computer notes

50¢

October '77

Volume 3 Issue 5

special software issue

This month:

Text Editors Save Time

Billing System Simplified for BASIC

Resequencer Revised

8K to Extended BASIC Convert Your Programs

SUBMITTAL SPECIFICATIONS

Articles submitted to **Computer Notes** should be typed, double-space, with the author's name, address and the date in the upper left-hand corner of each numbered page. Authors should also include a one-sentence autobiographical statement about their job, professional title, previous electronic and/or computer experience under the article's title. Authors should retain a copy of each article submitted.

All illustrations, diagrams, schematics and other graphic material should be submitted in black ink on smooth white paper. Prints and PMT's are acceptable. No pencil drawings unless properly "fixed." No halftone or wash drawings.

All artwork should be mailed flat, never folded. Unless requested, graphics are not returned. Sketches, roughs and "idea" drawings are generally not used.

Photos, charts, programs and figures should be clearly labelled and referred to by number within the text of the manuscript.

Only clear, glossy black and white photos (no Polaroid pictures) will be accepted. Photos should be taken with uniform lighting and sharp focus.

Program listings should be recorded with the darkest ribbon possible on blank white paper. A paper tape for each program submitted **must** also be included.

COMPUTER NOTES is published monthly by **MITS, Inc.**, 2450 Alamo SE, Albuquerque, NM, 87106, **(505) 243-7821.** A free year's subscription is included with every purchase of an Altair™ computer. Regular subscriptions can be ordered from the **MITS** Customer Service Dept. for **$5** per year in the U.S. and **$20** per year for overseas. Single copies are available for **50¢** each at all **Altair Computer Centers.** Entire contents copyright, 1977, **MITS, Inc.** Send articles, questions, comments and suggestions to **Editor, COMPUTER NOTES, MITS, Inc.**

a subsidiary of **Pertec Computer Corporation**
2450 Alamo S.E., Albuquerque, New Mexico 87106

BITS AND PIECES

By Sondra Pollini
MITS

Defective Software

If you receive MITS software that you believe to be defective, by all means return it to us. We want to correct or replace any defective software, but we need your help. Our technicians require as much background about the problem as you can supply. When returning defective software, please be sure to include the following information:

1. Name, address, and phone number
2. Detailed description of the problem (won't load, specific command,etc.)
3. Place and data of purchase (if other than MITS, include a copy of the invoice).
4. Whenever possible, include a computer listing of the problem.

Software Policy

1. We will only accept responsibility for original copies of MITS software. Please do **not** send in reprints of an original or copies of software that you haven't purchased at MITS or at one of our authorized dealers.
2. We only guarantee that MITS software will load and execute according to the specifications for each particular type and version of software. We do not guarantee **"Bug-Free"** software.
3. Replacements will only be made if the software does not load when checked out at MITS. If a particular bug prevents effective programming, patches will be supplied or the software will be totally replaced.

Software Discounts

There are two prices for software listed on the MITS suggested retail price list. The larger amount is for customers who haven't purchased a minimum system from MITS or one of our delears. (We do not honor second-hand purchases.) To qualify for a discount on software, you must meet the following purchase requirements for a minimum system for any version of MITS BASIC or assembly language packages. All software listed is designed for the Altair 8800 series of computers. Hardware requirements must be met with the Altair computer product line.

Altair 8800 Series of Computers

4K BASIC
8800 microcomputer, 4K of memory, I/0 board

8K BASIC
8800 microcomputer, 8K of memory, I/0 board

Extended BASIC
8800 microcomputer, 16K of memory, I/0 board

Disk BASIC
8800 microcomputer, 24K of memory, I/0 board

Minidisk BASIC (Included with Mini Disk System. Software not sold separately.)
Timesharing BASIC
8800 microcomputer, 32K of memory, 88-VI/RTC, I/0 board

Time Sharing Disk BASIC
8800 microcomputer, 32K of memory, 88-VI/RTC, 88-DCDD, I/0 board

Package II
8800 microcomputer, 8K of memory, I/0 board

DOS
8800 microcomputer, 16K of memory, I/0 board

Once you have purchased BASIC at the minimum system price, you can buy copies and updates to larger versions of BASIC at a reduced rate. Copies of any version of BASIC on paper tape or cassette are $25 plus $1 postage and handling. Copying fees for disk versions are $35 plus $1 for handling.

To determine an update charge, simply subtract the difference in price between your current version and the one you want to purchase, add a copying fee, plus $1 postage and handling. Updates can only be purchased at a reduced fee if you also meet all minimum system requirements for the new version of BASIC.

Editor
Andrea Lewis

Assistant Editor
Linda Blocki

Production
Al McCahon
Lucy Ginley
Beverly Gallegos
Alice Regan

Contributors
Don Chamberlain
Don Fitchhorn
Bruce Fowler
Doug Jones
Dr. James F. Morrison
Sondra Pollini
Robert Rossum
Thomas G. Schneider
Robert White

MITS, Inc. 1977
a subsidiary of
Pertec Computer Corporation
2450 Alamo S.E.
Albuquerque, New Mexico 87106

IN THIS ISSUE

Text Editor Prevents "Starvation" and Programming Hassles

By Donald Fitchhorn
MITS

"Bob. . . .your supper's ready."

"I'll be in as soon as I correct this bug in my program, dear. Shouldn't take more than a minute."

Bob now had one minute to find where A was being added to B. He scanned down the program listing to find an A=A+B. There were only 75 statements in the program, but somehow he missed it. He began again -- a little faster this time. "It must be here somewhere," he thought anxiously to himself. Again and again he scanned the program, but just couldn't find that illusive A=A+B.

"Your supper's getting cold . . . DEAR!"

"Just one more minute and I'll be in," he mumbled, knowing that just moments were left before his supper disappeared. He redoubled his efforts to locate those five little letters. WAIT! There they were in the middle of line 32. "Lets see now, I should have A=A*B not A=A+B," he thought.

Bob happily began retyping line 32. OOPS! Made a mistake. Quickly, he tried again. And again. Suddenly, the ominous sound of the garbage disposal drifted into the room. Slowly, he began typing again. (There was no hurry now.) AHHH. Got it right. Now to test it. HEY!! How come C3 has a 57 in it?

Does this sound familiar? The solution is a program to correct errors. If this sounds like what you have been looking for, look no further!

Why I wrote an editor

My main objective in writing an editor was to complement DISK EXTENDED BASIC's built-in EDIT feature. This feature is invaluable if the location of a needed change is already known. But what about our friend Bob's problem? He knew WHAT the problem was but not WHERE to find it. What he needs is a program that will search through the entire file until it finds what he wants. Since there isn't such an editor in Disk Extended BASIC, I wrote my own in BASIC so that it can be easily changed.

The search command and others like it were the beginnings of my editor, which now has 14 commands. But before we get into an explanation of this editor, let's review the definition of an editor.

What is an editor?

An editor is a program which permits the addition and deletion of lines and characters in one file to make another file. The second file is the new up-to-date file, and the first file is retained as the backup file. Some editors permit the creation of new files and/or use multiple input files. An editor can be as extensive or as minimal as is necessary for an application.

PROGRAM FILES

The EDIT program works on ASCII program files. A program file is any file that looks like a program to BASIC. (See EXAMPLE 1.) BASIC doesn't care if the whole file is REMark statements; it's only concerned with whether or not there is a line number at the beginning of each line. (See EXAMPLE 2.)

(NOTE: The EDIT program cannot handle files saved in binary. Save all files in ASCII, i.e. SAVE "FILENAME", Ø, A)

```
EXAMPLE #1
10 'THIS IS A PROGRAM FILE
20 FOR I = 1 TO 100000
30 PRINT I;RND(1)*1000;I*RND(I)
40 ' THIS PRINTS SOME NUMBERS
50 NEXT
60 END
EXAMPLE #2
10 'This is a document program file
20 'here is the
30 'text of the
40 'document !
50 'this is the end.
EXAMPLE #3
LOWER CASE IS USER TYPED
run"edit
EDIT -- VERSION 1.0
INPUT FILE NAME?time
>r
EOF1
CLEARING.....
>/1
5 LPRINT"MINUTES","HUNDRETHS",,"MINUTES","HUNDRETHS"
10 FORI=60TO30STEP-1
20 J=INT(I/60*100)
21 K=INT((I-30)/60*100)
30 LPRINTI,J,,I-30,K
40 NEXT
>gJ
20 J
>cL2
20 L2=INT(I/60*100)
>gJ
30 LPRINTI,J
>cL2
30 LPRINTI,L2,,I-30,K
>gJ
EOF
>x
BACKUP FILE NAME?time.bak
OK
load"time
OK
list
5 LPRINT"MINUTES","HUNDRETHS",,"MINUTES","HUNDRETHS"
10 FORI=60TO30STEP-1
20 L2=INT(I/60*100)
21 K=INT((I-30)/60*100)
30 LPRINTI,L2,,I-30,K
40 NEXT
OK
load"time.bak
OK
list
5 LPRINT"MINUTES","HUNDRETHS",,"MINUTES","HUNDRETHS"
10 FORI=60TO30STEP-1
20 J=INT(I/60*100)
21 K=INT((I-30)/60*100)
30 LPRINTI,J,,I-30,K
40 NEXT
OK
```

Saving documents as programs (EXAMPLE 2 format) allows them to be loaded with BASIC. This allows BASIC to be used to alter, delete, and add lines. Of course, line numbers on the finished document may not be wanted, so a short program that reads the file and PRINTs MID$(LINE$, INSTR(LINE$,"'")+1) will print everything to the right of the ('). Be sure to use LINEINPUT instead of INPUT when reading up LINE$ to prevent truncation because of commas in the text. The ('s in EXAMPLE 2 are necessary if the program file is to be loaded and saved by BASIC. Without the ('), BASIC will modify the text line.

BASIC won't allow lines to be moved around within the file without retyping each line. But with the EDIT program, line numbers can be changed to whatever is wanted. When the edited file is loaded into BASIC, BASIC will put the lines in numerical order.

Internal Structure

The EDIT program maintains a double-linked list a memory. Each line (L1$(X)) has a pointer to the previous line [M1(X,Ø)] and to the next line [M1(X,1)] added to it when it is read in. The array (L1$) that the lines are kept in is divided into two parts -- ACTIVE CELLS, which have data in them, and INACTIVE CELLS, available for use as data lines. Deleted lines are linked into the INACTIVE CELLS from the ACTIVE CELLS. Inserted lines are written into the first available INACTIVE CELL and then linked into the ACTIVE CELLS. Pointers are maintained for FIRST ACTIVE CELL(LN), FIRST INACTIVE CELL(IN), DOT or position within cell (H) and CELL that DOT is in (J).

Commands

My editor, like many others, uses a single letter to select a command. For example, A will advance DOT one line. Most commands may be preceded by a number or a slash (/) to indicate that they should be executed more than once. 13A will advance DOT 13 lines. OA will position DOT at the beginning of the current line. /A will advance DOT to the end of the page. All commands that allow a prefix will default to a one if none is specified. The following is an explanation of the commands. (See TABLE 1 for list of

commands in alphabetical order. See TABLE 2 for a list divided into four main groups.)

How to use edit's commands

The commands in EDIT are broken up into four groups, as shown in TABLE 2. The use of these commands will be explained in that order.

The commands that affect lines will work on one line at a time. Or, in the case of K&L, they may be preceded by a number or slash(/) to indicate that they are to be performed on several succeeding lines. To begin insertion of lines, type I ⟨return⟩ and then the lines to insert. To tell EDIT that the last line to be inserted has been entered, type backslash (\ ).

Of the commands that affect characters, I & C cannot have a # prefix. #D deletes # characters to the right of DOT. G & C work together to allow getting a string and then changing it to something else. G moves DOT ahead of the Nth occurrence of the string. Then C can be used to change the Nth occurrence to a new string. It works like C, except instead of changing one string for another, it inserts a new string ahead of DOT.

The commands that move DOT are A, B, E, and J. B & E require no other specifiers. They simply move DOT to the beginning or end of the current page. A & J move DOT forward or back a specified number of lines or characters.

The commands N, R, and X read and write the files. R reads the input file until EOF, until it has read 50 lines, or 2000 characters. It then clears and resets the INACTIVE CELLS. N writes out the current page and then executes an R (reads in the next page). X does a series of N's until the input file is EOF. Then it closes the files and renames them by giving the input file a backup name and the output file the input file's old name.

The best way to learn to use EDIT is to load it in and try a few commands. (See EXAMPLE 3.) Once you get the hang of it, the power and versatility will be well worth the time it took to type it in.

The other side of the coin

All of the above is really wonderful isn't it? But this program is not without its limitations.

1. The commands may or may not work the way the user expects them to. This is a typical problem in any new program because the commands take some getting used to. If, after trying it for a while, the user wants a command to work differently or wants to use another command, just remember that the program is written in BASIC, so modifications are easy.
2. The program allows working on large files by breaking the file up into pages. This works out well except for one thing. No matter what the user does, string space is used. Eventually, all available space will be used.

   At that time, BASIC has to look through all of the string space, shuffling things around and freeing up no-longer-used bytes so that the program will have some more space. This is commonly referred to as GARBAGE COLLECTING. It can happen at the most unlikely of times and can take as long as five minutes. Unfortunately, to the unsuspecting user, it looks as though the program has bombed BASIC out because CTRL-C and RESET don't help solve the problem. But patience is rewarded and the program does come back to life.

continued

TABLE #1

COMMAND	DESCRIPTION	OPERATES ON: LINE	CHAR	DOT	FILE	ALLOWED: Ø	#	-#	/
A	ADVANCE			*		*	*	*	*
B	BEGINNING			*					
C	CHANGE		*						
D	DELETE		*				*		
E	END			*					
G	GET		*				*		
I	INSERT	*	*						
J	JUMP			*		*	*	*	
K	KILL	*					*		*
L	LIST	*					*		*
N	NEXT				*				
R	READ				*				
V	VERIFY	*							
X	EXIT				*				

```
TABLE #2
COMMANDS THAT AFFECT LINES
I - Inserts lines until backslash (\) is entered.
K - Kills entire line.  #K & /K are legal.
L - Lists entire line.  #L & /L are legal.
V - Prints the current line up to DOT.
   (Verify's the position of DOT).
COMMANDS THAT AFFECT CHARACTERS
C - Changes last character string gotten with G command to
    something else.  Use: Ghere
                          Cthere
    will change here to there.
D - Delete a character.  #D legal.
G - Get string.  Searches for occurance of string.
    Use: 3Gstuff
    Finds third occurrance of stuff.  #G legal.
I - Insert characters at DOT.  Use: Iabcde
    Will insert abcde in current line at DOT.
COMMANDS THAT MOVE DOT
A - Moves DOT forward or back the number of lines
    specified.  ØA, #A, -#A, /A(same as E) legal.
B - Moves DOT to beginning of page.
E - Moves DOT to end of page.  (same as /A)
J - Moves DOT forward or back the number of characters
    specified.  ØJ, #J, -#J legal.
COMMANDS THAT READ AND WRITE THE FILE
N - Writes out current page and reads in next page.
R - Read in a new page.
X - Writes out current page then reads and writes until
    end of file.  Closes and renames files.
```

Altair KCACR Resequencer Revised

By Doug Jones
2271 North Mill
North East, PA 16428

When I bought the MITS Altair 680-KCACR (Kansas City Audio Cassett Recorder) board, I soon found that my 680 Resequencer program (see pp. 12-14, CN, June 1977) did not work with CSAVE BASIC. A modification to the original program could be made and will be explained later in the article. But I went ahead with a complete overhaul of the program. The KCACR Resequencer is still only 33 lines long, but it has a couple of new twists.

Why the 680 Resequencer Fails

As long as the original program is used with the original version of 680 BASIC (version 1.0, release 3.2), everything will work. But the program will not work with CSAVE BASIC. In CSAVE, someone opened up the command table right in the middle and dropped in two new commands. This shoved everything over two hexadecimal digits, causing bitter reactions, such as STR$ to be PEEK and MID$ to be LEFT$. It also made a mess of

Text Editor Prevents "Starvation"

continued

Modifications and improvements

I will leave these up to the reader because they are easy to make. For example, suppose a command is needed to exit the program gracefully without making any changes. Follow these steps:

1. Pick a command character. How about Q for quit?
2. Alter line 250 to reflect where the program should go if it sees a Q. Let's make this 1000.
3. Put in the necessary code to perform the command.

1000 CLOSE : CLEAR 200 : END

4. Save the new program.

So if you're like Bob and need to make a few program changes before supper, give my editor a try. Your wife/husband will love you for it. (And so will your stomach.)

```
10 '                 == WRITTEN BY D. L. FITCHHORN ==
15 '                =   *****  ****   *****  *****   =
20 '               =    *      *  *     *      *      =
25 '              =     ***    *  *     *      *       =
30 '               =    *      *  *     *      *      =
35 '                =   *****  ****   *****    *     =
40 '                 ==   PROGRAMMER - MITS, INC   ==
45 '
50 DEFINT A-Z
55 CLEAR 15000
60 DIM L1$(100),L2$(13),M1(100,1):FOR I=0 TO13:READ L2$(I):NEXT
65 DATA
A - ADVANCE,B - BEGINNING,C - CHANGE,D - DELETE,E - END,G - GET,I - INSERT,
J - JUMP,K - KILL,L - LIST,N - NEXT,R - READ,V - VERIFY,X - EXIT
70 PRINT"EDIT -- VERSION 1.0":PRINT
75 LINEINPUT"INPUT FILE NAME?";N1$
80 OPEN"I",1,N1$
85 N2$="EDIT.TMP"
90 OPEN"O",2,N2$:PRINT#2,""
95 I=1:J=1:H=1
100 '-----------------------------------------------------INPUT COMMAND
105 K=0
110 IF A$=""THEN FOR Q9=1TO5:PRINTCHR$(7);:NEXT:LINEINPUT">";A$
115 A=0:J1=1:U=0:T=1
120 IF A$="" THEN 105 ELSE I9=INSTR(A$,"\"):IF I9<>0THENS$=LEFT$(A$,I9-1):
                          A$=MID$(A$,I9+1) ELSE S$=A$:A$=""
125 SS=ASC(S$)
130 IF 64<SS THEN IF 96<SS THEN SS=SS-32 :GOTO170 ELSE GOTO 170
135 T=VAL(S$)
140 S=LEN(STR$(T))
145 IF T=0 THEN IF LEFT$(S$,1)="/" THEN T=400
150 S$=MID$(S$,S)
155 IF T<0 THEN S$=MID$(S$,2)
160 GOTO 125
165 '              .A  .B  .C  .D  .E  <F> .G  <H> .I  .J  .K  .L  <M>
170 ON SS-64 GOTO 195,235,245,265,280,180,290,180,320,360,385,425,180,
                  445,180,180,180,465,180,180,180,515,180,525,180,180
175 '              .N  <O> <P> <Q> .R  <S> <T> <U> .V  <W> .X  <Y> <Z>
180 FORN=0TO12:PRINT L2$(N):NEXT:GOTO105
185 '
190 '-----------------------------------------------------------------A COMMAND
195 H=1:IF T=0 THEN GOTO 105 ELSE IF T<0 THEN J1=-1
200 FOR I3=0 TO T-J1 STEP J1
205 IF M1(J,0)=0 AND J1=-1 THEN GOTO 105
210 IF M1(J,1)=-1 AND J1=1 THEN J=M1(J,0):GOTO105
215 IF J1=1 THEN J=M1(J,1) ELSE J=M1(J,0)
220 NEXT
225 GOTO 105
230 '-----------------------------------------------------------------B COMMAND
235 J=LN: H=1:GOTO 105
240 '-----------------------------------------------------------------C COMMAND
245 S$=MID$(S$,2):IF K=0 THEN S=LEN(S$):K=H
250 IF K=1 THEN L1$(J)=S$+MID$(L1$(J),K+S) ELSE L1$(J)=LEFT$(L1$(J),K-1)+
    S$+MID$(L1$(J),K+S)
255 PRINT L1$(J):H=K+LEN(S$):GOTO 105
260 '-----------------------------------------------------------------D COMMAND
265 IF H=1 THEN L1$(J)=MID$(L1$(J),H+T):
        ELSE L1$(J)=LEFT$(L1$(J),H-1)+MID$(L1$(J),H+T)
270 GOTO 105
275 '-----------------------------------------------------------------E COMMAND
```

the line numbers.

The program can be salvaged. But for reasons that will soon be apparent, I didn't try modifying it. If you care to try, here are several suggested patches.

1. (line 63968) Open up the DATA statements between WAIT and DEF, and add CLOAD, CSAVE.
2. (line 63973 Redimension the array to 67; also change the read loop boundary to 67.
3. (lines 63986, 63993) Change constant 158 to 160.

To the best of my knowledge, the above patches should work. But the problem is that the original (or modified) program was designed for paper tape environment. The new version presented here is for an ACR environment.

How to Save Resequencer

Type the program initially and save it on a mag-tape file as CSAVE"R". Check your file copy by doing a CLOAD?"R". These commands are covered in depth in the Altair KCACR Operating Manual.

On the same tape and in a second file, let's make an ASCII merge copy of the program. To do a merge, two subroutine calls must be modified in CSAVE BASIC. (This is not covered in the KCACR or BASIC manuals.) When preparing your merge tape, only the OUTCH call needs to be modified.

RESET the computer. Using the MONITOR'S (M) and (N) command, adjust the following address call to OUTCH.

.M 08BB BD N
.N 08BC FF FD
.N 08BD 81 F5

You have verified the JSR (BD) call to OUTCH (FF81) and changed the call to a similar routine (FDF5) on the KCACR ROM. Return to BASIC .J 0000.
OK
Carefully type in the following:
CAUTION You will not see echo of what you type.

```
NULL20:?:?:?:FORN=IT0100:?CHR$(0):
NEXT:?CHR$(15):LIST
```

Start the tape deck recording, leave a l-o-n-g leader, and **then** hit the carriage-return to the above commands. You can monitor the recording with an ear-plug in the external speaker jack.
RESET the computer and (M) and (N) the OUTCH call address to the former values.

continued

```
280 IF M1(J,1)=-1 THEN H=1: GOTO 105 ELSE J=M1(J,1):GOTO 280
285 '-----------------------------------------------------------G COMMAND
290 S$=MID$(S$,2):S=LEN(S$):IF S=0 THEN GOTO 105
295 K=INSTR(H,L1$(J),S$): IF K=0 THEN IF M1(J,1)=-1
    THEN PRINT"EOB":A$="":GOTO 105 ELSE J=M1(J,1):H=1:GOTO 295
300 U=U+1:IF U<T THEN H=K+S: GOTO 295
305 PRINT LEFT$(L1$(J),K+S-1):H=K
310 GOTO 110
315 '-----------------------------------------------------------I COMMAND
320 IF MID$(S$,2)<>"" THEN 345 ELSE I2=J:IF J=LN THEN LN=IN
325 LINEINPUT L1$(IN):IF L1$(IN)="\" THEN 105
330 I3=M1(IN,1):M1(IN,1)=I2
335 M1(IN,0)=M1(I2,0):M1(I2,0)=IN:I4=M1(IN,0)
340 M1(I4,1)=IN:IN=I3:M1(I3,0)=0:GOTO 325
345 IF H=1 THEN L1$(J)=MID$(S$,2)+L1$(J)
      ELSE L1$(J)=LEFT$(L1$(J),H-1)+MID$(S$,2)+MID$(L1$(J),H)
350 H=H+LEN(S$):GOTO105
355 '-----------------------------------------------------------J COMMAND
360 IF T=0 THEN H=1:GOTO 105
365 H=H+T :IF H<1THENH=1
370 IF H>LEN(L1$(J)) THEN H=LEN(L1$(J))
375 GOTO105
380 '-----------------------------------------------------------K COMMAND
385 H=1:I2=J:I3=M1(J,0):FOR J1=1 TO T
390 IF M1(J,1)=-1 THEN GOTO 410
395 I4=M1(J,1):L1$(J)="":M1(I4,0)=I3
400 M1(IN,0)=J:M1(J,0)=0:M1(J,1)=IN:IN=J
405 J=I4:NEXT
410 IF I2=LN THEN LN=J ELSE M1(I3,1)=J
415 GOTO105
420 '-----------------------------------------------------------L COMMAND
425 I2=J:FOR J1=1 TO T
430 PRINTL1$(I2): IF M1(I2,1)=-1 THEN GOTO 105
    ELSE I2=M1(I2,1): NEXT
435 GOTO 105
440 '-----------------------------------------------------------N COMMAND
445 I2=LN
450 IF M1(I2,1)=-1 THEN GOTO 465 ELSE PRINT#2,L1$(I2):I2=M1(I2,1)
455 GOTO 450
460 '-----------------------------------------------------------R COMMAND
465 J=1:A#=0:LN=1:I=1:FE=0:GOSUB470:GOTO 105
470 IF EOF(1) THEN PRINT"EOF1":I=I-1:FE=1:GOTO 495
475 LINEINPUT#1,L$:IF L$="" THEN GOTO 470
480 A#=A#+LEN(L$)
485 L1$(I)=L$:M1(I,0)=I-1:IF I=1 THEN 490 ELSE M1(I-1,1)=I
490 IF I=50 OR A#>2000  THEN GOTO 495 ELSE I=I+1: GOTO 470
495 M1(I,1)=I+1:I=I+1:L1$(I)="END OF BUFFER":M1(I,0)=I-1:M1(I,1)=-1:H=1:IN=I+1
500 FOR I2=IN TO 100:M1(I2,1)=I2+1:M1(I2,0)=I2-1:NEXT
505 M1(IN,0)=0:M1(I2-1,1)=-1:RETURN
510 '-----------------------------------------------------------V COMMAND
515 PRINT LEFT$(L1$(J),H):GOTO105
520 '-----------------------------------------------------------X COMMAND
525 I2=LN
530 IF M1(I2,1)=-1 THEN GOTO 535
    ELSE PRINT#2,L1$(I2):I2=M1(I2,1): GOTO 530
535 IF FE=0 THEN I=1 :A#=0: GOSUB470:GOTO 525
540 CLOSE:ONERROR GOTO 555:LINEINPUT"BACKUP FILE NAME?";N3$:KILL N3$
545 NAME N1$ AS N3$
550 NAME N2$ AS N1$:CLEAR 200:END
555 IF ERR = 53 THEN GOTO 545: ELSE :ONERROR GOTO 0
```

Altair KCACR Resequencer Revised

continued

```
.M 08BC FD FF
.N 08BD F5 81
JJ 0000
OK
NULLO
OK
```

Now you've created an ASCII format merge tape. It's just as if the computer were sending to a paper tape punch, except it's at 300 baud, and there are no paper tape chips on the floor. The command you typed (without echo) set nulls between lines to 20, put out a hundred-character leader, put out a CONTROL-0 to suppress echo when you reload the merge tape, and finally LISTed the program on mag tape.

How to Load Resequencer

If your BASIC buffer is initially empty and you'll be constructing a program as you go, then simply CLOAD"R" the first file. Check your load by rerunning the first file and CLOAD?"R". Then construct your program, but be careful not to type NEW. Directions for using the resequencing function follow in the next section.

If BASIC already has a program in the buffer that you might have CLOADed earlier, you will have to use the merge version to resequence it. To do a CLOAD at this pint would wipe out your program. So this time, we must temporarily change the INCH call in BASIC. RESET the computer. Using MONITOR, (M) and (N) the following:

```
.M 042D BD N
.N 042E FF FD
.N 042F 00 62
.J 0000
OK
```

Start the tape deck in the l-o-n-g-leader of file 2, the merge RESEQUENCER. Completion of the load is indicated by a computer response of:

```
?SN ERROR
```

Check the tape travel indicator (if you have one) and verify that the program did load.

RESET the computer. (M) and (N) the linkages back to INCH.

```
.M 042E FD FF
.N 042F 62 00
.J 0000
OK
```

You can check the merge-load of Resequencer by typing:

```
LIST63967
```

Although clumsy, the merge technique does allow you to append two programs. This can't be done with CLOAD due to the inferred NEW command that accompanies it.

The modification of the INCH and OUTCH address linkages could be done as a program, which might provide a smoother operation. But I didn't attempt it.

So, to append two CSAVEd programs, create a merge tape of one of them. Load the first program the normal CLOAD method. The merge copy must be loaded as described above. The program loaded last will predominate on overlapping line number.

How to Use Resequencer

Once the buffer is loaded with your program and Resequencer, the rest is simple:

```
RUN63967
(-)OLD LINE#, NEW#, STEP? 7,100,10
LINES 113
READY MAG-TAPE
"Z" IS READY
OK
```

Your response to the line query indicated that your old starting (indicated) line number was 7; you wanted the new first (absolute) line number to be 100 and the steps to be 10. Easy. It then told you your (indicated) program was 113 lines long and that you have about four seconds to start the mag-tape. At the end, it told you that your file "Z" was created.

An error response might appear like this:

```
LINES 113
READY MAG-TAPE
ERROR ON LINE # 211
OK
```

The number 211 refers to the line in the BASIC buffer.

```
LIST211
211 IF A=B THEN GOTO 437
```

Line number 437 probably does not exist in your program. Rewind the tape (it's junk anyway) and fix your program.

Now the piece-de-resistance (loose translation). It asked you (-)OLD LINE#, and your answer was 7,100,10. So it dumped only your resequenced program into "Z". If you answered with -7, 100,10, it not only dumped your resequenced program into "Z" file but appended the Resequencer program at the end of it. Resequencer remains intact with the original line numbers, which is useful if your program needs further work.

How to Load the "Z" File

It's simple, just type CLOAD "Z". Notice that it takes three times longer than normal to load a "Z" file. The Resequencer program does not buffer any of it's output. It does each charaacter the hard way: PEEK at it, inspect it, think about it, blink the lights a little bit, then finally POKE it. Meanwhile, your tape deck goes merrily along. With a GOTO or GOSUB, the computer goes nuts trying to figure out to which indexed depth in its array of possible line numbers you are referencing.

Once the file is loaded and you've verified program operation, dump it again with a straight CSAVE command. Then choose a file name. The dump time will soon seem more normal.

How the Resequencer Works

As described in my previous article (CN, June 1977), the Resequencer is a three-pass system. The passes are transparent to the user.

On the first pass, it looks for your first OLD LINE #. Once this is found, it continues counting lines in your program until it bumps up against itself.

On the second path, it gathers your present line numbers into an array. This array will be indexed into on future GOTO, BOSUB, and IF . . . THEN statements.

On the third pass, it writes your newly sequenced program out to mag-tape. Error detection will occur on this pass. Any non-existent but referenced line numbers will be flagged as an error, resequencing will stop. If, however, everything goes O.K., you will be told "Z" (the new file) IS READY.

Since there was no room on the listing for REMarks, perhaps the following will better explain the operation of RESE—QUENCER.

```
63967 Set aside string space;
      Gather pointer to first line of user's
        program;
      First line of RESEQUENCER;
      Variable for motor-off and command
        limit.
63968 Break off high-order byte of numer-
        ic;
      String this value;
      Break off low-order byte;
      String this value;
      Return.
63969 If mag-tape port is busy, loop on
        self.
63970 Send character to output port;
      Return.
63971 Left justify temporary string and
        add to main;
```

Null temporary string.

63972 Old line negative advise appending;
Readjust pointer to first pointer of RESEQUENCER
Lines equal RESEQUENCER length;
Old parameters to parameters of RESEQUENCER;
Flag is permanently set;
Go around again.

63973 String finishing header;
Dump header;
Stop motor.

63974 Gather starting parameters;
Adjust parameter is out of limits.

63975 Store line pointer;
Look for (indicated) first line of program;
Bump pointer;
Loop on self.

63976 Search for first line of RESEQUENCER;
Print how many lines (indicated) user program contains.

63977 Adjust line pointer;
Bump line counter.

63978 Advise to ready mag-tape;
Start motor;
Wait a while;
Dimension line-number array.

63979 Header record;
String numeric;
Header record;
String numeric;
Dump first string.

63980 Swap variables;

63981 Gather line numbers into array;
Readjust line pointer;
Set last line indicator;
Zero counter;
Swap beginning pointer.
Adjust input parameter if out of range.

63982 Break up string record;
Dump characters at output routine;
Set string to null;
Return.

63983 On last line, go to APPEND routine;
Advise that tape is ready;
End.

63984 Adjust line pointer;
Bump character pointer;
Calculate present new line number;
Go and string it;
Bump line character.

63985 Is character pointer equal to line pointer?

63986 Get a character;
If GOTO, GOSUB, THEN or flag is set, handle special.

63987 Bump character pointer;
Is character end-of-line character?

63988 Set variable equal to old pointer adjusted value;
Swap string variables;
Null main string;
Go string numeric just calculated;
Add old string to new;
Go dump entire string to port.

63989 Gather character;
Ad it to string.

63990 Bump character pointer;
Gather temporary character;
If numeric, add to string;
Set flag;
Loop on self.

63991 Is character SPACE or COMMA and is flag not set?

63992 Is flag set and is character not a command?

63993 If original character is not THEN and temporary character is not end-of-line and it is not COLON. . .

63994 Out of special range, handle normally.

63995 Stop motor;
Print error message;
Indicate line number that error is on;
End.

63996 Search through array for referenced line number;

63997 Compute indicated new value and string;

63998 Add (+ or −) line character-length difference to pointer adjuster;
Do special justification string;
Null numeric gathering string;
If temporary character is not SPACE and is not COMMA then expect more.

63999 Gather temporary character into string;
Go back for more.

```
63967 CLEAR400:P=256*PEEK(122)+PEEK(123):Y=63967:Z=127:GOTO63974
63968 F=INT(L/256):B$=B$+CHR$(F):F=L-256*F:B$=B$+CHR$(F):RETURN
63969 IFPEEK(61456)>127THEN63969
63970 POKE61457,ASC(X$):RETURN
63971 B$=B$+RIGHT$(X$,LEN(X$)-1):X$="":RETURN
63972 IFU<0THENPRINT"APPENDING":U=R:C=33:O=Y:W=Y:I=1:S=1:GOTO63980
63973 GOSUB63968:GOSUB63982:POKE61456,191:RETURN
63974 INPUT"(-)OLD LINE #, NEW #, STEP":U,W,I:I=ABS(INT(I))
63975 Q=P:IFABS(U)<>256*PEEK(P+2)+PEEK(P+3)THENP=P+1:GOTO63975
63976 IF256*PEEK(P+2)+PEEK(P+3)=WTHENPRINT"LINES =":C:GOTO63978
63977 P=256*PEEK(P)+PEEK(P+1):C=C+1:GOTO63976
63978 PRINT"READY MAG-TAPE":POKE61456,Z:FORX=1TO3000:NEXT:DIMA(C+32)
63979 L=54227:GOSUB63968:L=54106:GOSUB63968:GOSUB63982
63980 P=Q:FORN=1TOC:A(N)=256*PEEK(P+2)+PEEK(P+3)
63981 P=256*PEEK(P)+PEEK(P+1):NEXTN:T=C:C=0:P=Q:W=ABS(INT(W)):GOTO63983
63982 FORN=1TOLEN(B$):X$=MID$(B$,N,1):GOSUB63969:NEXT:B$="":RETURN
63983 IFC=TTHENL=0:GOSUB63972:PRINT"'Z'' IS READY":END
63984 R=256*PEEK(P)+PEEK(P+1):P=P+4:L=W+I*C:GOSUB63968:C=C+1:GOTO63986
63985 IFP=RTHEN63983
63986 GOSUB63989:IF(F=136ORF=140ORF=160)ANDS=0THEN63990
63987 P=P+1:IFF<>0THEN63985
63988 L=R+V:X$=B$:B$="":GOSUB63968:B$=B$+X$:GOSUB63982:GOTO63983
63989 F=PEEK(P):B$=B$+CHR$(F):RETURN
63990 P=P+1:G=PEEK(P):IFG>=48ANDG<=57THENC$=C$+CHR$(G):S=1:GOTO63990
63991 IF(G=32ORG=44)ANDS=0THEN63999
63992 IFS=1ANDG<ZTHEN63996
63993 IFF<>160ANDG<>0ANDG<>58THEN63995
63994 GOTO63986
63995 POKE61456,191:PRINT"ERROR ON LINE #":A(C):END
63996 FORK=1TOT:IFA(K)=VAL(C$)THENX$=STR$(W+I*(K-1)):S=0:GOTO63998
63997 NEXTK:GOTO63995
63998 V=V+LEN(X$)-LEN(C$)-1:GOSUB63971:C$="":IFG<>32ANDG<>44THEN63986
63999 B$=B$+CHR$(G):GOTO63990
OK
```

BASIC Text Editor Helps Optometrist with Research and Reports

By Dr. James F. Morrison
1770 West 6th St.
Colby, Kansas 67701

Note:

If you have an Altair Disk System or the initiative to change the following programs to be supported without a disk, then it would be worth your while to try Dr. Morrison's text editor. It allows you to write a new letter, read it, correct errors, and continue writing more. Mistakes in text can be edited by deleting or inserting characters or lines. You can also use his edit mode to search through the text for specific phrases. Once you've finished writing and editing, try his formatted output routine, which justifies the right margin. Of course, you can print out the final draft, formatted or not.

When the program asks for the width, it's asking for the formatted output width, not terminal width. The terminal width is set in line 20, which can be changed to suit your needs. Answering 0 to the text editor's width question will default the formatted output to a width of 65 with five spaces on the left margin. Once the width for a particular letter file is set, it's unadvisable to change it.

When writing a letter, make sure you type to the beep or bell sound of the terminal. After the beep, you have approximately nine spaces in which to complete the word you are typing before an error message tells you to re-type the line shorter. If you do not type to the beep, the line will be too short for the formatter to right justify it. While writing a line, make sure the escape (to end the file) is typed as the first character of the line. Otherwise, the line will be lost.

I highly recommend reading Dr. Morrison's documentation. He has done an excellent job of explaining how his program works as well as how to use it. However, both the text editor and the label program are device dependent. Portions of each program may have to be altered to suit your system.

Gale Schonfeld
MITS Software User Specialist

When I bought my Altair 8800b Computer for my office over two years ago, I had no previous experience with computers. Now, $28,000 worth of Altair computer equipment later, I've not only learned a lot about hardware, but have also written 160 different programs.

After seeing patients from 8:30-5:00 everyday, I usually spend six hour a night writing programs. In addition to the text editor and label program discussed in this article, I've done bookkeeping, tax summary and analysis, and general ledger programs, to name a few. All of them have been a tremendous help in the office. The tax summary and analysis program, which uses a real data base, provides details on my tax return for any combination of months or an entire year. Before I purchase anything, whether it's stock, a car, or office equipment, I can use this program to find out what effect that purchase will have upon my financial state.

My bookkeeping program does auditing and accounting. It seeks out math mistakes and won't let the user make any more entries until all errors have been corrected.

I also wrote a payroll program for a local nursing home which I partly own. It shows profits for individual departments and generates tax returns for the entire corporation. It simultaneously works on a cash and accrual basis, so it's easy to see where the cash goes.

In addition to the usual games, I've written some math and English programs for my four kids, ages seven through 12. They've been using the system at the office, since I have only a keyboard and TV monitor at home. But I do plan to buy another 8800b soon for a separate system at home. It will help the kids with their homework, and I want to use it to control my ham station.

I wrote the following text editor to help me with writing patient reports and research in evoked visual responses. I can do the patient reports so much faster with the text editor. Writing or even dictating them is such a slow and tedious process. It's also a lot easier to store them on floppy disks rather than in file cabinets.

The text editor is extremely useful for processing complex research information on evoked visual responses. In this research, through various light stimuli, the examiner can observe any defects within a patient's brain which may have even a minor effect on his/her vision. Often, the patient is not aware of any difference in vision. Many times, standard techniques cannot detect any problems either.

The text editor is set up to operate on my system, which consists of an Altair 8800b with 48L bytes of memory, Altair Floppy Disk, ADM-3, Decwriter II, and uses Altair Disk BASIC, version 4.0. I'll explain the operation of the text editor line by line, giving appropriate comments on the function of each line. Then I'll discuss how to modify your sytem to fit the editor and conclude with a summary of the codes used in the program.

LINE 20:
SETS UP THE MEMORY IN STRING SPACE, LEAVES SOME SCRATCHPAD OF 400 BYTES, GETS THE NAME OF THE LETTER OR DOCUMENT, TELLS THE OPERATOR HOW MANY BYTES HE HAS AVAILABLE IN THE MATRIX FUNCTIONS SETS THE PROPER WIDTH AND PROCEEDS INTO THE PROGRAM CONTROL SECTION (LINE 110-130).

LINE 30-100=WRITE SECTION:
LINE 30:
Y IS A COUNTER TO KEEP TRACK OF HOW MANY LINES THE TEXT CONTAINS. IT IS SET INITIALLY AT 2 BECAUSE RECORD NUMBER 1 CONTAINS THE LAST LINE WRITTEN AND THE TOTAL NUMBER OF LINES IN THE TEXT. IT IS NEVER PRINTED (NOR IS THE LAST LINE AS IT HAS ONLY THE NON-PRINTING ESC ON IT TO SIGNAL THE END OF THE TEXT). IT THEN OPENS THE FILE NAMED IN INITIALIZATION (GOSUB 820) AND FIELDS THE RANDOM BUFFER (GOSUB 800).
LINE 40:
PRINTS THE TERMINAL BELL (CHR$(7)), SETS THE NUMBER OF CHARACTERS PER LINE COUNTER (X)= TO 0 AND PUTS ALL ZEROS INTO THE DUMMY STRING (F$) USED TO INPUT EACH LINE OF TEXT INTO THE RANDOM BUFFER
LINE 50:
SIMPLY INCREMENTS THE CHARACTERS PER LINE COUNTER BY ONE UNIT.
LINE 60:
WAITS FOR INPUT FROM THE ADM-3 ANDS THE INPUT WITH 127 AND PLACES THE CHARACTER INTO F$. IT THEN CHECKS TO SEE IF THE NUMBER OF CHARACTERS INPUT ARE GREAT ENOUGH TO ALLOW RIGHT JUSTIFICATION OF THE TEXT. IF SO THE TERMINAL BELL IS SOUNDED FOLLOWING THE INPUT OF EACH TEXT CHARACTER WHICH TELLS THE OPERATOR THAT RIGHT JUSTIFICATION IS NOW POSSIBLE.
LINE 65:
CHECKS TO SEE IF THE NUMBER OF CHARACTERS INPUT EQUALS THE REQUESTED NUMBER OF CHARACTERS PER LINE. IF SO THEN TOO MANY CHARACTERS HAVE BEEN INPUT TO PROPERLY FORMAT THE TEXT THUS CAUSING THE PRINTING OF THE WARNING THAT YOU HAVE OVERTYPED THE MARGIN AND MUST THEN RETYPE THE ENTIRE LINE BUT SHORTER. THE NUMBER OF CHARACTERS (X) PER LINE IS THEN RESET TO 1 AND F$ IS CLEARED SO THAT THE RETURN TO LINE 60 WILL ALLOW REINPUT OF THE LINE WITH FEWER CHARACTERS.
LINE 70:
CHECKS TO SEE IF THE CHARACTER INPUT WAS THE 'AT' SIGN. IF IT WAS IT BACKSPACES THE TERMINAL (LIKE CONTROL-H) AND ALLOWS RETYPING (ERASING) OF THE PREVIOUSLY TYPED CHARACTER. YOU MAY BACKSPACE TO THE BEGINNING OF THE LINE IF YOU DESIRE. THAT IS THE ONLY METHOD ALLOWED FOR THE CORRECTION OF ERRORS DURING THE INITIAL TYPING OF THE TEXT. LATER USE OF THE EDITOR FUNCTIONS ENABLES CORRECTION OF PREVIOUSLY MISSED ERRORS.
LINE 80:
CHECKS TO SEE IF THE LAST CHARACTER WAS THE ESC KEY. IF IT WAS THE PROGRAM THEN ASSUMES THE END OF TEXT HAS BEEN REACHED AND BRANCHES TO LINE 100 WHERE THE FILES ARE PUT IN ORDER AND A RETURN TO PROGRAM CONTROL IS MADE. IT IS IMPORTANT TO NOTE THAT THE ESCAPE KEY MUST BE THE FIRST CHARACTER ON A NEW LINE (IN EFFECT THE LAST LINE OF THE TEXT) FOR THE PRINTOUT PROGRAMS TO WORK PROPERLY. DO NOT PUT ESC AT THE END OF A LINE OR THAT LINE WILL NOT APPEAR IN THE FINAL TEXT.
LINE 90:
ECHOS THE CHARACTER INPUT BY THE TERMINAL (ADM-3--PORT 0 OF MY 2 SIO BOARD) AND CHECKS TO SEE IF IT IS A CARRAIGE RETURN (13) IF SO IT BRANCHES TO LINE 100 AND PUTS THE LINE JUST TYPED ONTO THE PROPER DISC FILE RECORD NUMBER AND RETURNS TO LINE 50 TO ACCEPT FURTHER INPUT FROM LINE 60.
LINE 100:
APPROPRIATELY SETS B$=F$ AND C$=NUMBER OF LINES (Y) IN THE TEXT AND E$= NUMBER OF CHARACTERS IN THE CURRENT LINE (X) AND THEN PUTS THIS INFORMATION ONTO THE PROPER DISC FILE RECORD NUMBER TO STORE THAT LINE. IT THEN PUTS THE SAME LINE ON THE FIRST RECORD OF THE FILE SO THAT ONE CAN EASILY DETERMINE HOW MANY LINES OF TEXT THE LETTER CONTAINS. THEN IF AN ESC IS PRESENT THE PROGRAM BRANCHES TO PROGRAM CONTROL (LINE 110) OTHERWISE IT RETURNS TO LINE 40 FOR A NEW LINE INPUT.
PROGRAM CONTROL (100-130):

LINE 110:
W IS A FLAG TO INDICATE THAT THE EDITOR FUNCTIONS ARE USING THE READ FUNCTIONS AND FORMATTING FUNCTIONS. IF W IS NEGATIVE THEN THE EDITOR IS NOT IN USE. PROGRAM CONTROL SETS IT NEGATIVE--THE EDITOR MAKES IT POSITIVE. F=THE FUNCTION DESIRED AND IS SET INITIALLY TO 0 TO ALLOW A CARRIAGE RETURN TO PRINT A LIST OF THE FUNCTIONS AVAILABLE WITH THE TEXT EDITOR.

LINE 120:
IF F=9 THEN THE PROGRAM GOES TO A SUBROUTINE WHICH RESETS THE TEXT WIDTH AND THEN GOES BACK TO THE BEGINNING OF PROGRAM CONTROL.
LINE 130:
SIMPLY PRINTS THE FUNCTIONS TO WHICH YOU CAN BRANCH FROM PROGRAM CONTROL.

READ SECTION (140-180):
LINE 140:
X2 IS USED BY THE PAGING FUNCTION AS A LINE COUNTER SO IT IS SET TO 0. FILES ARE OPENED AND FIELDED. THE FIRST RECORD IS OBTAINED TO GET THE NUMBER OF LINES IN THE TEXT (VAL (C$)) AND Y IS SET TO THIS VALUE. PF$ IS USED AS A FORMATTING COMMAND BY THE EDITOR LIST FUNCTION.
LINE 150:
A FOR NEXT LOOP IS SET UP TO GET EACH LINE OF THE TEXT. THE LINE COUNTER IS INCREMENTED (X2) AND A CHECK IS MADE TO DETERMINE IF PAGING IS NOW NECESSARY (GOSUB 1150). THEN IF W IS A 0 OR POSITIVE VALUE THE PROGRAM GOES TO LINE 160 FOR THE EDITOR LISTING FUNCTION OTHERWISE IT GOES TO LINE 170.
LINE 160:
USED IN EDIT LISTING FUNCTIONS TO PROPERLY SPACE THE LINES TO MATCH THE NUMERICAL HEDDER THAT IS PRINTED.
LINE 170:
EACH LINE OF THE TEXT IS OBTAINED HERE AND THE NUMBER OF CHARACTERS CONTAINED IN EACH LINE (VAL (E$)) IS SET IN B. IF THERE ARE NO CHARACTER9 THEN B MUST BE SET TO 1 OR AN ERROR WILL OCCUR IN LINE 180.
LINE 180
PRINTS THE LINE PRINTING CHARACTERS AND GOES TO GET THE NEXT LINE OF TEXT. WHEN COMPLETED THE PROGRAM GOES BACK TO PROGRAM CONTROL.

FORMATTED PRINTING (190-230):
LINE 190:
THE BEGINNING IS ESSENTIALLY THE SAME AS LINE 140 BUT FOLLOWING THE FOR NEXT LOOP EACH RECORD IS OBTAINED AND B IS SET TO THE NUMBER OF CHARACTERS IN THE LINE. A BRANCH IS THEN MADE TO SEE HOW MANY SPACES ARE PRESENT AT THE BEGINNING OF THE LINE (LINE 940-1010). C IS THEN SET TO THE NUMBER OF PRINTING CHARACTERS IN THE LINE TO BE FORMATTED. A CHECK IS THEN MADE TO SEE IF THE LINE IS LONG ENOUGH TO BE FORMATTED--IF SO THE PROGRAM BRANCHES TO LINE 210 IF NOT IT GOES TO LINE 200.
LINE 200:
THE LINE COUNTER IS INCREMENTED A CHECK FOR PAGING IS MADE (GOSUB 1150) THE PROPER NUMBER OF SPACES FOR THE LEFT MARGIN (S1$) ARE PRINTED THEN THE CHARACTERS ARE PRINTED ALL AS A GROUP (NOT FORMATTED BY THIS LINE--PRINT AS IS)
LINE 210:
THE NUMBER OF CHARACTERS IS SET TO ONE (GOSUB 930) THE CHECK FOR PAGING IS MADE.
LINE 220:
PRINTS THE LEFT MARGIN (S1$) THEN GETS EACH CHARACTER ONE AT A TIME (P$) PRINTS IT AND CHECKS TO SEE IF IT IS A SPACE (THE CHR$(32)) AND THEN IF SPACES NEED FILLED TO FILL OUT THE LINE (C FLAG IS NOT 0) THEN PRINTS AN ADDITIONAL SPACE.
LINE 230:
GETS FIRST THE NEXT CHARACTER THEN PRINTS A CARRAIGE RETURN AND LINE FEED AND GETS THE NEXT LINE OF TEXT. WHEN TEXT IS ALL PRINTED IT GOES BACK TO PROGRAM CONTROL.
LINE 240:
THIS IS THE ONLY LINE FOR THE CONTINUE WRITING FUNCTION. IT OPENS AND FIELDS THE FILE AND BUFFER THEN GETS THE FIRST RECORD TO SEE HOW MANY LINES ARE IN THE TEXT. THEN IT CHECKS FOR AVAILABLE MEMORY FOR THE MATRIX FUNCTIONS (GOSUB 1120) AND THEN JUMPS TO LINE 40 (RATHER THAN 30) TO CONTINUE THE TEXT AS IF YOU HAD NOT LEFT IT.

EDITOR SECTION (250-290):
LINE 250:
PRINTS A CARRIAGE RETURN-LINE FEED THEN SETS G=0 SO THAT IF A CARRAIGE RETURN IS GIVEN TO THE FOLLOWING INPUT STATEMENT THE PROGRAM WILL LIST THE EDITOR FUNCTIONS (G).
LINE 260:
SIMPLY GIVES THE FUNCTIONS AVAILABLE IF YOU REQUEST THEM BY HITTING RETURN TO THE INPUT PROMPT.
LINE 270:
OPENS THE FILE FIELDS THE BUFFER GETS THE FIRST RECORD AND THEN PRINTS HOW MANY LINES YOU HAVE IN THE TEXT (THE -2 IS THERE BECAUSE OF THE USE OF THE FIRST AND LAST RECORDS FOR HOUSEKEEPING) IT THEN GOES INTO THE EDITOR INSERT MODE AND ASKS IF YOU WANT TO ENTER A LINE OR A CHARACTER. IF YOU WISH TO ENTER A LINE IT BRANCHES TO LINE 550 OTHERWISE IT CONTINUES ON.
LINE 280:
CHECKS TO SEE IF YOU WANT A CHARACTER EDIT--IF SO THEN IT GOES ON TO LINE 300.
LINE 290:
IF THE PROGRAM GETS TO HERE THE PROGRAM ABORTS AND GOES BACK TO PROGRAM CONTROL ON LINE 110.

EDIT CHARACTER INSERT (300-360):
LINE 300:
D$ IS SET TO THE NULL STRING TO PREVENT ANY ERROR IN THE INPUT QUERRY WHICH OCCURS LATER. THE BUFFER IS THEN FIELDED AND THE LINE AND CHARACTER POSITION FOR THE EDIT IS OBTAINED (GOSUB 830). IF THE LAST CHARACTER (TO EDIT) IS THE FIRST CHARACTER (0 IN OTHER WORDS) IT THEN PRINTS THE WARNING THAT THE EDIT IS AT THE FIRST OF THE LINE.
LINE 302:
IF THE EDIT IS TO BEGIN AT THE FIRST OF THE LINE THEN THAT LINE IS PULLED FROM THE DISC FILE AND PRINTED TOTALLY. THE CHARACTER TO BE INSERTED IS THEN REQUESTED (I$) AND THEN THAT CHARACTER AND THE ORIGINAL LINE STRINGS ARE SWAPPED (EXCHANGED) AND THE CONCATENATION OF THE TWO IS SET TO D$ AND PRINTED AS THE CORRECTED LINE. IF THAT IS CORRECT THE PROGRAM IS ALLOWED TO CONTINUE.
LINE 305:
IF THE EDIT DOES NOT BEGIN AT THE FIRST OF THE LINE THEN GOTO LINE 330 OTHERWISE CONTINUE.
LINE 310:
IF THE CORRECTED LINE AS PRINTED DOES NOT READ CORRECTLY THEN TRY AGAIN (BACK TO LINE 300) OTHERWISE PUT THE CORRECTED LINE ONTO DISC (GOSUB 840) AND RETURN TO THE PROGRAM CONTROL SECTION.

continued

BASIC Text Editor Helps Optometrist

continued

```
LINE 330:
GET THE LINE TO BE EDITED (SINCE EDIT IS NOT AT THE BEGINNING OF
THE LINE) AND SET H$ EQUAL TO THE LINE. THEN SET J$ EQUAL TO
THE LINE AFTER THE EDIT IS TO OCCUR (H$ IS ALL BEFORE THE EDIT)
THEN PRINT H$ (LINE BEFORE EDIT) + \" "\+J$ (ALL AFTER EDIT).
LINE 340:
IF THE LINE AS PRINTED BY LINE 330 IS CORRECT THEN CONTINUE.
OTHERWISE GO BACK TO LINE 300 AND TRY AGAIN.
LINE 350:
H$ (BEGINNING OF LINE) IS PRINTED AND THE ALTAIR THEN WAITS
FOR THE CORRECTIONS TO BE TYPED IN (I$). A CHECK IS THEN MADE
TO DETERMINE IF THE EDIT OCCURED AT THE END OF THE LINE AND IF
SO J$( LAST OF LINE AFTER EDITED CHARACTER) IS SET TO THE NULL
STRING.
LINE 360:
D$ IS THEN SET TO THE H$+I$+J$ CORRECTED LINE AND PRINTED. IT
IS THEN PUT ON DISC AND RETURNED TO THE PROGRAM CONTROL.

CHARACTER CHANGE (370-400):
LINE 370:
THE FILE IS OPENED AND THE BUFFER IS FIELDED. A REQUEST FOR THE
LINE TO BE EDITED IS THEN MADE (LN). THAT LINE IS THEN OB-
TAINED FROM DISC AND PRINTED. THEN YOU ARE ASKED TO SUPPLY THE
WORD WHICH NEEDS CHANGED (SP$). USING THE SUBSTRING FUNCTION
(INSTR) Z IS THEN SET TO INDICATE THE NUMERICAL POSITION OF THE
OCCURANCE OF THE WORD TO BE EDITED.
LINE 380:
IF THE WORD IS THE FIRST WORD THEN AA$ (USED LATER TO PUT THE
CORRECTED LINE BACK TOGATHER) IS SET TO THE NULL STRING OTHER-
WISE AA$ IS SET TO THE LINE PREVIOUS TO THE OCCURANCE OF THE
EDITED WORD.
LINE 390:
AC$ IS SET TO THE LINE FOLLOWING THE OCCURANCE OF THE EDITED
WORD.
LINE 400:
THEN THE LINE WITH THE WORD TO BE EDITED CONTAINED BETWEEN SLASH
MARKS IS THEN PRINTED AND THE CORRECT WORD (OR CHARACTERS) IS
REQUESTED AND PUT IN I$. THE LINE WITH THE CORRECTION IS THEN
PRINTED AS D$, PUT ON DISC. AND RETURNED TO PROGRAM CONTROL.

EDITOR DELETE (410-440):
LINE 410:
REQUESTS YOU TO STATE A LINE OR CHARACTER DELETION (LINE=LI
AND CHARACTER=CH). IF IT IS A LINE GOTO 450 OTHERWISE IF IT
IS A CHARACTER THEN CONTINUE TO LINE 420.
LINE 420:
GETS THE LINE AND CHARACTER NUMBER FOR THE DELETION AND THEN
ASKS HOW MANY TO DELETE. THEN THE PROPER LINE IS TAKEN FROM THE
DISC FILE. IF THE CHARACTER TO BE DELETED IS THE FIRST CHAR-
ACTER THEN I$ IS SET TO THE NULL STRING AGAIN OTHERWISE I$ BE-
COMES THE FIRST PART OF THE STRING BEFORE THE DELETION IS TO
OCCUR.
LINE 430:
P$ IS SET TO EQUAL THE CHARACTER TO BE DELETED AND L$ IS SET TO
THE LINE FOLLOWING THE DELETION. THE LINE WITH THE CHARACTER
YOU REQUESTED DELETED IS THEN PRINTED WITH THE DELETED CHARAC-
TER SET BETWEEN SLASHES (\). IF THE LINE IS CORRECT WITH THE
CHARACTER INDICATED DELETED THEN THE CORRECTED LINE IS PUT ON
DISC AND RETURNS TO PROGRAM CONTROL (IN THE NEXT LINE 440).

LINE DELETE (450-540):
LINE 450:
GOSUB 900 CAUSES THE TEXT TO BE PLACED IN ITS ENTIRETY INTO A
MATRIX (M$(Y)). Z (USED AS A CHARACTER COUNTER) IS SET TO 1
AND YOU ARE THEN ASKED WHICH LINE TO DELETE (L). THEN L IS
INCREMENTED BY ONE AND THEN CHECKED TO SEE IF THE LINE IS GR-
EATER THAN THE NUMBER OF LINES IN THE TEXT. IF SO THE PROGRAM
IS ABORTED BACK TO PROGRAM CONTROL. OTHERWISE THE LINE (M$
(L)) IS PRINTED AND YOU ARE ASKED IF THAT IS THE CORRECT LINE.
IF IT IS THEN THE PROGRAM CONTINUES.
LINE 460:
MAKES SURE THAT THE ANSWER IS YES AND IF NOT ERASES THE M$ AND
RESTARTS AT LINE 450.
LINE 470:
IF THIS IS THE LAST LINE THIS LINE CAUSES A SKIP TO LINE 540.
LINE 480:
MOVES THE LINES IN THE MATRIX DOWN (LINE 40 BECOMES LINE 39 ETC)
FOR THE LENGTH OF THE ENTIRE TEXT.
LINE 490:
QUERIES TO SEE IF YOU WANT TO DELETE THE NEXT LINE ALSO--THIS
CAN CONTINUE TO THE END OF THE TEXT IF YOU SO DESIRE.
LINE 500:
IF YOU DO WISH TO DELETE THE NEXT LINE ALSO THEN THIS LINE DEC-
REMENTS THE PROPER LINE COUNTERS TO ELIMINATE THAT NEXT LINE
AND THEN GOES BACK TO LINE 470 TO DETERMINE IF THE END OF TEXT
HAS BEEN REACHED--IF NOT ALL THE LINES ARE AGAIN MOVED DOWN.
LINE 510:
FIELDS THE RANDOM BUFFER GETS THE FIRST RECORD OF THE FILE AND
DECREMENTS THE TOTAL NUMBER OF LINES CONTAINED IN THE LINE COU-
NTER FLAG (VAL(C$)) BY THE VALUE OF Z (NUMBER OF LINES DELETED).
THAT IS THEN PUT INTO THE FIRST RECORD OF THE FILE. THE FIELD
IS THEN CHANGED IN LINE 810.
LINE 520:
PUTS THE MATRIX AS MODIFIED ONTO THE DISC.
LINE 530:
CLEARS M$ (FREES UP THE MEMORY) AND RETURNS TO PROGRAM CONTROL.
LINE 540:
USED TO INCREMENT THE Y COUNTER AND RETURNS TO LINE 510.

NEW LINE INSERT (550-730):
LINE 550:
SETS UP THE MATRIX FUNCTION AGAIN.
LINE 560:
GETS THE APPROPRIATE LINE AND CHARACTER NUMBER FOR THE LINE TO
BE INSERTED.
LINE 570:
MAKES CERTAIN THAT YOU HAVE THE CORRECT LINE OR CAUSES THE INSERT
TO BE ABORTED.
LINE 580:
FIELDS THE BUFFER AND GETS THE FIRST RECORD OF THE TEXT.
LINE 590:
CHECKS TO SEE IF YOU ARE TRYING TO INSERT A LINE AFTER THE LAST
LINE OF THE TEXT. IF YOU ARE IT TELLS YOU TO USE THE CONTINUE
PROGRAM RATHER THAN THE INSERT FUNCTION OF THE EDITOR. THAT
CAN BE VERY HELPFUL BECAUSE YOU HAVE NO GUIDES IN THE EDITOR
FUNCTIONS AS TO WHEN THE LINE CAN OR CANNOT BE PROPERLY JUST-
IFIED.
LINE 600:
MAKES SURE YOU HAVE THE CORRECT INFORMATION AND DESIRE TO CONT-
INUE WITH THE INSERTION OF A NEW LINE.
LINE 610:
INCREMENTS Y (WILL NOW HAVE MORE LINES OF TEXT THAN BEFORE AS
YOU ARE INSERTING MORE LINES) AND THEN EXPANDS THE MATRIX
ACCORDINGLY.
LINE 620:
SIGNALS THAT THE MATRIX AND COMPUTER ARE NOW READY TO ACCEPT
THE NEW LINE FOR INSERTION INTO THE MATRIX (D$=NEW LINE).
LINE 630:
REPRESENTS THE ACTUAL INSERTION OF THE NEW LINE (THE NEW LINE
MAY CONTAIN ANY CHARACTER).
LINE 640:
ADDS A CARRIAGE RETURN (CHR$(13)) TO THE INPUTED LINE. THIS IS
NECESSARY TO BE PROPERLY PRINTED OUT AT A LATER TIME.
LINE 650:
SETS EVERYTHING UP TO BE PLACED ONTO THE DISC IN THE FIRST REC-
ORD (FILE MANAGEMENT RECORD #1).
LINE 660:
PUTS THE MATRIX INTO PROPER ORDER SO AS TO NOT ACCIDENTLY RE-
PLACE THE WRONG LINE RATHER THAN INSERTING A LINE.
LINE 670:
INCREMENTS THE COUNTER AND CHECKS TO SEE IF YOU HAVE YET ARRIVED
AT THE LAST LINE OF THE TEXT. (POSSIBLY NOT NEEDED IN THIS CASE).
LINE 680:
SETS UP THE RANDOM BUFFER TO ACCEPT FOR INPUT THE MATRIX WITH
THE NEW LINE INSERTED.
LINE 690:
PREPARES THE MATRIX TO BE INSERTED ONTO THE DISC.
LINE 700:
PUTS THE MATRIX LINE BY LINE ONTO THE DISC.
LINE 710-720:
ERASES THE MATRIX AND RETURNS TO PROGRAM CONTROL.

EDIT LIST (LINE 740):
LINE 740:
THE ONLY LINE OF THE EDIT LIST FUNCTION OF THE EDITOR ASKES IF YOU
WANT THE LISTING ON THE T.V. (PORT 0 OF MY 2-SIO) OR THE PRINTER
(PORT 1 OF MY 2-SIO) SETS W TO A POSITIVE VALUE AND TO PRINT 7
LINES OF NUMBERS REPRESENTING THE HEDDER FOR THE EDITOR LISTING
FUNCTION. THEN THE PROGRAM BRANCHES TO LINE 140 AND PRINTS THE
TEXT WITH LINE NUMBERS (REMEMBER W IS NOW NOT A NEGATIVE VALUE SO
THE PRINT FORMATTING OF PF$ FUNCTIONS).

EDIT SEARCH SECTION (750-770):
LINE 750:
ASKS FOR YOU TO GIVE THE PHRASE YOU WISH TO HAVE LOOKED FOR IN
THE TEXT. SETS UP THE FILES AND GETS THE FIRST RECORD TO DE-
TERMINE HOW MANY LINES OF TEXT TO SEARCH WITHIN.
LINE 760:
THIS LINE GETS EACH LINE OF TEXT AND SEARCHES IT FOR THE SEARCH
PHRASE. IF IT FINDS THE PHRASE TO BE PRESENT Z RETURNS A VALUE
GREATER THAN 0 AND THE PHRASE IS PRINTED AS BEING FOUND ON THAT
LINE AT THE POSITION INDICATED BY Z.
LINE 770:
GETS THE NEXT LINE UNTIL ALL LINES ARE SEARCHED. WHEN COMPLET-
ED THE PROGRAM RETURNS TO PROGRAM CONTROL.

FINAL DRAFT CONSOLE (780):

LINE 780:
THIS SINGLE LINE SIMPLY SWITCHES TO THE PRINTER (MY CONSOLE
ON PORT 1 2-SIO) AND GOES TO THE READ ROUTINE ON LINE 140 TO
PRINT THE TEXT ON THE PRINTER. (CONSOLE 18,1 SPECIFIES 1 STOP
BIT--YOU MAY HAVE TO CHANGE IT TO SUIT YOUR SYSTEM).

SUBROUTINES GENERALLY FINISH OUT THE PROGRAM:
LINES 800-810:
FIELD STATEMENTS FOR THE RANDOM BUFFER TO SET UP THE PROPER
STRING SPACE.
LINE 820:
CLOSES ALL FILES AND REOPENS THE FILE NAMED AS THE LETTER (A$)
IN INITIALIZATION.
LINES 830-890:
THESE SHOULD BE SELF-EXPLANITORY.
LINES 900-920:
LOAD THE LETTER INTO A MATRIX FOR LINE INSERTION AND DELETION.
```

```
LINES 940-1020:
ENABLE THE FORMATTER TO LOOK AT THE BEGINNING OF A LINE AND DE-
TERMINE IF THERE ARE MORE THAN FIVE LEADING SPACES. IF SO THEN
PRINT THEM OTHERWISE THE TEXT IS SET FOR BLOCK PRINTING. L$ IS
USED BY THE FORMATTER AS WELL AS X AND C TO KEEP TRACK OF THE
TOTAL NUMBER OF CHARACTERS ON THE LINE AND THE NUMBER OF SPACES
WHICH NEED TO BE FILLED IN ORDER TO RIGHT JUSTIFY THE LINE.
LINE 1030:
CLOSES THE PROGRAM AND RESTORES MEMORY
LINE 1090:
ALLOWS YOU TO SET YOUR OWN DEFALTS FOR MARGIN AND LINE WIDTH.
LINE 1150:
LP IS THE NUMBER OF LINES PER PAGE BEFORE PAGING WILL OCCUR.
SET THIS FOR THE NUMBER OF LINES YOU DESIRE FOR PAGING. YOU
MAY ALSO THEN HAVE TO CHANGE THE NUMBER 67 IN THE FOR-NEXT LOOP
ON THAT SAME LINE TO GET IT TO PROPERLY PAGE.
```

If your system does not consist of an Altair 8800 with at least 28K bytes of memory, Altair 2SIO board, ADM-3 (on port), Decwriter II (on port 1), and an Altair Floppy Disk and doesn't use Altair Disk BASIC, you'll probably have to make the following changes in your system.

1. Change the Wait 16,1 on line 60 as well as INP(17) to the address of the input terminal you plan to use.
2. Change the console commands on lines 110, 780, 870, 880, and 1050 to the correct type for your terminal(s).

Remember, initialization of the default at line 1090 will set your normal line width and left margin. So typing 0 to the width ? will set everything up as required. The paging on line 1150 (LP) can be changed from 55 to any other number of lines per page required.

```
B       =       # CHARACTERS PER LINE (IF LESS THAN 5 SPACES=BLOCK)
S1$     =       # SPACES ON LEFT MARGIN
X2      =       PAGE COUNTER
A$      =       LETTER NAME
Y       =       VAL(C$)=#LINES IN TEXT
X       =       VAL(E$)=# CHARACTERS/LINE
F$      =       INPUT STRING FOR EACH LINE
D       =       EACH ASCII INPUT
LL      =       LINE WIDTH
B$      =       LEFT SET F$ AS IS 'PUT'
W       =       FLAG FOR EDIT LIST (IF <0 THEN IS EDIT LIST)
F       =       FUNCTION INPUT
PF$     =       PRINT USING STRING FOR EDIT LIST FUNCTION
C       =       # CHARACTERS/LINE LESS THAN TOTAL WIDTH REQUESTED
P       =       UNIT IN FOR-NEXT LOOP
P$      =       SINGLE CHAR IN FORMATTED LINE
G       =       INPUT FOR TYPE OF EDIT
H$      =       INPUT FOR EDITOR LINE OR CHARACTER
LC      =       LAST CHARACTER IN EDITOR
LN      =       LINE NUMBER IN EDITOR
I$      =       NEW CHARACTER
D$      =       NEW LINE AFTER CONCATENATION
J$      =       1ST PART OF LINE BEFORE EDITED CHARACTER
SP$     =       SEARCH PHRASE
P$      =       DELETED CHARACTER
L$      =       ALL AFTER DELETED CHARACTER
I$      =       ALL BEFORE DELETED CHARACTER
M$      =       MATRIX FUNCTION
Z       =       INSTR(B$,SP$)
QQ$     =       REQUEST FOR FORMATTED OUTPUT
LP      =       LINES/PAGE
A       =       #SPACES ON LEFT MARGIN
Z1      =       FREE STRING DURING FORMATTING
Q$      =       QUERRY USED IN LINE 300.
AA$     =       1ST PART OF LINE BEFORE EDIT CHARACTER CHANGE
AC$     =       2ND PART OF LINE AFTER EDIT CHARACTER CHANGE
L       =       QUERRY FOR LINE OR CHARACTER
Z       =       VALUE RETURNED BY INSTR (EDIT SEARCH)
```

```
10 REM                  DISC BASIC 4.0 TEXT EDITOR
                        WRITTEN BY:
                          DR. JAMES F. MORRISON
                          P.O. BOX 341
                          COLBY, KANSAS 67701


20 CLEAR 400:X=FRE(A):IF X>32767 THEN CLEAR 32767ELSECLEAR X:
   Z1=FRE(""):X=2:LINE INPUT"WHAT IS THE NAME OF THE LETTER "$A$:
   GOSUB 1080:WIDTH 132:GOSUB 820:GOSUB 800:GET1,1:Y=VAL(C$):
   GOSUB 1130:CLOSE:GOTO 110:REM

                        WRITE SECTION


30 Y=2:GOSUB 820:GOSUB 800
40 PRINT CHR$(7):X=0:F$=STRING$(128,0)
50 X=X+1
60 WAIT 16,1:D=INP(17):D=DAND127:MID$(F$,X,1)=CHR$(D):
   IF X=>LL-INT(LL/5)+2 THEN PRINT CHR$(7)
65 IF X=LL THEN PRINT:
   PRINT"OVERTYPED RIGHT MARGIN IN A "LL" WIDTH LINE--RETYPE":
   X=1:F$=STRING$(128,0):GOTO 60
70 IF D=64 THEN PRINT CHR$(8):X=X-1:GOTO 60
80 IF D=27 GOTO 100
90 PRINT CHR$(D):IF D=13 THEN GOTO 100ELSE GOTO 50
100 LSET B$=F$:LSETC$=STR$(Y):LSET E$=STR$(X):PUT 1,1:
   PUT 1,Y:GOSUB 1120:Y=Y+1:IF D=27 THEN GOTO 110:
   ELSE PRINT:GOTO 40:REM

                        PROGRAM CONTROL


110 W=-1:CONSOLE 16,1:PRINT:F=0:INPUT"FUNCTION "$F:
    PRINTON F GOTO 20,140,30,240,1030,
    250,1040,190
120 IF F=9 THEN GOSUB 1080:GOTO 110
130 PRINT"FUNCTIONS ARE: ":PRINT"1.NEW LETTER":PRINT"2.READ":
                                                PRINT"3.WRITE":
    PRINT"4.CONTINUE WRITING": PRINT"5.END":PRINT"6.EDIT":
                                              PRINT"7.FINAL DRAFT":
    PRINT"8.FORMATTED OUTPUT":PRINT"9.CHANGE WIDTH":GOTO 110
    REM
                        READ SECTION


140 X2=0:GOSUB 820:GOSUB 800:GET 1,1:Y=VAL(C$):PF$="###"
150 FOR X=2 TO Y-1:X2=X2+1:GOSUB 1150:IF W=>0 GOTO 160 ELSE GOTO 170
160 PRINT USING PF$;X-1;:PRINT"    ";
170 GET 1,X:B=VAL(E$):IF B=0 THEN B=1
180 PRINT LEFT$(B$,B):NEXT:GOTO 110: REM

                        FORMATTED PRINTING


190 X2=0:GOSUB 820:GOSUB 800:GET 1,1:Y=VAL(C$):X=1:
    FOR Z=2 TO Y-1:GET 1,Z:B=VAL(E$):GOSUB 940:C=LL-B:
    IF B<LL THEN IF B>LL-INT(LL/5) GOTO 210
200 X2=X2+1:GOSUB 1150:PRINTS1$;:PRINT LEFT$(B$,B):NEXT:GOTO 110
210 GOSUB 930:X2=X2+1:GOSUB 1150
220 PRINTS1$;:FOR P=X TO LL:P$=MID$(B$,P,1):PRINT P$;:
    IF P$=CHR$(32) THEN IF C>1 THEN PRINT" ";:C=C-1
230 NEXT:PRINT:NEXT:GOTO 110:REM

                        CONTINUE WRITING SECTION


240 GOSUB 820:GOSUB 800:GET1,1:Y=VAL(C$):GOSUB 1120:GOTO 40:REM

                        EDITOR SECTION


250 PRINT:G=0:INPUT"TYPE OF EDIT "$G:ON G GOTO 270,410,740,110,750,370
260 PRINT"TYPES OF EDITING ARE: ":PRINT:PRINT"1.INSERT":PRINT"2.DELETE":
    PRINT"3.LIST":PRINT"4.END":PRINT"5.SEARCH":PRINT"6.CHANGE":GOTO250
270 GOSUB 820:GOSUB 800:GET 1,1:PRINT"LINES IN TEXT="VAL(C$)-2:
    INPUT"INSERT A LINE OR CHARACTER "$H$:IF LEFT$(H$,1)="L" GOTO 550
280 IF LEFT$(H$,1)="C" THEN GOTO 300
290 GOTO 110:REM

                        EDIT CHARACTER INSERT


300 Q$="":GOSUB800:GOSUB 830:
    IF LC=0 THEN PRINT"INSERT STARTS AT FIRST CHARACTER POSITION:"
302 IF LC=0 THEN GET 1,LN:H$=LEFT$(B$,VAL(E$)):PRINTH$:LINE INPUTI$:
                                                SWAP H$,I$:D$=H$+I$:
    PRINT "LINE NOW READS AS: "D$:INPUT"CORRECT "$Q$
305 IF LC>0 THEN GOTO 330
310 IF Q$<>"Y" THEN GOTO 300 ELSE GOSUB 840:GOTO 110
330 GET1,LN:H$=LEFT$(B$,LC):
    J$=MID$(B$,LEN(H$)+1,VAL(E$)-LEN(H$)):PRINTH$"\" "\"J$
340 INPUT"CORRECT "$Q$:IF LEFT$(Q$,1)<>"Y" THEN GOTO 300
350 PRINTH$;:LINE INPUT I$:IF VAL(E$)-LEN(H$+I$)<0 THEN J$=""
360 D$=H$+I$+J$:PRINT D$:GOSUB 840:GOTO 110:REM

                        CHARACTER CHANGE


370 GOSUB 820:GOSUB800:INPUT"WHICH LINE "$LN:LN=LN+1:
    GET 1,LN:PRINT B$:
    LINE INPUT"WHICH WORD (SPELL EXACTLY AS IN ERROR) "$SP$:
    Z=INSTR(B$,SP$)
```

BASIC Text Editor Helps Optometrist

continued

```
380 IF Z=1 THEN AA$="" ELSE AA$=LEFT$(B$,Z-1):
390 AC$=MID$(B$,Z+LEN(SP$),VAL(E$)-Z+1-LEN(SP$))
400 PRINT AA$+"\"+SP$+"\"+AC$:LINE INPUT "CORRECT WORD= ";I$:
    PRINT"LINE "LN-1" NOW READS AS: ":D$=AA$+I$+AC$:
    PRINT D$:GOSUB 840:GOTO 110:REM

                        EDITOR DELETE


410 INPUT"DELETE A LINE OR CHARACTER ";H$:
    IF LEFT$(H$,1)="L"THEN GOTO 450 ELSE IF LEFT$(H$,1)<>"C"THENGOTO110
420 GOSUB 830:GOSUB820:INPUT"HOW MANY TO DELETE ";N:
    GET1,LN:IF LC=0 THEN I$="" ELSE I$=LEFT$(B$,LC):GOTO 430
430 P$=MID$(B$,LEN(I$)+1,N):L$=MID$(B$,LEN(I$)+N+1,VAL(E$)-(LEN(I$)+N)):
    PRINTI$+"\"+P$+"\"+L$:INPUT"CORRECT ";Q$:
    IF Q$ <> "Y" GOTO 420 ELSE D$=I$+L$:GOSUB 840
440 CLOSE:GOTO 110:REM

                        LINE DELETE


450 GOSUB 900:Z=1:INPUT"WHICH LINE TO DELETE ";L:
    L=L+1:IF L>Y THEN GOTO 110 ELSE PRINT M$(L):INPUT"CORRECT LINE ";Q$:
460 IF LEFT$(Q$,1)<>"Y" THEN ERASE M$:GOTO 450
470 IF L>Y THEN 540
480 FOR X=L TO Y:M$(X)=M$(X+1):NEXT
490 Q$="":INPUT"NEXT LINE ALSO ";Q$
500 IF LEFT$(Q$,1)="Y" THEN Y=Y-1:Z=Z+1:GOTO 470
510 GOSUB 800:GET 1,1:LSET C$=STR$(Y-Z):PUT 1,1:GOSUB 810
520 FOR X=2 TO Y:LSET M$=M$(X):PUT 1,X:NEXT
530 ERASE M$:GOTO 110
540 Y=Y+Z+1:GOTO 510:REM

                        NEW LINE INSERT


550 GOSUB 900
560 INPUT"INSERT NEW LINE AFTER WHICH LINE ";X:X=X+1:
    PRINT M$(X):INPUT"CORRECT LINE ";Q$:IF Q$="N" THEN GOTO 560
570 IF Q$<>"Y" THEN GOTO 710
580 GOSUB 800:GET1,1
590 IF X+1=>Y THEN PRINT"CAN'T INSERT ON LAST LINE USE CONT.":GOTO 110
600 INPUT"CONTINUE ";Q$:IF Q$<>"Y" THEN GOTO 680
610 Y=Y+1:FOR Z=Y TO X+1 STEP -1:M$(Z)=M$(Z-1):NEXT
620 PRINT "START NEW LINE NOW:":D$=""
630 LINE INPUT D$
640 D$=D$+CHR$(13)
650 LSET B$=D$:LSET C$=STR$(Y):LSET E$=STR$(LEN(D$)):PUT 1,1
660 M$=B$+C$+E$:M$(X+1)=M$
670 X=X+1:GOTO 590
680 GOSUB 810
690 FOR X=2 TO Y:LSET M$=M$(X)
700 PUT 1,X:NEXT
710 ERASE M$
720 GOTO 110
730 REM

                        EDIT LIST


740 GOSUB 860:PRINT"       ";:FOR W=1 TO 7:
    PRINT"123456789 ";:NEXT:PRINT:W=0:GOTO140:REM

                        EDIT SEARCH SECTION


750 INPUT"SEARCH PHRASE: ";SP$:GOSUB 820:GOSUB 800:GET1,1:Y=VAL(C$)
760 FOR X=1 TO Y:GET 1,X:Z=INSTR(B$,SP$):
    IF Z>0 THENPRINT "'"SP$"' " CAN BE FOUND ON LINE "X-1" POSITION #"Z:
770 NEXT:GOTO 110:REM

                        FINAL DRAFT CONSOLE


780 CONSOLE 18,1:GOTO 140
790 REM

                        SUBROUTINES


800 FIELD 1,117 AS B$,6 AS C$,5 AS E$:RETURN
810 FIELD 1,128 AS M$:RETURN
820 CLOSE:OPEN"R",1,A$,0:RETURN
830 INPUT"ON WHICH LINE ";LN:LN=LN+1:
    INPUT "AND AFTER WHICH CHARACTER ";LC:RETURN
840 LSET E$=STR$(LEN(D$)):LSET B$=D$:PUT 1,LN:IF LC>0 THEN LC=0
850 RETURN
860 INPUT"WANT PRINTED ON THE TV (1) OR THE PRINTER (2) ";C
870 IF C=1 THEN CONSOLE 16,1:PRINT:RETURN
880 IF C=2 THEN CONSOLE 18,1:PRINT:RETURN
890 GOTO 860:REM

                        LOAD MATRIX SUBROUTINE


900 GOSUB820:GOSUB 800:GET 1,1:Y=VAL(C$):GOSUB 1120: DIMM$(Y+5):GOSUB 810
910 FOR X=1 TO Y:GET 1,X:M$(X)=M$:NEXT
920 RETURN
930 X=1:F=0
940 IF B=<1 THEN B=2
950 IF MID$(B$,B-1,1)=" " THEN B=B-1
960 IF LEFT$(B$,1)=" " THEN L$=" ":X=2:C=C+1
970 IF LEFT$(B$,2)="  " THEN L$="  ":X=3:C=C+1
980 IF LEFT$(B$,3)="   " THEN L$="   ":X=4:C=C+1
990 IF LEFT$(B$,4)="    " THEN L$="    ":X=5:C=C+1
1000 IF LEFT$(B$,5)="     " THEN L$="     ":X=6:C=C+1
1010 IF LEN(L$)=0 THEN L$=""
1020 RETURN
1030 CLEAR 100:END
1040 Q0$="":INPUT"WANT FORMATTED OUTPUT ";Q0$
1050 IF Q0$="Y" THEN CONSOLE 18,1:GOTO 190
1060 IF Q0$="N" THEN GOTO 780
1070 PRINT"A YES (Y) OR NO (N) WILL DO FINE ":GOTO 1040
1080 INPUT"WIDTH ";LL
1090 IF LL=0 THEN LL=68:S1$="     ":RETURN
1100 INPUT"# SPACES ON LEFT MARGIN=";A:IF A<1 THEN A=1
1110 S1$=STRING$(A,32):RETURN
1120 IF Y>INT(Z1/128)-20 THEN GOTO 1125 ELSE RETURN
1125 IF INT(Z1/128)-14<=Y-2 THEN RETURN
1130 PRINTCHR$(7):PRINT"CAPACITY IS"INT(Z1/128)-14" LINES.
                                YOU HAVE USED "Y-2" LINES!"
1140 RETURN
1150 LP=55:IF X2=>LP THEN IF Y-X<6 THEN RETURN
                                ELSE FOR X1=1 TO 67-LP:
     PRINT:NEXT:X2=1     'LP=LINES/PAGE
1160 RETURN
OK
```

Label Program

The following label program permits the storing of 2045 labels on one disc. It's written to support a single row of gum labels. If your labels are different, you'll have to change the format accordingly. Additional labels can also be set up as a separate file or subfiles.

A variety of clubs and civic organizations to which I belong use this program to generate their mailing lists. The label program not only helps save time and improves efficiency, but it makes the lists easy to maintain and update.

Line 260 allows you to print data as a label or as a list of information. Some of the data asked for in the program will not be printed on the labels, i.e. the phone number. If you want such information printed, simply add it to the appropriate print statement on line 260.

The change section (line 300-375) requires that the entire label and information be retyped completely. This can be changed by adding an argument which specifies what part or parts of the record should be changed to the prompt at line 300.

This program can be easily altered to accomodate your system by changing the various console commands. You can also change the field subroutines and data inputs to suit your particular needs.

```
LN$     =       LABEL NAME
IN$     =       FUNCTION INPUT (R,W,C,D OR END)
X       =       VAL(NU$)--THE NUMBER OF LABELS WRITTEN
Y       =       VAL(NU$) IN THE READ SECTION
X       =       LOWEST RECORD TO BE READ IN THE READ SECTION
Q$      =       TYPE OF READOUT (R=REGULAR LABELS, L=LINE LISTING)
Z       =       CONSOLE COMMAND (HARD COPY OR ON T.V.)
M$      =       NAME ON THE LABEL (ADDRESSEE)
MM$     =       ADDRESS OF ADDRESSEE
CTZ$    =       CITY, STATE AND ZIP CODE
PH$     =       PHONE NUMBER
OT$     =       OTHER INFORMATION
P$      =       SPARE 3 CHARACTERS
NU$     =       TOTAL NUMBER OF LABELS WRITTEN AND CURRENT LABEL #

10 REM


20 CLEAR 200:WIDTH132:INPUT"WHAT IS THE LABEL NAME--";LN$:IF LN$="END" THEN END
30 OPEN"R",1,LN$
40 CONSOLE 16,1:INPUT"READ THE LABELS (R), WRITE NEW LABELS (W)CHANGE A LABEL (C) OR OPEN A DIFFERENT FILE (D) (TYPE'END' TO STOP)";
IN$
50 IF IN$="R"GOTO190
60 IF IN$="W" GOTO110
70 IF IN$="C"GOTO290
80 IF IN$="D" THEN CLOSE:GOTO20
90 IF IN$="END" THEN PRINT"PROG FILES ALL CLOSED--FINISHED":END
100 GOTO40
110 REM

                WRITE SECTION


120 GOSUB 390:GET 1,1:X=VAL(NU$)        '# LABELS WRITTEN =X
130 PRINT"HIGHEST RECORD # WAS-",X-1
140 IF X<=1 THEN X=2:
150 PRINT"READY NOW FOR--",X     '# CURRENT LABEL
160 GOSUB400:GOSUB410:GOSUB390
170 X=X+1:RSETNU$=STR$(X)
180 PUT1,1:GOTO160
190 REM

                READ SECTION


200 GOSUB 390:GET1,1:Y=VAL(NU$)
210 PRINT"HIGHEST RECORD# IS",Y:INPUT"HIGHEST TO BE READ-";Y
220 INPUT"LOWEST TO BE READ";X:IF X<=1 THEN X=2
225 INPUT"DO YOU WANT REGULAR LABELS(R) OR LISTING(L)";Q$
230 INPUT"PRINT ON TV(1) OR PRINTER(2)";Z
240 IF Z=2 THEN INPUT"POSITION PRINTER % RETURN";Q:CONSOLE 18,1
250 GOSUB 400
260 GET 1,X:IF Q$<>"R" THEN PRINT M$TAB(31)MM$TAB(62)CTZ$TAB(97)PH$TAB(112)
    OT$TAB(124)NU$ ELSE PRINT M$:PRINTMM$:PRINTCTZ$:PRINTNU$:PRINT:PRINT
270 X=X+1:IF X=>Y THEN CONSOLE 16,1:GOTO 40
280 GOTO 260
290 REM

                CHANGE SECTION


300 INPUT"WHICH RECORD DO YOU WISH TO CHANGE";X:GET1,X
310 GOSUB400:PRINTM$:PRINTMM$:PRINTCTZ$:PRINTNU$:PRINTPH$:PRINTOT$
320 INPUT"CORRECT LABEL TO CHANGE";Q$:IF Q$="END" THEN GOTO 40
330 IF LEFT$(Q$,1)="N" GOTO300
340 GOSUB410
350 GET1,X:GOSUB400:PRINT"LABEL NOW READS AS--"
360 PRINTM$:PRINTMM$:PRINTCTZ$:PRINTNU$:PRINTPH$:PRINTOT$:PRINT:PRINT
370 GOTO40
380 REM

                SUBROUTINES


390 FIELD 1,124 AS Z$,4ASNU$:RETURN
400 FIELD 1,30 AS M$,30 AS MM$,34 AS CTZ$,14 AS PH$,13 AS OT$,3 AS P$,4ASNU$:
    RETURN
410 LINEINPUT"NAME:  ";NA$:IF NA$="END" THEN GOTO40
420 LINEINPUT"ADDRESS:  ";AD$:LINE INPUT"CITY STATE AND ZIP CODE:  ";C$
422 LINE INPUT"PHONE NUMBER: ";P$
424 LINE INPUT"OTHER INFORMATION (13 CHAR): ";O$
430 LSETM$=NA$:LSETMM$=AD$:LSETCTZ$=C$:LSETPH$=P$:LSETOT$=O$:RSETNU$=STR$(X):PUT 1,X:RETURN
OK
CONSOLE 16,0
JAMES F. MORRISON, KOCVY        1770 WEST 6TH STREET            COLBY, KANSAS 67701             1-913-462-2242 1234567891023   2
JOHN M. MORRISON                SOMEWHERE-ANYWHERE              CHICAGO, ILLINOIS 77707         1-999-222-3333 MPI             3


JAMES F. MORRISON, KOCVY
1770 WEST 6TH STREET
COLBY, KANSAS 67701
   2


JOHN M. MORRISON
SOMEWHERE-ANYWHERE
CHICAGO, ILLINOIS 77707
   3
```

Run 8K BASIC Programs with Extended BASIC Conversion Technique Makes It Easy

By Robert White
8530 Stonehaven
Boise, Idaho 83704
Bus. (208) 384-3150
Home (208) 377-0336

Robert White is the CICS/VS System Software Specialist for the State of Idaho Auditor's Office. He owns an ALTAIR 8800 with 24K bytes of memory and LA-36 DECWRITER II.

Now that you've gotten Altair Extended BASIC from MITS, why don't you load up one of your old 8K version programs and try it out? Well, you're in for a surprise if you're using the ACR for program storage and don't have a Teletype™ with paper tape. Programs that have been CSAVE'd using 8K BASIC will not load in Disk BASIC and vice versa. But there is a solution.

After consulting MITS and the Altair Computer Store in Beaverton, Oregon, I decided to put out an ASCII program tape from one version and read it in on the other. The conversion technique is quite simple and can save hours of repunching programs.

The process is based on allowing the ACR to perform half of the normal terminal I/0. To output a tape, change the output side of the BASIC terminal I/0 routines to use the ACR. To reload the program, change the input side of the BASIC terminal I/0 routines to use the ACR. After either procedure, you may want to change BASIC back to do further processing. This can all be done using the following two techniques.

To Input 8K (Extended) ASCII Tapes

1. Load 8K (Extended) BASIC
2. Enter <CR>
3. Press stop on the computer.
4. Single step until you are at 2526 Q (7020 Q) where 333Q should be displayed.
5. Examine 2627Q (7021Q) (Input Status Port #).
6. Record the value.
7. Deposit 006Q.
8. Examine 2536Q (7030Q) (Input Data Port #).
9. Record the value.
10. Deposit 007Q.
11. Examine 2526Q (7020Q).
12. Press run on cassette.
13. Press run on the computer.
14. As the program is read, it will be listed. When the final "OK" is typed, press stop on the computer.
15. Press stop on the cassette.
16. Single step until you are at 2526Q (7020Q).
17. Examine 2527Q (7021Q).
18. Deposit the value recorded in step 6.
19. Examine 2536Q (7030Q).
20. Deposit the value recorded in step 9.
21. Examine 2526Q (7020Q).
22. Press run on the computer.

To Output 8K (Extended) ASCII Tapes

1. Load 8K (Extended) BASIC.
2. Enter NULLb15 <CR> .
3. Read in the program.
4. Enter the "LIST" keyword on the console. Do not enter the CR
5. Press stop on the computer.
6. Single step the computer until the address lights read 2526Q (7020 Q). The data should be a 333Q.
7. Examine 2510Q (7002 Q) (Output Status Port #).
8. Record the data.
9. Deposit 006Q.
10. Examine 2520Q (7012 Q) (Output Data Port #).
11. Record the data.
12. Deposit 007Q.
13. Examine 2526Q (7020Q).
14. Start the cassette in record mode.
15. Press run on the computer.
16. Wait about 1 minute then press run on the computer.
17. Press <CR> on the console.
18. Wait until the lights quit blinking. They should show 2537Q (7027Q).
19. Press stop on the cassette recorder.
20. Press stop on the computer.
21. Single step as in step 6.
22. Examine 2510Q (7002Q).
23. Deposit the data recorded in step 8.
24. Examine 2520 (7012Q).
25. Deposit the data recorded in step 11.
26. Examine 2526Q (7020Q).
27. Press run.

One note of caution - the addresses used were taken from version 4.0 of Altair 8K and Extended BASIC. They could change from version to version, so to double check them, here is a routine that will print the beginning addresses of the terminal input and output routines. It's based on the information given in Appendix L of the Altair 8800 BASIC Reference Manual.

By accident, I found that the terminal input area resides just prior to those addresses. If your input is too large, it will overlay the address list. So run the program right after you load BASIC and keep the input lines short. I also assumed that your ACR was addresses by parts 6,7. If they aren't, change the procedures to reflect the correct parts.

```
LIST
 1 REM PRINT TERMINAL I/0 ADDRESSES
 2 REM
 3 REM RUN THIS RIGHT AFTER YOU
 4 REM LOAD BASIC AND DON'T USE
 5 REM LARGER LINES THAN SHOWN
 6 REM OR THEY WILL OVERLAY THE
 7 REM I/0 LIST ADDRESSES.
10 DEF FNA (I)=PEEK (I)+256*PEEK (I+1)
20 B = FNA (57)
30 OA = FNA (B)
40 IA = FNA (B+2)
50 PRINT "INPUT ROUTINE=";IA;
60 PRINT "OUTPUT ROUTINE=":OA
70 END
OK
RUN
INPUT ROUTINE= 1351 OUTPUT ROUTINE = 1366
OK
```

After you've run the terminal I/0 address routine, convert the decimal addresses to octal. This can be done with the following small conversion routine.

```
 1 REM ***CONVERT DECIMAL TO OCTAL***
 5 A=O
10 INPUT "DECIMAL=";X
20 IF X <O OR X> 65536 THEN 10
30 I=O
40 C=(X-(8*INT(X/8)))
50 A = A+(C*10-I)
55 X = INT X/8)
60 IF X > O THEN I=I+1 : GOTO 40
70 PRINT "OCTAL =";A
80 END
OK
RUN
DECIMAL =? 1351
OCTAL = 2507
OK
RUN
DECIMAL =? 1366
OCTAL = 2526
OK
```

continued

Simplified Billing System . . . in BASIC for the Small Business

By Carl Denver Warren II
2980 W 235th St., Apt. 2
Torrance, CA 90505

Several months ago a small transportation company contacted me regarding a billing problem they had. They wanted a system that could take multiple entries, keep track of them, give a total of all, plus break down the totals of the different types of deliveries. There was no need to save data on tape or build data up over a period of time, as the data would be presented just prior to the billing date. They did need from 2 to 4 copies with the ability of making this change with little trouble.

After analyzing the problem, a set of specifications was drawn up and agreed upon before creating the actual program. (In application programming, this is a must and saves a lot of trouble later.) The actual program was written in MITS 8K 680 BASIC Version 1.0 Revision 3.2. The program was designed to run on the following system configuration.

a. MITS ALTAIR 680B with 17K memory.
b. SOUTHWEST AC 30 cassette interface.
c. SOUTHWEST CT 1024 terminal modified for 64 characters and scrolling.
d. AXION EX-800 electrosensitive printer, interfaced for RS-232C serial input and strapped for 1200 baud.

The program incorporates many cursor control functions to clean up the screen and prevent possible errors by displaying previously entered data. The print ASCII characters (PRINT CHR$), 2910 or 3010, is interpreted by the AXIOM printer and changes the size of type between 40 and 80 columns. This provides for maximum control of formating the output. Null is used to slow the execution time of BASIC as it is presented to the printer and prevents garbling.

The application called for 13 separate variables which were to be stored then read to the output in a first in, first out manner. Therefore, each variable was dimensioned, and the initial value set to zero. Lines 66 through 140 are the input portion of the program and is the part of the program where the work of building the array is done. The array grows until terminated by the input of 99 for a date - this was done since the date could never be less than 1 or greater than 31 and provided a means of leaving the loop. Once 99 is input for the date, the output routine which starts at line 700 takes over. The work is done in lines 903 to 945. The array is decremented each time through. However, the data is not destroyed but remains intact. When the last piece of data is read the totals are printed, the values of which are saved in the array as specified in the input lines. Lines 1500 to 1520 are a subroutine that prints the totals in the specified order. In the program I have specified 2 copies. Therefore, after the totals are printed the program then tests the value of I in lines 1310 and 1320 (which determines the number of copies) to determine where to

continued

Run 8K BASIC Programs with Extended BASIC

continued

The input routine address should be the same as the 2526Q (8K) or 7020Q (Extended) addresses mentioned in the ASCII tape procedures and should display 333Q for data. If you press EXAMINE NEXT, you should now be looking at the input status port #. Its address corresponds to the 2527 (8K) or 7021K procedure address. Now press EXAMINE NEXT about six more times. Then press it until 333Q is displayed in the data lights. Press it once more, and it will show the address of the input data port # which relates to 2536Q (8K) Or 7030Q (Extended). To find the addresses for the output side, follow the same procedure using the output routine address as the starting point. The only differences are to step about seven steps and search for 323Q when trying to locate the output data port #.

```
                                BILLING COPY
QUICK FOX TRANSPORTATION SERVICE
8921 SOUTH SEPULVEDA BOULEVARD
LOS ANGELES, CALIFORNIA 90045

AIRLINE: UNITED                 STATION: LAX

PERIOD: DEC 1 TO DEC 15

*************************************************************************

DATE    ORDER #   PICK-UP TIME   DESTINATION   DELV TIME   CHG$   CODE   BAGS

*************************************************************************

1       127486    0700           TORRANCE      1100        5      R      1
1       127495    0800           GLENDALE      1200        5      R      1
3       326145    1500           VENTURA       2355        35     SP     1
4       347898    1100           VAN NUYS      1302        10     SC     3
6       347888    1300           INGLEWOOD     1645        0      COD    3
15      491876    2200           HYATT LAX     2245        4      SP     5

2 REGS + 2    SPLS    + 1    CODS + 0    NO CHG    + 1    SVC CHG

TOTAL DELIVERIES = 6
TOTAL CHARGES = 59
TOTAL BAGS = 14

**********                      CUT                      **********
```

Fig. 1. Sample run of program.

Simplified Billing System . . . in BASIC for the Small Business

continued

send the program next. When the required amount of copies is created, the program reinitializes itself by returning to line 9, which clears all variables.

Although this program may never be of any use to anyone other than a company doing similar work, it does demonstrate one method used to save data in an array and what I feel is an interesting method of printing the final output.

```
1 REM          MULTIPLE ITEM STATEMENT
2 REM                  BY
3 REM       CARL DENVER WARREN II / AUG 1976
5 REM CLEAR STRING SPACE AND DIMENSION ARRAY FOR NAMED VARIABLES
9 NULLO:CLEAR 4000:GOSUB62
11 DIMD(100),O$(100),P$(100),D$(100),L$(100),C(100),E$(100),B(100)
15 REM INPUT NONE MATRIXED ITEMS
20 INPUT"AIRLINE: ";A$:INPUT"STATION: ";S$
30 GOSUB 62
40 INPUT"MONTH ";M$:GOSUB62:INPUT"STARTING DATE: ";W:GOSUB 64
50 INPUT"END DATE: ";U:GOTO70
55 REM SCREEN CONTROL SUBROUTINES (62) HOME CLEAR SCREEN
56 REM (64) GENERATES A SPACE
62 PRINTCHR$(22):RETURN
64 PRINTCHR$(32):RETURN
65 REM SET VARIABLES TO 0
66 A=0:C=0:Z=0:X=0:M=0:Q=0:V=0:H=0
69 REM START OF MATRIXED VARIABLES
70 INPUT"DATE: ";D(A):GOSUB62
71 IF D(A)=99THEN 790
72 IF D(A))99THEN70
73 A=A+1
80 INPUT"ORDER #";O$(C):GOSUB 62:C=C+1
90 INPUT"PICK-UP TIME: ";P$(Z):GOSUB62:Z=Z+1
100 INPUT"DESTINATION: ";D$(X):GOSUB62:X=X+1
110 INPUT"DELV TIME: ";L$(M):GOSUB62:M=M+1
120 INPUT"CHG: ";C(Q):GOSUB62:TC=TC+C(Q):Q=Q+1
130 INPUT"CODE: ";E$(V):GOSUB62
131 IFE$(V)="R"THEN R=R+1
132 IFE$(V)="SP"THENSP=SP+1
133 IFE$(V)="NC"THEN NC=NC+1
134 IFE$(V)="COD"THENCOD=COD+1
135 IFE$(V)="SC"THENSC=SC+1
139 V=V+1
140 INPUT"#OF BAGS: ";B(H):TB=TB+B(H):H=H+1:GOTO70
700 REM BEGIN OUTPUT ROUTINE
701 REM CONTROL CHR ARE FOR FORMATING OUTPUT OF AXIOM PRINTER
702 REM NULL PROVIDES SUFFICIENT TIME FOR FIFO OF PRINTER TO
703 REM CLEAR.
750 PRINTCHR$(29);TAB(32)"BILLING COPY":GOTO800
790 NULL20:PRINTTAB(36)"FILE COPY":GOTO800
800 PRINTCHR$(30);"QUICK FOX TRANSPORTATION SERVICE"
810 PRINT "8921 SOUTH SEPULVEDA BOULEVARD"
820 PRINT "LOS ANGELES, CALIFORNIA 90045":GOSUB64
830 PRINT"AIRLINE: ";A$;TAB(25)"STATION: ";S$:GOSUB64
832 PRINT"PERIOD: ";M$;" ";W;" ";"TO";" ";M$;" ";U;CHR$(29)
870 GOSUB1900:GOSUB64
880 PRINT"DATE     ORDER #      PICK-UP      DESTINATION      ";
890 PRINT"DELV   CHG$    CODE     BAGS":PRINTTAB(21)"TIME";TAB(50)"TIME"
900 GOSUB1900:GOSUB64
903 A=0:C=0:Z=0:X=0:M=0:Q=0:V=0:H=0
920 PRINTD(A);TAB(9);O$(C);TAB(21);P$(Z);TAB(34);D$(X);TAB(50);L$(M);
921 PRINTTAB(57);C(Q);TAB(65);E$(V);TAB(71);B(H)
930 A=A+1:C=C+1:Z=Z+1:X=X+1:M=M+1:Q=Q+1:V=V+1
935 H=H+1
940 IF D(A)=99 THEN 1100
945 GOTO 920
1100 GOSUB62:GOSUB64:GOSUB1500:PRINTCHR$(29)
1200 REM CUT LINE PROVIDES FOR DIVISION OF COPIES
1220 PRINT"**********";TAB(39)"CUT";TAB(70)"**********"
1221 GOSUB64
1290 REM THIS ROUTINE COUNTS THE NUMBER OF COPIES OUTPUT
1300 I=I+1
1310 IF I=1 THEN 750
1320 IF I=2 THEN 9
1400 GOSUB64
1490 REM THIS ROUTINE PROVIDES THE OUTPUT TOTALS OF ALL THE
1491 REM MATRIXED VARIABLES. EACH VARIABLE WAS INCREMENTED BY
1492 REM 1 DURING INPUT. WHEN THE OUTPUT ROUTINE IS BEGUN
1493 REM EACH ITEM IS DECREMENTED BY ONE.
1500 PRINTR;" REGS";" + ";SP;" SPLS";" + ";COD;" CODS";" + ";NC;
1501 PRINT" NO CHG";" + ";SC;" SVC CHG";GOSUB62
1510 PRINT"TOTAL DELIVERIES = ";A:PRINT"TOTAL CHARGES = ";TC
1520 PRINT"TOTAL BAGS = ";TB:RETURN
1900 FOR J=0TO79:PRINT"*";:NEXTJ:RETURN
```

Program A. Listings for Multiple Item Statement Program.

Altair 8800A

Running a rental agency can be a real headache. Keeping track of what is rented to whom and when, due dates, inventory, billing statements, and receipts is time-consuming and very confusing even if you have several people helping. But Paul Dontje, who started his own TV and microwave oven rental business in Wheat Ridge, Colorado several years ago, has found the perfect assistant to help him keep things organized -- a Altair micro-computer.

Dontje's system consists of an Altair 8800 with 32K bytes of memory, two Altair Disk Drives, a Teletype,™ an Info-Tek video generator, and a TV monitor. Dontje is very proud of the fact that neither he nor any of the other six people in his organization had any previous computer experience before he bought the system about a year ago and designed most of the software to meet his business's needs. "One of the reasons for designing the disk drive programs was so that in less than five minutes, I can show anyone how to use the system -- even if he doesn't know a thing about computers," he said. However, Dontje credits his friend Lawrence Costa, Senior Systems Analyst for the Denver Public School System, with the initial system layout and programming. Although Dontje didn't have any computer experience prior to purchasing the Altair 8800, he was a radio repairman in the military and has also taught electronics.

Dontje said his primary purpose for using the system is for automatic typing of weekly billing and delinquency statements and keeping track of customers and due dates. This is done with two statement printing programs. The first program types statements for the week, so we can mail them at least 10 days before rent is due. Every Monday the other program types delinquent statements for the past 30 days. This list allows us to contact delinquent customers every two days," he said. "When a customer's rent is two days overdue, our policy is to contact him or her by phone. When someone is seven days delinquent, we pick up the TV set or oven, or if that's not possible, we swear out a warrant for his arrest," he explained.

Dontje said the automatic typing of statements assures accurate and on-time

Keeps Rental Agency Running Smoothly

By Linda Blocki

mailing and also saves three to five hours of manual typing each week. "Customers are impressed with the computerized billing, too," he said.

After implementing the Altair computer system, Dontje discovered several other benefits it provided that he hadn't originally planned. One is that the Altair 8800 can be used to do a complete inventory list, which shows what's in stock, if it's rented, to whom, and the due date. "This list is not only necessary for our records, but also for bank records, since some of the TV sets are mortgaged. The computer saves me four hours a month in manually checking the inventory," he said.

continued

Photograph by Steve Wedeen

"The Altair 8800 saves me hours of manually checking the inventory. That's just one of several benefits I hadn't planned on when I bought the system."

MITS Training Department and Media Relations Manager Chuck Olsen discusses a billing program with store owner Paul Dontje.

Photograph by Steve Wedeen

Need a Quick Word Processing System?

By Thomas G. Schneider
MITS

One of the often overlooked features of a microcomputer system, especially the Altair computer system, is its ability to handle literal information in practical situations. For most people, the transfer of thoughts to the printed page is greatly simplified when even a simple text editor is used. Ideas may be almost instantly transferred to the computer and then saved permanently on disk. Any item of text which has been saved on disk may be easily printed on a line printer or edited for future use. Altair Disk BASIC lends itself well to the processing of literal information since it has a powerful set of functions for handling string manipulations in addition to flexible line editing features.

I wrote the following text editor when I found myself being snowed under by extensive amounts of paperwork. The program, written in Disk BASIC, is short and simple, but incorporates such important features as disk storage, line editing, and hard copy printout.

Text editor commands

The test editor will print 'READY...' when it is waiting for a command.

The commands and their functions are:

*B - Clears the text buffer and lets you start entering text. If a control command is typed at the beginning of a line, the editor will accept that command. After the command function has been executed, you will be returned to the text entry mode.

*L - Lists, with line numbers, the present contents of the text buffer

*P - Prints the contents of the test buffer on the line printer

*S - Saves the contents of the text buffer on disk

*R - Recalls text from the disk and puts it in the text buffer

*I - Lets you insert a line

*D - Lets you delete a line

*N - Lets you add text starting at the specified line number

*C - Lets you replace a specified line

To immediately edit a just-typed line, type CTRL A. This will drop you into BASIC's line editor and allow editing of that line. The BASIC manual contains a full description of the features of this editor.

Altair 8800A Keeps Rental Agency Running Smoothly continued

Another unplanned Altair 8800 time-saving application is the compiling of statistics on the rental equipment. Dontje said these statistics include: the average life of an item, how many customers rent it during that time, the company's total income for the life of the item, how many days it was rented, the company's average income per month and per customer, etc. He said this statistacal report is run on four different groups of TV sets on the 15th and the 30th of every month. "Before buying the Altair microcomputer, it took hours and hours of research to get these statistics. We did research then only rarely and on a random basis when we wanted to find out something specific," he said. "Now we can look at the statistics in a few seconds, and do a lot better analysis of our rental program," he added.

Dontje said the system is also useful for typing customer rental receipts. The Teletype™ types a receipt while a clerk rings up the payment on the cash register, he explained. "So we don't have to write out a rental receipt invoice. Customers also like having their receipts automatically typed."

Software for the system is written in Altair BASIC, version 3.4. It consists of two disk driver programs which allow access to all other programs. Dontje said the first program is call "Selector" and allows running any of the programs that read from the disk and typing them out either on the Teletype™ or the video terminal. Driver program number 2 allows accessing any of the programs that write on disk. "The reason for laying the system out this way was so that in driver number 2, we could unload and remount the disk to guard against anyone writing on an unmounted disk," he said.

There are also a "mammoth" number of subroutines, Dontje said, which include customer and item history files. "These files access each other via pointers internally in their records," he explained. "We also have an old customer history file and a sub-let TV sets file, which contains information regarding sub-lets or commercial account customers that rent groups of sets."

Future plans for the system include adding programs for service contract customers, Dontje said. "This will enable automatic billing for service contract payments." Dontje said other software includes a program for new merchandise inventory, a list of any new or used equipment purchased, and the names of people who have had major repair jobs on their electronic equipment, which will be used for solicitation of service contract customers. "This will also be used for solicitation of rentals and anything else that we want for business promotion." he added.

Overall, Dontje said he's extremely pleased with how the Altair microcomputer system has helped improve the efficiency and effectiveness of his business. "There's no question in my mind that a similar system would be very useful for anyone in any type of rental business," he said. Dontje said he figures that the cost and time of putting in a microcomputer system can be justified if the information stored in the computer is used at least four times in a year. He said that he uses the information for his business many times more often. Dontje is currently writing an article on the topic of justifying the purchase of an in-house microcomputer. He said that he hopes to have it published by one of the trade magazines in the near future.

Try This One.

```
100 CLEAR4000:I=1:DIMA$(400):PRINT:PRINT:PRINT
130 PRINT"READY...":PRINT
140 LINEINPUTA$
160 IFLEFT$(A$,2)="*L"THEN230
170 IFLEFT$(A$,2)="*P"THEN280
180 IFLEFT$(A$,2)="*C"THEN320
190 IFLEFT$(A$,2)="*S"THEN400
200 IFLEFT$(A$,2)="*R"THEN470
210 IFLEFT$(A$,2)="*B"THEN550
211 IFLEFT$(A$,2)="*N"THEN700
212 IFLEFT$(A$,2)="*I"THEN800
213 IFLEFT$(A$,2)="*D"THEN900
215 A$(I)=A$:I=I+1:CT=I-1:GOTO140
230 PRINT:PRINT:PRINT:FORK=1TOCT:PRINTK;A$(K):NEXTK
270 PRINT:PRINT:GOTO130
280 LPRINT:LPRINT:LPRINT
300 FORK=1TOCT:LPRINTA$(K):NEXTK:GOTO130
320 INPUT"LINE TO CHANGE";R
330 PRINT"PRESENT LINE IS:":PRINT:PRINTA$(R)
350 PRINT:PRINT"ENTER YOUR NEW LINE:":LINEINPUTA$(R)
370 PRINT:PRINT"NEW LINE IS:":PRINTA$(R):GOTO130
400 INPUT"FILE NAME";N$:OPEN"O",1,N$,0:PRINT#1,STR$(CT)
410 FORD=1TOCT:PRINT#1,A$(D):NEXTD:CLOSE:GOTO130
470 INPUT"FILE NAME";N$:OPEN"I",1,N$,0:INPUT#1,CT$:CT=VAL(CT$)
480 PRINT"THERE ARE ";CT$;" LINES IN THIS FILE"
490 FORD=1TOCT:LINEINPUT#1,A$(D):NEXTD:CLOSE:GOTO130
550 ERASEA$:DIMA$(400):CT=1:GOTO130:INPUT"START AT LINE #";V:I=V:GOTO130
700 INPUT"START AT LINE NUMBER";I:GOTO130
800 PRINT:INPUT"INSERT AFTER LINE NUMBER";AL:CT=CT+1
810 FORI=CTTOAL+1STEP-1:A$(I)=A$(I-1):NEXT
820 PRINT"ENTER YOUR NEW LINE":LINEINPUTA$(AL+1):GOTO130
900 PRINT:INPUT"LINE TO DELETE";LD
910 FORI=LDTOCT:A$(I)=A$(I+1):NEXT
920 CT=CT-1:GOTO130
```

GLITCHES

By Bruce Fowler
MITS

This month's column is a potpourri of information based on customer's questions over the past few months.

8800B Interface Board and 74LS20

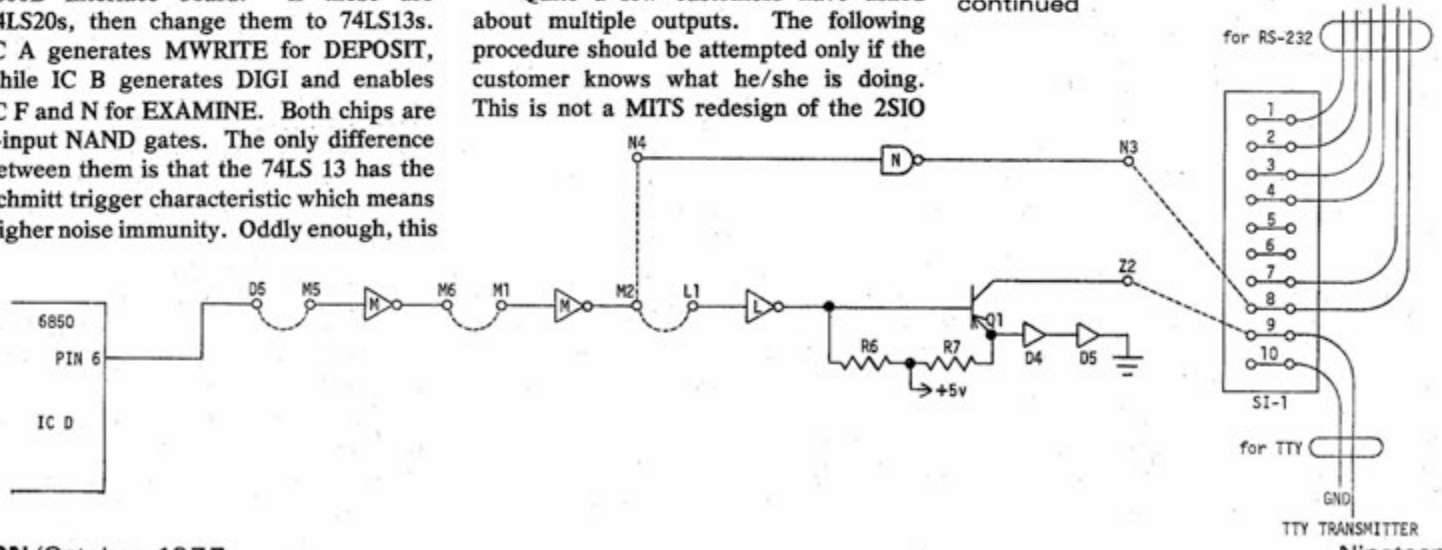
If you have to hit EXAMINE several times to read the contents of location 170008 or if you can't deposit 3778 into memory, then look at IC A and B on the 8800B Interface board. If these are 74LS20s, then change them to 74LS13s. IC A generates MWRITE for DEPOSIT, while IC B generates DIGI and enables IC F and N for EXAMINE. Both chips are 4-input NAND gates. The only difference between them is that the 74LS 13 has the Schmitt trigger characteristic which means higher noise immunity. Oddly enough, this problem only occurs when an address attaining a byte of ones (e.g. 17[illegible]0 or 000377) is examined or when a byte of ones (i.e. 377) is deposited. Change the indicated IC chips **only** if this problem appears. Kits and assembled units will use 74L 5153 for ICA and B.

Multiple Outputs from the Same 2SIO Port:

Quite a few customers have asked about multiple outputs. The following procedure should be attempted only if the customer knows what he/she is doing. This is not a MITS redesign of the 2SIO BOARD. This rewiring is not very critical, but radical changes in MITS boards go beyond MITS support.

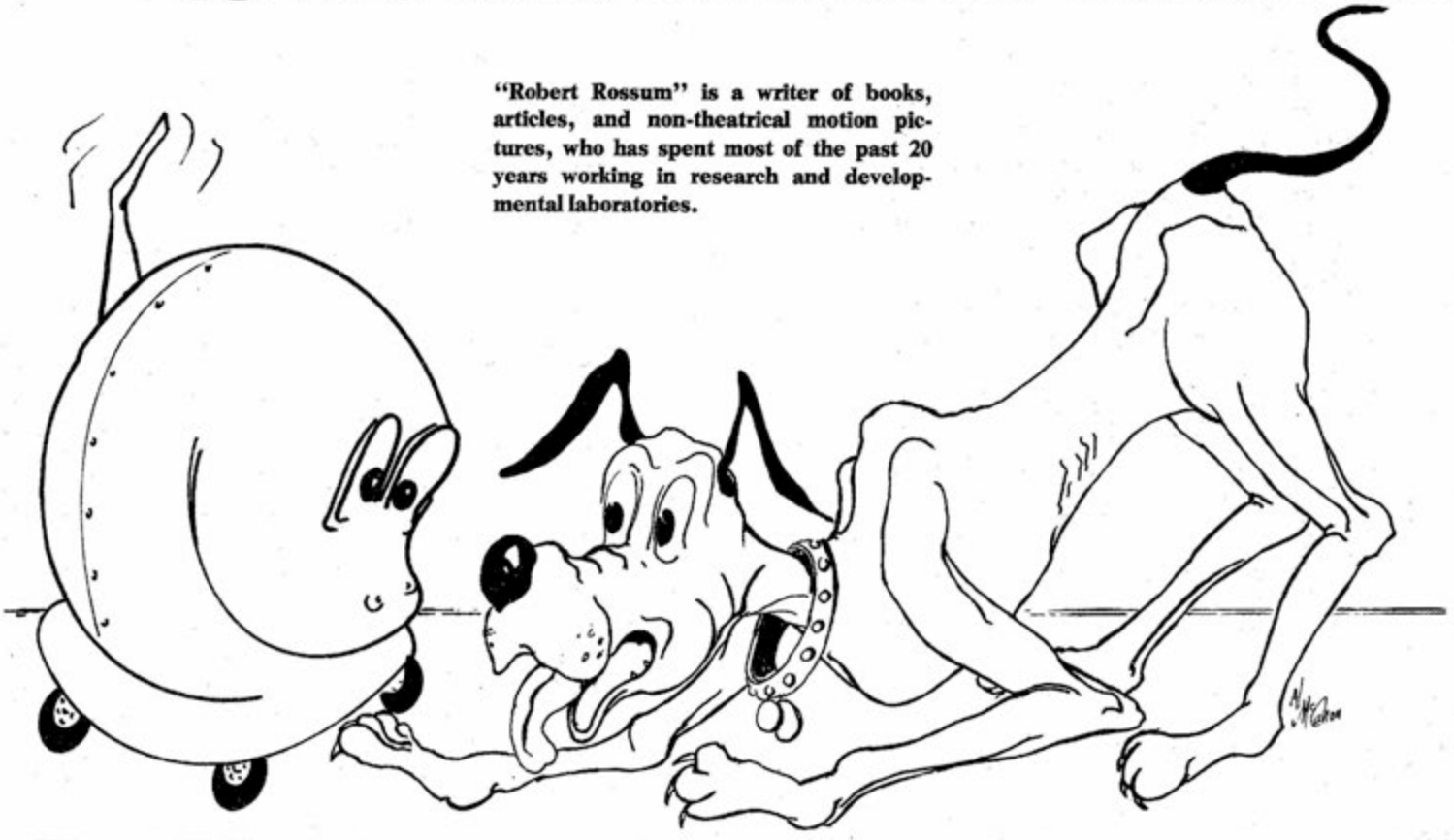
Some customers use a CRT terminal for most of their program entry and debugging, and a Teletype™ printer for the hard copy. The problem lies in trying to obtain hard copy. Consoling over to the Teletype

continued



The Patter of Little Feet
A Cheap Approach to the Mechanics of Robotics

By Robert Rossum



"Robert Rossum" is a writer of books, articles, and non-theatrical motion pictures, who has spent most of the past 20 years working in research and developmental laboratories.

GLITCHES continued

won't work because it doesn't have a keyboard for sending commands to the Altair computer. Obviously, without any commands, the Altair computer cannot output. To get around this problem with software, locate the output routines in BASIC and change the I/0 address of these output routines to that of the second port of the 2SIO. The input subroutines of BASIC would retain the I/0 address of the first 2SIO port to receive instructions from the first port (CRT). Notice that the input terminal (CRT) will not echo what is typed. To see what you type, switch the input terminal (CRT) to half duplex, so that the terminal itself echoes what you type.

Poking BASIC's LLIST routines might also work, but an extensive system is necessary to have BASIC with LLIST. Either way, a lot of poking and repoking must be done. So a hardware solution seems easier than a software one.

The 2SIO board is designed to permit changing the transmission levels. Figure 1 shows a multiple output setup for a CRT (RS-232 level terminal) and a Teletype (20 ma. current loop). the 2SIO inputs and outputs from the CRT as usual. When hard copy is needed, simply turn on the Teletype printer. Notice that whatever is echoed back on the CRT is printed on the Teletype printer at the same time. When no hard copy is desired, simply turn off the Teletype printer. The only annoying thing about this is that the CRT and the Teletype printer must be set to the same data transfer rate, which is 110 baud for most Teletypes.

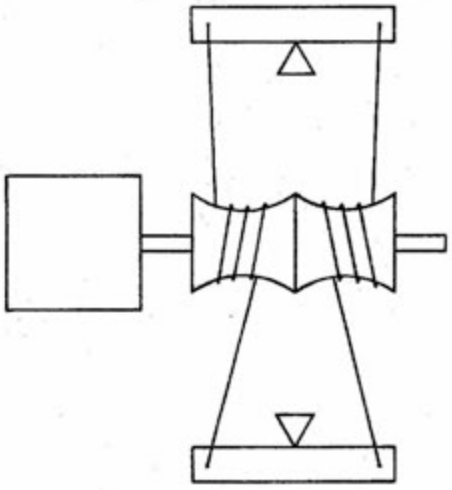
Wire the first port for RS-232, except for transmit, as shown on page 24 of the 2SIO manual. Wire transmit as shown in Figure 1. Note that if IC M is used for port 2, another free inverter must be used. Connect the TTY output from Z2 to an unused Moles pin like S1-9. Add two wires to the 10-pin female Molex connector that corresponds to this TTY output pin (S1-9) and ground (S1-10). Then connect these to the TTY receive and ground, pins 3 and 2 respectively of the 25 pin connector. Notice that the transmit output (pin 6 of the 6850 ACIA) goes through two inverters. This is because the 6850 output current capability (fanout) is not enough for both a 1488 and a 74LS04, IC N and M. Notice also that this can't be done to input on two terminals. The TTL gates would fight each other, and perhaps damage something. The same conflict occurs on the data bus when you input on two I/0 boards which have the' same I/0 address; the data itself would be erroneous.

If anyone learns how to poke BASIC's LLIST routines for a 2SI0 or BASIC's output subroutines to achieve multiple outputs with software, why not write an article about it for CN? Information about any other alterations for specific cases would also make an interesting article. Chances are, if one user has to do an alteration, dozens of others would like to do it, but don't know how.

Part II in a three-part series on building a robot.

RECAP: In Part I of this series, I described the double windlass mechanism as a simple, cheap approach to the design and construction of mobile systems by poverty-stricken non-engineers.

The basic idea is that a very small amount of energy applied to one of either Lever A or Lever B (See Figure 1) pulls the cord snug around the pulley and thereby draws energy from the main motor to move the opposite lever until tension on the cord is relieved. In this way, very small amounts of energy can be used for control of a large number of paired mechanisms along a shaft driven by a single main motor. The

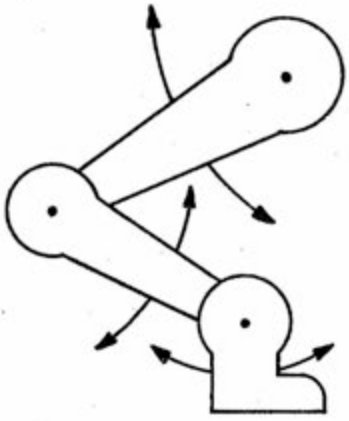


power of the motor is shared by various mechanisms as it's needed. So any particular mechanism can obtain as much energy as it needs up to the full capacity of the motor, while the average load on the main motor is comparatively low. The elements of the system are not much more complex than Tinkertoys, allowing the frustrated amateur to build his/her own experimental systems at a modest cost.

When Nature designed animals to move around, she gave them legs instead of wheels. For such animals as snails, caterpillars, and snakes, "tracks" were provided. However, complex as they are, legs are the big winners in locomotion. Wheels just don't seem to work well unless the terrain is fairly level and smooth. It seems a shame to design robots with systems Nature has rejected, but we can usually control our robots' environments more easily than we can build flexible robots.

Perhaps the most discouraging fact to those who hope to develop anti-gravity devices is that no living creatures employ them. If antigravity were possible, surely some animals would have evolved practical anti-gravity capabilities. The competitive advantage of such capabilities would be enormous. In this case, even to the born optimist, the fact that something hasn't already been done strongly suggests that it can't be done. There's no evidence either, that animals can do well on home-grown wheels. Lovable as **Star Wars'** R2D2 may be, it's hard to believe that his little wheels carried him smoothly through those sand dunes. So flapping wings, waving fins, walking legs, and perhaps a few tracks seem inevitable in the future of robots.

Since legs are the most practical, let's talk about them. We can construct the best design by copying Nature. "Maybe the only sensible thing to do," commented one engineer, "is to find a dog skeleton and make an investment casting of his legs. That's a great design." Basically, the dog's leg looks like this:



This is a simplified sketch, but it shows that only three moves are required to make the leg functional. Each section of the leg can be treated as "Lever B" in the double windlass mechanism described here. Each section can be powered and controlled individually, drawing power from the single main motor.

Power can be transmitted from the rotating main shaft to the levers by way of a system similar to an old-fashioned dentist's drill; three double-windlass mechanisms operate together to control the movement of one leg.

In this case, it may be desirable to use pulleys of different sizes on the shaft, since the longest section of the limb may need more power but less speed applied to it, while the shortest section may work best with lots of speed and limited power.

Clearly, the coordination of movements for effective use of such a leg is complex, requiring accurate timing and logical operations dependent on feedback of information from the real world. This article deals only with the principles of the mechanical system, not with the logic required. The computerist can look forward to many years of developing logic that lets the clumsy baby mechanism grow up into a coordinated artificial animal. However, the logic necessary is not mysterious; it's largely a matter of timing and sequencing, influenced by feedback.

Commentators at the 1976 Olympics talked a great deal about the improbability that a tensed-up sprinter could get away from the starting blocks in as short a time as 10 milliseconds. That's enough time for 10,000 cycles of operation of one-megaherz computer. Real-time operation seems like no great problem, especially since the robots may not need the coordination of Olympic athletes for some time.

Examining this proposed system, one enthusiast commented: "It would be the essence of simplicity to make the robot tap dance!" The remark was followed by a strong remonstrance from Glenn Norris, president of USRS, who has made a career out of conquering hideously difficult projects, producing things that wild-eyed inventors regard as trivially easy. After Norris's heated rejoinder, the enthusiast said, "Let me rephrase that. With enough effort, it could be made to tap dance."

The fact is that dogs, from whom this leg design is borrowed, are seldom good dancers. Certainly, that's not only due to lack of interest, but also to the design of the leg, which has fewer degrees of freedom in its motion than does the human leg or arm. Even so, the dog can do a number of practical things that the average robot can't do. For example, a dog can walk around a hill without toppling over or walk up a flight of steps.

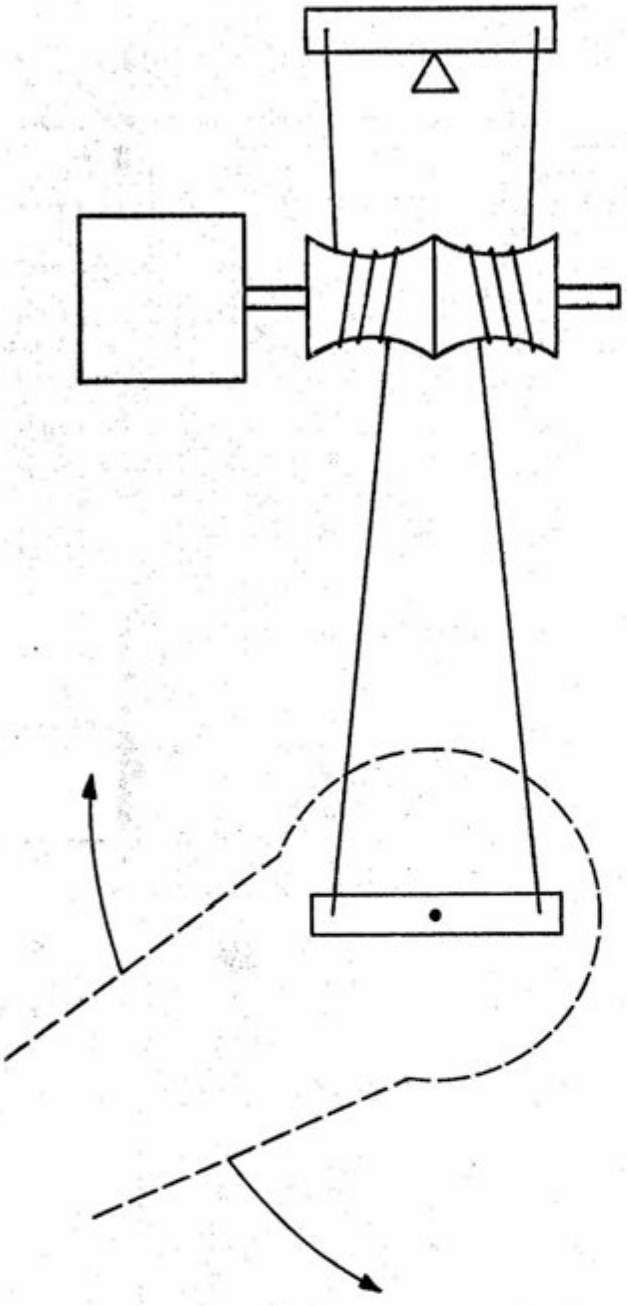
He can step over obstacles, squeeze between objects, squirm under fences. A robot might do worse than to emulate the dog. But by using this scheme, many configurations of the robot are possible.

The first important consideration is how many legs your robot needs. It's generally true that the larger and heavier

continued

The Patter of Little Feet

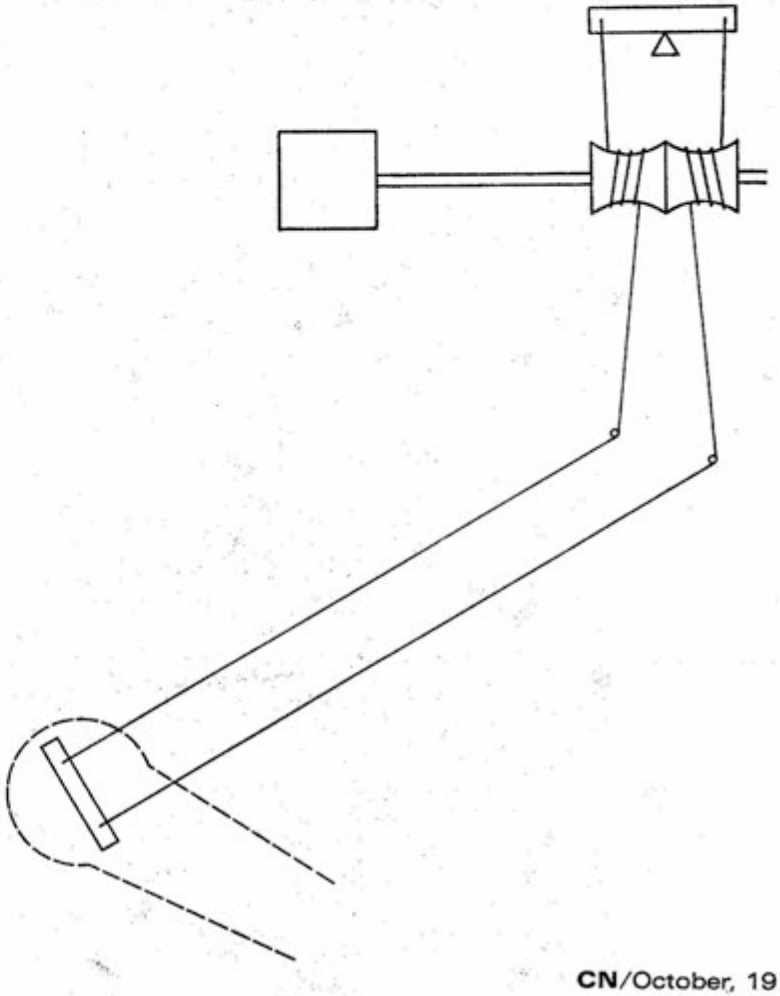
continued



an animal is, the less likely he is to have all four feet off the ground at once. An elephant moves fast, but he always has two feet on the same side on the ground at the same time. Smaller animals can more safely use a variety of gaits. While walking animals move their legs in synchrony much of the time, they tend not to move them in phase to avoid oscillation that could flip them right over.

Bugs with eight legs tend to turn like caterpillar tractors, stopping movement on one side while continuing movement on the other to cause pivoting. Of course, an animal with legs all around might be able to rotate smoothly around the center point of its body as wheeled robots (consider Papert's Turtle, Hollis's Newt) often do.

Legs-all-around may not too difficult to construct. We've already discussed the use of a flexible shaft on our main motor. It may be possible to run that shaft all the way around so that it connects with the motor again at the other end. Pulleys could be arrayed along the circular shaft so that power could be pulled off at any point. But how many legs and how many mechanisms can be fitted along the loop?



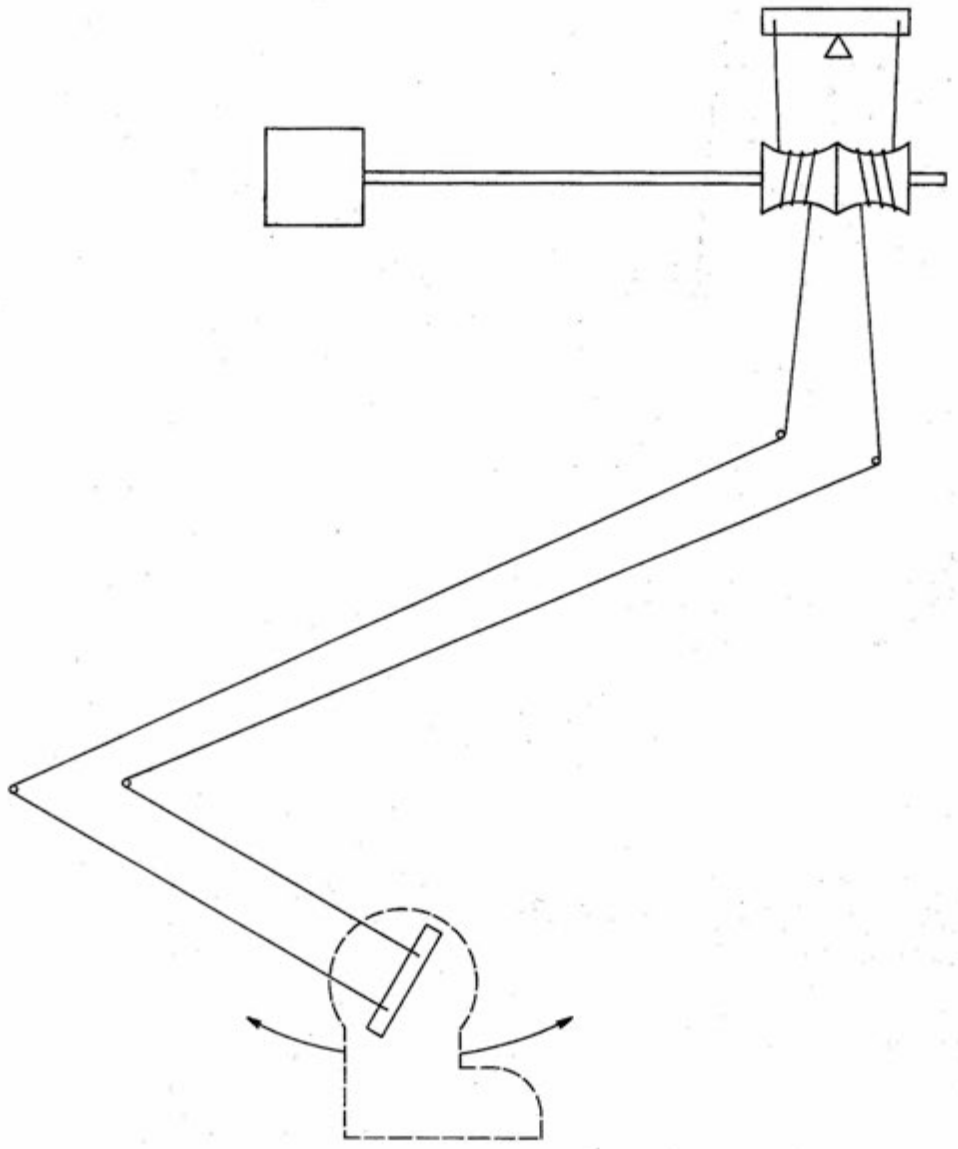
Operating limbs can reach out in any direction from the main body of the creature. If you lean strongly to emulation of the dog, you may decide to give the beast a waggable tail. That would require only a single double-windlass mechanism. To give the tail more expressiveness, add another double-windlass so it can move up and down as well as back and forth. A dog with its tail between its legs is ordinarily indicating submission. You may want your robot to appear submissive when it's being scolded. The ability of a robot to express gloom as well as pleasure can prove extremely useful to its handler, who needs signs that let him anticipate what the robot will do next. Clues to "emotional state" are extremely useful.

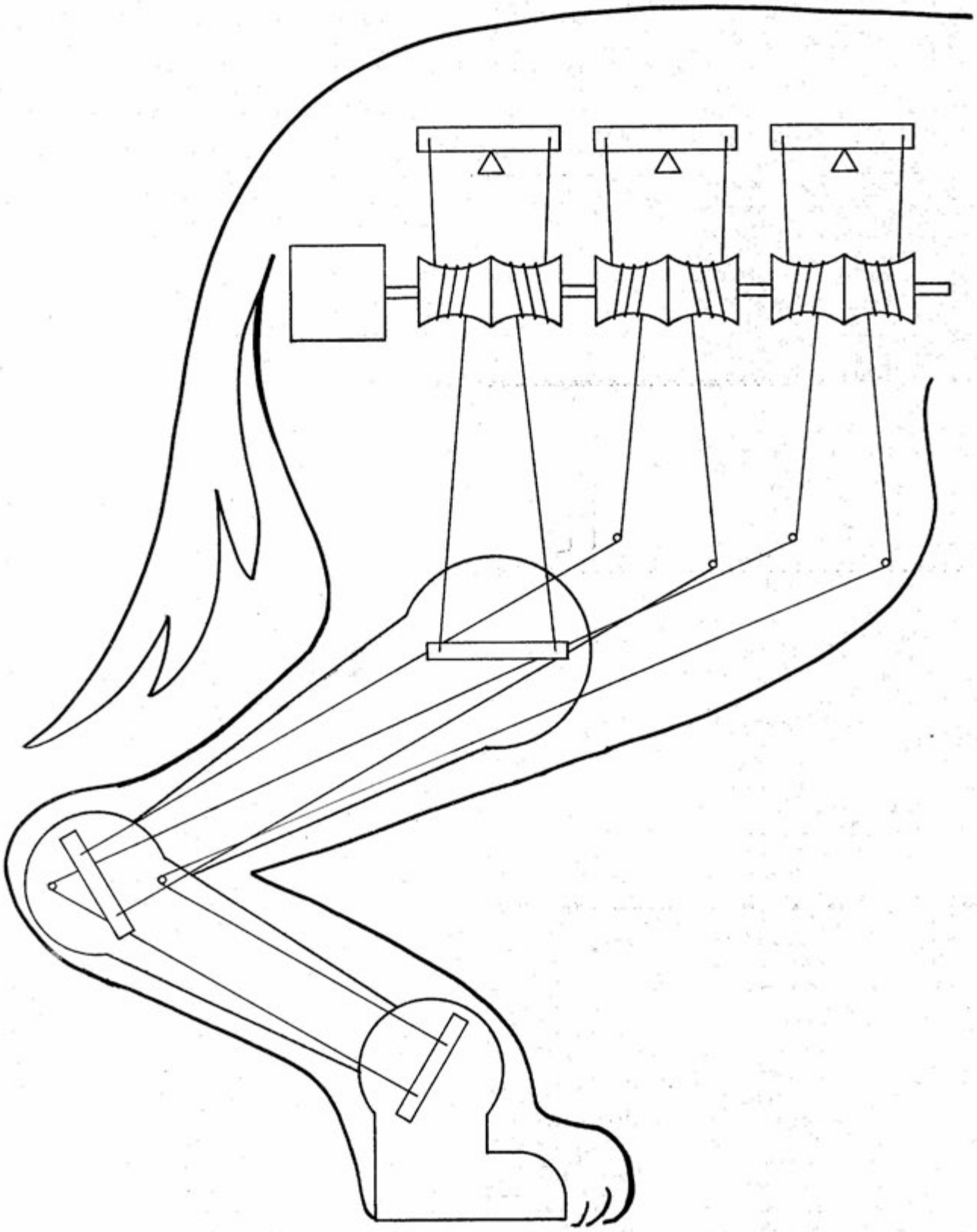
In past articles copyrighted by the United State Robotics Society (USRS), robots have been evaluated as household pets; the point being that if robots are to be welcome in society, they must display features and characteristics that society has already accepted. Since household pets are clearly acceptable, roboticists may well study them in detail and make use of their physical and behavioral characteristics. For example, animals whose hooves scratch hardwood floors and damage carpets aren't often invited into the parlor. Animals that weigh more than 50 pounds ordinarily live outside. Those that move faster than about two miles an hour in the house are sent outside to release their energies.

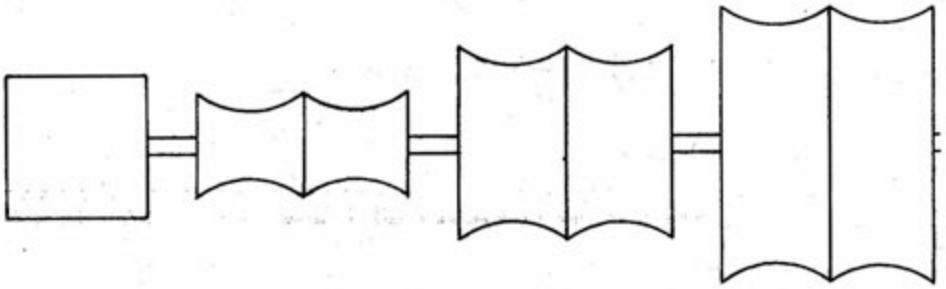
Taking these things into consideration, the practical pet robot will probably weigh 30-90 pounds, will use one to three cubic feet of space, and will be more spherical than otherwise. This configuration suits the double-windlass mechanism very well.

An engineer listening to this discussion might suggest off the top of his head that a 1/6 to 1/4 horsepower motor is probably adequate to move such a robot around at not more than two mph, assuming no great acceleration demanding extra surges of energy is required. Moving upstairs and downstairs should be all the same to a machine designed to be in no particular hurry.

continued



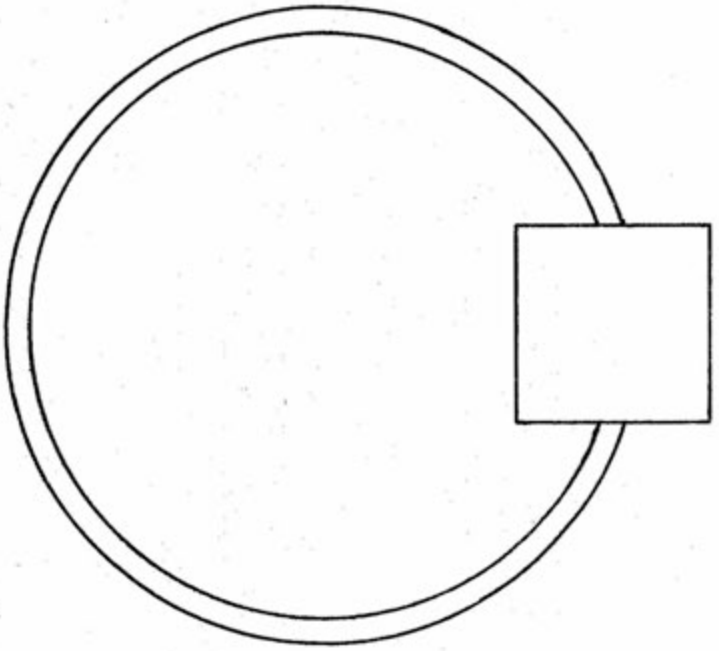




If you're eager for robots that slither or creep instead of walk, this same approach should work. Aquatic creatures with flippers and arboreal types with arms designed for tree-climbing are very similar. A flying robot with flapping wings seems improbable, considering the aeronautical failure with ornithopters, but the field is wide open to experiment.

A major feature of the double-windlass is its push-pull capability. It not only pulls a lever up in one direction, but can also pull it back the other way. This reciprocating movement is very important to animals; Nature makes sure that they can back out of trouble as well as push into it, thus adapting to new situations. To survive, the robot must be able to put its foot down as well as pick it back up without cycling through a full rotation movement. Control of this action must be in the "brain" of the robot, in its logic circuitry.

Again, this involves philosophy and logic beyond the scope of this article, but the mechanical design must not preclude solutions to logical problems.



Briefly consider one of Nature's tricks--the human knee-jerk reflex. When you get a physical exam, the doctor taps just under your kneecap with his silly little rubber hammer and is gratified when your knee jerks. In fact, your brain doesn't become involved with that knee-jerk, except as an observer. The whole thing is handled in a subcircuit of your nervous system. You receive a sharp stimulus to a sensitive point in your knee, and your system responds by yanking on your lower leg. (If something is biting you, or you are being burned, this reflex tends to pull you out of danger.)

When the reaction is complete, where is your leg? Well, it's hanging there, loose again, ready for something else of interest to happen. Note that your leg doesn't fly up and kick the doctor into the next room. It also doesn't snap back sharply to its original position, or lock into position at the peak of its reaction movement. It stops the reaction after doing something that your brain can detect after making a protective move. Then it's ready for the next action, relaxed and not committed to anything in particular.

The "nervous system" in the robot may be designed with similar reflexes. The mechanical system must be able to work in accordance with the nervous system. This leads to questions about "normal" positions for limbs. A horse ordinarily sleeps on his feet, because he has a leg at each corner to keep him balanced, and his legs lock normally into a standing position. Similarly, he sleeps with his head erect. He must expend energy to put his head down to the grass to graze. When he relaxes, his head is pulled naturally up to a level at which he can see most things that attack horses.

What's the normal position for the robot? A design can be chosen to suit any rationale. Springs can be used to hold limbs in a "normal" position. However, it's not necessary to use a single spring to hold a leg in position. Doing so expends a lot of the motor's energy in overcoming that spring whenever the limb moves. Instead, you can use paired springs that hold the limb at the point of equilibrium between them. When the motor pulls the limb, it fights one of the springs as it releases energy stored under tension. This wastes some energy, but the overall cost may be far outweighed by the savings in maintaining a normal "power-off" position, which is better than having your robot collapse in a heap on the floor.

With small motors, pulleys, springs, and levers, your construction of a satisfactory mobile system at modest cost is possible. The patter of little feet around the house may soon be produced not by kids and cats, but walking machines trying their new legs.

In next month's CN, part III will discuss applications of robots.

NEW BOOKS

GAME PLAYING WITH BASIC

Hayden Book Company has just released Donald Spencer's GAME PLAYING WITH BASIC -- the book that challenges computer users with such games and puzzles as Nim, Roulette, Slot Machines, Magic Squares, Keno, Baccarat, and 3-D Tic-Tac-Toe.

Spencer writes for all beginners in an easy, nontechnical style so that almost anyone can understand computerized game playing. The 176-page paperback book includes a clear explanation of the rules of each game, how each game works (with illustrative flowcharts and diagrams), and the actual output produced by each program.

The last chapter contains 26 games for reader solution. Games included are: Hexapawn, Poker Dice, Nine Men's Morris, The 50 Puzzle, Boule, and Craps. All of these will challenge users' best programming skills.

GAME PLAYING WITH **BASIC,** by Donald Spencer, is available for $6.95 from local computer stores, bookstores, or Hayden Book Company, Inc. (50 Essex Street, Rochelle Park, N.J. 07662).

YOUR HOME COMPUTER

This new book is the first comprehensive microcomputer user's guide. Written in clear and understandable language, YOUR HOME COMPUTER tells you everything that you want to know about home computing and gives the computer novice a painless introduction to microcomputer terminology and technology. This book requires no prior knowledge or experience on electronics or computing.

YOUR HOME COMPUTER provides information about home computer kits, guidelines for selecting and building your own microcomputer, addresses of computer stores and clubs, lists of periodicals, how to use your home computer, suggested applications and much more.

With over 100 illustrations, YOUR HOME COMPUTER is fun and easy to read. The book is available for $6 from local computer stores, bookstores, or DYMAX.

INSTANT BASIC

For the microcomputer enthusiast or the user of DEC's BASIC PLUS language, there is finally a new book to teach you BASIC -- Jerald Brown's INSTANT BASIC.

INSTANT BASIC teaches Altair-style BASIC to beginners using interesting programming ideas and applications that will be easily understood by the home computer programmer. BASIC Plus users know that the two languages are very similar, so this book can be used by them as well.

INSTANT BASIC is an "active participation" workbook, designed to be used with your home computer so you can learn by doing. Ideas are slowly introduced in a non-mathematical context so the beginner can quickly learn good programming techniques. Brown's crazy graphics, reminiscent of the popular MY COMPUTER LIKES ME, not only help illustrate ideas but make the book fun to read.

For those with some previous knowledge of BASIC, the workbook has handy reference summary boxes of most statements (8K Altair BASIC), including differences between Altair BASIC and BASIC PLUS.

INSTANT BASIC is available for $6 from local computer stores, bookstores, or DYMAX, (P.O. Box 310, Menlo Park, CA 94025).

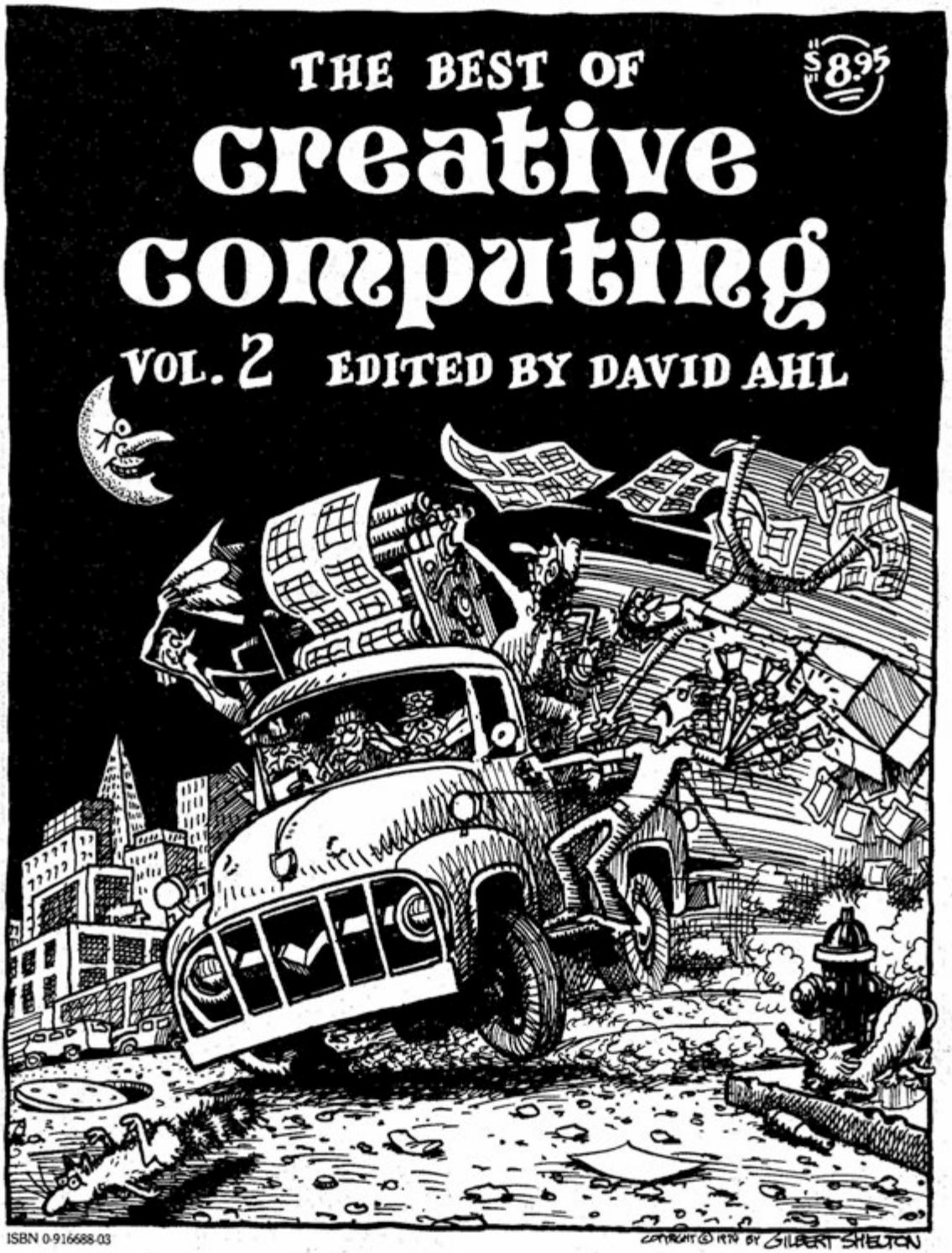
THE BEST OF CREATIVE COMPUTING, VOL. 2

This fascinating book contains the best of the articles, fiction, foolishness, puzzles, programs, computer games, and reviews from the Volume 2 (1976) issues of CREATIVE COMPUTING magazine. During this period, home computers just began coming on the scene. This volume vividly reflects the transition from minis and timesharing terminals to the new micro systems, although the diversity of the contents guarantees something for just about everyone.

Also included are 15 new computer games with complete listings and sample runs for each, and 67 pages of puzzles, problems, programs, and things to do with your computer or terminal. Frederik Pohl is just one of 10 imaginative storytellers whose work appears in this collection.

The staggering diversity of this book makes it an ideal point from which to jump in to the amazing world of recreational or educational computing.

THE BEST OF CREATIVE COMPUTING - VOLUME 2 is available for $9.95 postpaid from CREATIVE COMPUTING, Attn: Pamela, P.O. Box 789-M, Morristown, N.J. 07960.



ISBN 0-916688-03

continued

NEW BOOKS

continued

THE COLOSSAL COMPUTER CARTOON BOOK

Gee whiz! Someone has finally done it and collected all the jokes, jibes and cartoons about the world of computers under one cover. Would you believe 15 chapters ranging from cartoons about robots to computer dating to computers in everyday life ("Daily Data") to computers in the office ("Keypunch Lines") to the inevitable malfunctions. A full-page movie ad for The Fortran Monster -- scary! Four noted cartoonists are highlighted (each with their own section) -- Dave Harbaugh (very droll), Paul ("The Robot and the Professor") Swan, Sandy Dean, and Al Johns. We didn't count them, but there must be several hundred cartoons, ranging from puns to the sophisticated.

Mere words just can't described this books humor. Try it; you'll like it. A terrific gift for the non-computer freak in your life too!

THE COLOSSAL COMPUTER CARTOON BOOK is available for $5.95 postpaid from CREATIVE COMPUTING.

COMPUTER RAGE

Tired of Monopoly, Aggravation and Sorry? Looking for a game that teaches something about computers and is still sheer fun? Then try Computer Rage. Two to four players use three binary dice, to move from zero to seven spaces per turn. There are priority interrupts, restricted use input and output channels, power failures, program bugs and branch points. The objective is to get your three programs (shaped like miniature disk packs) from the input to the output by weaving through a maze of program steps, checkpoints, I/O queues, interrupts, and decision points.

The game comes with a large ("19 x 19") colorful board, 12 playing pieces, three binary dice, 38 interrupt cards, rules, and a booklet describing how to use the game as an educational tool. With several possible playing variations, Computer Rage is lots of fun even when the computer is down. Recommended for ages nine to adult, it makes a great gift for anyone.

COMPUTER RAGE is available for $8.95 postpaid from CREATIVE COMPUTING.

THE BEST OF BYTE, VOL. 1

This 384-page blockbuster of a book contains the majority of material from the first 12 issues of BYTE magazine. 146 pages are devoted to "Hardware" and are crammed full of how-to articles on everything from TV displays to joysticks to cassette interfaces. But a home computer without software might as well be a boat anchor, so there are 125 pages of "Software and Applications," ranging from on-line debuggers to games to a complete small business accounting system. A section on "Theory" examines the how and why behind the circuits and programs. and a final section, "Opinion," looks at where this explosive new home computer hobby is heading.

Early issues of BYTE are selling for as much as $15. If you're not a fanatic collector, this book represents a much better value per dollar. THE BEST OF BYTE - VOLUME 1 is available for $12.95 postpaid from CREATIVE COMPUTING.

Why Aren't There Any Altairs™ on Arcturus II?

By Henry Melton
7307 E. Riverside Dr.
Lot #13
Austin, TX 78741

The personal computer is here now, with a potential to open up society to private initiative in a manner almost unprecedented in history, and nobody wrote a story predicting it. Are science fiction writers losing their crystal balls?

I have been writing science fiction for some time now, and I've been reading it forever, so it doesn't strike me as being very odd that I'm now into computers. If you mention your computer to a friend, you're likely to get a HAL joke back. It's an automatic association: Computers and science fiction just seem to go together. It's logical too. For as long as I can remember the science fiction stories I read had ideas. Spaceflight, nuclear power, robots and a hundred other marvels were accepted features of those adventures. A good science fiction yarn would drag me off to some strange time and place where there would be far too much going on for me to ponder over the particular space drive involved, or to puzzle about the program that might be running beneath the robot's polished skull. But when I got back to Earth, how things had changed. So many marvels seemed so possible in what had been such a drab mundane world.

Well, the moonflights and the nuclear power plants and the Viking Lander robots happened. Everyone speaks of today as being a science fiction world. And, in a sense, the wonders of today were spelled out years ago in the books and magazines of science fiction. Science fiction writers are treated as modern day prophets and people expect them to have a handle on the future. In another sense, however, a closer look at some of these predictions shows a less flattering picture. Only the correct predictions are remembered, and even those weren't outstandingly accurate. Nobody predicted such a complicated first moon flight, with command modules and orbiting stages, descent and ascent staged landers. Few stories gave the first moonship an onboard computer. None included television cameras.

But no one worries too much about that. After all, one person sitting at a typewriter can't really compete with a team of engineers in working out the best way to put a man on the moon. A science fiction writer isn't a true prophet; all he or she can do is choose a possible future to write about. Just as the human race, by its actions, chooses a possible future in which to live. If a writer knows humans well enough, he/she can second guess the race and look like a pretty fair prophet. Such is the way of the game. Looking back at the hundred million words of science fiction I have read to date, I think I have gotten a good return. Future shock holds no terror for science fiction readers. No new technological marvel can sneak up and go boo. The science fiction reader has seen it all before.

Or, rather, almost all of it.

There does seem to be a big black gap in this flood of prophecy. The science fiction reader who is familiar with the common picture of a computer as portrayed in science fiction is due for a big shock when he/she runs up against the powerful little critter called a microprocessor. It's here, now, with a potential to open up society to private initiative in a manner almost unprecedented, and nobody wrote a story predicting it.

That is frightening in itself. What is wrong with science fiction that something so technological as a revolution in computers could go unheralded? Are the science fiction writers losing their touch? Is technology developing too fast for them to keep up?

An overview of modern science fiction, however, indicates that this gap seems to be a strange, localized thing. In other areas of human knowledge, science fiction is still riding high on the far edge of the barely possible. Interstellar ramjets, galactic core explosions, Kerr black holes, gene grafting, just about any conjecture of physics, cosmology, biology or whatever, is likely to be found in a modern science fiction story. Particularly in physics, some stories have come out in the science fiction magazines before the original research on which they were based has even made it to the professional journals. Science fiction can still make its claim to be the literature of ideas.

But, Alfred Bester's excellent novel, **The Computer Connection,** has a room sized malevolent computer trying to take

continued

over the world. Roger Zelazny's **Home Is The Hangman**, which has just won the Hugo award for the best science fiction novella of 1975, has the main character hinting down a possibly murderous robot, before the robot can find his creators. Isaac Asimov's "The Bicentennial Man" follows a robot, originally designed to be a butler, in his lifelong quest to become a human being.

Now, these are good stories, some of the best of the past couple of years. But the computers and robots in them are no more sophisticated than those in science fiction stories of twenty years ago.

Science fiction has had its stereotypes. The two computer types constantly used are either the huge device, something that would have been a good university computer back in the sixties, except for its disconcerting tendency to chuckle evilly when no one is looking, and the robot. The robot, moreover, usually has some kind of magical (i.e. Asimov's positronic) brain that is really nothing more than technological handwaving on the author's part to let him have a human brain sized computer inside a human looking robot body.

The real shame is that these stereotypes haven't changed in the twenty years that they have been used. A reader of BYTE can sit down with one of today's science fiction books and rest his/her reading arm on top of a logic machine considerably more sophisticated than anything he/she is likely to encounter in the pages of the book. It is a shame, if for no other reason than some otherwise good stories are going to be unpalatable to a lot of personal computer owners. This mental gap in science fiction writing really shows up in some places. Can you imagine what Spock would have done if there had been a microprocessor in his tricorder as there should have been?

The more one thinks about it, the clearer it becomes that science fiction writers are really behind schedule when it comes to computers; actually behind the times, when they should be well ahead. It is a blank spot in science fiction, and it has been there for years. It might have stayed blank for many more years if it hadn't been for the invention of the microprocessor. The only people who can notice that blank spot are those who know better. In other words, only you and I and the people we talk to.

But there are signs that the writers are waking up. I think the pocket calculators caught us all offguard. Poof! There it was, a gadget with major sociological implications in everybody's pocket, and nobody had really predicted it. I can remember only one old story that used pocket calculators, and even then, they were finely machined, motorized slide rules. The story was good, though. It explored what a world would be like if everybody had forgotten that math could be done in one's head. If the story had come out only five years ago, it would have been hailed as a prime example of the predictive value of science fiction. But no modern story had even considered the concept.

That was a shock for the writers; and signs that this stagnation is breaking up are starting to appear in print. But why was that mental block there at all, and why has it persisted for so long? Was it just a fluke?

I don't think so. I'll tell you why.

A quick look at all of the computer stories that have come out since computers and science fiction writers discovered each other back in the forties will show one dominant theme: artificial intelligence.

A writer of fiction stories has to be concerned with people. Even a gadget story is only a story inasmuch as it affects people. For a writer to write anything worth reading, he/she has to consider the characters above all else. When the intelligent computer first appeared on the science fiction scene, everybody took the concept to heart.

The intelligent computer made a beautiful character, from the storyteller's point of view. The full range of personalities, from purely logical to insanely demonical, were all available for use, and all essentially believable. Believability is a prime quality in science fiction. It has to be there if the story is going to be any good. And anyone can believe a computer with a bad case of misprogramming.

And so the computer became part of the stable of characters available for a science fiction writer, right up there with human beings, mythological creatures, and aliens from the planet_____. (The symbolic name of your favorite mythical planet is a parameter to be supplied by the user of a science fiction writing program.) A computer was a character. The concept became so fixed that writers forgot that real computers exist because they are beautiful tools. And while science fiction writers told their tales of pensive robots and planet killing Berserker machines from some war lost long ago and light centuries away, the macrocomputers begat minicomputers, and minis begat micros. Out of the same labs came the pocket calculators, which jumped out on the market and us all a little taste of future shock.

The shock has done some good. Some of the better writers are already into the swing of things. Take two examples: You must read **The Mote In God's Eye** by Larry Niven and Jerry Pournelle, and **Imperial Earth** by Arthur C. Clarke. In both of these novels everyone has a pocket computer. These gadgets are pocket sized with large memories and very easy to program. They can store text, graphics, and sound, occasionally tying into larger computers by radio, thus serving as diary, library, calculator, and who knows what else. This pocket computer is such a logical development that you can bet that other writers will pick up on the idea. Here is a beautiful tool in science fiction, and not a hint of an evil chuckle out of it.

Here is another example: **Shockwave Rider,** by John Brunner, extrapolates a world where any computer can be reached and programmed from any telephone. (This is an extrapolation?) In this future world, the programming genius with the right password is like the one eyed man in the country of the blind. What can be forbidden to the man who can change his identity, profession, and financial status, with just a couple of hours tapping touchtones on his phone? **Computerworld!** In this story, Brunner introduced the concept of a tapeworm to the science fiction readership.

A tapeworm is a software life form. It lives in memory space, eating processor time. And they exist. I've seen them. So have you if you've watched a program blow up, using a video display as a window into your memory space. I've watched about five naturally occuring species in a friend's 8080. If you are doubtful that they are alive, go back and reread the definition of life and think again.

Exciting things are starting to happen as science fiction takes another look at computers and what they can do. But only you can determine how fast these stories can come out. You can't wake up a writer if

you haven't met him/her. So put on your BYTE tee-shirt and go to the next science fiction convention and meet the people. Talk about your machines.

The computer as a beautiful character will never go away; you can be sure of that But the beautiful tool is here, and all it takes is for science fiction writers to realize it. Then, finally, you might see an Altair computer on Arcturus II.

CLASSIFIED ADS

FOR SALE

4 Teletype terminals:

ASR - 33 Punch & Reader	$500
KSR - 35	$750
KSR - 33	$350
ASR-33 Tractor Feed with automatic Punch and Reader	$650

45-day warranty included.

WANTED

Used Altair 8800b or partially/nonassembled Altair 8800b computer.

Contact:
Bill Shelton
6327 Acoma, SE
Albuquerque, NM 87108
(505) 255-8180

VDM Linkage for 4.1 BASIC

By Don Chamberlain
10301 Alpine Drive, #1
Cupertino, CA 95014
Chamberlain is currently a manufacturing engineer for Amdahl Corp.

EDITOR'S NOTE: MITS has NOT tested and will thus not support the following fix. Readers should send all questions directly to the author.

When using Processor Technology's VDM-1 for a console display in a system, you will encounter some difficulties with Altair Extended BASIC, version 4.1. After several weeks of hair-tearing and single-stepping, I finally isolated the problem. The VDM couldn't handle the LF character and kept the I/0 routine in a loop looking for the next character. I solved the problem by outputting all LF characters to the primary terminal, a COMTER in my case. A similar fix allows the system to output BELL (control G). The following is a listing of the critical portion of the I/0 routine:

```
TRYOUT   : JMP LINK        ; Go to linkage routine
           •
           •
           •
           RET
LINK     : IN 377          ; Get sense status
           CPI 1           ; Output to COMTER?
           JZ COMOUT       ; Yes? Goto COMTER
                             output routine
           CPI 3           ; Output to printer
           JZ PNTOUT       ; YES? Goto printer
                             output routine
           POP PSW         ; Look at character
           CPI 7           ; Ring bell?
           JZ SAVE         ; LINEFEED?
           JNZ VDMOUT      ; If not output on screen
SAVE     : PUSH PSW        ;
           JMP COMOUT      ; Goto COMTER out-
                             put routine
VDMOUT   :   •             ; Bulk of VDM linkage
             •
             •
COMOUT   : IN O            ; Get statue
           ANI 200         ; Ready to output?
           JNZ COMOUT      ; If busy loop
           POP PSW         ; Get character
           OUT 1           ; Output character to
                             COMTER
           JMP GOBAC       ;
PNTOUT   : IN 20           ; Get status
           ANI 2           ; Ready to output
           JZ PNTOUT       ; If busy loop
           POP PSW         ; Get character
           OUT 21          ; Print character
GOBAC    : RET             ; Go back to BASIC
```

ROBOTS

AN ESTIMATE OF THE STATE OF THE ART, AND AN INVITATION TO PERSONS OF ADVENTUROUS SPIRIT AND INQUIRING MIND

We believe that the key discoveries necessary to the art of robotics have already been made. **We believe** that behind various national borders, behind the doors of various scientific disciplines from biochemistry to microelectronics, all of the primary technical obstacles have been overcome, all feasibilities been proven, all methods become known.

We believe that what remains to be achieved is principally the refinement of systems applying existing technologies — **and that this work proceeds apace.** We believe the world is about to encounter (where? when?) machines that truly simulate the intellectual and physical behavior of human beings: robots.

Robots are on our doorstep. Robots are almost within our reach. And we within theirs.

Robots are as frightening as they are alluring, as threatening as they are promising. Yet whatever reservation anyone may feel, there is now no turning back, no possibility of their denial or prohibition. The development of artificial intelligence proceeds not only in the laboratories of governments and industries, but also among the thousands of individual amateurs and hobbyists, free citizens exercising their freedom with experiments in the fascinating field of personal computing. **We believe** that since they are possible, robots are inevitable — "for good or ill."

The United States Robotics Society is established "for good" — for the good of mankind — not in opposition, for opposition is idle, and not in advocacy, for advocacy is unnecessary. **We invite** the support and active participation of all persons who can face the Age of the Robot with the appropriate curiosity and spirit of adventure.

Intelligent machines for production and service — tireless, able to understand commands and carry them out sensibly without feeling a need to make policy for themselves — may become the long-heralded boon to humanity, lifting ancient burdens of toil and suffering. But if they were to be developed "in the dark" — if they were to be sprung upon us full-blown, without our preparation — the reaction might be disastrous. The survival of our own society may depend quite soon (how soon?) on our ability to deal even with "friendly" robots. If we ignore them, if we are incompetent in their fields, we are surely not serving our own interests.

Intelligent weapons now appear practicable within the next decade or two — systems, for example, that can differentiate between friend and foe automatically, through their own sensors and judgement. If such weapons are developed anywhere in the world, they will be extraordinarily dangerous to any society which has not learned how to deal with them.

Robotics has charm not only for trained technicians and professionals but also for millions of persons without the skills and resources to participate directly in the work. Communication about robotics, like robots themselves, is inevitable, — through publicity, rumor, espionage, and now through The United States Robotics Society. This organization will assume the important task of identifying discoveries, gathering supporting data from the hidden recesses where they rest, collating, publishing, becoming a center of information for all parties seeking knowledge of current and historical activity in robotics. We urge you to be one of us — for just $12/year.

Benefits to USRS Members Growing Year by Year

- Certificate of Registration as USRS Member.
- USRS Newsletter, USRS bulletins, other correspondence from the Society as occasion demands.
- Aid in contacting other USRS Members in home regions, toward establishing USRS events.
- Opportunity (Qualified) to officiate as USRS Representative at Regional and National robotics shows and exercises.
- Service (Optional) as USRS Contributing Correspondent.
- Participation in the determination of procedures for investigating, reporting, archiving, and disseminating information relevant to robotics . . . and
- Privileged access to the Library of Robotics to be established by USRS.
- Discounts as may from time to time be arranged by USRS on behalf of members — with publishers, manufacturers/distributors of robotics related materials. (Note: this benefit alone can be expected to repay the moderate USRS Membership costs many times over.)

United States Robotics Society

A Non-Profit Organization
Glenn R. Norris, President
Box 26484 Albuquerque, New Mexico 87125

Application for Charter Membership

USRS

United States Robotics Society
Box 26484 Albuquerque, NM 87125

Enclosed is my check for $12 for enrollment and first-year dues.

The following information is requested (OPTIONAL) to help ensure your full participation in the benefits of the Society.

My interest in robotics derives from ( )intellectual curiosity ( )academic training ( ) professional/business

Please tell us more: _____

( ) I am interested in joining with others in local USRS activity.
I might serve as ( ) Correspondent ( ) Official at USRS functions.

NAME _____
ADDRESS FOR USRS COMMUNICATIONS _____ CITY _____ STATE _____ ZIP _____ PHONE _____

Fill in the blanks

with **Computer Notes.** Cover the bare spots in your microcomputer library with the information on the latest hardware, software and applications **CN** provides each month. **CN** is bound in a standard format designed to be kept easily in a three-ring binder as a ready reference for the computer club or individual user.

To insure that you never miss an issue, simply **fill in the blanks** on the coupon below and send it along with the subscription fee to:

computer notes

mits a subsidiary of **Pertec Computer Corporation**
2450 Alamo S.E.
Albuquerque, New Mexico 87106

Please send me a 1 year subscription to **Computer Notes.**
$5.00 per year in U.S. $20.00 per year overseas.

NAME:_____

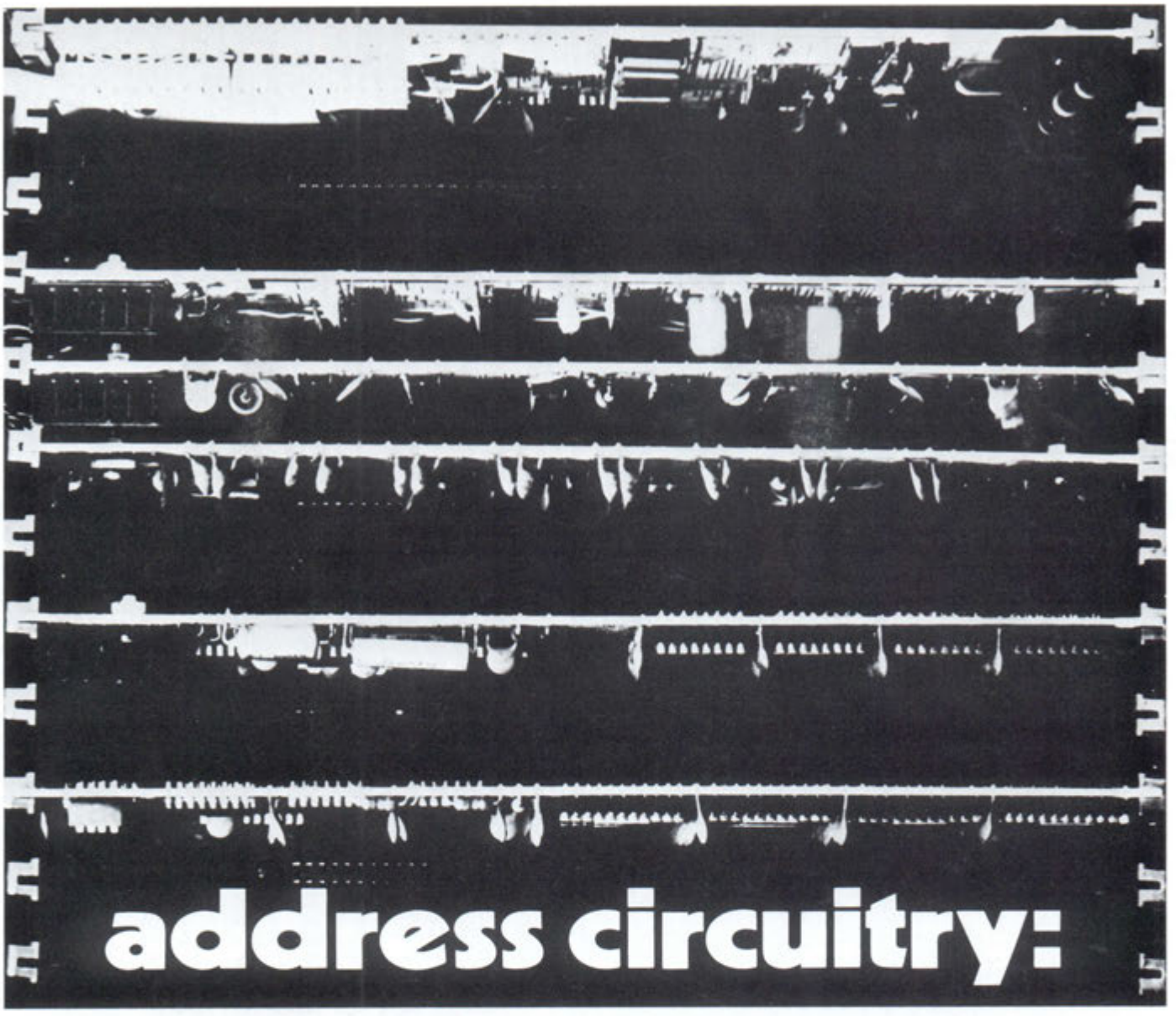
ADDRESS:_____

CITY:_____ STATE:_____ ZIP_____

COMPANY/ORGANIZATION_____

☐ Check Enclosed
☐ Master Charge
☐ BankAmericard/Visa

MC or BAC/Visa #_____
Exp Date _____
Signature_____



Altair Computer Centers offer you direct access to the complete line of Altair microcomputer products. For demonstrations, information or service, visit the experts at your local Altair Computer Center today. They're located at these addresses:

ALTAIR COMPUTER CENTER
4941 East 29th St.
TUCSON, AZ 85711
(602)-748-7363

COMPUTER KITS (S.F. area)
1044 University Ave.
BERKELEY, CA 94710
(415)-845-5300

THE COMPUTER STORE
(Arrowhead Computer Co.)
820 Broadway
SANTA MONICA, CA 90401
(213)-451-0713

THE COMPUTER STORE, INC.
(Hartford area)
63 South Main Street
WINDSOR LOCKS, CT 06096
(203)-627-0188

GATEWAY ELECTRONICS, INC.
OF COLORADO
2839 W. 44th Ave.
DENVER, CO 80211
(303)-458-5444

THE COMPUTER SYSTEMCENTER
3330 Piedmont Road
ATLANTA, GA 30305
(404)-231-1691

CHICAGO COMPUTER STORE
517 Talcott Rd.
PARK RIDGE, IL 60068
(312)-823-2388

THE COMPUTER STORE, INC.
120 Cambridge St.
BURLINGTON, MA 01803
(617)-272-8770

THE COMPUTER STORE
OF ANN ARBOR
310 East Washington Street
ANN ARBOR, MI 48104
(313)-995-7616

COMPUTER STORE OF DETROIT
505-507 West 11 Mile St.
MADISON HEIGHTS, MI 48071
(313)-545-2225

THE COMPUTER ROOM
3938 Beau D'Rue Drive
EAGAN, MN 55122
(612)-452-2567

GATEWAY ELECTRONICS, INC.
8123-25 Page Blvd.
ST. LOUIS, MO 63130
(314)-427-6116

ALTAIR COMPUTER CENTER
5252 North Dixie Drive
DAYTON, OH 45414
(513)-274-1149

ALTAIR COMPUTER CENTER
110 The Annex
5345 East Forty First St.
TULSA, OK 74135
(918)-664-4564

ALTAIR COMPUTER CENTER
8105 SW Nimbus Ave.
BEAVERTON, OR 97005
(503)-644-2314

ALTAIR COMPUTER CENTER
611 N. 27th St. Suite 9
LINCOLN, NB 68503
(402)-474-2800

COMPUTER STORES
OF CAROLINA, INC.
1808 E. Independence Blvd.
CHARLOTTE, N.C. 28205
(704)-334-0242

COMPUTER SHACK
3120 San Mateo N.E.
ALBUQUERQUE, NM 87110
(505)-883-8282, 883-8283

THE COMPUTER STORE
269 Obsorne Rd.
ALBANY, NY 12211
(518)-459-6140

THE COMPUTER STORE
OF NEW YORK
55 West 39th St.
NEW YORK, NY 10018
(212)-221-1404

ALTAIR COMPUTER CENTER
3208 Beltline Road
Suite 206
DALLAS, TX 75234
(214)-241-4088 Metro-263-7638

ALTAIR COMPUTER CENTER
5750 Bintliff Drive
HOUSTON, TX 77036
(713)-780-8981

COMPUTERS-TO-GO
4503 West Broad St.
RICHMOND, VA 23230
(804)-335-5773

MICROSYSTEMS (Washington, D.C.)
6605A Backlick Rd.
SPRINGFIELD, VA 22150
(703)-569-1110

THE COMPUTER STORE
Suite 5
Municipal Parking Building
CHARLESTON, W.VA. 25301
(304)-345-1360

mits