complete system plans, based on factors such as your applications and the language(s) you prefer. Also, keep in mind that there are more products in the works. According to preliminary information we have received, within the next few months there will likely be four or more memory expansion cards available that will provide memory in excess of 32K, with some sporting up to 512K of memory and some supporting RAMDISK (RAM that emulates, or acts like, a disk drive, making some applications much faster). The new memory capabilities should eliminate the only remaining significant complaint about using the TI-99/4A for very serious processing. As an example of planning your configuration, an Assembly programmer with significant word processing requirements would likely want an 80 Column Card and 512K. On the other hand, owners who use mostly commercial game packages would likely be very satisfied with a standard TI configuration.

Should a "compatible" become available (yes, unconfirmed rumors are still flying), expansion of your TI-99/4A would be greatly enhanced. Because of the upward compatibility of machine code for TI processors (this does not apply to the TI Professional computer, which uses an Intel processor), there would be a lot of transportable software, which would yield benefits both to the compatible owners and the TI-99/4A users. Peripherals may or may not be compatible, but sharing files would be no problem for owners of expanded systems. Additionally, we see the possibility that if one compatible comes along, more could follow. In short, the effective life of your TI-99/4A would be greatly extended by a compatible.

As for the cost of software for a card, the cost will depend primarily on the number of cards sold and the application for the specific card. The Foundation Z80A Card and a card produced by Morningstar are both designed for CP/M™ usage (note that there are several popular versions of CP/M™). A large base of software for CP/M™ already exists, having been developed for use on other computers. Many public domain programs exist and there are numerous publications on the market with programs ready to be keyed in. Therefore, software for the Z80A Card can be expected to initially be much more readily available and at a lower cost than software for the 80 Column Card, for example. Of course, another way to save money on software is to write your own.

Next month we'll get into the specific claims of Morningstar and Foundation.

-->

Using Strings

STANDARD: 1A 9A

In computer jargon, "strings" are simply sentences, words, letters, or other characters. Strings are very useful in writing programs. Strings can be stored in string variables. The name of a string variable always ends with the "$". Here is a BASIC program that sets up a string as a person's name and then PRINT's the string on the screen:

```
>100 A$="BILL JONES"
>110 PRINT A$
```

As you can see, using the string variable used fewer keystrokes than using the entire name in line 110. There are many other advantages to be gained in using string variables.

Strings can be introduced into your program in several ways: through use of LINPUT (Extended BASIC only) or INPUT from diskette, cassette tape or the screen, from DATA statements, or directly by use of LET (line 100 above uses an implied LET. The full statement is LET A$="BILL JONES").

On page II-99 of your TI BASIC Reference Manual, you will find a section titled "Built-In String Functions", which references 7 string functions available in console BASIC. By using the string functions and some other tools available, strings can be manipulated for many useful purposes.

Strings can be combined by using the "&" (concatenate) operator. Here is a simple program example:

```
>100 A$="BILL"
>110 B$="JONES"
>120 PRINT A$&" "&B$
```

When you RUN the program, you will see "BILL JONES" on the screen.

Now comes the really fun part. Now that you understand the basics, you can get started in doing almost anything you want with a string.

The following program will READ DATA statements and tell you the length of every person's name:

```
>100 READ A$
>110 IF A$=" " THEN 999
>120 PRINT A$;LEN(A$)
>130 GOTO 100
>999 END
>1000 DATA FRANK SMITH
>1010 DATA BILL JONES
>1020 DATA MARY WILLIAMS
>1030 DATA BETH BUHL
>1040 DATA " "
```

If we want to now sort the above list by last name, we can change the first few lines and retain lines 999 through 1040:

```
>100 OPTION BASE 1
>110 DIM A$(5)
>120 N=1
>130 READ A$(N)
>140 IF A$(N)=" " THEN 170
>150 N=N+1
>160 GOTO 130
>170 FLAG=0
>180 FOR I=1 TO N-1
>190 IF SEG$(A$(I),POS(A$(I),
 " ",1)+1,1)<=SEG$(A$(I+1),PO
 S(A$(I+1)," ",1)+1,1)THEN 24
 0
>200 SWITCH$=A$(I)
>210 A$(I)=A$(I+1)
>220 A$(I+1)=SWITCH$
>230 FLAG=1
>240 NEXT I
>250 IF FLAG=1 THEN 170
>260 FOR I=1 TO N
>270 PRINT A$(I)
>280 NEXT I
```

Whew! That's sure a lot more advanced than the previous examples. In order to get beyond writing simple programs, you'll find that you will sometimes have to take a quantum leap forward, figuring to pick up a full understanding through repeatedly using your new knowledge. Let's take a look at what that last program does.

Line 100 merely establishes that there will be no A$(0), the start is

-->

at A$(1). Line 110 allows a maximum of 5 items. Lines 120 through 160 count the number of names to be sorted. A sort routine, known as a bubble sort, is at lines 170 through 250. If you are a beginner, you are not expected to necessarily fully understand sorts yet. Just try to be able to recognize the bubble sort when you see it. Although the bubble sort is not always the best sort, it is one of the shortest and easiest to learn sort algorithms. The tricky line in the example is line 190, which uses the SEG$ function so that only the last name is used in the sort. In studying and working with lines such as line 190, you may find it useful to segregate your thoughts in dealing with the SEG$ and the POS, which is imbedded within the SEG$. Lines 260 through 280 PRINT the sorted list on the screen.

If you want to study some very innovative uses of strings, refer to "Tips from the Tigercub", the monthly tips produced by Tigercub's Jim Peterson. Jim is an expert at manipulating strings and almost every BASIC programmer can learn from his examples.

EXTENDED BASIC

Using REDO

STANDARD: 1A 2A 9A

REDO is a special function for Extended BASIC that is not available in console BASIC.

REDO is used to repeat or edit the line most recently entered. The Extended BASIC manual does not point out that the line does not necessarily have to be a program line. For example, key in:

```
RUN DSK1.MYPROGRAM
```

Extended BASIC gives an error because the file requested must be enclosed in quotes when using RUN. Instead of keying the entire line over, you can use REDO, INS and <FCTN> <D> (the right arrow) to make the line like this:

```
RUN "DSK1.MYPROGRAM"
```

There are also some special benefits in using REDO with program lines. By entering the previous line number and <FCTN> <E> or <FCTN> <X>, followed by REDO (which is <FCTN> <8>), you can edit not only the program coding, but also the line number and you can also proceed beyond the usual fifth line of code. In editing the line number, be sure to remember that you generate two lines, the old one and the edited one, which is sometimes a big help and at other times a minor burden.

Using REDO can greatly speed up and simplify your computing.

ASSEMBLY

Disk Sector Addressing

STANDARD: 1A 2C 4B 5A 6B 7A 9A

Last month we indicated we would begin discussing file construction this month. In the meantime, we received a routine from a reader that ties in with the topic of files. So, before actually going into the files, we'll give you advanced users the opportunity to look at everything on your disk directly, which should help you to better understand your files.

William C. Peregrin, of Cincinnati, Ohio, wrote saying that thanks to information in the FORTH source code and Computer Shopper™, he was able to write a program that accesses a TI disk by sector, instead of by file as is normally done.

Mr. Peregrin offered the following information about the program.

Once the control block is set up starting at >834A, just update the word value at >8350 with the sector

-->

that you want to read. To write a sector, change the byte value at >834D to a zero. To access a different drive, change the byte value at >834C to the drive that you want to access. The word value contained at >834E is a pointer to the location in VDP RAM that contains the 256 byte buffer for the data that is written to or read from the disk. This routine always transfers 256 bytes (one sector is 256 bytes).

Once a read/write operation is completed, the status is returned in location >8350 and the sector accessed is (a word value) returned in location >834A.

Continual reads or writes can be performed by making the following code a subroutine and executing a BL @name:

```
name    EQU  $
        LI   R1,PAB          POINTER TO SUBPROGRAM NAME LENGTH
*                            IN THIS CASE VDP ADDRESS >F80
        MOV  R1,@>8356       THIS ADDRESS IS USED BY THE DSR
        BLWP @DSRLNK
        DATA >A              USED TO INDICATE A SUBPROGRAM
        RT                   RETURN
```

The sector address (>8350) must be updated before each read/write operation is performed.

As a very simple example of how to use the routine, we'll now describe how to read the first disk sector, which includes the name of the disk, directly from the screen. Change the VDP ADDRESS FOR SECTOR, sometimes known as the PAB buffer, from >1000 to >0000, which is the beginning of the Screen Image Table. Note that this will not always give you ASCII code that is readable from the screen, but it will allow you to read the name of the disk. Add the following lines in the space designated in the source listing as YOUR PROGRAM:

```
*****************
* YOUR PROGRAM *
*****************************
KEY     CLR  @STAT           AWAIT PRESSING A KEY
        BLWP @KSCAN          SO THAT SCREEN CAN BE
        MOV  @STAT,@STAT     READ BEFORE PROGRAM
        JEQ  KEY             ENDS
*****************************
```

Here is the complete source listing:

```
*************************************
*  UPDATER - JANUARY 4, 1985        *
*      BY WILLIAM C. PEREGRIN       *
*        CINCINNATI, OHIO           *
*************************************
        DEF  SECTOR
        REF  DSRLNK,VSBW,VMBW,KSCAN
PAB     EQU  >F80
FAC     EQU  >834A
SUBPTR  EQU  >8356
STAT    EQU  >837C
READ    DATA 0                SECTOR # READ OR WRITTEN
UNIT    BYTE 1                UNIT # 1, 2, 3
*                                    4 FOR CORCOMP
```

```
RORW    BYTE >FF                READ OR WRITE
*                               READ  = FF
*                               WRITE = 0
        DATA >1000              VDP ADDRESS FOR SECTOR
        DATA 0                  SECTOR # TO READ OR WRITE
*                                 STORED AT >8350
*                                 CONTAINS ERROR CODE ON RETURN
SROW    DATA >0110              SUBPROGRAM NAME LENGTH AND NAME
MYREG   BSS  32                 WORK AREA
RTREG   BSS  2                  R11 SAVE AREA
SECTOR  MOV  R11,@RTREG
        LWPI MYREG
        LI   R0,PAB             FETCH VDP RAM ADDRESS
        LI   R1,SROW            FETCH POINTER TO READ/WRITE
        LI   R2,2               NUMBER OF BYTES
        BLWP @VMBW              WRITE TO VDP RAM
        LI   R3,8               LOAD BYTE COUNT
        LI   R1,READ            FETCH POINTER TO CONTROL BLOCK
        LI   R2,FAC             FETCH POINTER TO CPU RAM
SETUP   MOVB *R1+,*R2+          MOVE A BYTE OF DATA
        DEC  R3                 DECREASE BYTE COUNT
        JNE  SETUP              JUMP IF NOT FINISHED
        LI   R1,PAB             FETCH POINTER TO VDP RAM
        MOV  R1,@SUBPTR         STORE TO NAME LENGTH POINTER
        BLWP @DSRLNK            DO A DSR
        DATA >A                 SUBPROGRAM IDENTIFIER
**************
* YOUR PROGRAM
* Insert your own codings in this area before assembling to object code.
**************
        CLR  R0                 CLEAR R0
        MOVB R0,@STAT           CLEAR STATUS FLAG
        LWPI >83E0              FETCH CPU WORK AREA
        MOV  @RTREG,R11         RESTORE R11
        RT                      RETURN
        END
```

The above direct sector addressing routine could be used to repair disks that are blown (but normally not physically damaged disks), to identify the disk structure of files, and to investigate the structure of the header sectors of disks. We'll try to return to this routine in a future issue with some advanced programs for performing some of the above-mentioned operations.

TI-WRITER

Value File Form Letters

By Charles M. Robertson
Lake Charles, Louisiana

STANDARD: 1A 2E 4B 5A 6A 7A 9A

Computer literacy is not just knowing the difference between a bit and a byte, a disk and a desk. In its broadest sense, computer literacy is also knowing what can be done with a computer. In theory, anything that can be described in sufficient detail to be modeled can be performed by a computer with suitable interfaces and peripherals. Making the leap from "real world" problems to solutions by computer is the true challenge of achieving computer literacy. These ideas, paraphrased from the writings of Jerry Pournelle, a science and science fiction author and computer magazine editor, best capture the spirit with which I like to approach the keyboard. With this spirit in mind, I began to wonder how I could expand the use of TI-Writer from electronic typewriter and word jiggler to my "lazy man's definition" of word processor: something that processes words so I don't have to!

While assisting a client that I consult for set up a computer system in San Francisco, I began to compare the features of TI-Writer with the word processing program the client had already purchased. I noted, among other differences, a thought provoking aspect of TI-Writer in which the other program was substantially deficient. In the mail merge option of TI-Writer the the value file (often thought of as a name and address or "mail" list) can

-->

contain up to 100 variables of up to 77 columns each, in each file! The wheels began to spin.

I work as a psychotherapist for a community health clinic and enjoy everything about my job except paperwork. I doubt that I am unique in this aspect of job satisfaction. I frequently must prepare case summaries for transfers, case closures, letters to other agencies and even documents for our own data processing department for keying into the dumb-terminal that is part of the network that ties the state's clinics together. This would be tedious enough, but it became maddening when I realized how many times I was sending essentially the same information, in many different forms, for one client. The solution was to establish a "data base file" for each client, utilizing the merge option of TI-Writer.

I began by examining the documents I had been preparing for data they had in common. After selecting the items for my "fields", I determined a way of wording the data at entry that would allow it to fit into several different contexts in the different documents it would be merged with. Most awkward sentences could be reworked in the "form letter" to utilize the data as worded in the value file for insertion. The final step was to set up my value file with prompts for required formats, such as double spacing on the social security number to fit the blocks on some forms I would be printing on and preceeding words in the "form letter".

An asterisk marks the end of the record for the mail merge program and the next record would begin below. The numbers preceding each data field identify this field for entry into text. Optional lines can be used when the destination document is printed in either "adjust, fill" or "no-adjust, no-fill". In the first case, a blank entry is ignored and text wraps to the next field so items that must be longer than 77 characters can be entered by sequentially dedicating fields to them. In the latter case, a null line may be entered into text if no data is present.

A sample of patient data, once entered into the record is reproduced below (patient data is fictitious).

```
1 06 24 83
2 12 11 84
3 12 27 84
4 36 year old white male
5 "I was having problems with the job...I started seeing differences in things".
6 The Pt. apparently developed Sxs. of paranoia which led to conflict at work.
7 a rather heavy use of marijuana prior to onset.
8
9 hostility, denial, and suspiciousness were noted. Depression was also present.
10
11 Eh-35, TP-30, PB-30, RP-30.
12 his denial of Sxs., compliance with treatment, and drug education.
13
14 3 times after intake and currently.
15 "Closure is being processed at this time : "
16 he had accepted his illness, denied drugs, and demonstrated fair remission.
17 "The closing LCAR ratings are "
18 OPTIONAL LINE
19 SCHIZOPHRENIFORM DISORDER                                    :
20 2 9 5 .4 0
21 NONE                                                        :
22 V 7 1 .0 9
23 SAME                                                        :
24 SAME
25 SAME                                                        :
26 SAME
27 JOHN DOE
28 2 2 0 0 0
29 8 8 8  7 7  6 6 6 6
30  2
31 9 9
32 1 5
33 2 5
34 1 5
```

-->

```
35 3 0
36 3 5
37 3 0
38 2 0
39 3 0
40 1 5
41 D2-2-PA #1
51 2120 ANYSTREETNAME
52 LAKE CHARLES, LA 70605
*
```

This data is merged into any of several existing form letters and documents.I have created and saved to disk with these files. A particular record is selected by answering "Y" to the MAILING LIST question in TEXT FORMATTER and selecting the desired record(s) in response to WHICH LETTERS?. Incidentally, each file holds fifteen such records and I have room on a SS/SD disk for three such files and the six documents I merge them with. Data item 41 holds the disk, file, and record number for each patient and is reported as Mail List name and "page" number when TI-Writer prompts for this data in in preparation to print. Files can be searched by.any field in Editor mode through use of the FIND STRING command. A sample of one of the documents my computer now prepares for me (so I don't have to) is shown below.

```
.LM 7;RM 73;SP 5;PL 55;FO
.DP 99:ENTER DATE:;DP
98:CITY/STATE/ZIP;DP 97:ST.
ADDRESS?;DP 96:LETTER TO? .DP
42:OPTIONAL LINE(28);DP 43:OPTIONAL
LINE(28) .DP 44:OPTIONAL LINE;DP
45:OPTIONAL LINE .CE[CR]
   *99*[CR]
   .SP 4[CR]
       *96*[CR]
       *97*[CR]
       *98*[CR]
   .SP 2[CR]
                                                  Re: *27*[CR]
                                                  *29*[CR]

   .SP 2[CR]
       Gentlemen:[CR]
   .SP 3;AD;FI;IN +5[CR]
The above captioned entered our clinic
*1* with a complaint of *5* *6*
Contributing factors were *7* And,
symptoms of *9*[CR]
The client was assigned to the
Short-Term Treatment unit and seen
*14* Treatment has focused on *12* The
diagnosis at admission was *19* DSM
III code:*20*[CR]
   *42* *43* *44* *45*[CR]
   .SP 2[CR]
   .IN 7;NA;NF[CR]
       Sincerely,
   .SP 3[CR]
       CHARLES M. ROBERTSON, HSW III[CR]
```

The use of DEFINE PROMPT allows me to "customize" each letter at print time to answer unforeseen questions or make special recommendations.

This is my idea of word processing! You should hear the grumbles from my co-workers as I applaud the arrival of each new request for information I receive! What was once written over and over in a dozen different ways is now only written once (by me, that is).

In a future article, I will share a program in Extended BASIC that allows me to create these files on my unexpanded console at work when I have taken the Peripheral Expansion Box home for more fun projects.

-->

99 POTPOURRI

News, Corrections, Updates, Editorials, Kudos, and Come-what-may

CORRECTIONS:

DECEMBER: Well, we hope to finally get this one right. On page 8, the Extended BASIC test program, lines 140, 200 and 260, the ends of the lines should read "*768)" instead of "8)^". Sorry, but our January correction was wrong, too.

JANUARY: In "Solitaire Checkers", on page 3, line 1170 should read as follows ("STEP 2" was omitted):
>1170 FOR I=8 TO 22 STEP 2

In the FORTH article, we found no errors, but we failed to fully explain the scoring. The score represents the square of the pixel distance of one shot from the target, as if each shot were a new game, with the better score being the lower score (best is 1). Also, we did not list the dash options to be used, which were -GRAPH and -GRAPH1.

In the TI-Writer article, page 7, the FIX/80 file can be read directly from Extended BASIC's output file to TI-Writer's Editor, thus retaining control characters.

In the future, we promise to use the computer more in assisting editing!

This issue represents a departure from our typical issue. This is not to be considered a change in policy! We simply had more advanced and special material available than usual and many readers were requesting more advanced topics. We haven't forgotten the beginners! Next month we will return to our normal wide range of subject matter. Our recent mail has been in favor of beginner topics, program listings, children's programs, game reviews, spreadsheeting, and word processing. We'll be sure to cover these topics in upcoming issues.

COMING IN MARCH:
Game review!
Important Multiplan™ tips!
More TI-Writer discoveries!
Tips on writing graphics programs!
Information on the new CP/M™ cards!

Amerisoft has recently released several very promising programs. The programs include a compiler (!), an Assembly program that links to Editor/Assembler BASIC, Extended BASIC, or Mini Memory BASIC for plotted graphics that can be dumped to a printer (disk version only), a "painting" program with printer dump, and a "3-D" program for rotating objects on 3 planes. Most of the major wholesalers are handling the programs, which should be available through most dealers.

IMPORTANT NOTICE: Now that we have released a number of issues, we have discontinued our policy of automatically beginning subscriptions with Volume 1, Issue 1 (Sept. '84). If you are a new subscriber and would like to receive back issues, just drop us a line and we'll change the beginning month of your subscription to any previous month you desire. Since we often reference back issues and we have offered a lot of valuable information, we're sure most of you will want the previous issues. Allow one to three weeks for receiving your copies. If you have any requests for topics, please write, as we make every possible effort to provide the type of information being requested.

Multiplan is a registered trademark of Microsoft Corp.
Computer Shopper is a registered trademark of Patch Publishing Co., Inc.
CP/M is a registered trademark of Digital Research Corp.

-->

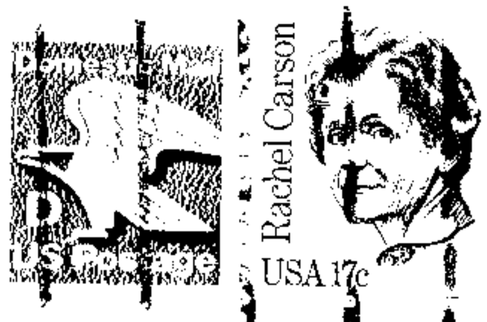
Super 99 Monthly is published monthly by Bytemaster Computer Services, 171 Mustang Street, Sulphur, LA 70663. Subscription rate in U.S. and possessions is $12.00 per year; all other countries $16.00 U.S. funds for surface mail. All correspondence received will be considered unconditionally assigned for publication and copyright and subject to editing and comments by the editors of Super 99 Monthly. Each contribution to this issue and the issue as a whole Copyright 1984 by Bytemaster Computer Services. Bytemaster Computer Servies assumes no liability for errors in articles.

STANDARD KEY

1	Computer	A	TI-99/4A
2	Cartridge	A	Extended BASIC
		C	Editor/Assembler
		E	TI-Writer
4	Disk Drive	B	TEAC 55B
5	Expansion Box	A	TI
6	Disk Controller	A	TI
		B	CorComp
7	32K Card	A	TI
9	Monitor or TV	A	TV & RF Modulator

Bytemaster Computer Services
171 Mustang Street
Sulphur, LA 70663

FIRST CLASS MAIL



FIRST CLASS MAIL

SUPER 99 MONTHLY