OSI-TEMS by OSUNY

OCT. ~~APRIL~~ 1982 VOL.IV NO.3

MITS Altair Computer Report II

MITS Announces Lower Memory Prices!

On July 1, 1975, **MITS lowered the price** of the Altair 1K Static Memory Card (88-1MCS). The kit price was dropped from $176 to **just $97** while the assembled price was dropped from $209 to $139.

This price reduction was made possible by a reduction in the price of the Altair 1K 8101 memory chips.

Also affected was the price of 88-MM 256 byte (word) memory modules. The $53 kit price was lowered to just $14 and the $61 assembled price to $26.

Altair BASIC—Not Just Anybody's BASIC

Altair BASIC is an easy-to-use programming language that can solve applications problems in business, science and education.

You will find that with only a few hours of using BASIC that you can already write programs with an ease that few other computer languages can match.

Altair BASIC doesn't compromise power for simplicity. While it is one of the simplest computer languages in existence, it is also a very powerful language.

ALTAIR BASIC comes in three versions. The first of these is a 4K BASIC designed to run in an Altair with as little as 4,000 words of memory. This powerful BASIC language has **6 functions** (RND, SQR, SIN, ABS, INT, and SGN) in addition to **15 statements** (IF . . . THEN, GOSUB, RETURN, FOR, NEXT, READ, INPUT, END, DATA GOTO, LET, DIM, REM, RESTORE, PRINT, STOP) and **4 commands** (LIST, RUN, CLEAR, SCRATCH).

The second ALTAIR BASIC option is the 8K BASIC designed to run in an Altair with as little as 8,000 words of memory. This BASIC language is the same as the 4K BASIC only with **8 additional functions** (COS, LOG, EXP, TAN, ATN, INP, FRE, POS) and **4 additional statements** (ON . . . GOTO, ON . . . GOSUB, OUT, DEF) and **1 additional command** (CONT). This BASIC has a multitude of advanced STRING functions and it can be used to control low speed devices—**features not normally found in many BASIC languages.**

The third ALTAIR BASIC is the EXTENDED BASIC version designed to run on an Altair with as little as 12,000 words of memory. It is the same as the 8K BASIC with the addition of PRINT USING, DISK I/O, and double precision (13 digit accuracy) add, substract, multiply and divide.

Altair BASIC is only the beginning. MITS is currently engaged in an extensive software development program. Other software now available includes an Assembler, System Monitor, and Text Editor.

Altair software comes with complete documentation.

One Month Specials

The Altair Users Group is quite possibly the largest computer hobbyist organization in the World. It is both a means of communication among Altair Users and a method of building a comprehensive library of Altair programs. All Altair 8800 owners are entitled to a free, one year membership in this group.

For one month only, you can become an Associate Member for one year at a reduced rate of $10 (regularly $30). Among other benefits you will receive a subscription to the monthly publication, **Computer Notes,** which contains complete update information on Altair hardware and software developments, programming tips, general computer articles and other useful information.

Now available is the **Altair Software Documentation Book I** which contains technical data on the Altair Assembler, Text Editor, System Monitor and BASIC language software. This documentation is free to purchasers of Altair BASIC. For one month only, it is being offered for only $7.50 (regularly $10).

Offers good until September 30, 1975.

The 1K Static Memory Card contains 1024 bytes of memory with a maximum access time of 850 nanoseconds.

Now ready for production is the new Altair 2K Static Memory Card (88-2MCS) with 2048 bytes of memory. Like the 1K Static Memory this new card contains memory protect features and provisions for disabling the ready.

It has a maximum access time of 850 nanoseconds and is engineered with the finest components available. It is inexpensively priced at $145 kit and $195 assembled.

HARDWARE PRICES:

Altair Computer kit with complete assembly instructions	$439
Assembled and tested Altair Computer	$621
1,024 Byte Static Memory Card	$97 kit and $139 assembled
2,048 Byte Static Memory Card	$145 kit and $195 assembled
4,096 Byte Dynamic Memory Card	$264 kit and $338 assembled
Full Parallel Interface Card	$92 kit and $114 assembled
Serial Interface Card RS232	$119 kit and $138 assembled
Serial Interface Card (TTL or Teletype)	$124 kit and $146 assembled
COMTER II*	$780 kit and $920 assembled

*The Comter II Computer Terminal has a full alpha-numeric keyboard and a highly readable 32-character display. It has its own internal memory of 256 characters and complete cursor control. Also has its own built-in audio cassette interface that allows you to connect the Comter II to any tape recorder for both storing data from the computer and feeding it into the computer. Requires an RS232 Interface Card.

SOFTWARE PRICES:

Altair 4K BASIC	$350
Purchasers of an Altair 8800, 4K of Altair Memory, and Altair Serial I/O or Audio-Cassette I/O	ONLY $60
Altair 8K BASIC	$500
Purchasers of an Altair 8800, 8K of Altair Memory, and Altair Serial I/O or Audio-Cassette I/O	ONLY $75
Altair EXTENDED BASIC	$750
Purchasers of an Altair 8800, 12K of Altair Memory, and Altair Serial I/O or Audio-Cassette I/O	ONLY $150
Altair PACKAGE ONE (assembler, text editor, system monitor) Purchasers of an Altair 8800, 8K of Altair Memory, and Altair I/O	ONLY $30

NOTE: When ordering software, specify paper tape or cassette tape.

Warranty: 90 days on parts for kits and 90 days on parts and labor for assembled units. Prices, specifications, and delivery subject to change.

MAIL THIS COUPON TODAY!

☐ Enclosed is check for $_____

☐ BankAmericard #_____ ☐ or Master Charge #_____

☐ Altair 8800 ☐ Kit ☐ Assembled ☐ Options (list on separate sheet)
Include $8 for postage & handling

☐ Altair Users Group Associate ☐ Software Documentation

☐ Please send free literature

NAME_____

ADDRESS_____

CITY_____STATE & ZIP_____

MITS/6328 Linn N.E., Albuquerque, NM 87108 505/265-7553 or 262-1951

MITS

"Creative Electronics"

MITS/6328 Linn N.E., Albuquerque, NM 87108 505/265-7553 or 262-1951

OSItems VOL III

October 1980

TABLE OF CONTENTS

OSItems is a monthly publication produced by the Ohio Scientific Users Group of New York, OSUNY.

OSItems is edited by Shell Sacks. all articles and inquiries should be mailed to him at: 2 Eldorado Blvd. Plainview, N.Y. 11803

OSItems is produced by Dale Jones

OSUNY BBS: (914) 725-4060
meeting first Thursday of each month at:
INTECHNOLOGY SERVICE ORGANIZATION
23 East 20 st. New York, N.Y. 10003 tel (212) 673-6310

LINE INPUT FOR OSI - by Dan Schwartz

One feature found in some Basics but absent from OSI's is the LINE INPUT statement, which inputs an entire line into a string variable without regard to commas, colons, quote marks, or leading spaces (and without an automatic question mark prompt). While this function is not directly supported in OSI Basic, it is actually quite easy to accomplish with a very short USR function, as I will show in this article.

Listed below are two versions of this function, one for ROM Basic and one for Disk Basic. They both are actually the same, only the addresses of the routines and buffers used differ. Did you know that you can use USR to return string values? I figured out the procedure for doing so by looking at the code for the STR$ function, which can be found at $B08C (ROM) or $12E9 (disk).

To use this function in a program, just include a statement such as A$=USR(X) or B$=USR(B$) in your program. The argument in parentheses is meaningless and can be either string or numeric. The function will stop your program, wait for a line to be entered on the keyboard, and assign this string to the variable given on the left side of the equals sign. You can use this just like any other function that returns a string value; the only thing you should not do is use it in immediate mode, since typing in the input string would destroy the statement being executed.

ROM Version	Comments	Disk Version
JSR $A357	Input a line from the keyboard	JSR $0558
PLA	Pull off one subroutine return address	PLA
PLA		PLA
LDA #$13	Low byte input buffer addr. (Y=high byte=0)	LDA #$1B
LDX #0	Only valid delimiter is a 0 (Null=EOL)	LDX #0
JMP $B0B0	Go construct a string descriptor and return it to the caller	JMP $130D

Sample program for ROM Basic

```
10 FOR X=1 TO 12:READ A:POKE 545+X,A:NEXT
20 DATA 32,87,163,104,104,169,19,162,0,76,176,176
30 POKE 11,34:POKE 12,2
40 PRINT:PRINT"TYPE ANYTHING:";:A$=USR(X)
50 PRINT"YOU ENTERED THIS:":PRINT A$
```

REVIEW - BMC MONITOR

Terry Terrance

Another new monitor has invaded our shores from those distant lands to the East. The BMC model BM-12A shares many features with its breathren, 12" screen size, the now ubiquitous green phosphor, a warmed-over TV case and 80 character capacity. Of course to provide competition to the already established entrants in the "green screen" monitor field the BMC must offer something, and the something is low price; the retail price is only $99.00.

From a looks-only standpoint the BMC is better than most, but not as attractive as some. The case is obviously a product of a TV assembly line, complete with unpunched speaker grill and tuner knob access holes; which are mericfully covered with a single metal cover plate bearing the BMC logo, the now common "IC - Solid State" legend (as if we didn't know), and an unbelievably hokey atom-model emblem. In my opinion the case is not as bad looking as most, but not nearly as cosmetically sucessful at hiding its TV origins as the Zenith. Had BMC deleted the vestigial tuner/speqker grill area and packaged its monitor in a trimmer case it would be a handsome unit indeed. The colors are light and dark tan which I'm sure are a perfect match for the Apple, but go well with OSI tan and brown anyway.

Input to the BMC is handled with a standard RCA female jack located on the rear panel. Horizontal and vertical hold controls are of the normal TV variety and found in their usual TV place - at the rear of the set. The On/Off, brightness and contrast controls are located on the right side of the unit, a place that is common for TVs but can make the controls hard to adjust on the computerist's usually crowded desk.

I have not had the chance to test the BMC with the output of the SEB-3 or any other 80 column source. The 64 characters output by the C8 nestle within the visible sweep of the beam so that 80 characters should be accomodated. The display seems to be sharp and square right out to the edges and top of the field with no evidence of tearing or shear.

My only substantial complaint with the BMC might be confined to the one unit that I received. When I hooked the monitor up to my C1, which I knew to have a well-adjusted video section producing maximum contrast output, all I saw on the screen, with the brightness cranked up all of the way, was VERY faint characters - and these were only visible when there were only a few characters on the screen.

Fortunately, BMC does something very uncharacteristic for consumer electronics manufacturers these days - they include schematics for their unit. Reference to the schematic showed that there is an internal "sub-brightness" control, a

potentiometer marked VR203 that might be of help. Why it's called sub-brightness when it's twice the value of the brightness control is beyond me. Inside the case I found this control to be turned two-thirds of the way counterclockwise. Whether this is the correct position for this control or not I have no way of knowing. But setting the pot to the mid-point gave a good display with all of the OSI machines that I own after their own video sections were set for optimum output. If your computer is still set with the usual indifferent factory video contrast setting, and you don't want to mess around in your computer, you may want to set the BMC's pot to about two-thirds clockwise. By the way, the BMC's case is a pain to reassemble unless you have a helper handy to lend an extra hand.

My only other complaint about the BMC is that its display tends to vary in brightness more than most with the number and type of characters displayed on the screen. This effect is quite small, however, and you probably would not notice it unless you're specifically looking.

One thing to note (this is not strictly under the heading of a complaint) about the operation of the BMC and the Zenith and probably most other monitors of this price catagory is that their power transformers are always connected to the power lines. The On/Off switches in these units operates on the secondary side of the transformers. Since these units are always dissipating a small (probably exceedingly small) amount of current due to heating in the transformer core, you might want to disconnect them entirely when not in use.

In all, the BMC BM-12A has no objective faults (that can't be fixed anyway) and certainly is a good monitor for the price.

P.S. After I completed the review above and before I sent it in to OSI-tems, I received a flyer about a new "greenie" monitor from BMC. This new unit, designated BM-12EN has all of the features of the 12A plus a 20 MHz video bandwidth, a non-glare screen (probably an etched glass tube) and (Lord be praised) an honest-to-goodness, good looking case WITHOUT space for tuner knobs or a speaker grille. The retail price was not specified but my guess is that it will be about $40 to 50 higher than the 12A.

SNAKE GAME

By Mike Cohen

This game was originally written for an IBM mainframe and a 3270 terminal. It will run on any system with 65D V3.3, including serial systems. The field can be enlarged by increasing W% and H%. The object is to get the highest score by 'eating' numbers without running into yourself.

```
10 REM SNAKE    6/9/82
20 REM
30 W%=23:H%=22
40 DIM S%(W%,H%),K%(9),X%(9),Y%(9)
50 DATA 32,204,50,168,173,137,58,76,24,18
60 DATA 113,119,101,97,115,100,122,120,99
70 DATA -1,0,1,-1,0,1,-1,0,1
80 DATA -1,-1,-1,0,0,0,1,1,1
90 DEF FNR(X)=INT(X*RND(1))
100 L%=H%+1:C%=W%/2+1
110 PRINT!(28);"Loading";
120 FOR I=0 TO 9:READ X:POKE14974+I,X:NEXT I:PRINT".";
130 FOR I=1 TO 9:READ K%(I):NEXT I:PRINT".";
140 FOR I=1 TO 9:READ X%(I):NEXT I:PRINT".";
150 FOR I=1 TO 9:READ Y%(I):NEXT I:PRINT".";
160 FORI=0TOW%:FORJ=0TOH%:S%(I,J)=FNR(6)+1:NEXTJ:PRINT".";:NEXTI
170 POKE574,126:POKE 575,58:POKE13026,42:POKE13743,32:PRINT!(20)
180 SK=0
190 FOR Y=0 TO H%:FOR X=0 TO W%:PRINT&(X,Y);CHR$(S%(X,Y)+48);
200 NEXT X:POKE22,0:NEXT Y:X%=FNR(W%):Y%=FNR(H%):S%(X%,Y%)=0
210 PRINT&(X%,Y%);".";
220 PRINT&(X%,Y%);
230 KT=0
240 KT=KT+1:KY=USR(0):IF KY THEN 270
250 IF KT>1E8 THEN PRINT!(21);"TIME OUT":PRINT"SCORE=";SK:END
260 GOTO 240
270 KY=KY OR 96:FORI=1TO9:IFKY=K%(I)THEN290
280 NEXTI:GOTO 230
290 XX%=X%(I):YY%=Y%(I)
300 IF X%+XX%<0 OR X%+XX%>W% OR Y%+YY%<0 OR Y%+YY%>H% THEN 220
310 Q%=S%(X%+XX%,Y%+YY%)
320 X%=X%+XX%:Y%=Y%+YY%:IF X%<0 OR X%>W% OR Y%<0 OR Y%>H% THEN 380
330 IF S%(X%,Y%)=0 THEN 390
340 SK=SK+S%(X%,Y%)
350 S%(X%,Y%)=0:PRINT&(X%,Y%);".";
360 Q%=Q%-1:IF Q%>0 THEN 320
370 PRINT&(0,L%);"SCORE=";SK;!(15);:POKE22,0:GOTO 220
380 X%=X%-XX%:Y%=Y%-YY%:GOTO 370
390 PRINT&(0,L%);"SCORE=";SK;&(C%,L%);"*** END ***"
400 END
```

Requires 12 bytes reserved below the workspace.

MACOMNET

THE SYSTEMS ARCHITECT FOR PRIVATE BUSINESS COMMUNICATIONS NETWORKS

Necessary and frequent face to face meetings, and a need for data and information exchange are problems faced by most multi-location organizations. Macomnet solves this problem with the most technologically advanced, private business communications network available. Reduced-travel, increased face-to-face meetings become possible with audio/video teleconferencing. Data, document and graphic images can be exchanged in seconds utilizing our systems network.

Own or lease a system tailored to your needs.

You specify the requirements, and our staff will respond with a custom designed system...a system that will handle your interactive communications needs with ultra-high reliability and efficiency.

With Macomnet, turn-key means just that.

You receive all the network elements from Macomnet. Turn-key installation, including systems engineering, hardware, software, total test and integration, training of your personnel, and continued maintenance.

We provide prompt service assistance from one of our 65 full-staffed service locations.

Proven, edge-of-the-future technology.

Macomnet operates its own satellite TDMA video information conferencing network at five company sites nationwide...for a monthly space segment cost of $8,000. Modest cost. Demonstratively efficient technology.

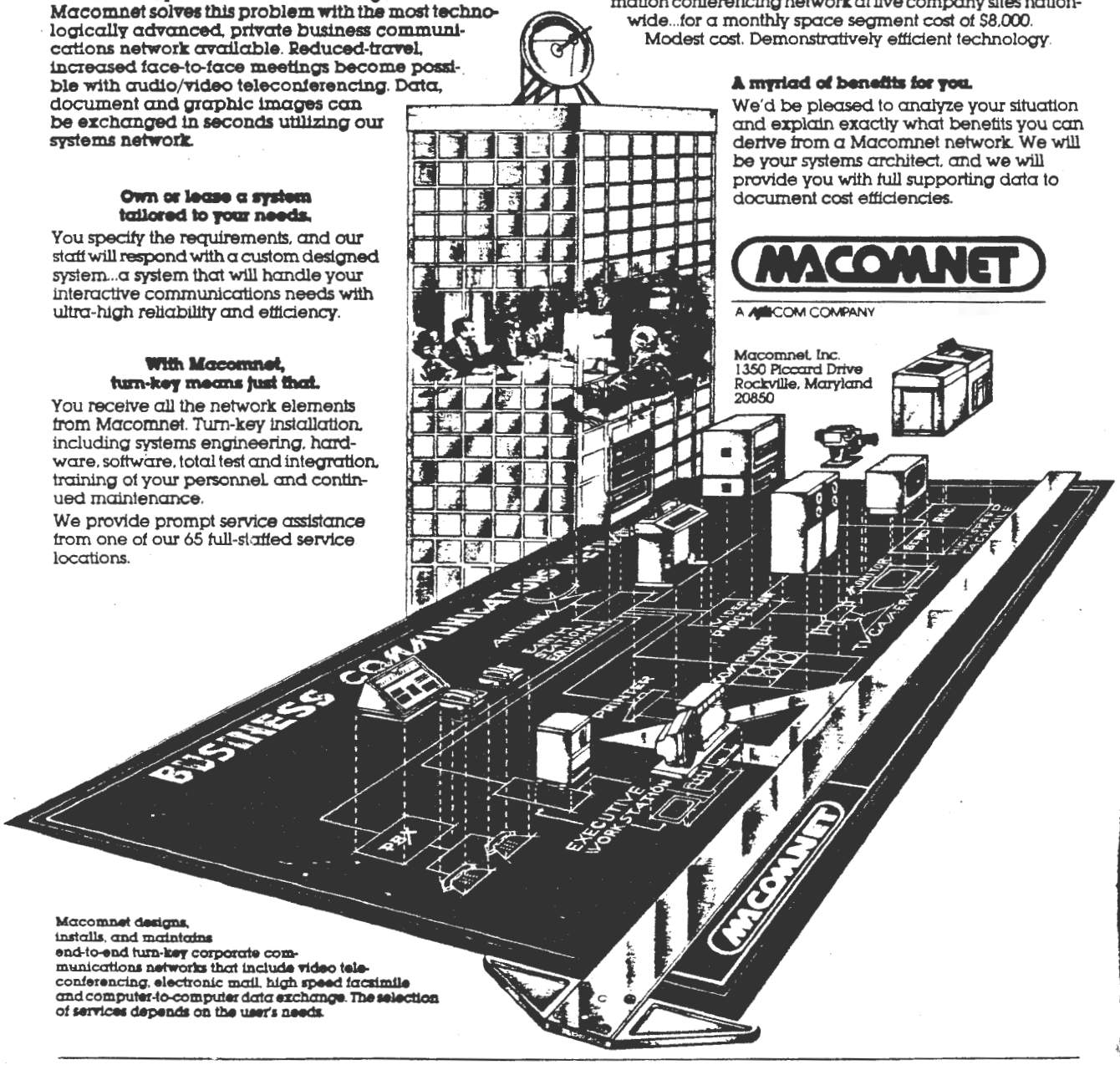
A myriad of benefits for you.

We'd be pleased to analyze your situation and explain exactly what benefits you can derive from a Macomnet network. We will be your systems architect, and we will provide you with full supporting data to document cost efficiencies.

MACOMNET

A AMECOM COMPANY

Macomnet, Inc.
1350 Piccard Drive
Rockville, Maryland
20850



Macomnet designs, installs, and maintains end-to-end turn-key corporate communications networks that include video teleconferencing, electronic mail, high speed facsimile and computer-to-computer data exchange. The selection of services depends on the user's needs.

To learn more, call Jim McKenna, Vice President, Sales, at **301/258-8858**, or write him at Macomnet, Inc.

Circle 1 on Reader Service Card

HOW TO ADD A SINGLE KEY BACKSPACE TO THE OSI C1P

Albert J. McCann, Jr.
30 South Elmwood Avenue,
Glenolden, PA 19036
215-583-8171

I have recently modified my C1P so that I no longer have to use SHIFT 0 for a backspace. I also moved the BREAK switch further away from the keyboard. The main reasons I did this was for easier typing, and preventing the computer from being accidentaly reset. (I'm a fumble fingered typist.) Let me give you some background on this.

I bought Aardvark Technical Services C1S monitor EPROM for my C1P about two years ago. When using this monitor, releasing the SHIFT LOCK key gives you a typewriter mode keyboard. Both SHIFT keys do the same thing, and there are no more garbage characters. There is still one problem left. Typing SHIFT 0 gives capital letter O instead of backspace. To backspace you must depress SHIFT LOCK to get upper case mode, then use SHIFT 0 to backspace, and unlock SHIFT LOCK to get back into typewriter mode. In my mind this is too many keystrokes. With all the extra key-strokes, three times out of ten I forget to unlock SHIFT LOCK and type in a few upper case letters where they are not wanted. I also kept hitting the BREAK key and occasionally losing programs

and data. The frustrationof these problems resulted in a hardware and software solution.

The first item needed to implement this is the most important. You must have an EPROM monitor ROM instead of the original ROM. This modification involves changing one byte of code in the monitor input routines. The next important item is access to an EPROM programmer. The other items needed are: anew 2716 EPROM, one small momentary pushbutton switch with normally open contacts, three feet of wire-wrap wire, schematic diagram of the computer - preferably the SAMS FOTOFACT, soldering iron, solder, and an X-ACTO knife.

In effect we will replace the BREAK switch with a new push-button switch mounted further away from the keyboard, and use the BREAK key as our new backspace key. After unpluging your CIP, dissasemole it by removing the six large screws from the bottom of the cabinet and removing the five screws from the cpu board. Gently lift the board out. Unplug the power supply and the video/cassette connectors plus any other connectors pluged in. Lay the cpu board down, right side up, with the keyboard facing you. Locate the BREAK key. Notice the small trace running out from under it at about forty-five degrees towards the rear. (All work will be on the top side of the board.) Where it bends to head straight back, cut it with the X-ACTO knife. In the same trace, make a second cut next to it, about 1/32 of an inch away from it. Gently using the knife point peel away the small bit of the trace between the cuts. Be carful not to lose the small bit of copper into the computer. Throw it away. Make two more cuts in the large ground trace, to the

right of the BREAK key, below the BREAK switch about 1/4 inch from the switch and two cuts in the ground trace above the switch about 5/8 inch from it. Dispose of these small bits of copper also. Take a good look at all your cuts and make sure that the BREAK switch is isolated from the ground trace on both sides and not touching the small trace beyond the elbow where you cut it. Take a piece of wire about three inches long and solder one end to the ground trace below the BREAK switch below the cut. Solder the other end to the ground trace above the switch above the cut. Take two pieces of wire about seven inches long, soldering one end of both wires to the new push button switch terminals. Solder one of the switch wires to the ground trace where we have already soldered a wire. The other wire gets soldered to the small trace coming from the BREAK switch above the cut we made in it. The pushbutton is our new BREAK switch. Now take the case to another location and drill a 1/4 inch hole midway between the cutout for the BREAK key and the edge of the case. You could put the switch elsewhere if you want. Make sure that all the metal chips are cleaned out of the case and check the hole for burrs. Once both sides of the new hole are clean and smooth return to the cpu board.

Solder one end of a one inch wire to the small trace coming out from under the BREAK switch below the cut. Solder the other end to the plated-through-hole which is about 1/4 of an inch from the small trace, right next to the return key. Solder one end of a five inch wire to the fat trace coming out from under the BREAK key, above the switch. This was ground before we made our cuts. There is a row of resistors above the 8, 9, and Ø keys, standing on end.

The fifth resistor in from the right side is the one we want. Solder the other end of the five inch wire to the trace that touches this resistor (top side of board). If the resistor is mounted so that the exposed lead touches the trace, solder to the resistor lead instead. This finishes our hardware changes.

Electricaly speaking, what we have done is add a new switch to the keyboard matrix in one of the three blank places. Look at your schematic. The switch connects keyboard row five with column one. The byte that translates the keyboard output into ASCII for this switch location is stored at FDE9 hex in Aardvark's CIS monitor EPROM. It presently is a zero or an ASCII null. To be a backspace character it must be changed to a DF hex. DF hex is ASCII 5F hex or underline (OSI's backspace) with bit seven set to one. Bit seven set to one tells the input routine not to unshift this key when the SHIFT LOCK key is released. The data for all the keys in the CIS monitor is stored in a table from FDCD hex to FDFD hex. Looking through this data you will notice three locations with zero in it. If you look at the schematic of the keyboard, You will see three keys missing on rows five and six (shift key row doesn't count). The zero bytes refer to these three switches.

You may have to reassemble your CIP to program the 2716 EPROM. Or you could us a friends computer or EPROM programmer to do it. What ever way you choose, you are going to need your old CIS EPROM in memory. Using a new 2716 EPROM is the easiest way to make the one byte change. I used Aardvark's EPROM burner because it reads out from the RAM memory on the computer. Listing one will copy the monitor into memory where the one byte can be changed. Answer 6143

to memory size when cold starting the computer. This loads the monitor into the RAM memory 1800 hex to 1FFF hex and makes the change needed for backspace. Some other changes can also be made at this point. Monitor locations F80B hex to F80E could be changed to EA hex (NOP). These four bytes strip off bit seven from the character to be printed. It does this to correct the error message output and keeps you from printing graphics characters. If you make this change be aware that the odd error messages will be back, but you can print graphics characters again. Address F8EA hex is the cursor character, F88A hex to F8C5 hex is the 'ready' prompt, FF7E hex is the number of nulls output after the cariage return when in SAVE mode. It is 0A hex or ten decimal but I have changed mine to 01 hex. This makes for faster printing at 300 baud. To set nulls for saving to tape use NULL command or POKE 13, N with N = number of nulls, up to 255.

Once you have the monitor set into RAM memory and have changed the bytes you want, warm start, type NEW, and load in the EPROM programmer program. Start adress is 1800 hex and end address is 1FFF hex. EPROM start address is zero (0). After the EPROM has been programmed and verified, install it in the ROM monitor socket U13. Make sure you orient it the same way that the old one was. Reassemble the CIP the way it came apart, Taking care not to crush anything. Put the new pushbutton switch in the 1/4 inch hole. Don't forget to plug all jacks back in. When done, plug in and turn the computer on. Cold start and type something in. Try your new backspace key. You should have no problems.

If you don't have an EPROM programmer or don't want to atempt this

n

modification your self, you can send me your CIS monitor and one dollar to cover return postage and I will reprogram it for you. Wrap the EPROM in foil an insert it inside a cassette box. Put this on a larger piece of cardboard and tape in place, and put it in a large envelope. Mark FRAGILE in a few different locations.

Now that I have a new backspace key, word processing is much easier. If you can think of any other character code to use for the two remaining blank key locations, Write a letter and let the rest of us know about it.

Listing One

```
4  REM A=1800,B=F800 HEX
5  A=6144:B=63488
10 FOR I=0 TO 2047
14 REM GET MONITOR CODE
15 C=PEEK(I+B)
19 REM STORE IT
20 POKE I+A,C
30 NEXT
39 REM MAKE IDE9 (FDE9) DF HEX
40 POKE 7655,223
50 END
```

OSI CA-23 EPROM PROGRAMMER

Terry Terrance

If you want to program some EPROMs and you have a current 48 pin buss machine (C8P DF or C4P MF or DF) with the 16 pin I/O buss you're in luck, OSI has something that just might work for you. Introduced in early 1981, the CA-23 will program most single source (i.e. +5V only) EPROMs up through the 64K (8Kx8) size units.

The CA-23 is a bare circuit board with attached rubber feet, much like a Superboard but smaller. The user must supply +5V and +25.6V at 0.5 amps each to operate the board. Connection to the host computer is by means of a 16 pin DIP jumper to the host machine's 16 pin I/O buss, it does not go directly onto the machine's buss like the older 450 board. So if you have an older 48 pin buss machine, you'll need an operational A-15 card which usually comes only on current-model disk machines.

The board is well-laid-out with all chips socketed and 4 ZIP (Zero-Insertion-Pressure) sockets for Master and Copy EPROMs in both 24 pin and 28 pin varieties. LED indicators are provided for indicating Programming, Error and Complete modes as well as power on. Three slide switches control power on/off, read/write, and type A/B.

The programmer is accompanied by a disk of routines designed to make the thing go. The software does work, but the routines are so rudimentary as to be the weak link in an otherwise good system.

The documentation included is somewhat unusual for OSI, copious in volume - but of typically poor quality. The manual gives enough information to get the programmer running in the limited manner that the software allows, but gives no insight into how the programmer works, making life difficult for the user who would like to make hardware or software modifications.

The first problem that the user of this board will have to overcome is the power supply. The +5 is easy enough to supply, either from the computer itself or an easily obtained external supply. The +25.6V, however, is quite another matter. This value is not available in a fixed-output, one-chip voltage regulator, nor is it possible to add diodes in the ground lead of a fixed output regulator to fake the chip into producing this output. The only way I could think of for supplying this voltage was to use a variable output supply set at 25.6V. This can be very expensive if your only alternative is to buy a variable supply to accomodate this one board. My solution was to build a dedicated supply for the programmer using a MG 7805 for the +5V, and a LM 317 regulator for the +25.6V. Ordinarily the family of EPROMs

that this board is designed to handle requires only 25V to do the actual "burning", but OSI added a blocking diode into the high voltage line to prevent damage from a reverse polarity hook-up and the voltage drop inherent to the diode adds the extra 0.6V.

Once you have a power supply, connecting the board is straightforward. Operation could have been made simple enough for an untrained user, but alas, you'll need to read the manual before you try to use the software even though the programs appear to be menu driven. The software allows five basic functions: Duplicate, Verify, List, Program from memory, Load memory and Edit/Save memory. All of these basic functions allow several options in their implimentation. For instance, all of the modes which actually allow programming a ROM give the user the option of checking the ROM for all $FF's or not (when an EPROM is erased all locations should be in the $FF state), allowing the user to make multiple attempts to program a reluctant ROM or to program a partially pre-programmed ROM. The software functions acceptably and the ROM can be manipulated in any way that one could want - even though it may take several operations to do so. Duplicate, Verify and List modes should be self-explainatory. Program from memory allows the copy EPROM to be programmed from a memory buffer that the software creates starting at $5000. Load memory allows that memory buffer to be loaded either from the copy ROM socket, from the master ROM socket or from a disk file. Edit/Save mode allows the buffer to be manipulated and/or saved to a disk file.

The software is written in BASIC with a machine language device handler to actually do the reads and writes to the board. The programmer programs rather slowly so the necessity of a machine language routine escapes me. User changes to the BASIC sections of the programs are easily accomplished and some of the modifications that the user might consider could be: a real menu-driven format, dedicated versions eliminating repetitive entries necessary to set-up the programmer for a specific ROM type, and possibly a production mode. This last possibility is particularly interesting since the host computer can detect the operation of the switches on the programmer, and the CA-23 can flash status lights at the operator, so it should be possible to write software so that all that would be necessary to do "production" runs of ROMs would be to watch the lights and operate the On/Off switch.

The hardware is a simple, clean design centered around a pair of 6821's, it's much simpler and, in my opinion, much less likely to fail than the design recommended by Intel. Also, since the programmer is basically software driven, it should be adaptable to higher density EPROMS, like the recently introduced 128K (16Kx8) units, with little hardware change. The two master and two copy sockets are high quality ZIP units. A laudable feature of the CA-23 is that programming voltage is never present on the master socket making accidental modification of the master ROM impossible.

The only problem that I had with the harware end of the system was that some individual ROMs seemed to read poorly when they were in the programmer. I suspect that this is due to a timing problem, specifically the fact that my C8 runs at 2 Mhz and most EPROMs and even the 6821's on the programmer are 1 Mhz components. The result of this possible incompatibility was that out of about 27 EPROMs that I tried to program, one would not program correctly no matter how many times I tried.

In all, the CA-23 works and with suitable software will be a really top-notch EPROM programmer for the hobbyist, designer or small production runs. Since I got my CA-23 over a year ago and I received mine at dealer cost, you'd better consult a current OSI price list for today's cost.

OS65D V3.3 KEYBOARD ROUTINE

By Mike Cohen

OS65D Version 3.3 has a keyboard scanning routine at $32CC which doesn't wait for a key to be pressed. It will return the ASCII value of the key or a zero if no key is pressed. This makes it ideal for real-time games and terminal programs.

Here is a small BASIC program which demonstrates how it can be used. The machine language program gets imbedded in line 0 of the program. This technique of burying machine language programs in a BASIC program, which has been documented in Steve Hendrix's HEXDOS newsletter, saves memory space since none has to be reserved, and protects the program from BASIC. When this is done, however, the program *MUST* not contain any zero bytes since BASIC will mistake it for a new line. This is the reason for the LDA $3A7E when a zero is needed.

```
0 REM ............... : DO NOT EDIT THIS LINE!
10 AD=14975:RESTORE
20 IFPEEK(AD)<>46 THEN AD=AD+1:GOTO 20 :REM '.'
30 ad=ad+1:FOR I=0 TO 9:READ X:POKE AD+I,X:NEXT I
40 U1=AD AND 255:U2=INT(AD/256)
50 POKE 574,U1:POKE 575,U2: REM USR VECTOR
60 DATA 32,204,50,168,173,126,58,76,24,18,-1
70 READ X:IFX<>-1 THEN PRINT"**** DATA ERROR ****":END
80 REM DEMO PROGRAM ------------
90 C=0:L=300
100 C=C+1:X=USR(0):IF X<>0 THEN 130
110 IF C=L THEN PRINT:PRINT"Time limit expired":END
120 GOTO 100
130 PRINTCHR$(x);:GOTO 90
900 REM ASSEMBLER LANGUAGE ROUTINE IS:
910 REM   JSR $32CC ; 65D3.3 KEYSCAN
920 REM   TAY       ; PUT CHR IN 'Y' FOR BASIC
930 REM   LDA $3A7E ; WHICH CONTAINS ZERO
940 REM   JMP $1218 ; BASIC OUTVAR ROUTINE
950 REM USR(X) WILL RETURN 0 IF NO KEY PRESSED
960 REM ROUTINE GETS IMBEDDED IN LINE 0 AND
970 REM CANNOT CONTAIN ANY ZERO BYTES.
```

```
OK

 99 REM-"GO"BY JOHN KULA--COPYRIGHT 1982
 100 GOTO800
 105 GOSUB225:GOSUB275:GOSUB200:GOSUB400:RETURN
 110 GOSUB225:GOSUB275:GOSUB200:RETURN
 200 FORQ=1TO32:PRINT:NEXT:DR=D:FORVT=0TO18:FORHT=0TO18
 205 POKEDR+HT,MI(HT,18-VT):NEXTHT:DR=D+(32*(VT+1)):NEXTVT:FORQ=0TO9
 210 POKED+607-(32*Q),48+Q:POKED+286-(32*Q),49:POKED+287-(32*Q),48+Q
 215 NEXT:POKED+607,32:FORQ=0TO18:POKED+640+Q,65+Q:NEXT:RETURN
 225 FORP=1TOGM:IFS(P)=0THENG=P:GOTO235
 230 NEXT:GM=GM+1:G=GM
 235 MG(H,V)=G:S(G)=MI(H,V)
 240 FORP=1TO4:F(P)=51:NEXT:IFH)0THENF(1)=MG(H-1,V)
 245 IFH(18THENF(2)=MG(H+1,V)
 250 IFV)0THENF(3)=MG(H,V-1)
 255 IFV(18THENF(4)=MG(H,V+1)
 260 RETURN
 275 FORP=1TO4:IFF(P))0ANDS(F(P))()MI(H,V)THENGT=F(P):GOSUB325
 280 NEXTP:FORP=1TO4:IFS(F(P))()MI(H,V)THEN295
 285 FORVT=0TO18:FORHT=0TO18:IFMG(HT,VT)=F(P)THENMG(HT,VT)=G
 290 NEXTHT,VT:L(F(P))=0:S(F(P))=0:F(P)=G
 295 NEXTP:FORP=1TO4:IFF(P))0THENGT=F(P):GOSUB325
 300 NEXTP:GT=G:GOSUB325:RETURN
 325 L(GT)=0:FORVT=0TO18:FORHT=0TO18:IFMG(HT,VT)()GTTHEN365
 330 IFHT)0THENHL=HT-1:VL=VT:FL=1:GOTO355
 335 IFHT(18THENHL=HT+1:VL=VT:FL=2:GOTO355
 340 IFVT)0THENHL=HT:VL=VT-1:FL=3:GOTO355
 345 IFVT(18THENHL=HT:VL=VT+1:FL=0:GOTO355
 350 GOTO365
 355 IFMG(HL,VL)=0THENMG(HL,VL)=-1:L(GT)=L(GT)+1
 360 ONFLGOTO335,340,345
 365 NEXTHT,VT:FORVT=0TO18:FORHT=0TO18:IFMG(HT,VT)=-1THENMG(HT,VT)=0
 370 NEXTHT,VT:IFL(GT))0THENRETURN
 375 FORVT=0TO18:FORHT=0TO18:IFMG(HT,VT)()GTTHEN385
 380 MI(HT,VT)=46:S(GT)=0:MG(HT,VT)=0
 385 NEXTHT,VT:GT=0:RETURN
 400 GOTO435
 405 FORVT=0TO18:FORHT=0TO18:IFMG(HT,VT)()ZTHEN430
 410 IFHT)0ANDMG(HT-1,VT)=0THENH=HT-1:V=VT:RETURN
 415 IFVT)0ANDMG(HT,VT-1)=0THENH=HT:V=VT-1:RETURN
 420 IFHT(18ANDMG(HT+1,VT)=0THENH=HT+1:V=VT:RETURN
 425 IFVT(18ANDMG(HT,VT+1)=0THENH=HT:V=VT+1:RETURN
 430 NEXTHT,VT
 435 FORZ=1TOGM:IFL(Z)()1ORS(Z)=BTHEN445
 440 GOSUB405:RETURN
 445 NEXTZ:FORZ=1TOGM:IFL(Z)()1ORS(Z)=WTHEN460
 450 GOSUB405:GOSUB240:Y=0:FORX=1TO4:IFF(X)=0THENY=Y+1
 455 NEXTX:IFY)2THENRETURN
 460 NEXTZ:Q=0:FORVT=V-2TOV+2:FORHT=H-2TOH+2:Q=Q+1:F(Q)=0
 465 IFHT)-1ANDHT(19ANDVT)-1ANDVT(19THENF(Q)=MI(HT,VT)
 470 NEXTHT,VT
 471 IFF(18)=46ANDF(17)=BANDF(19)=BTHEN535
 472 IFF(14)=46ANDF(9)=BANDF(19)=BTHEN530
 473 IFF(8)=46ANDF(7)=BANDF(9)=BTHEN520
 474 IFF(12)=46ANDF(7)=BANDF(17)=BTHEN525
 475 IFF(18)=46ANDF(19)=46ANDF(17)=BANDF(20)=BTHEN535
 476 IFF(14)=46ANDF(9)=46ANDF(4)=BANDF(19)=BTHEN530
 477 IFF(8)=46ANDF(7)=46ANDF(6)=BANDF(9)=BTHEN520
 478 IFF(12)=46ANDF(17)=46ANDF(7)=BANDF(22)=BTHEN525
 479 IFF(18)=46ANDF(17)=46ANDF(16)=BANDF(19)=BTHEN535
 480 IFF(14)=46ANDF(19)=46ANDF(9)=BANDF(24)=BTHEN530
 481 IFF(8)=46ANDF(9)=46ANDF(7)=BANDF(10)=BTHEN520
 482 IFF(12)=46ANDF(7)=46ANDF(2)=BANDF(17)=BTHEN525
 483 IFF(18)=46ANDF(17)=46ANDF(19)=46ANDF(16)=BANDF(20)=BTHEN535
 484 IFF(14)=46ANDF(9)=46ANDF(19)=46ANDF(4)=BANDF(24)=BTHEN530
 485 IFF(8)=46ANDF(7)=46ANDF(9)=46ANDF(6)=BANDF(10)=BTHEN520
 486 IFF(12)=46ANDF(7)=46ANDF(17)=46ANDF(2)=BANDF(22)=BTHEN525
 487 IFF(12)=46ANDF(17)=46ANDF(16)=BANDF(18)=BTHEN525
 488 IFF(18)=46ANDF(19)=46ANDF(14)=BANDF(24)=BTHEN535
 489 IFF(14)=46ANDF(9)=46ANDF(8)=BANDF(10)=BTHEN530
 490 IFF(8)=46ANDF(7)=46ANDF(2)=BANDF(12)=BTHEN520
 491 IFF(14)=46ANDF(19)=46ANDF(18)=BANDF(20)=BTHEN530
 492 IFF(8)=46ANDF(9)=46ANDF(4)=BANDF(14)=BTHEN520
 493 IFF(12)=46ANDF(7)=46ANDF(6)=BANDF(8)=BTHEN525
 494 IFF(18)=46ANDF(17)=46ANDF(12)=BANDF(22)=BTHEN535
 495 IFF(18)=46ANDF(17)=BAND(F(14)=BORF(15)=B)THEN535
```

```
496 IFF(14)=46ANDF(19)=BAND(F(8)=BORF(3)=B)THEN530
497 IFF(8)=46ANDF(9)=BAND(F(12)=BORF(11)=B)THEN520
498 IFF(12)=46ANDF(7)=BAND(F(18)=BORF(23)=B)THEN525
499 IFF(18)=46ANDF(12)=BAND(F(19)=BORF(20)=B)THEN535
500 IFF(14)=46ANDF(18)=BAND(F(9)=BORF(4)=B)THEN530
501 IFF(8)=46ANDF(14)=BAND(F(7)=BORF(6)=B)THEN520
502 IFF(12)=46ANDF(8)=BAND(F(17)=BORF(22)=B)THEN525
503 IFF(18)=46ANDF(14)=BANDF(16)=BANDF(17)=46THEN535
504 IFF(14)=46ANDF(19)=46ANDF(8)=BANDF(24)=BTHEN530
505 IFF(8)=46ANDF(9)=46ANDF(10)=BANDF(12)=BTHEN520
506 IFF(12)=46ANDF(7)=46ANDF(2)=BANDF(18)=BTHEN525
507 IFF(8)=46ANDF(14)=46ANDF(7)=BANDF(15)=BTHEN520
508 IFF(12)=46ANDF(8)=46ANDF(3)=BANDF(17)=BTHEN525
509 IFF(18)=46ANDF(12)=46ANDF(11)=BANDF(19)=BTHEN535
510 IFF(14)=46ANDF(18)=46ANDF(9)=BANDF(23)=BTHEN530
515 GOTO550
520 V=V-1:RETURN
525 H=H-1:RETURN
530 H=H+1:RETURN
535 V=V+1:RETURN
550 FORZ=1TOGM:IFL(Z)<>2ORS(Z)=BTHEN560
555 GOSUB405:A=1:RETURN
560 NEXTZ:FORZ=1TOGM:IFL(Z)<>2ORS(Z)=WTHEN570
565 GOSUB405:RETURN
570 NEXTZ:F=-1:IFV<10THEN580
575 F=F+1
580 IFH<10THENF=F+2:GOTO590
585 F=F+4
590 ONFGOTO595,610,625,640
595 IFH>0ANDV>0ANDMG(H-1,V-1)=0THENH=H-1:V=V-1:RETURN
600 IFH>0ANDV>1ANDMG(H-1,V-2)=0THENH=H-1:V=V-2:RETURN
605 IFH>1ANDV>0ANDMG(H-2,V-1)=0THENH=H-2:V=V-1:RETURN
610 IFH>3ANDV<18ANDMG(H-1,V+1)=0THENH=H-1:V=V+1:RETURN
615 IFH>1ANDV<18ANDMG(H-2,V+1)=0THENH=H-2:V=V+1:RETURN
620 IFH>0ANDV<17ANDMG(H-1,V+2)=0THENH=H-1:V=V+2:RETURN
625 IFH<18ANDV>0ANDMG(H+1,V-1)=0THENH=H+1:V=V-1:RETURN
630 IFH<18ANDV>1ANDMG(H+1,V-2)=0THENH=H+1:V=V-2:RETURN
635 IFH<17ANDV>0ANDMG(H+2,V-1)=0THENH=H+2:V=V-1:RETURN
640 IFH<18ANDV<18ANDMG(H+1,V+1)=0THENH=H+1:V=V+1:RETURN
645 IFH<17ANDV<18ANDMG(H+2,V+1)=0THENH=H+2:V=V+1:RETURN
650 IFH<18ANDV<17ANDMG(H+1,V+2)=0THENH=H+1:V=V+2:RETURN
655 H=0:V=0:RETURN
800 FORQ=1TO32:PRINT:NEXT:PRINTTAB(10);"GO":PRINT:PRINT
805 PRINT"COPYRIGHT 1981 JAN KULA":PRINT"2522 BELMONT, VICTORIA
810 PRINT"B.C., CANADA, V8R 4A4":FORQ=1TO5:PRINT:NEXT
815 CLEAR:B=233:W=232:D=53415:DIMMI(18,18),MG(18,18),S(30),L(30),F(30)
820 FORV=0TO18:FORH=0TO18:MI(H,V)=46:NEXTH,V:FORV=3TO15STEP6
825 FORH=3TO15STEP6:MI(H,V)=111:NEXTH,V:GOSUB200
830 PRINT"WHAT HANDICAP DO YOU":INPUT"WANT ME TO HAVE (1-9)";R
835 ONR-RGOTO845,850,855,860,865,870,875,880
840 MI(9,9)=B
845 MI(3,9)=B
850 MI(15,9)=B
855 MI(9,15)=B
860 MI(9,3)=B
865 MI(15,3)=B
870 MI(3,15)=B
875 MI(15,15)=B:MI(3,3)=B:GOTO885
880 MI(16,4)=B
885 FORV=0TO18:FORH=0TO18:IFMI(H,V)<>BTHEN895
890 G=G+1:S(G)=B:MG(H,V)=G:L(G)=4
895 NEXTH,V:GM=G:GOSUB200
900 PRINT"YOUR MOVE:";:INPUTH$,V$:IFH$="0"THENGOSUB200:END
905 IFASC(H$)>64THENH=ASC(H$)-65:V=VAL(V$)-1:GOTO915
910 H=ASC(V$)-65:V=VAL(H$)-1
915 IFH<0ORH>18ORV<0ORV>18THEN940
920 IFMG(H,V)<>0THEN940
925 GOSUB240:FORQ=1TO4:IFS(F(Q))<>BTHEN945
930 NEXT:PRINT"NO LIBERTIES!":INPUT"CONFIRM MOVE";Q$
935 IFLEFT$(Q$,1)="Y"THEN945
940 PRINT"ILLEGAL MOVE":INPUT"READY TO TRY AGAIN";Q$:GOSUB200:GOTO900
945 MI(H,V)=W:GOSUB105
950 MI(H,V)=B:GOSUB110
955 PRINT"MY MOVE:";CHR$(65+H);V+1;:IFA=1THENPRINT"ATARI!":A=0
960 GOTO900
OK
```

```
20000 REM-WPsoon-COPYRIGHT 1982, SHEL SACKS
20020 REM-10.5.82
20100 FORI=0TO50:PRINT:NEXT:GOSUB30000:GOTO20400
20200 FORJ=0TOTW-1:MEM=MEM+1:Y=PEEK(UL+J):POKEMEM,Y:NEXT
20220 PRINT:IFP(2THENMEM=MO
20240 IFL/LI=INT(L/LI)THENL=0:P=P+1
20260 L=L+1:POKESL+7,INT(L/10)+48:POKESL+8,L-10*INT(L/10)+48
20280 POKESL+20,INT(P/10)+48:POKESL+21,P-10*INT(P/10)+48
20300 IFMA=1OR2THENPOKEUL-1,62
20320 RETURN
20400 FORI=0TO50:PRINT:NEXT:FORI=0TO23:READX:POKESL+I,X:NEXT:LN=LL-TW
20500 REM-INPUT ROUTINE
20520 POKE11,0:POKE12,253:FORI=0TOTW-1
20560 POKEST+I,CU:X=USR(X):X=PEEK(531):II=I+LN
20600 IFI(0ANDMA()3ANDX()127ANDII/LL=INT(II/LL)THENPOKEST+I,32:I=I+LN
20640 IFX=127THEN21000
20660 IFX=10ORX=13THEN22000
20680 IFX=27THEN23000
20700 POKEST+I,X:MEM=MEM+1:POKEMEM,PEEK(UL+I)
20800 NEXT:GOSUB20220:GOTO20500
21000 REM-RUBOUT ROUTINE
21100 POKEST+I,32:I=I-1:IF(I+1)/LL=INT((I+1)/LL)THENIFMA=1OR2THENI=I-L
21140 GOTO20560
22000 REM-LF&RETURN
22100 POKEST+I,32:GOSUB20200:IFX=13THEN20500
22160 GOTO20560
23000 PRINT"EDIT ROUTINE":GOTO20500
30000 POKE15,255:REM-PARAMETERS
30020 PRINT"MACHINE IN USE          1P-1;1PII-2;2,4,8P-3"
30040 INPUTMA:PRINT:INPUT"DEFAULT SCREEN(1)OR SET LIMITS(2)";DE:PRINT
30080 INPUT"STD CURSOR(1)OR CHAR CODE";CU:PRINT:IFCU=1THENCU=95
30100 MO=2817:MEM=MO:L=1:P=L:PRINT:INPUT"K OF RAM";RAM
30120 PRINT:INPUT"PRINTER LINES/PAGE";LS:INPUT"PRINTER CHARS/LINE";CL
30200 IFDE=2THEN30500
30220 IFMA=1THENTW=24:LI=24:ST=54085:UL=53317:LL=32
30240 IFMA=2THENTW=48:LI=12:ST=54091:UL=53323:LL=64
30260 IFMA=3THENTW=64:LI=30:ST=55104:UL=53248:LL=64
30280 SL=ST+2*LL:RAM=RAM*1024-100
30300 RETURN
30500 PRINT:INPUT"TERM WIDTH";TW:PRINT:INPUT"LOCATION,UP/LEFT";UL
30520 PRINT:INPUT"#LINES";LI:PRINT:INPUT"LOC,LOW/LEFT";ST
30540 LL=64:IFMA=1THENLL=32
30560 SL=ST+2*LL
30800 RETURN
31000 D TA108,105,110,101,35,32,32,32,32,32,32,32,32,32,32
31020 DATA112,97,103,101,35,32,32,32,32
39000 REM-OUTPUT
39100 FORK=1TOLI:GOSUB20200:NEXT
39120 INPUT"OUT OF MEM--OUTPUT TO TAPE(1) OR PRINTER(2)";OU
39140 IFOU=1THENSAVE:GOTO39900
39160 INPUT"HIT 1 WHEN PRINTER READY";R:SAVE
39200 GOTO39920
39900 INPUT"HIT 1 WHEN RECORDER READY";R:FORK=MOTORAM
39920 PRINTCHR$(PEEK(K));:NEXT:POKE517,0:FORI=1TO10:PRINT:NEXT
39960 INPUT"OUTPUT FINISHED:HIT 1 TO CONTINUE INPUT";R
39980 MEM=MO:FORI=1TO50:PRINT:NEXT:GOTO20500
OK
```