computer notes

Nov '77

50¢

Volume 3 Issue 6

New!

88-MU1 Music Card See Story on Page 3

Photograph by Steve Wedeen

Compose Yourself with the New Altair 88-MU1

By Thomas G. Schneider
MITS

Through the gray gloom and the midnight mist swirling around the gnarled branches of long-dead vegatation, the castle loomed dark and foreboding on the edge of a huge cliff. I viewed the scene with some apprehension, but called to the driver to move on. When the ancient creaky carriage finally rumbled into the cobblestoned courtyard, I thought that I heard swells of medieval organ music booming ominously through the stone walls. "How gothic," I quipped to myself, jumping down from the carriage and peering suspiciously at the "KILOBAUD Sold Here" sign in the window.

Approaching the heavy wooden door with large brass knockers, I had a funny feeling of deja vu. Hmm. Maybe it was that Gene Wilder movie about monsters I had seen recently. Just then the door opened abruptly, and a black-cloaked gentleman with pointed teeth appeared. Bowing, he introduced himself as the count.

"You've probably heard this line before," he said in a slow, thick accent, "but, good evening. Welcome to my castle. Your rooms are awaiting. Dinner will be served at 8:00. Afterwards, we will give the demonstration," he said with a ghoulish smile as he turned to leave.

continued page 2

Editor
Andrea Lewis
Assistant Editor
Linda Blocki
Production
Al McCahon
Susan Blumenthal
Tom Antreasian
Contributors
Thomas G. Schneider
Bennett Inkles
Susan Blumenthal
Robert Lopez
Steve Grider
Thomas Durston
Gale Schonfeld
Gary Runyon
Lee Wilkinson
Doug Jones
Ken Knecht
Doyl Watson

in this issue

*Important Note On Page 19

©mits Inc. 1977
a subsidiary of Pertec Computer Corporation
2450 Alamo SE, Albuquerque, NM 87106

As I prepared for dinner, I wondered what he had in store for me. Strange man, this count . . .I couldn't help but think I knew him from somewhere else. Oh well, the demonstration would be interesting.

After a delicious repast of undetermined substance, the count led me down a wooden cobwebbed stairway to what I assumed could only be the dungeon. "Don't mind the bats," he said. "They give the place character." He fumbled with the heavy iron padlock and pushed against the old dungeon door. My heart raced. Finally, the door gave way and slowly creaked open to reveal an amazing spectacle.

I had expected to see an immense pipe organ of the kind usually seen only in well-preserved European cathedrals, but I was wrong. Occupying all four walls of the dungeon and reaching almost to the ceiling was the largest collection of sound equipment I had ever laid eyes upon. Completely covering three walls were woofers, tweeters, midranges, folded horns, ring radiators, and all sorts of sound reproducing devices. The fourth wall was obscured by racks and racks of high-power audio amplifiers, tape machines, equalizers, and other audio processing equipment. "Listen carefully," he said, flipping up a bat-handle toggle switch.

The machinery clicked, popped, and buzzed for several mintues before I finally heard what I had come all this way to experience. Emanating simultaneously from hundreds of speakers came the most musically precise rendition of Johann Sebastian Bach's **Toccata and Fugue in D Minor** that I had ever heard. Every massive chord, every subtle passage was accurately reproduced. But from where??? None of the tape machines were running... something strange was going on here. As strains of the Fugue floated through the dungeon I asked the count how it was all done.

"Very simply," he replied, pointing to an object in the corner.

"An Altair? What are you doing with an Altair? Counting bats?!"

"Let's not be silly, my good man," he said, somewhat miffed. "Nowadays, what self-respecting vampire would be without a computer? Besides, how else could I make such splendid music?"

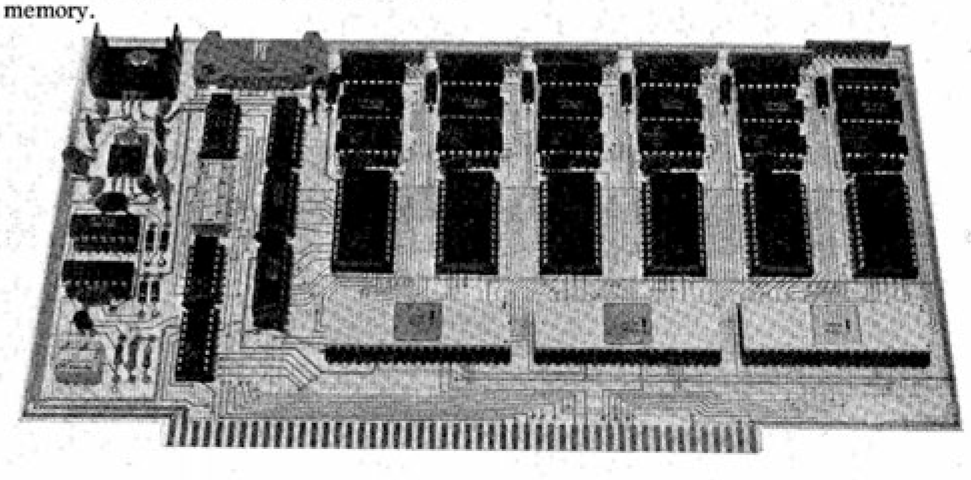
"You must be joking. How can a microcomputer do all this?"

"Very easily," he said. "Since my friends at MITS came up with the 88-MU1 and the MOS-DOS software for composition, I can play just about anything using my Altair!"

"Tell me more," I implored.

"Very well," he sighed and provided me with the following information.

The Altair 88-MU1 is a polyphonic six-channel note generator card. With it, the user can generate, under complete software control, six independent musical sequences all running simultaneously in real time. The 88-MU1 comes with a sophisticated, high-level software package with full composition and editing capabilities. It also includes output connectors designed to connect to most stereo amplifiers. The software package will run in any Altair disk system with at least 16K of memory.



Altair™ Note Synthesizer Board (88-MU1)

Composition using the 88-MU1 software is simple. The software allows the creation of six independent text files which can be saved and recalled from disk. Each group of six files can be given a common name up to eight characters long. The 88-MU1 software also incorporates a powerful text editor for listing files, inserting or deleting lines, and renumbering files.

Listing 1 is a sample listing for one channel of a six-channel composition. Each line contains three fields describing note, octave and timing parameters. For example, line 1 specifies a C note in the fourth octave lasting 1/8 of a second. Line 2 specifies a D note in the fifth octave lasting 1/8+1/16 of a second. (The period after the eight specifies a dotted eighth note.) Line 3 specifies an F# note in the seventh and eighth octaves lasting one second. The length of each channel of a composition is limited only by the amount of memory in the user's machine.

```
Listing 1
1  C, 4, 8
2  D, 5, 8
3  F#, 78, 1
```

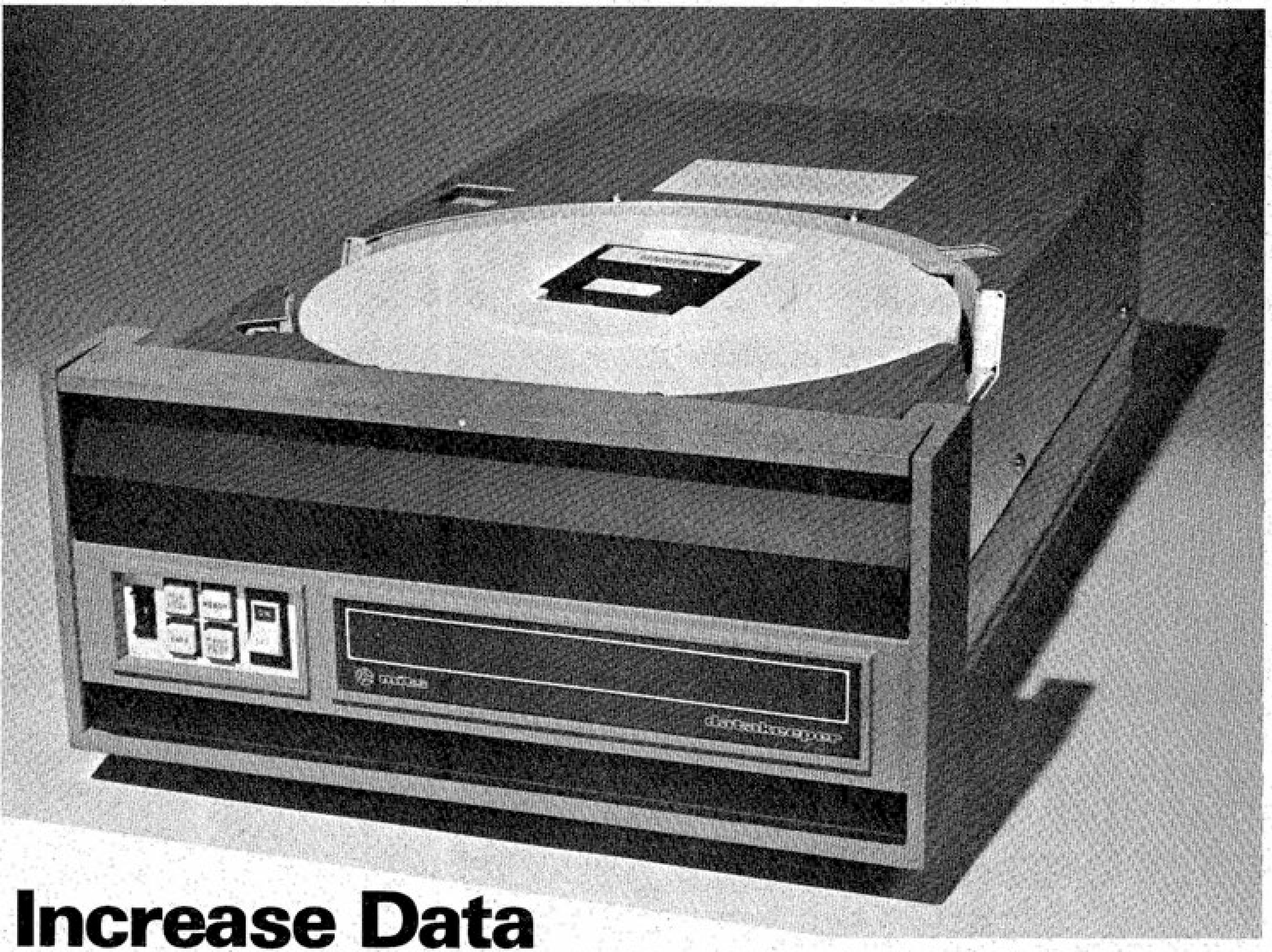
As the system is expanded, special characters may be added to the end of each line. These characters will control such functions as envelope shaping, filtering, and vibrato effects. After all channels of the composition have been entered, the composition can be played at a variety of tempos determined by the user.

For those users desiring musical effects, the 88-MU1 can also be easily accessed by user routines written in machine code. Figure 1 shows what the 88-MU1 looks like to software. The base address can be set from 0 to octal 360 in increments of 16. For even more flexibility, the 88-MU1 can accept two external signals: one is the reference frequency for the 88-MU1's pitch generator. This signal is normally derived from the Altair 8800's two MHZ clock, but can also be externally applied by the user. For example, inputting a one MHZ signal will cause the 88MU1's entire range to be shifted down one octave. The other signal is the software synchronization signal. It normally occurs at a frequency of 128 HZ, but can be externally applied, giving the user control of the rate of the composition execution speed.

"This 88-MU1 is fascinating," I said to the count.

"Yes indeed, most remarkable. . .but unfortunately, I must be leaving you now," he said. "It's getting close to dawn, so I must retire. I trust the demonstration pleased you." he remarked as he escorted me to the courtyard where the same black carriage was waiting. "Most impressive. I enjoyed every bit of it."

As the carriage started rolling, I couldn't help but lean out the window and shout, 'Fangs a lot for everything!" The count grimaced painfully as the carriage moved through the castle gate. But I hurried on, eager to get home and treat my Altair to a brand new 88-MU1.



Increase Data Storage up to 80 MBytes with Altair™ Hard Disk System

By Bennett Inkeles
MITS

The new Datakeeper Hard Disk System (88-HDSK) from MITS offers a unique form of expanded mass storage for Altair 8800 series microcomputers. It consists of the Altair Datakeeper Controller and a Pertec D3422 Hard Disk Drive. The 88-HDSK has a data storage capacity of approximately 10 MBytes.
(A 20 MByte drive option is also available. Business management, education, and scientific applications are among the numerous possibilities in which the 88-HDSK may be incorporated.

The following components make up and are included with the purchase of the Datakeeper Hard Disk System:

A. Altair Datakeeper Controller in a self-contained cabinet.
B. 1 pair of interconnect cables for controller to computer connection
C. 1 cable assembly for controller to Pertec Hard Disk Drive connection.
D. 1 Pertec D3422 Hard Disk Drive with Fixed Platter.
E. 1 5440 Removable Top Loading Cartridge with Altair Datakeeper BASIC.
F. 1 set of Bootstrap Loader PROMs for system initialization.
G. Datakeeper Hard Disk System Documentation

The Datakeeper Controller acts as the interface between the Hard Disk Drive and the Altair 8800 computer. Up to four disk drives may be interfaced with one controller allowing a total storage capacity of approximately 40 MBytes. The controller unit includes a five-slot, bus-oriented motherboard, three plug-in interface boards and power supply. The plug-in Interface boards are:

A. Processor Board--contains a 8 x 300 bipolar processor, TTL ROM, 1K byte of buffer RAM for data transfers, and two bidirectional I/0 ports for communicating with the computer.

continued page 4

Increase Data Storage

continued

B. Disk Data Board--has serial to parallel and parallel to serial converters, FIF0 Registers, CRC generator/checker, and bit counters.

C. Disk Interface Board--includes the write data rate clock, I/0 ports, and line drivers for communicating with the Hard Disk Drive.

The Altair computer communicates to the Datakeeper Controller through two ports of an 88-4-PI0.

The 88-HDSK utilizes the Pertec D3422 Hard Disk Drive with 24 sectored format. It allows for approximately 5 MBytes of storage using the Fixed Platter and increases to 10 MBytes when the Removable Top Loading Cartridge is added.

To properly implement the 88-HDSK, the Altair 8800 series mainframe requires:

A. 48 K bytes of RAM memory (three each of either the Altair 88-16MCD or 88-16MCS)

B. 2 parallel ports (one each of Altair 88-4 PI0 and 88-PP)

C. 1 PROM Memory Card (Altair 88-PMC)

D. Serial I/0 Board for terminal communication (Altair 88-2SI0)

E. Terminal--CRT or Teletype ™

The Datekeeper Hard Disk System design emphasizes operational reliability and user convenience. Turnkey Operation assures fast and efficient power-up and program loading. Modular construction permits future expansion and easy component access. The Pertec D3000 series Hard Disk Drives have been proven in the field in a wide variety of applications and environments. This combination of optimum design and "state of the art" technology further extends the programming and data manipulation possibilities for the Altair 8800 series.

Controller Specifications

A. Power Requirements

70 watts typical, 120 watts maximum
Wired for 105-130V, 50/60 HZ
210-260 V, 50/60 Hz available on request

B. Physical Specifications

Size - Height 5.3 in (13.5 cm)
Width 16.85 in (40.5 cm)
Depth 17.3 in (41.5 CM)
Weight 20 lbs. (9.1 Kg)

Cabinet styling matches the Altair 8800b and 8800b Turnkey. A keyswitch on the front panel controls the power switch, and CPU Reset and Run mode.

Drive Specifications

A. Drive Type
Pertec D3422-E024-MWU

B. Data Storage Capacity
1 each Fixed Platter
4,988,928 Data Bytes
1 each 5440 type Removable Cartridge
4,988,928 Data Bytes
TOTAL 9,977,856 Data Bytes

C. Physical Format

Tracks per inch	200
Cylinders	406
Disk Surfaces	4
Tracks	1624
Sectors	24
Data Bytes/Sector	256

D. Serial Data Transfer Rate
2.5 MBits/second, determined by:
Spindle speed - 2400 RPM
Density - 2200 BPI

E. Access Time
1. Latency - Maximum 25.0 ms± 1%
- Typical 12.5 ms ± 1%
2. Seek Time - Minimum (Adjacent Track) 10 ms, Max.
Average (⅓ Full Stroke) 40 ms, Max.
Maximum (Full Stroke) 65 ms, Max.
3. Total maximum access time to read a Sector: 92 ms (25 ms Latency, 65 ms Seek, 2 ms Read)

F. Power Requirements
1100 watts Peak (start/stop cycle only)
400 watts typical
95-125V
or Must specify nominal voltage
190-250 V
48 to 52 Hz
or Must specify if nominal line
58 to 62 Hz frequency is 50 Hz

G. Physical Specifications

Height 8 ¾ inches	(22.2 cm)
Width 19 inches	(48.3 cm)
Depth 29 ¼ inches	TOTAL (74.3 cm)
Weight 130 lbs.	(59 Kg)

H. Reliability
Meantime between failure - MTBF - 4000 hrs.
Service life 5 years or 24,000 hrs.
Meantime to repair - 1 hr.

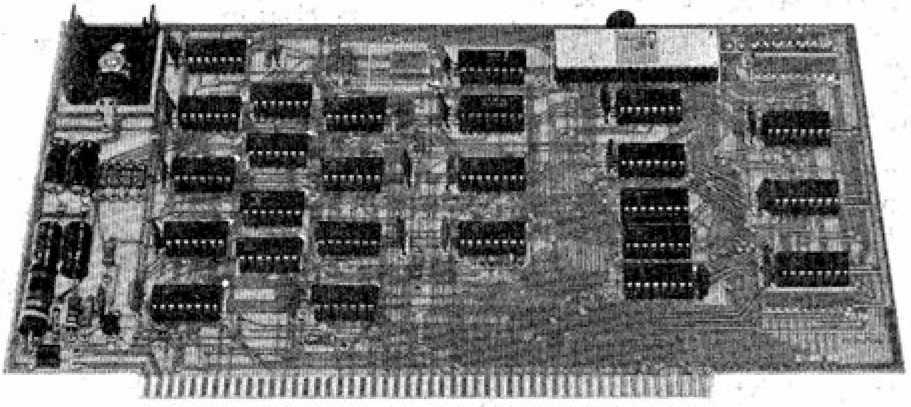
I. Recommended Preventive Maintenance
-Alignment check using CE pack recommended after moving or every 3 months/1000 hrs.
-1000 hr/3 months inspection and cleaning recommended
-2000 hr/6 months replace air filter, inspect for wear

NOTES

1. If using the Altair 8800 Turnkey, the 88-PMC and 88-2SI0 are not required.
2. The 88-HDSK System is not designed to run with the Altair Floppy Disk or Minidisk Systems.

Z-80 CPU Increases Processing Capabilities

By Susan Blumenthal MITS



Z-80 CPU

MITS introduces a Z-80-based Control Processing board to increase the processing capabilities of the Altair™ 8800 series microcomputers.

Designed as a replacement for the 8080 CPU, the Z-80 contains a powerful extended instruction set in addition to the standard 8080 instruction. It is compatible with any Altair 8800 series microcomputer with complete compatibility. (The Z-80 CPU Board is not compatible with the 88-PMC 8, 8K Prom Memory Card.) No hardware modifications are necessary to accomodate the board.

The internal hardware of the Z-80 microprocessor consists of:

--12 General purpose registors
-- 2 Accumulators
-- 2 Index registers
-- 2 Flag registers.

The Z-80 operates under a variety of software which includes:

Z-80 BASIC - a modified version of Altair BASIC (all current versions 4K, 8K, Extended and Disk)

DOS (Disk Operating System) Current available versions of DOS will operate with the Z-80.

The Z-80 CPU provides all 78 of the 8080 microprocessor instructions and an additional 80 instructions. Some of these added valuable instructions include:

--A block transfer group
--A block search group
--Individual bit manipulation group.

The Z-80 includes all 8080 addressing modes plus indexed and bit modes. With the increased capabilities of a more comprehensive instruction set and addressing modes, the amount of memory required for machine language programs decreases.

The Z-80 CPU is available for $295 fully assembled and $275 in Kit form. It's also available in a fully assembled Altair microcomputer.

Specifications

Power Requirements:
5 vdc at 500 MA
+12vdc at 40 MA

Instruction Cycle:
2 microseconds (minimum)

Block Transfer rate:
95,000 bytes per second including increment and decrement overhead

Dimensions:
10" x 5"

Altair™ 88-16MCD Compatible with 8800A

By Robert Lopez
MITS

Since the introduction of the Altair 88-MCD, there has been some confusion among many of our customers about whether or not it's compatible with the 8800A and other Altair computer plug-in boards. With a simple power supply modification to the 8800A, the 16MCD becomes compatible with both the 8800A and all Altair 8800 series plug-in boards.

The Power supply lines of the Altair Bus System are unregulated supply lines, i.e. the voltage present can vary depending upon input A.C. line voltage and frequency and the load power demand. Regulation for each supply line is done individually on each printed circuit board. An Altair 8800A should have bus lines #1 and #51 not less than +7v. (+7.5 NOMINAL), bus line #2 not less than +14v (+15 Nominal), and Bus Line #52 not less than -14v (-15 Nominal).

Changes in technology lead to printed circuit boards which loaded down the +7.5v line to less than +7v. voltages less than +7v cannot be regulated to a clean +5v. The power supply modification printed in the September 1975 CN allowed increased loading.

Several changes have since been made in the Altair 8800B which weren't incorporated in the 8800A. Bus lines #1 and #51 in the 8800B should be not less than +7v (+8 Nominal), line #2 should be not less than +17v (+18 Nominal), and line #52 should be not less than -17v (-18 Nominal).

The 16MCD was designed to run in the Altair 8800B and the Altair 8800B Turnkey, which has the same bus specifications as the 8800B. The requirement of the 16MCD which limits its operation to the 8800B is the +15V necessary for the Mostek 4096 Rams. A 7815 regulator is used to regulated the +15v. For complete regulation, a 7815 requires a minimum of +17v.

So to use the 16MCD in an 8800A, it's necessary to convert to 8800A power supply to 8800B specifications. In order to accomplish this conversion, the 8800A power transformer must be replaced with MITS part #102621. Owners of Altair 8800A's who purchase a 16MCD will receive the new power transformer at no cost.

Use the Interrupt Vector in Single-Level Interrupt Systems

By Steve Gride
MITS Engineering Dept.

A number of new Altair™ computer users have said that they don't understand how the interrupt system is used in the Altair 8800 series. This has led to a misunderstanding concerning single-level interrupts; how are they generated, and what happens during their acknowledgement? Users also ask, "How can I change a single-level interrupt to jump to a location other than 070(8)?" This article will attempt to address these questions.

The Altair 8800 microcomputers use an eight-level vectored interrupt system. This system is based on the interrupt-response vector built into the 8080 CPU chip. It has the following effect: When an interrupt occurs, the device generating the interrupt creates a vector address, which the CPU uses as a restart address during the interrupt-acknowledge cycle. This results in a call to one of the low-memory restart areas

In the Altair system, the restart vector address is usually created by the 88-VI board (vectored interrupt board). This board allows the prioritizing of up to eight levels of interrupts in the restart area. When this board is absent, however, it is the responsibility of the interrupting device to generate the interrupt address. This is usually not done, resulting in a "floating" input to the CPU during interrupt-acknowledge time. These "floating" inputs look like a vector-7 to the CPU, which acknowledges with a restart to 070(8). So most single-level interrupt systems automatically generate a restart to level 7.

(Note: All MITS standard software recognizes single-level interrupts at level 7, therefore, any hardware modifications will require a corresponding change in software.)

The way to jump to a different location in the interrupt vector is illustrated schematically in Figure 1. During the interrupt-acknowledge cycle, the CPU generates the status signals M1 and SINTA. When these two signals occur concurrently, the restart vector is gated onto the data bus.

This circuit may be built up "piggy-back" on the I/0 or other board which will use it, or it may be built on a separate breadboard and plugged into the bus.

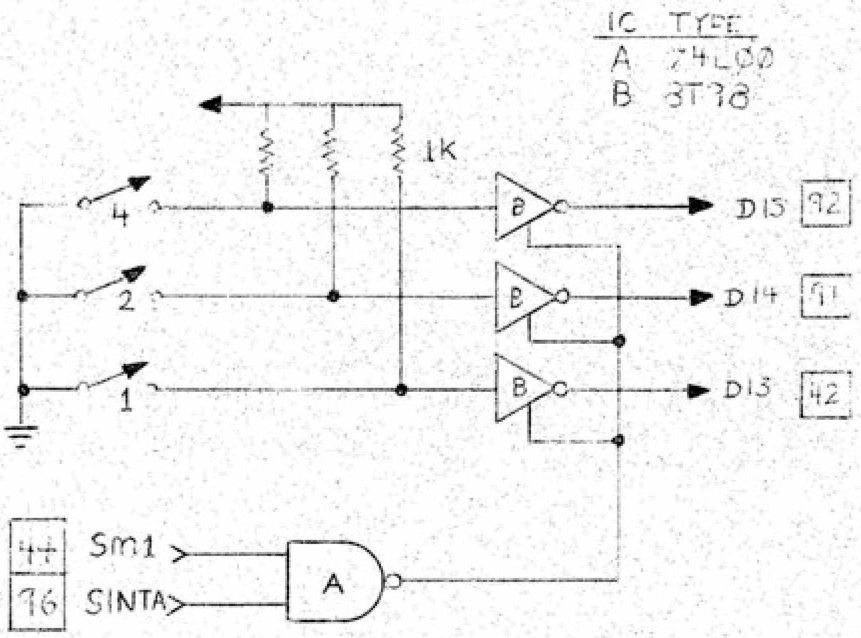


FIG.1

FLOPPY DISK:

Does Your Drive Buzz During a Mount?

By Thomas Durston

If your Floppy Disk Drive makes a loud buzzing noise during Mounting of a diskette, the problem can be eliminated by adjusting a resistor on Floppy Disk Controller Board #2.

The buzzing is caused by the Drive's head trying to step in farther than it should. This occurs during a Mount if an error is detected when reading the track number. The track number error causes the track counter (software) to think it is farther out than it should be, stepping the head in and against the stop at the end of the stepping shaft. The result is the buzzing noise.

This buzzing noise occurs only on certain diskettes if the Head Load time constant is less than 45 ms. It is a function of the Mount routine which reads every eight sectors.

To correct the problem, adjust R8 on Controller Board #2 to yield a 50ms ± 4ms pulse at I.C. B1 pin 13 (TP-6) during a Mount command. The value of R8 will be approximately 16K, and a 20K or 50K trimpot may be used for adjustment in place of R8.

Program Allows Disk Timesharing to Read Non-Timesharing Diskettes

By: Gale Schonfeld
MITS

Many of you are now sharing our excitement over the new Altair Timesharing BASIC. Those of you who have the disk version may be perturbed about a problem with loading 4.0 or 4.1 Disk BASIC program files under Timesharing. However, with only a few minutes of your time and the computer's, the problem can be solved.

In the disk version of Timesharing BASIC, an optional password may be specified during SAVEing of a program. In regular Disk BASIC, the password facility is not provided. Therefore, the problem may occur when a LOAD or RUN command is issued in Timesharing for a program on a regular BASIC disk. Timesharing may respond to the command with PASSWORD FOR FILE "XXX. . ."?, and the user will not know with what password to answer.

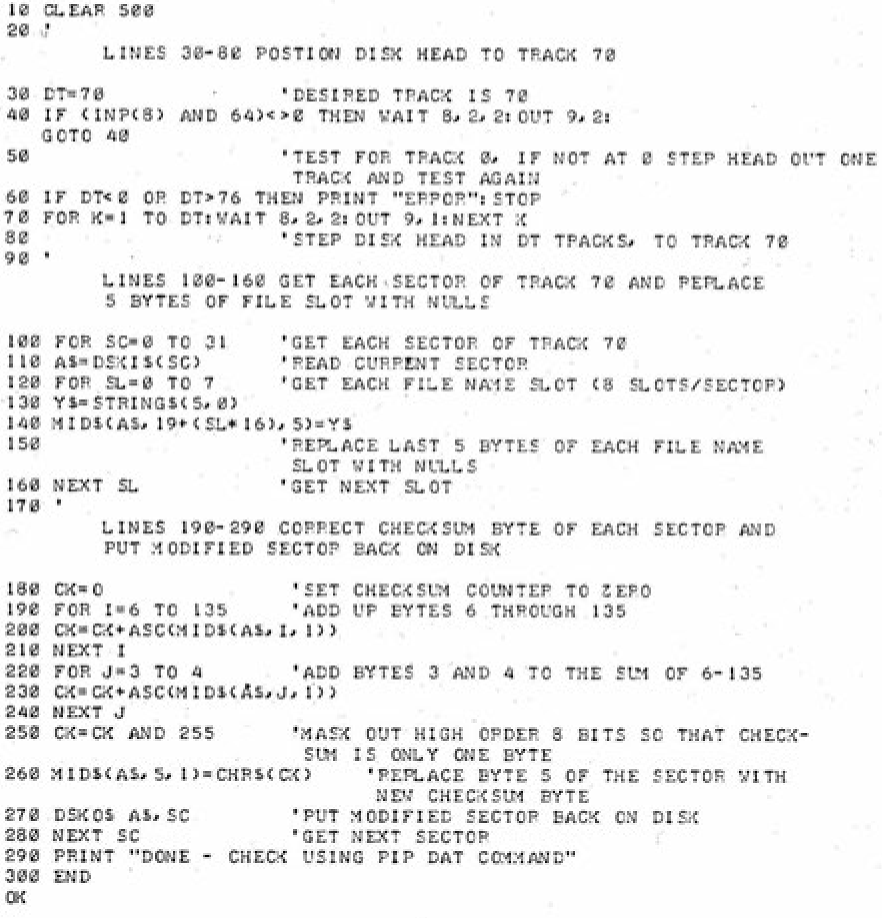
This problem is due to the format of the directory track on the diskettes. To review, each sector of the directory track is comprised of eight file name slots. Each slot contains 16 bytes--eight bytes for the file name, one byte for the track pointer, one byte for the sector pointer, one byte indicating whether the file is random or sequential and in regular Disk BASIC, and five unused bytes normally set to nulls. In Timesharing Disk BASIC, these extra five bytes are used for passwords. Occasionally, "garbage" can get into these extra bytes on the normal BASIC diskettes. When Timesharing tries to access these files, it "sees" a password which the user is unaware. If all five bytes are null, Timesharing realizes that a password is not required.

The following program, when executed in 4.0 or 4.1 Disk BASIC, will correct the directory track of a 4.0 or 4.1 diskette. The functions of PASSCHEK are to set the last five bytes of the file name slots to nulls and recalculate the checksum of the sector so it can be read by Timesharing. The program PASSCHEK contains detailed comments regarding its execution. The remark statements can be left out when entering the program in order to utilize a minimum amount of memory.

To use PASSCHEK, enter it into memory using 4.0 or 4.1 Disk BASIC. (It will **not** run in Timesharing.) Place the diskette you need to correct in **Disk Drive** and **MOUNT** it. Now type RUN. PASSCHEK will run for approximately two to three minutes, printing "DONE - CHECK USING PIP DAT COMMAND" when it's finished. If you wish to check using PI0, the format of the floppy disk is described in Appendix H of the Altair BASIC Manual.

For those of you who have old 3.4 Disk BASIC program files that you want to run under Timesharing Disk BASIC, a few extra steps are needed before running PASSCHEK on the 3.4 diskette. Since Timesharing will read only 4.0 or 4.1 formatted files, you must convert your 3.4 files to the 4.0 format. This is easily done by first LOADing and then re-SAVEing all 3.4 program files in ASCII (e.g. SAVE "XXX", O, A), using 3.4 Disk BASIC, and then using the 4.0 PIP CNV command on the diskette to convert the files to the 4.0/4.1 format. After this, you can run PASSCHEK.

Program



```
10 CLEAR 500
20 '
      LINES 30-80 POSTION DISK HEAD TO TRACK 70

30 DT=70                  'DESIRED TRACK IS 70
40 IF (INP(8) AND 64)<>0 THEN WAIT 8,2,2:OUT 9,2:
   GOTO 40
50                        'TEST FOR TRACK 0, IF NOT AT 0 STEP HEAD OUT ONE
                           TRACK AND TEST AGAIN
60 IF DT<0 OR DT>76 THEN PRINT "ERROR":STOP
70 FOR K=1 TO DT:WAIT 8,2,2:OUT 9,1:NEXT K
80                        'STEP DISK HEAD IN DT TRACKS, TO TRACK 70
90 '
      LINES 100-160 GET EACH SECTOR OF TRACK 70 AND REPLACE
      5 BYTES OF FILE SLOT WITH NULLS

100 FOR SC=0 TO 31        'GET EACH SECTOR OF TRACK 70
110 A$=DSKI$(SC)          'READ CURRENT SECTOR
120 FOR SL=0 TO 7         'GET EACH FILE NAME SLOT (8 SLOTS/SECTOR)
130 Y$=STRING$(5,0)
140 MID$(A$,19+(SL*16),5)=Y$
150                       'REPLACE LAST 5 BYTES OF EACH FILE NAME
                           SLOT WITH NULLS
160 NEXT SL               'GET NEXT SLOT
170 '
      LINES 190-290 CORRECT CHECKSUM BYTE OF EACH SECTOR AND
      PUT MODIFIED SECTOR BACK ON DISK

180 CK=0                   'SET CHECKSUM COUNTER TO ZERO
190 FOR I=6 TO 135         'ADD UP BYTES 6 THROUGH 135
200 CK=CK+ASC(MID$(A$,I,1))
210 NEXT I
220 FOR J=3 TO 4           'ADD BYTES 3 AND 4 TO THE SUM OF 6-135
230 CK=CK+ASC(MID$(A$,J,1))
240 NEXT J
250 CK=CK AND 255          'MASK OUT HIGH ORDER 8 BITS SO THAT CHECK-
                            SUM IS ONLY ONE BYTE
260 MID$(A$,5,1)=CHR$(CK)     'REPLACE BYTE 5 OF THE SECTOR WITH
                               NEW CHECKSUM BYTE
270 DSKO$ A$,SC            'PUT MODIFIED SECTOR BACK ON DISK
280 NEXT SC                'GET NEXT SECTOR
290 PRINT "DONE - CHECK USING PIP DAT COMMAND"
300 END
OK
```

PRACTICAL PROGRAMMING

By Gary Runyon
MITS

This new column will discuss some of the things we're learning in the MITS Computing Services Department about how to program in Altair™ Basic. Although the articles will be aimed at the beginning programmer, even the most advanced programmer should find the column useful and interesting. Complete listings of programming aids we've developed (cross, reference list program, variable name replacement programs, etc.) will be included when necessary. But,there will be nothing about programming in machine code, except possibly a few USR routines.

Each month's column will become a chapter of the Computing Services Standard Practices Manual, which will be used by programmers here at MITS.

LINE COUNTING

One of the first problems the beginning programmer tangles with is line counting, i.e. how to tell that you're at the bottom of the page when printing a report so that you know when to space to the top of the next page. After much work, the beginner's report program can decide when to space to the next page, but for some reason it spaces too far or not far enough. By adding a patch, everything works fine, except for an extra space between the first and second pages. A hokey patch is added and all works well until the program needs its first modification.

The solution? Adopt a convention, understand it, and stick to it. Here at MITS the variable name L9 is reserved for line counting in all programs.

L9 points to the next line to be printed. It is initialized to one plus the number of lines printed at the exit of the page header routine. L9 is incremented by one for every line printed thereafter. For L9=L9T066: LPRINT:NEXT is the routine for getting from the bottom of a page to the top of the next page.

The 66 in the routine comes from six lines per inch, 11 inches per page. If you're printing special forms (checks, invoices, W2, etc.), or have a printer that doesn't print six lines per inch, replace the 66 with the appropriate lines per page. If you need to print a really oddball form, such as three ¼" checks, the trick is to throw in an extra line every other check. The following will handle three ¼" forms on a standard printer:

FORL9=L9T019:LPRINT:NEXT:IF A THEN LPRINT:A=O ELSE A=1.

Test for bottom of the page when you have something to print. Testing for bottom of page after printing can result in an occasional sloppy header with no data at end of report.

The usual test for bottom of page is: IF L9>XX THEN GOSUB [space up and print heading]. This results in XX lines printed per page with 66-XX spaces between the bottom and top of each page.

The test for bottom of page before printing n lines when n is greater than one is: IF L9>XX+1-n THEN GOSUB[]. For example, if a report has three lines per item, five lines of totals, and is not to go below line 64, the test before printing each item would be: IF L9>62THEN GOSUB[]; the test before printing the totals would be: IF L9>60 THEN GOSUB[].

In those cases where n is not a fixed constant, the test for bottom of page will appear in the form IF L9+n XX+1 THEN GOSUB [] (see example program). The concept is, "Will the hokey patch work well until the program allowed value (XX+1) after these n lines are printed?"

The example program PROGLIST demonstrates how to line count. The program reads a program saved in ASCII and prints a listing with the program name, the current date, and page YY of pages ZZ at the top of each page. In order to provide at least three blank lines between each page, the program does not print past line 63.

The two clear statements in line 70 grab off as much string space as is available. This holds to a minimum the time lost to string space garbage collection. Line 100 allows you to input a file name ending with a comma and number to specify files on other than disk drive zero. Line 120 checks for the null string that is at the beginning of every ASCII file. Lines 140-190 read through the file, duplicating what will happen to L9 and the page count when the file is listed. Line 220 prints the heading at the top of the first page.

The FORL9=L9T0132 in line 250 spaces the printer to the top of page twice, leaving the listing where it can be easily torn off.

Lines 290 and 300 show the standard print out for one-line:

1. Test for bottom of page when ready to print
2. Print
3. Increment the line counter

Lines 320-350 determine how many lines will actually print when a program line with the line feeds prints. Each part of the line is loaded into the array L$ so that it can be printed separately. This avoids problems caused by line printers reacting differently to the line feed carriage return embedded in program lines.

Lines 360-370 show the standard print out for more than one-line :

1. Test for bottom of page when ready to print
2. Print
3. Increment the line counter

Line 390 is the standard to-to-top-of-page routine.

Line 420 sets L9 to one plus the number of lines printed in the header (one information line and one blank line) before exiting the heading routine.

To summarize, L9 is the next line on the page to be printed. L9 is initialized to one plus the number of header lines at the exit from the header routine. L9 is incremented by one after each line printed. The test for bottom of page is executed when the program is ready to print. The space to top of page routine is:

FORL9=L9T066:LPRINT:NEXT

Letter Writing Program Solves Photographers Mailing Problems

By: Lee Wilkinson
2308 New Walland Hwy.
Maryville, Tennessee 37801

Wilkinson currently runs his own photography studio. For the past 15 years he has been an avid ham radio hobbyist, but had no previous computer experience before purchasing an Altair 8800 to use in his business. In addition to the mainframe, his system now consists of 24K memory, a Teletype, ADM-3, 8-PMC, 88-ACR, 88-SIOA, 88-SIOB and wire wrap board for morse code. Wilkinson has also recently published three other software articles in KILOBAUD.

One of the most beneficial and frequently used programs in my collection of software is a letter writing program. When used in conjuction with our regular direct mail promotion program, it has been an invaluable advertising aid.

Originally, we were sending about 200 letters each month to parents of new babies, one year olds, and two year olds. The parent's names were compiled from the local newspaper, and the letters were prepared on our printing press. Records of appointments made show about a three percent rate of response to this promotion. This is about the national average for direct mail advertising.

We used the Altair computer for printing mailing labels for our children's promotion campaign and for writing personalized letters. Our first mailing brought a 17% return. Needless to say, we continued with this personalized type of mailing, and are still enjoying the same increased response.

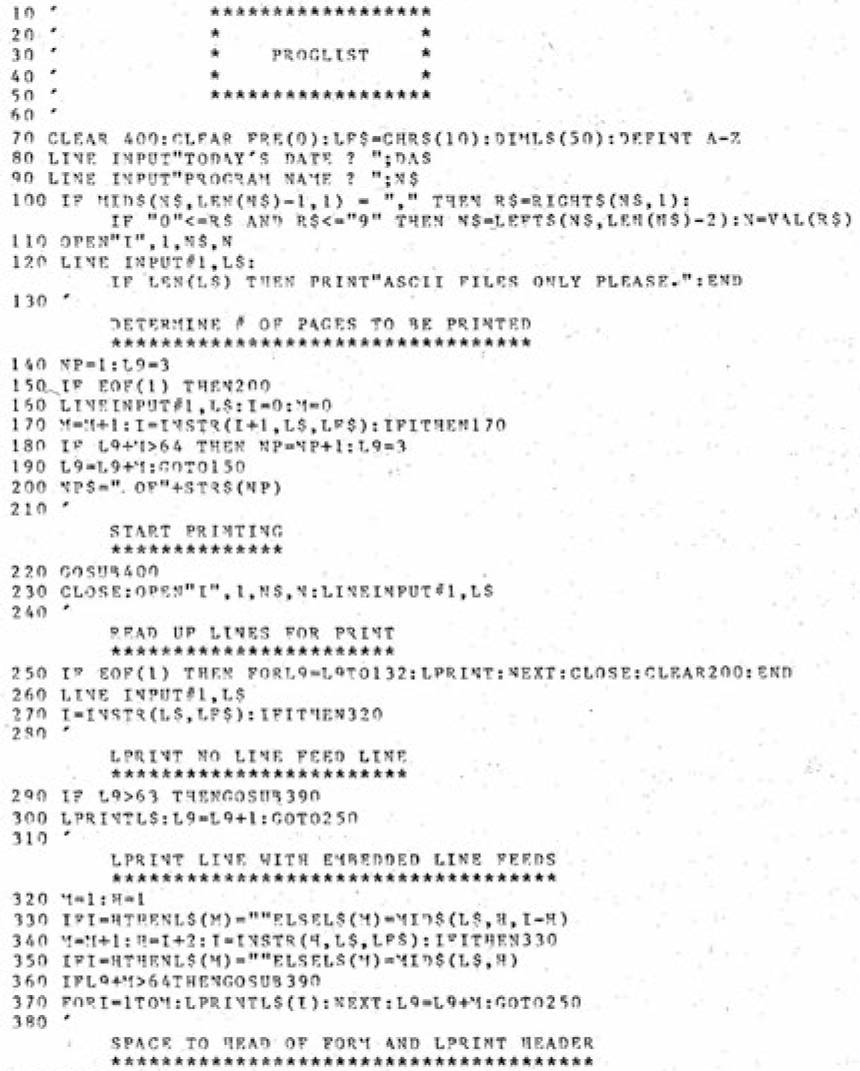
However, there were several problems in preparing the mailings. First, the type style of the Teletype wasn't appropriate, and the standard roll paper wasn't a very high quality. Remembering an old cliche, "lemons can be turned into lemonade", an idea came to mind. Why not get a rubber stamp made that said "STUDI-O-GRAM" and imprint each letter so that it would look like a telegram? By using this stamp and placing the letter in a window envelope we created a personalized package that the recipient felt compelled to open.

We've used the "STUDI-O-GRAM" for the local births for about a year now and still enjoy excellent success. We've expanded the "STUDI-O-GRAM" to include about every conceivable list we've ever stored on cassette. This includes doctors, realtors, past patrons, businessmen, little league coaches, and churches, just to mention a few.

For those interested in adapting the program for their own use, a sample listing is enclosed. There's nothing really exotic about the program, and users should have no trouble following it. The body of the letter is inserted from lines 200-279. Lines 500-580 print the title (Mr., Mrs., Rev., etc.) and the last name. Mailing labels can be generated by the subroutine 600-690. The label format can be altered by changing lines 620 and 650-670. The inclusion of the subroutine at lines 700-745 allows a "town code" to be typed for the local area post offices and saves much time and a great deal of memory when typing local lists. However, any city, state, and zip may be typed on any data line (1000 and up), and the program will recognize it. The subroutine at 10000 switches from CRT (port 000// and 00/) to TTY (port 024 and 025 Q) and back to the CRT in my MITS 8K, Ver. 4.0 BASIC.

One of these days I hope to replace the ACR with a disk and a faster printer and then really increase sales.

Practical Programming



```
10 '            ******************
20 '            *                *
30 '            *    PROGLIST    *
40 '            *                *
50 '            ******************
60 '
70 CLEAR 400:CLEAR FRE(0):LF$=CHR$(10):DIML$(50):DEFINT A-Z
80 LINE INPUT"TODAY'S DATE ? ";DA$
90 LINE INPUT"PROGRAM NAME ? ";N$
100 IF MID$(N$,LEN(N$)-1,1) = "," THEN R$=RIGHT$(N$,1):
        IF "0"<=R$ AND R$<="9" THEN N$=LEFT$(N$,LEN(N$)-2):N=VAL(R$)
110 OPEN"I",1,N$,N
120 LINE INPUT#1,L$:
        IF LEN(L$) THEN PRINT"ASCII FILES ONLY PLEASE.":END
130 '
        DETERMINE # OF PAGES TO BE PRINTED
        **********************************
140 NP=1:L9=3
150 IF EOF(1) THEN200
160 LINEINPUT#1,L$:I=0:M=0
170 M=M+1:I=INSTR(I+1,L$,LF$):IFITHEN170
180 IF L9+M>64 THEN NP=NP+1:L9=3
190 L9=L9+M:GOTO150
200 NP$=". OF"+STR$(NP)
210 '
        START PRINTING
        **************
220 GOSUB400
230 CLOSE:OPEN"I",1,N$,N:LINEINPUT#1,L$
240 '
        READ UP LINES FOR PRINT
        ***********************
250 IF EOF(1) THEN FORL9=L9TO132:LPRINT:NEXT:CLOSE:CLEAR200:END
260 LINE INPUT#1,L$
270 I=INSTR(L$,LF$):IFITHEN320
280 '
        LPRINT NO LINE FEED LINE
        ************************
290 IF L9>63 THENGOSUB390
300 LPRINTL$:L9=L9+1:GOTO250
310 '
        LPRINT LINE WITH EMBEDDED LINE FEEDS
        ************************************
320 M=1:H=1
330 IFI=HTHENL$(M)=""ELSEL$(M)=MID$(L$,H,I-H)
340 M=M+1:H=I+2:I=INSTR(H,L$,LF$):IFITHEN330
350 IFI=HTHENL$(M)=""ELSEL$(M)=MID$(L$,H)
360 IFL9+M>64THENGOSUB390
370 FORI=1TOM:LPRINTL$(I):NEXT:L9=L9+M:GOTO250
380 '
        SPACE TO HEAD OF FORM AND LPRINT HEADER
        ***************************************
390 FORL9=L9TO66:LPRINT:NEXT
400 PG=PG+1:PG$="PAGE"+STR$(PG)+NP$
410 LPRINTN$;" LISTED ";DA$;TAB(75-LEN(PG$));PG$
420 LPRINT:L9=3:RETURN
```

continued page 25

By Doug Jones
2271 North Mill
North East, PA 16428

Trace Program Simplifies Debugging for Altair™ 680b

The software interrupt instruction (SWI hex 3F) in the Altair™ 680b computer permits a rather unique method of setting program breakpoints for debugging. The PROM MONITOR manual contains a rather good discussion of this routine in Section V, which also includes a very short program to print out the contents of the processor's registers each time a program breakpoint occurs.

There are two methods of handling a SWI by the MONITOR. (1) If you haven't set a bit 7 of BRKADR (00F2), anytime a SWI is executed in the assembled code, a return is made to the MONITOR. Using the (N)ext command, all registers may be inspected and, if you wish, modified. Continuation of the program is made by the (P)roceed command. Everything is returned back from the stack, and processing continues. (2) If bit 7 of BRKADR is set, upon execution of the SWI, control is vectored to address 0000 where a user routine, such as the print register routine, must be waiting.

Consider the program shown in the sample run. Assume that this program is giving you trouble, or perhaps you would like to watch the values loaded into the A register. To use the SWI, the program would have to be opened up just before the BEQ instruction, a SWI inserted, and then one of the two methods described above used to watch the A register contents.

Once the program error has been corrected, it must either be reassembled to remove the SWIs that you have used, or they must be NOPed out.

DEBUG TRACE will co-exist in memory with your program. It will wrap itself around your program so to speak and allow you to control its running. It will replace every instruction encountered in your program with a SWI, give you a dump of register content if you want it, replace your original instruction, and continue processing through that instruction.

In abbreviated format, here are particulars of the program:

Length 1K.

Starting address (j) 4000.

Commands:

D Dump registers while in the command mode.

M Return to MONITOR. After (M) and (N)ing any part of memory, a (P)roceed will return control to DEBUG.

J Jump to program. You will be queried about the starting address. Program execution from that point on the will be under control of DEBUG.

A/B/C/X allows you to set the indicated register.

I Set instruction breakpoint. Zero (0000) for none.

O Set operand breakpoint. Zero for none.

T Set trace on and trace off addresses. To kill trace, set to FFFF and 0000 respectively.

(ESC) Escape can be used any time during controlled program run or register dump for return to command mode.

****CAUTION****

Any address set or register set MUST be valid hex characters or you will return to MONITOR. A (J)ump command must be executed back to DEBUG to return operation to normal.

PRINTOUTS

Type of dump:

D called by dump command (extended);

T trace dump;

B dump due to I or 0 breakpoint (extended)

X illegal operation attempted (extended).

I The instruction you are about to process.

Operand will show none, one, or two bytes, depending on the instruction.

Stack will show where the user's program placed it.

Program counter will normally show the address of the instruction you are **going into**. It will show the destination address if a jump or conditional branch is executed.

Illegal operations are RTI ($3B), WAI ($3E). RTS ($39) will also be an illegal operation if the number of returns exceeds the number of subroutine calls.

Any return to DEBUG command mode will normalize and cancel all subroutine linkages. User program must be restarted with a (J) XXXX.

Legal calls to MONITOR subroutines OUTCH, INCH, OUTS, and OUT2H are allowed, executed, and printed (with echo), but are not traced.

As shown in Table 2, wherever the user program defines the stack, approximately 11 bytes will be utilized by DEBUG. All pointers will be returned to where you left them.

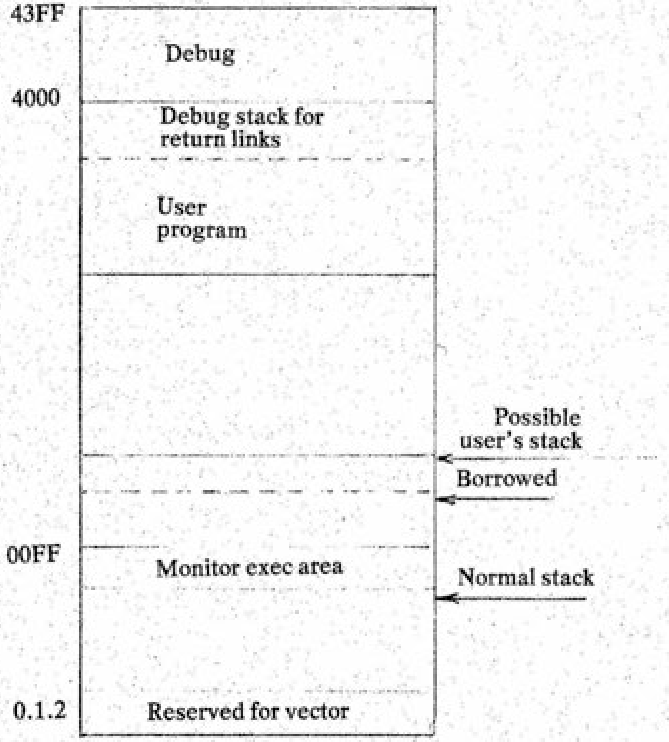
DEBUG is volatile. In order to keep the program length to 2 K or under, many checks and cross-checks had to be eliminated. One, for example, was a range check that would stop all activity equal to or above DEBUG's stack area. Some bells and whistles also had to be excluded; for example, the ability to proceed from a breakpoint or an (ESC)ape.

The user's program will run with no trace or breakpoints established and is interruptable by (ESC). You will, however, notice a 100-fold increase or greater in user program run time.

Table 1 Printout Format.

```
          Trace Only            (extended)
TII0000SSSSCCBBAAXXXXPPPPTTTT
TT TT II II 00 00
Operand breakpoint
Instruction breakpoint
Trace off
Trace on
Program counter
X-register
A-register
B-register
Condition code register
Stack pointer
Operand
Instruction
Type of dump
```

Table 2 Memory Map.



OBJECT CODE

```
S00B000044454255472020202D
S10400F3FF09
S11E4000BF439D07B743A6CE43788D57BE439DBF43A4CE3FFFFF43967F439A6A
S11E401BFE439BB643A0A7007F439BCE43728D38CE4239DF01867E9700439727
S11E4036F2BD4307CE43B5E600270EF1439F270508080820F2EE016E00CE43FA
S11E40518EBD0F20B6B7439F8611B7434DBD431A20F1E6002706BDFF81082036
S11E406CF639D7F3D7F23F7E40078D18FF43AD8D13FF43AF20DF8D0CFF43B1BF
S11E408720F78D05FF43B320F0CE4381BD40637E42F8BD4313F743A620DFBD13
S11E40A24313F743A720F6BD4313F743A820EE8DE3FF43A920E78DD6A600B78D
S11E40BD43A07E4256BD42E1F7412C7F412BFE412BBD4293C6022019BD42EED2
S11E40D8B643A0FE43A1FF412BBD4293817E271C81BD2721C603FE43AB5D2750
S11E40F304085A20F9FF42CD7E42B4FE43ABBD4147FE412BFF43AB5F20E18C39
S11E410EFF81270F8CFF6D270A8CFF0027058CFF8226DDBD42FFB643A8F64314
S11E4129A7BD0000B743A8F743A7BD4302BD4274FE43AB080808A600B743A0D7
S11E41447E4256080808FF42CDBF4398BE4396B642CE36B642CD36BF4396BEA2
S11E415F43987C439A39B643A0818D2715818C270B818E270781CE27037E403E
S11E417AC2BD42EE7E40ECFE43AB8DC27E4212BD42DCF643A0C1392716C13BD9
S11E4195270DC13E2709C13F2705C6017E40EE86587E40567D439A27F67A43E3
S11E41B09ABF4398BE439632B742CD32B742CEFE42CDFF43ABBF4396BE439809
S11E41CB7E40F8BD42E1FE43A9FF412B0C5FB643A18D17B643A081AD270781CB
S11E41E66E27097E40CEFE43ABBD41487E4104BB412CF9412BB7412CF7412B87
S11E420139BB412C2405FB412B20EFFB412B5A20E9BD42E1FE43AB0808FF41B8
S11E421C2BB643A0B74227B643A606000220BE0C5FB643A12A038DCE8C8DBEB6
S11E423720B9FE439BB643A0A7008607CE43A633E700084A26F9BF43A48D1A57
S11E4252FE43AB09FF43ABB643A084F0444444CE43D3084A2AFCEE006E00BD1D
S11E426DFF24240ABDFF04C11B26037E400739BC43B1272EB643AEF643AD800C
S11E428801C200B0412CF2412B2506BC43B3271739B643B0F643AFB0412CF2E5
S11E42A3412B25F18654B7439F7E431A86427E4056BE43A48607CE43ACE6007B
S11E42BE37094A26F9FE43ABFF412BBD427CCE0000A600B743A0863FA700FFED
S11E42D9439B3B4FB743A33986018DF8FE43ABE601F743A13986028DF1E602A7
S11E42F4F743A2398D05BDFF62200386038C86FF97F3398DF6BDFF00F7439F4E
S11E430F8D5220EF8DEABDFF5320E8CE438ABD4063F6439F8D3BB643A08D430F
S11E432AB643A32714B643A1BDFF6DB643A34A270AB643A2BDFF6D20048D242A
S11E43458D228D20CE43A4C609270AA600378D1833085A20F48609B7434D390E
S11E4360BDFF81BDFF827E426C8DF820F6BDFF6D20F10D0AFF4020000D0AFF36
S11E437B4445425547002041444452203F20000D0AFF002A4552524F522A000E
S11E43963FFF00000000000000003F000000000000000000000000FFFF00008D
S11E43B1000000004D406E4340994240A14140A95840B15440764F4089494095
S11E43CC824A40B8444059004189418942124189418941894 1CE40D5416540E1
S11043E7C241CE40D5416540C241CE40D513
S10400F30305
S9030000FC

TOTAL ERRORS 00000

ENTER PASS
```

continued on page 12

Trace Program Simplifies Debugging

Source Listing

```
  FFEF$$
  NAM DEBUG
 *
 *SOURCE 1.2.0
 *

 *JUNE 1977 DLJ
 *
  OPT NOG
 *
   ORG $00F3
   FCB $FF
 *

 * INSTRUCTIONS:

 *
 * D = (D)UMP REGISTERS
 * M = (M)ONITOR RETURN
 * J = (J)UMP
 * A/B/C/X/I/O/T =
 * SET REGISTERS/BREAKPOINTS/TRACE
 *
 BADDR EQU $FF62
 BRKADR EQU $00F2
 BYTE EQU $FF53
 ECHO EQU $00F3
 INCH EQU $FF00
 OUT2H EQU $FF6D
 OUTCH EQU $FF81
 OUTS EQU $FF82
 POLCAT EQU $FF24
 *
   ORG $4000
 START STS STKSV SAVE IT
   TPA
   STA A CCREG
 *

 DEBUG LDX #MES1  SEND 'DEBUG'
   BSR MSG
 *
 EXEC LDS STKSV
  STS STKHI
  LDX #START-1
  STX MYSTK
  CLR SUBCNT
  LDX SWIADR
  LDA A INST
  STA A X
  CLR SWIADR
   LDX #PRMPT  POP OUT A @
   BSR MSG
   LDX #RUNVCT  SET RUN VECTOR
   STX 1  STORE AT SWI
   LDA A #$7E  LOAD A JMP
   STA A 0 STORE IT AT SWI
   COM A  SET HIGH BIT
   STA A BRKADR  AT BREAK ADDR
   JSR IN  GET A CHRCTR
  LDX #JMPTB JUMP TABLE
 EXEC1 LDA B X GET LTR
  BEQ BUM DONE=
  CMP B WHAT MATCH?
  BEQ JMPCMD
  INX TO NEXT LTR
  INX
  INX
  BRA EXEC1
 JMPCMD LDX 1,X TAKE IT
  JMP X
 *
 BUM LDX #EM BUMMER
  BSR MSG
 BUM1 BRA EXEC  BACK YOU GO
 *
 DMP1 STAA WHAT
 DMP LDA A #$11
   STA A HMNY  SET FOR BIG DMP
 DMP3  JSR PRNTRG
 DMP2 BRA BUM1 EXEC
```

```
 *
 MSG LDA B 0,X
  BEQ MSG1
  JSR OUTCH
  INX
  BRA MSG
 MSG1 RTS
 *
 MONIT  STA B ECHO
   STA B BRKADR
   SWI  BACK TO MONITOR
   JMP DEBUG  READY FOR (P)ROCEED
 *
 TSET BSR ADPRM TRACE SET  GET ADDR
   STX TON  TRACE ON ADR
   BSR ADPRM
   STX TOFF  TRACE OFF ADR
 TS1 BRA DMP2 EXEC
 *
 BI BSR ADPRM  INST BREAKPT
   STX BIADR
   BRA TS1  EXEC


 *
 BO BSR ADPRM   OPRND BKPT
   STX BOADR
  BRA TS1
 *
 ADPRM LDX #MES2
 ADPRM1 JSR MSG
 ADPRM2 JMP BAD & RTRN
 *
 STC JSR BY CNDTN REG
   STA B CCREG
 STC1  BRA TS1
 *
 STB JSR BY BREG
   STA B BREG
   BRA STC1
 *

  STA JSR BY AREG
   STA B AREG
   BRA STC1
 *
 STX BSR ADPRM2 XREG
   STX XREG
 *
 ST5 BRA STC1 EXEC
 *
 JMPXX BSR ADPRM GET ADR
  LDA A X GET INST
  STA A INST
  JMP RUN2
 *
 DIR JSR POP1 LOAD OPRND
  STA B CKADR+1
  CLR CKADR
  LDX CKADR
 DIR3
       JSR EXMOP
 DIR2 LDA B #2 NEXT SWI
  BRA EXT1A
 *
 EXT JSR POP2 LOAD OPRND
   LDA A INST
   LDX INST+1 GET ADR
  STX CKADR
  JSR EXMOP
   CMP A #$7E  JMP?
   BEQ EXT2
  CMP A #$BD JSR?
   BEQ EXT3
 EXT1  LDA B #3 NEXT SWI
 EXT1A LDX PCREG
 EXT1B TST B
  BEQ EXT1C
  INX
  DEC B
  BRA EXT1B
 EXT1C STX HERE
  JMP REPAK
 EXT2 B LDX PCREG
  JSR SAVLK3
 EXT2  LDX CKADR
```

```
  STX PCREG  SWAP
  CLR B NEXT SWI
  BRA EXT1A
 EXT3 CPX #OUTCH
  BEQ DOIT
  CPX #OUT2H
  BEQ DOIT
  CPX #INCH
  BEQ DOIT
  CPX #OUTS
  BNE EXT2B
 DOIT  JSR EON
  LDA A AREG
  LDA B BREG
 **********
  FCB $BD JSR
 CKADR FCB 0,0

 **********
  STA A AREG
  STA B BREG
  JSR EOF
 JSR CKHUM3 ESCAPE?
 LDX PCREG NO
 INX PAST JSR
 INX
 INX
 LDA A X
 STA A INST
 JMP RUN2
*
SAVLK3 INX SAVE LINK
SAVLK2 INX
SAVLK INX
 STX HERE
 STS STKTMP
 LDS MYSTK
 LDA A HERE+1
 PSH A
 LDA A HERE
 PSH A
 STS MYSTK
 LDS STKTMP
 INC SUBCNT
 RTS
*
IMM LDA A INST
  CMP A #$8D BSR?
  BEQ BSIMM
  CMP A #$8C CPX?
  BEQ IMM3
  CMP A #$8E  LDS?
  BEQ IMM3
 CMP A #$CE  LDX?
 BEQ IMM3
 JMP DIR
IMM3 JSR POP2  OK
 JMP EXT1
BSIMM  LDX PCREG
 BSR SAVLK2
 JMP REL
*
INHER JSR POP0 FILL OPRND
 LDA B INST
 CMP B #$39 RTS
 BEQ INH1
 CMPB #$3B RTI
 BEQ INHOUT
 CMPB #$3E WAI
 BEQ INHOUT
 CMP B #$3F SWI
 BEQ INHOUT
 LDA B #1
 JMP EXT1A
INHOUT LDA A #'X  WON'T ALLOW
 JMP DMP1 PRINT & EXEC
INH1 TST SUBCNT
 BEQ INHOUT TOO MANY RTS?
 DEC SUBCNT
 STS STKTMP
 LDS MYSTK
 PU
   L

    A
 STA A HERE
 PUL A
 STA A HERE+1
```

continued

```
  LDX HERE
  STX PCREG
  STS MYSTK
  LDS STKTMP
 JMP EXTIC
* .
INDX JSR POP1  LOAD OPRND
   LDX XREG
   STX CKADR
   CLC
   CLR B
   LDA A INST+1   LOAD INDEX VALUE
 BSR ADDM
INDX2  LDA A INST
   CMP A #$AD
                JSR?
 BEQ INDX4
 CMP A #$6E JMP
 BEQ INDX5
INDX3 JMP DIR3
INDX4 LDX PCREG
 JSR SAVLX2
INDX5 JMP EXT2
*
ADDM ADD A CKADR+1 LS BITS
   ADCB CKADR MS BITS
ADDM1 STA A CKADR+1
   STA B CKADR
 RTS
*
SUBM  ADD A CKADR+1
 BCC SUB1
 ADD B CKADR
 BRA ADDM1
SUB1 ADD B CKADR
 DEC B
 BRA ADDM1
*
REL JSR POP1 OPRND
 LDX PCREG
 INX
 INX
 STX CKADR
   LDA A INST  GET READY FOR JUMP
   STA A PSEUDO
   LDA A CCREG  LOAD CNDTNS
   TAP
**********
PSEUDO FCB Ø,2
**********
 BRA INDX3  DOES NOT JMP
REL2 CLC  DOES JMP
   CLR B
   LDA A INST+1
   BPL REL3  IS JMP POS OR NEG
   BSR SUBM
   FCB $8C CPX
REL3 BSR ADDM
REL4 BRA INDX5  MAKE SWAP
*
RUNVCT  LDX SWIADR RESTORE INSTR
 LDA A INST
 STA A X
 LDA A #7
 LDX #CCREG
SAV1 PUL B
 STA B X
 INX
 DEC A
 BNE SAV1
 STS STKHI
  BSR CKHUM  CHECK HUMAN
RUN LDX PCREG
 DEX DUE TO SWI
RUN2 STX PCREG
 LDA A INST
 AND A #$FØ CLEAR JNK
 LSR A
 LSR A
 LSR A
 LDX #TABLE-1 SET FOR JMP
R1 INX
 DEC A
 BPL R1
 LDX X
 JMP X TAKE JMP
*
```

```
CKHUM JSR POLCAT  HUMAN WANT CONTROL?
  BCC CKHUM2  NO
CKHUM1 JSR INCH+4
CKHUM3  CMP B #$1B  ESCAPE?
  BNE CKHUM2  NOPE
  JMP DEBUG  SCRAM
CKHUM2 RTS  BACK YOU GO
*
EXMDR  CPX BIADR  INST BKPNT?
  BEQ BKPT
 LDA A TON+1
 LDA B TON
 SUB A #1 CRRCT FOR CARRY
 SBC B #Ø
 SUB A CKADR+1
 SBC B CKADR
 BCS EX2
EXMOP CPX BOADR OPRND BKPNT?
 BEQ BKPT
EX1 RTS
EX2 LDA A TOFF+1
 LDA B TOFF
 SUB A CKADR+1
 SBC B CKADR
 BCS EX1
EX3  LDA A #'T
  STA A WHAT
 JMP PRNTRG DMP & RTRN
*
BKPT LDA A #'B
 JMP DMP1 PRINT & EXEC
*
REPAK  LDS STKHI  REPAK STACK
  LDA A #7
 LDX #PCREG+1
REP1  LDA B X
  PSH B
  DEX
  DEC A
  BNE REP1
 LDX PCREG ANYTHING GOING ON?
 STX CKADR
 JSR EXMDR GO SEE
 FCB $CE LDX #
HERE FCB Ø,Ø
 LDA A X
 STA A INST
 LDA A #$3F
 STA A X
 STX SWIADR
 RTI
*
POPØ CLR A  NO OPRND
  STA A ASCFG
  RTS
POP1 LDA A #1
  BSR POPØ+1
  LDX PCREG
  LDA B 1,X
  STA B INST+1
  RTS
POP2 LDA A #2
  BSR POP1+2
  LDA B 2,X
  STA B INST*2
  RTS
*
BAD BSR EON  ECHO ON
  JSR BADDR  GET ADDR
 BRA EOF
*
EON LDA A #$Ø3
  FCB $8C CPX
EOF LDA A #$FF
 STA A ECHO
  RTS
*
IN BSR EON
  JSR INCH
 STA B WHAT
 BSR PNTS
  BRA EOF
*
BY BSR EON
  JSR BYTE
  BRA EOF
*
```

```
PRNTRG LDX #MES4
 JSR MSG
   LDA B WHAT  WHAT TYPE DMP
   BSR PNT1
   LDA A INST  INST
 BSR OUT2
  LDA A ASCFG  OPRND?
   BEQ PRN3  NONE
   LDA A INST*1
   JSR OUT2H
   LDA A ASCFG  MORE?
  DEC A
   BEQ PRN2  NOPE
   LDA A INST+2
   JSR OUT2H
   BRA PRN1
PRN3 BSR XX
PRN2 BSR XX
PRN1 BSR XX
 LDX #STKHI
**********
  FCB $C6  (LDA B #)
HMNY FCB 9
**********
PRNLP BEQ PRN4
 LDA A X
 PSH B
 BSR OUT2
 PUL B
 INX
 DEC B
 BRA PRNLP
PRN4 LDA A #9 FORM RESET
 STA A HMNY
 RTS
*
PNT1 JSR OUTCH
PNTS JSR OUTS
PNTC JMP CKHUM
*
XX BSR PNTS
 BRA PNTS
*
OUT2 JSR OUT2H
 BRA PNTS
*
PRMPT FCB $ØD,$ØA
 FCB $FF
  FCC /@ /
  FCB Ø
*
MES1 FCB $ØD,$ØA
 FCB $FF
  FCC /DEBUG/
  FCB Ø
*
MES2 FCC / ADDR ? /
  FCB Ø
*
MES4 FCB $ØD,$ØA
 FCB $FF,Ø
*
EM FCC /*ERROR*/
  FCB Ø
*
MYSTK FDB START-1
STKTMP FCB Ø,Ø
SUBCNT FCB Ø
SWIADR FCB Ø,Ø
STKSV FCB Ø,Ø
*
WHAT FCB Ø
INST FCB $3F,Ø,Ø
ASCFG FCB Ø
STKHI FCB Ø,Ø
CCREG FCB Ø
BREG FCB Ø
AREG FCB Ø
XREG FCB Ø,Ø
PCREG FCB Ø,Ø
TON FCB $FF,$FF
TOFF FCB Ø,Ø
BIADR FCB Ø,Ø
BOADR FCB Ø,Ø
*
JMPTB FCC /M/ MONITOR
 FDB MONIT
 FCC /C/ CREG
 FDB STC
 FCC /B/ BREG
```

continued on page14

Trace Program Simplifies Debugging

Source Listing
continued

```
 FDB STB
 FCC /A/ AREG
 FDB STA
 FCC /X/ XREG
 FDB STX
 FCC /T/ TRACE
 FDB TSET
 FCC /O/ OPR BKPT
 FDB BO
 FCC /I/ INST BKPT
 FDB BI
 FCC /J/ JMP
 FDB JMPXX
 FCC /D/ DMP REG
 FDB DMP
 FCB Ø
*
TABLE FDB INHER
 FDB INHER
 FDB REL
 FDB INHER
 FDB INHER
 FDB INHER
 FDB INDX
 FDB EXT
 FDB IMM
 FDB DIR
 FDB INDX
 FDB EXT
 FDB IMM
 FDB DIR
 FDB INDX
 FDB EXT
*
 ORG $ØØF3
 FCB $Ø3
*
 END

@
```

Assembled Listing

```
                     NAM       DEBUG
             *
             *SOURCE 1.2.Ø
             *
             *JUNE 1977 DLJ
             *
                     OPT       NOG
             *
ØØF3                 ORG       $ØØF3
ØØF3 FF              FCB       $FF
             *
             * INSTRUCTIONS:
             *
             * D = (D)UMP REGISTERS
             * M = (M)ONITOR RETURN
             * J = (J)UMP
             * A/B/C/X/I/O/T =
             * SET REGISTERS/BREAKPOINTS/TRACE
             *
        FF62 BADDR   EQU       $FF62
        ØØF2 BRKADR  EQU       $ØØF2
        FF53 BYTE    EQU       $FF53
        ØØF3 ECHO    EQU       $ØØF3
        FFØØ INCH    EQU       $FFØØ
        FF6D OUT2H   EQU       $FF6D
        FF81 OUTCH   EQU       $FF81
        FF82 OUTS    EQU       $FF82
        FF24 POLCAT  EQU       $FF24
             *
4ØØØ                 ORG       $4ØØØ
4ØØØ BF 439D START   STS       STKSV     SAVE IT
4ØØ3 Ø7              TPA
4ØØ4 B7 43A6         STA A     CCREG
             *
4ØØ7 CE 4378 DEBUG   LDX       #MES1     SEND 'DEBUG'
4ØØA 8D 57           BSR       MSG
             *
4ØØC BE 439D EXEC    LDS       STKSV
4ØØF BF 43A4         STS       STKHI
4Ø12 CE 3FFF         LDX       #START-1
4Ø15 FF 4396         STX       MYSTK
4Ø18 7F 439A         CLR       SUBCNT
4Ø1B FE 439B         LDX       SWIADR
4Ø1E B6 43AØ         LDA A     INST
4Ø21 A7 ØØ           STA A     X
4Ø23 7F 439B         CLR       SWIADR
4Ø26 CE 4372         LDX       #PRMPT    POP OUT A @
4Ø29 8D 38           BSR       MSG
4Ø2B CE 4239         LDX       #RUNVC1   SET RUN VECTOR
4Ø2E DF Ø1           STX       1         STORE AT SWI
4Ø3Ø 86 7E           LDA A     #$7E      LOAD A JMP
4Ø32 97 ØØ           STA A     Ø         STORE IT AT SWI
4Ø34 43              COM A               SET HIGH BIT
4Ø35 97 F2           STA A     BRKADR    AT BREAK ADDR
4Ø37 BD 43Ø7         JSR       IN        GET A CHRCTR
4Ø3A CE 43B5         LDX       #JMPTB    JUMP TABLE
4Ø3D E6 ØØ   EXEC1   LDA B     X         GET LTR
4Ø3F 27 ØE           BEQ       BUM       DONE?
4Ø41 F1 439F         CMP B     WHAT      MATCH?
4Ø44 27 Ø5           BEQ       JMPCMD
4Ø46 Ø8              INX                 TO NEXT LTR
4Ø47 Ø8              INX
4Ø48 Ø8              INX
4Ø49 2Ø F2           BRA       EXEC1
4Ø4B EE Ø1   JMPCMD  LDX       1,X       TAKE IT
4Ø4D 6E ØØ
                     JMP       X
             *
4Ø4F CE 438E BUM     LDX       #EM       BUMMER
4Ø52 8D ØF           BSR       MSG
4Ø54 2Ø B6   BUM1    BRA       EXEC      BACK YOU GO
             *
4Ø56 B7 439F DMP1    STA A     WHAT
4Ø59 86 11   DMP     LDA A     #$11
4Ø5B B7 434D         STA A     HMNY      SET FOR BIG DMP
4Ø5E BD 431A DMP3    JSR       PRNTRG
4Ø61 2Ø F1   DMP2    BRA       BUM1      EXEC
             *
4Ø63 E6 ØØ   MSG     LDA B     Ø,X
4Ø65 27 Ø6           BEQ       MSG1
4Ø67 BD FF81         JSR       OUTCH
4Ø6A Ø8              INX
4Ø6B 2Ø F6           BRA       MSG
4Ø6D 39      MSG1    RTS
             *
```

continued

for Altair™ 680b continued

```
ØØØ86 4Ø6E D7 F3    MONIT  STA B   ECHO
ØØØ87 4Ø7Ø D7 F2           STA B   BRKADR
ØØØ88 4Ø72 3F              SWI             BACK TO MONITOR
ØØØ89 4Ø73 7E 4ØØ7         JMP     DEBUG   READY FOR (P)ROCEED
ØØØ9Ø               *
ØØØ91 4Ø76 8D 18    TSET   BSR     ADPRM   TRACE SET  GET ADDR
ØØØ92 4Ø78 FF 43AD         STX     TON     TRACE ON ADR
ØØØ93 4Ø7B 8D 13           BSR     ADPRM
ØØØ94 4Ø7D FF 43AF         STX     TOFF    TRACE OFF ADR
ØØØ95 4Ø8Ø 2Ø DF    TS1    BRA     DMP2    EXEC
ØØØ96               *
ØØØ97 4Ø82 8D ØC    BI     BSR     ADPRM   INST BREAKPT
ØØØ98 4Ø84 FF 43B1         STX     BIADR
ØØØ99 4Ø87 2Ø F7           BRA     TS1     EXEC
ØØ1ØØ               *
ØØ1Ø1 4Ø89 8D Ø5    BO     BSR     ADPRM   OPRND BKPT
ØØ1Ø2 4Ø8B FF 43B3         STX     BOADR
ØØ1Ø3 4Ø8E 2Ø FØ           BRA     TS1
ØØ1Ø4               *
ØØ1Ø5 4Ø9Ø CE 4381  ADPRM  LDX     #MES2
ØØ1Ø6 4Ø93 BD 4Ø63  ADPRM1 JSR     MSG
ØØ1Ø7 4Ø96 7E 42F8  ADPRM2 JMP     BAD     & RTRN
ØØ1Ø8               *
ØØ1Ø9 4Ø99 BD 4313  STC    JSR     BY      CNDTN REG
ØØ11Ø 4Ø9C F7 43A6         STA B   CCREG
ØØ111 4Ø9F 2Ø DF    STC1   BRA     TS1
ØØ112               *
ØØ113 4ØA1 BD 4313  STB
                           JSR     BY      BREG
ØØ114 4ØA4 F7 43A7         STA B   BREG
ØØ115 4ØA7 2Ø F6           BRA     STC1
ØØ116               *
ØØ117 4ØA9 BD 4313  STA    JSR     BY      AREG
ØØ118 4ØAC F7 43A8         STA B   AREG
ØØ119 4ØAF 2Ø EE           BRA     STC1
ØØ12Ø               *
ØØ121 4ØB1 8D E3    STX    BSR     ADPRM2  XREG
ØØ122 4ØB3 FF 43A9         STX     XREG
ØØ123               *
ØØ124 4ØB6 2Ø E7    ST5    BRA     STC1    EXEC
ØØ125               *
ØØ126 4ØB8 8D D6    JMPXX  BSR     ADPRM   GET ADR
ØØ127 4ØBA A6 ØØ           LDA A   X       GET INST
ØØ128 4ØBC B7 43AØ         STA A   INST
ØØ129 4ØBF 7E 4256         JMP     RUN2
ØØ13Ø               *
ØØ131 4ØC2 BD 42E1  DIR    JSR     POP1    LOAD OPRND
ØØ132 4ØC5 F7 412C         STA B   CKADR+1
ØØ133 4ØC8 7F 412B         CLR     CKADR
ØØ134 4ØCB FE 412B         LDX     CKADR
ØØ135 4ØCE BD 4293  DIR3   JSR     EXMOP
ØØ136 4ØD1 C6 Ø2    DIR2   LDA B   #2      NEXT SWI
ØØ137 4ØD3 2Ø 19           BRA     EXT1A
ØØ138               *
ØØ139 4ØD5 BD 42EE  EXT    JSR     POP2    LOAD OPRND
ØØ14Ø 4ØD8 B6 43AØ         LDA A   INST
ØØ141 4ØDB FE 43A1         LDX     INST+1  GET ADR
ØØ142 4ØDE FF 412B         STX     CKADR
ØØ143 4ØE1 BD 4293         JSR     EXMOP
ØØ144 4ØE4 81 7E           CMP A   #$7E    JMP?
ØØ145 4ØE6 27 1C           BEQ     EXT2
ØØ146 4ØE8 81 BD           CMP A   #$BD    JSR?
ØØ147 4ØEA 27 21           BEQ     EXT3
ØØ148 4ØEC C6 Ø3    EXT1   LDA B   #3      NEXT SWI
ØØ149 4ØEE FE 43AB  EXT1A  LDX     PCREG
ØØ15Ø 4ØF1 5D       EXT1B  TST B
ØØ151 4ØF2 27 Ø4           BEQ     EXT1C
ØØ152 4ØF4 Ø8              INX
ØØ153 4ØF5 5A              DEC B
ØØ154 4ØF6 2Ø F9           BRA     EXT1B
ØØ155 4ØF8 FF 42CD  EXT1C  STX     HERE
ØØ156 4ØFB 7E 42B4         JMP     REPAK
ØØ157 4ØFE FE 43AB  EXT2B  LDX     PCREG
ØØ158 41Ø1 BD 4147
                           JSR     SAVLK3
ØØ159 41Ø4 FE 412B  EXT2   LDX     CKADR
ØØ16Ø 41Ø7 FF 43AB         STX     PCREG   SWAP
ØØ161 41ØA 5F              CLR B           NEXT SWI
ØØ162 41ØB 2Ø E1           BRA     EXT1A
ØØ163 41ØD 8C FF81  EXT3   CPX     #OUTCH
ØØ164 411Ø 27 ØF           BEQ     DOIT
ØØ165 4112 8C FF6D         CPX     #OUT2H
ØØ166 4115 27 ØA           BEQ     DOIT
ØØ167 4117 8C FFØØ         CPX     #INCH
ØØ168 411A 27 Ø5           BEQ     DOIT
```

continued on page 18

Classified Ads

For Sale
Altair™ 8800
Fully assembled, tested--runs beautifully.
16K memory, serial I/0, RS232 I/0,
Clock Vectored Interrupt, flexible disk.
All documentation and many programs.
including Op. Sys., Assem., Edit,
BASIC and games.
$3200
Contact:

Computer Solutions
17922 Sky Park Cr.
Suite L
Irvine, CA 92714
(714) 751-5040

Correction

GLITCHES, p. 19, Oct. CN

The last line in the second paragraph should read, "Kits and assembled units will use **74LS13** for ICA and B. There's no such chip as a 74SL5153.

Also, note that a separate 25-pin DB connector is used for RS-232 (wired as before), and a separate 25DB connector is used for the TTY printer.

Destroying Klingons Can

Audio Star Trek Using the 88-MU1
By Thomas G. Schneider
MITS

Bleep-Bleep!

Klingon at sector 4-8, Captain. I recommend immediate action.

Blow him away, Sulu!

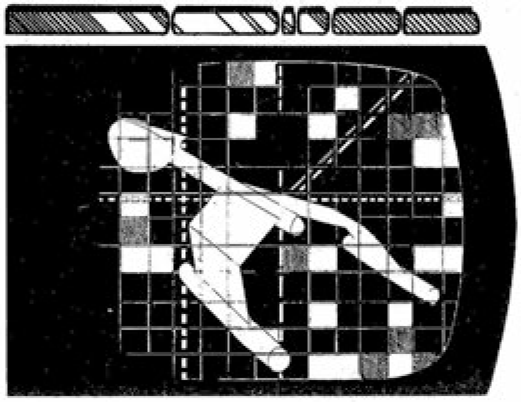
BZZZZZZZZZZZZZZT. . .Poot!

Klingon destroyed, Captain!

Wouldn't computer Star Trek be really far-out if it actually made those sounds? Let's face it, watching those K's disappear on your screen quietly and undramatically leaves a lot to be desired. But now, with the new Altair™88-MU1, you can produce almost any sound effects for practically any purpose, including Star Trek.

Listing 1 is a version of Star Trek modified for sound effects. These effects are generated by the subroutines listed at the end of the program. Sounds are produced for maps, warp engines, photon torpedos, phasors, destruction of stars and klingons, and command prompts. As an added feature, an appropriate melody is played to insult the user who misses a klingon. If you want to modify Star Trek even more radically, refer to listing 2, which shows where the sound routines are called.

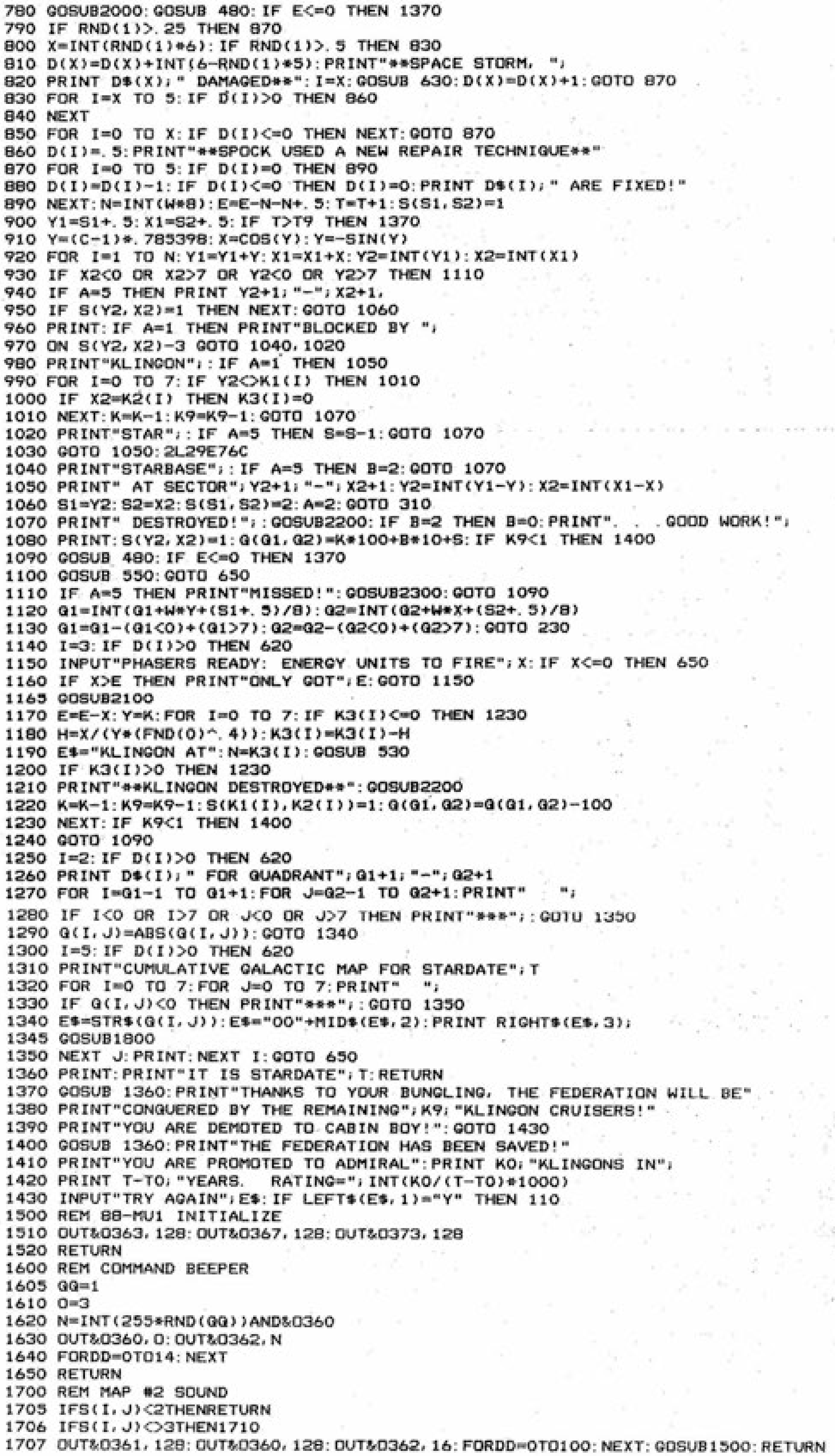
So plug in your new 88-MU1, load up audio Star Trek, turn up your amplifier, and get those klingons.



```
9 GOSUB1500
10 DIM D(5),K1(7),K2(7),K3(7),S(7,7),G(7,7),D$(5)
20 G$=".EKB*"
30 D$(0)="WARP ENGINES"
40 D$(1)="SHORT RANGE SENSORS"
50 D$(2)="LONG RANGE SENSORS"
60 D$(3)="PHASERS"
70 D$(4)="PHOTON TORPEDOES":D$(5)="GALACTIC RECORDS"
80 INPUT"PLEASE ENTER A RANDOM NUMBER";E$:I=ASC(E$)
90 I=I-11*INT(I/11):FOR J=0 TO I:K=RND(1):NEXT:PRINT"WORKING-"
100 DEF FND(N)=SGR((K1(I)-S1)^2+(K2(I)-S2)^2)
110 GOSUB 610:GOSUB 450:G1=X:G2=Y:X=8:Y=1:X1=.2075:Y1=6.28:X2=3.28
120 Y2=1.8:A=.96:C=100:W=10:K9=0:B9=0:S9=400:T9=3451:GOTO 140
130 K=K+(N<X2)+(N<Y2)+(N<.28)+(N<.08)+(N<.03)+(N<.01):K9=K9-K:GOTO 160
140 T0=3421:T=T0:E0=4000:E=E0:P0=10:P=P0:FOR I=0 TO 7
150 FOR J=0 TO 7:K=0:N=RND(Y):IF N<X1 THEN N=N*64:K=(N<Y1)-Y:GOTO 130
160 B=(RND(Y)>A):B9=B9-B:G(I,J)=K*C+B*W-INT(RND(Y)*X+Y):NEXT J,I
170 IF K9>(T9-T0) THEN T9=T0+K9
180 IF B9>0 THEN 200
190 GOSUB 450:G(X,Y)=G(X,Y)-10:B9=1
200 PRINT LEFT$("STARTREK ADAPTED BY L.E.COCHRAN 2/29/76",8):K0=K9
210 PRINT"OBJECTIVE: DESTROY";K9;"KLINGON BATTLE CRUISERS IN";T9-T0;
220 PRINT"YEARS.":PRINT" THE NUMBER OF STARBASES IS";B9
230 A=0:IF G1<0 OR G1>7 OR G2<0 OR G2>7 THEN N=0:S=0:K=0:GOTO 250
240 N=ABS(G(G1,G2)):G(G1,G2)=N:S=N-INT(N/10)*10:K=INT(N/100)
250 B=INT(N/10-K*10):GOSUB 450:S1=X:S2=Y
260 FOR I=0 TO 7:FOR J=0 TO 7:S(I,J)=1:NEXT J,I:S(S1,S2)=2
270 FOR I=0 TO 7:K3(I)=0:X=8:IF I<K THEN GOSUB 460:S(X,Y)=3:K3(I)=S9
280 K1(I)=X:K2(I)=Y:NEXT:I=S
290 IF B>0 THEN GOSUB 460:S(X,Y)=4
300 IF I>0 THEN GOSUB 460:S(X,Y)=5:I=I-1:GOTO 300
310 GOSUB 550:IF A=0 THEN GOSUB 480
320 IF E<=0 THEN 1370
330 I=1:IF D(I)>0 THEN 620
340 FOR I=0 TO 7:FOR J=0 TO 7:PRINT MID$(G$,S(I,J),1);" ";:GOSUB1700:NEXT J
350 PRINT"  ";:ON I GOTO 380,390,400,410,420,430,440
360 PRINT"YEARS =";T9-T
370 NEXT:GOTO 650
380 PRINT"STARDATE=";T:GOTO 370
390 PRINT"CONDITION: ";C$:GOTO 370
400 PRINT"QUADRANT=";G1+1;"-";G2+1:GOTO 370
410 PRINT"SECTOR =";S1+1;"-";S2+1:GOTO 370
420 PRINT"ENERGY=";E:GOTO 370
430 PRINT D$(4);"=";P:GOTO 370
440 PRINT"KLINGONS LEFT=";K9:GOTO 370
450 X=INT(RND(1)*8):Y=INT(RND(1)*8):RETURN
460 GOSUB 450:IF S(X,Y)>1 THEN 460
470 RETURN
480 IF K<1 THEN RETURN
490 IF C$="DOCKED" THEN PRINT"STARBASE PROTECTS ENTERPRISE":RETURN
500 FOR I=0 TO 7:IF K3(I)<=0 THEN NEXT:RETURN
510 H=K3(I)*.4*RND(1):K3(I)=K3(I)-H:H=H/(FND(0)^.4):E=E-H
520 E$="ENTERPRISE FROM":N=E:GOSUB 530:NEXT:RETURN
530 PRINT H;"UNIT HIT ON ";E$;" SECTOR";K1(I)+1;"-";K2(I)+1;
540 PRINT" (";N;"LEFT)":RETURN
550 FOR I=S1-1 TO S1+1:FOR J=S2-1 TO S2+1
560 IF I<0 OR I>7 OR J<0 OR J>7 THEN 580
570 IF S(I,J)=4 THEN C$="DOCKED":E=E0:P=P0:GOSUB 610:RETURN
580 NEXT J,I:IF K>0 THEN C$="RED":RETURN
590 IF E<E0*.1 THEN C$="YELLOW":RETURN
600 C$="GREEN":RETURN
610 FOR N=0 TO 5:D(N)=0:NEXT:RETURN
620 PRINT D$(I);" DAMAGED.";
630 PRINT" ";D(I);"YEARS ESTIMATED FOR REPAIR.":PRINT
640 IF A=1 THEN RETURN
650 FORLL=1TO7:PRINTMID$("COMMAND",LL,1);:GOSUB1600:NEXT:GOSUB1500:INPUTA
660 IF A<1 OR A>6 THEN 680
670 ON A GOTO 710,310,1250,1140,690,1300
680 FOR I=0 TO 5:PRINT I+1;"= ";D$(I):NEXT:GOTO 650
690 IF D(4)>0 THEN PRINT"SPACE CRUD BLOCKING TUBES.";:I=4:GOTO 630
700 N=15:IF P<1 THEN PRINT"NO TORPEDOES LEFT":GOTO 650
710 IF A=5 THEN PRINT"TORPEDO ";
720 INPUT"COURSE (1-8.9)";C:IF C<1 THEN 650
730 IF C>=9 THEN 710
740 IF A=5 THEN P=P-1:GOSUB1900:PRINT"TRACK:";:GOTO 900
750 INPUT"WARP (0-12)";W:IF W<=0 OR W>12 THEN 710
760 IF W<=.2 OR D(0)<=0 THEN 780
770 I=0:PRINT D$(I);" DAMAGED, MAX IS .2 ";:GOSUB 630:GOTO 750
```

continued

Bring Music to Your Ears



```
780 GOSUB2000: GOSUB 480: IF E<=0 THEN 1370
790 IF RND(1)>.25 THEN 870
800 X=INT(RND(1)*6): IF RND(1)>.5 THEN 830
810 D(X)=D(X)+INT(6-RND(1)*5): PRINT"**SPACE STORM, ";
820 PRINT D$(X);" DAMAGED**": I=X: GOSUB 630: D(X)=D(X)+1: GOTO 870
830 FOR I=X TO 5: IF D(I)>0 THEN 860
840 NEXT
850 FOR I=0 TO X: IF D(I)<=0 THEN NEXT: GOTO 870
860 D(I)=.5: PRINT"**SPOCK USED A NEW REPAIR TECHNIQUE**"
870 FOR I=0 TO 5: IF D(I)=0 THEN 890
880 D(I)=D(I)-1: IF D(I)<=0 THEN D(I)=0: PRINT D$(I);" ARE FIXED!"
890 NEXT: N=INT(W*8): E=E-N-N+.5: T=T+1: S(S1,S2)=1
900 Y1=S1+.5: X1=S2+.5: IF T>T9 THEN 1370
910 Y=(C-1)*.785398: X=COS(Y): Y=-SIN(Y)
920 FOR I=1 TO N: Y1=Y1+Y: X1=X1+X: Y2=INT(Y1): X2=INT(X1)
930 IF X2<0 OR X2>7 OR Y2<0 OR Y2>7 THEN 1110
940 IF A=5 THEN PRINT Y2+1;"-";X2+1,
950 IF S(Y2,X2)=1 THEN NEXT: GOTO 1060
960 PRINT: IF A=1 THEN PRINT"BLOCKED BY ";
970 ON S(Y2,X2)-3 GOTO 1040,1020
980 PRINT"KLINGON";: IF A=1 THEN 1050
990 FOR I=0 TO 7: IF Y2<>K1(I) THEN 1010
1000 IF X2=K2(I) THEN K3(I)=0
1010 NEXT: K=K-1: K9=K9-1: GOTO 1070
1020 PRINT"STAR";: IF A=5 THEN S=S-1: GOTO 1070
1030 GOTO 1050: 2L29E76C
1040 PRINT"STARBASE";: IF A=5 THEN B=2: GOTO 1070
1050 PRINT" AT SECTOR";Y2+1;"-";X2+1: Y2=INT(Y1-Y): X2=INT(X1-X)
1060 S1=Y2: S2=X2: S(S1,S2)=2: A=2: GOTO 310
1070 PRINT" DESTROYED!";: GOSUB2200: IF B=2 THEN B=0: PRINT". . .GOOD WORK!";
1080 PRINT: S(Y2,X2)=1: G(G1,G2)=K*100+B*10+S: IF K9<1 THEN 1400
1090 GOSUB 480: IF E<=0 THEN 1370
1100 GOSUB 550: GOTO 650
1110 IF A=5 THEN PRINT"MISSED!": GOSUB2300: GOTO 1090
1120 G1=INT(G1+W*Y+(S1+.5)/8): G2=INT(G2+W*X+(S2+.5)/8)
1130 G1=G1-(G1<0)+(G1>7): G2=G2-(G2<0)+(G2>7): GOTO 230
1140 I=3: IF D(I)>0 THEN 620
1150 INPUT"PHASERS READY:  ENERGY UNITS TO FIRE";X: IF X<=0 THEN 650
1160 IF X>E THEN PRINT"ONLY GOT";E: GOTO 1150
1165 GOSUB2100
1170 E=E-X: Y=K: FOR I=0 TO 7: IF K3(I)<=0 THEN 1230
1180 H=X/(Y*(FND(0)^.4)): K3(I)=K3(I)-H
1190 E$="KLINGON AT": N=K3(I): GOSUB 530
1200 IF K3(I)>0 THEN 1230
1210 PRINT"**KLINGON DESTROYED**": GOSUB2200
1220 K=K-1: K9=K9-1: S(K1(I),K2(I))=1: G(G1,G2)=G(G1,G2)-100
1230 NEXT: IF K9<1 THEN 1400
1240 GOTO 1090
1250 I=2: IF D(I)>0 THEN 620
1260 PRINT D$(I);" FOR QUADRANT";G1+1;"-";G2+1
1270 FOR I=G1-1 TO G1+1: FOR J=G2-1 TO G2+1: PRINT"   ";
1280 IF I<0 OR I>7 OR J<0 OR J>7 THEN PRINT"***";: GOTU 1350
1290 G(I,J)=ABS(G(I,J)): GOTO 1340
1300 I=5: IF D(I)>0 THEN 620
1310 PRINT"CUMULATIVE GALACTIC MAP FOR STARDATE";T
1320 FOR I=0 TO 7: FOR J=0 TO 7: PRINT"  ";
1330 IF G(I,J)<0 THEN PRINT"***";: GOTO 1350
1340 E$=STR$(G(I,J)): E$="00"+MID$(E$,2): PRINT RIGHT$(E$,3);
1345 GOSUB1800
1350 NEXT J: PRINT: NEXT I: GOTO 650
1360 PRINT: PRINT"IT IS STARDATE";T: RETURN
1370 GOSUB 1360: PRINT"THANKS TO YOUR BUNGLING, THE FEDERATION WILL BE"
1380 PRINT"CONQUERED BY THE REMAINING";K9;"KLINGON CRUISERS!"
1390 PRINT"YOU ARE DEMOTED TO CABIN BOY!": GOTO 1430
1400 GOSUB 1360: PRINT"THE FEDERATION HAS BEEN SAVED!"
1410 PRINT"YOU ARE PROMOTED TO ADMIRAL": PRINT K0;"KLINGONS IN";
1420 PRINT T-T0;"YEARS.   RATING=";INT(K0/(T-T0)*1000)
1430 INPUT"TRY AGAIN";E$: IF LEFT$(E$,1)="Y" THEN 110
1500 REM 88-MU1 INITIALIZE
1510 OUT&0363,128: OUT&0367,128: OUT&0373,128
1520 RETURN
1600 REM COMMAND BEEPER
1605 GG=1
1610 O=3
1620 N=INT(255*RND(GG))AND&0360
1630 OUT&0360,0: OUT&0362,N
1640 FORDD=0TO14: NEXT
1650 RETURN
1700 REM MAP #2 SOUND
1705 IFS(I,J)<2THENRETURN
1706 IFS(I,J)<>3THEN1710
1707 OUT&0361,128: OUT&0360,128: OUT&0362,16: FORDD=0TO100: NEXT: GOSUB1500: RETURN
```

continued on page 18

```
1710 OUT&0361, S(I, J)
1720 OUT&0362, 2^I
1730 GOSUB1500
1740 RETURN
1800 REM MAP #3 AND #6 SOUND
1805 IFG(I, J)<100THEN1810
1806 OUT&0361, 128: OUT&0360, 128: OUT&0362, 16: FORDD=0TO100: NEXT: GOSUB1500: RETURN
1810 OUT&0361, G(I, J)
1820 OUT&0362, 2^I
1830 GOSUB1500
1840 RETURN
1900 REM PHOTON TORPEDO SOUND
1905 O=128
1910 O=O/2
1920 FORN=0TO11
1930 OUT&0362, N: OUT&0361, O
1940 NEXT: IFO<>1THEN1910
1945 GOSUB1500
1950 RETURN
2000 REM WARP SOUND
2005 FORKK=1TO3
2010 OUT&0361, &0300
2015 OUT&0360, &040
2020 FORN=0TO11
2021 NN=N*16: OUT&0362, NN+N
2025 FORDD=0TO50: NEXT
2040 NEXT
2045 NEXT
2050 OUT&0360, 0: OUT&0361, 0: RETURN
2100 REM PHASOR SOUNDS
2110 FORPP=1TO200
2112 OUT&0361, 3
2115 PN=ABS(PN-1)
2116 OUT&0362, PN
2130 NEXT
2140 OUT&0361, 0
2150 RETURN
2200 REM DEAD ITEM SOUND
2205 OUT&0361, &0300
2210 FORN=11TO0STEP-1
2215 FORDD=0TO40: NEXT
2220 OUT&0362, N
2230 NEXT
2240 OUT&0361, 0: RETURN
2300 REM INSULT MELODY
2310 READN, TT
2315 IFTT=0THEN2350
2320 OUT&0361, &010: OUT&0362, N
2330 FORD=0TOTT: NEXT
2340 GOTO2310
2350 OUT&0361, 0: RESTORE: RETURN
3000 DATA3, 100
3001 DATA12, 4
3002 DATA3, 100
3003 DATA0, 100
3004 DATA5, 100
3005 DATA3, 200
3006 DATA0, 200
3010 DATA0, 0
```

TRACE PROGRAM

Assembled Listing continued

```
ØØ169 411C 8C FF82         CPX        #OUTS
ØØ170 411F 26 DD           BNE        EXT2B
ØØ171 4121 BD 42FF DOIT    JSR        EON
ØØ172 4124 B6 43A8         LDA A      AREG
ØØ173 4127 F6 43A7         LDA B      BREG
ØØ174              ***********
ØØ175 412A BD              FCB        $BD       JSR
ØØ176 412B ØØ      CKADR   FCB        Ø,Ø
ØØ177              ***********
ØØ178 412D B7 43A8         STA A      AREG
ØØ179 413Ø F7 43A7         STA B      BREG
ØØ18Ø 4133 BD 43Ø2         JSR        EOF
ØØ181 4136 BD 4274         JSR        CKHUM3    ESCAPE?
ØØ182 4139 FE 43AB         LDX        PCREG     NO
ØØ183 413C Ø8              INX                  PAST JSR
ØØ184 413E Ø8              INX
ØØ186 413F A6 ØØ           LDA A      X
ØØ187 4141 B7 43AØ         STA A      INST
ØØ188 4144 7E 4256         JMP        RUN2
ØØ189              *
ØØ19Ø 4147 Ø8      SAVLK3  INX                  SAVE LINK
ØØ191 4148 Ø8      SAVLK2  INX
ØØ192 4149 Ø8      SAVLK1  INX
ØØ193 414A FF 42CD         STX        HERE
ØØ194 414D BF 4398         STS        STKTMP
ØØ195 415Ø BE 4396         LDS        MYSTK
ØØ196 4153 B6 42CE         LDA A      HERE+1
ØØ197 4156 36              PSH A
ØØ198 4157 B6 42CD         LDA A      HERE
ØØ199 415A 36              PSH A
ØØ2ØØ 415B BF 4396         STS        MYSTK
ØØ2Ø1 415E BE 4398         LDS        STKTMP
ØØ2Ø2 4161 7C 439A         INC        SUBCNT
ØØ2Ø3 4164 39              RTS
ØØ2Ø4              *
ØØ2Ø5 4165 B6 43AØ IMM     LDA A      INST
ØØ2Ø6 4168 81 8D           CMP A      #$8D      BSR?
ØØ2Ø7 416A 27 15           BEQ        BSIMM
ØØ2Ø8 416C 81 8C           CMP A      #$8C      CPX?
ØØ2Ø9 416E 27 ØB           BEQ        IMM3
ØØ21Ø 417Ø 81 8E           CMP A      #$8E      LDS?
ØØ211 4172 27 Ø7           BEQ        IMM3
ØØ212 4174 81 CE           CMP A      #$CE      LDX?
ØØ213 4176 27 Ø3           BEQ        IMM3
ØØ214 4178 7E 4ØC2         JMP        DIR
ØØ215 417B BD 42EE IMM3    JSR        POP2      OK
ØØ216 417E 7E 4ØEC         JMP        EXT1
ØØ217 4181 FE 43AB BSIMM   LDX        PCREG
ØØ218 4184 8D C2           BSR        SAVLK2
ØØ219 4186 7E 4212
                           JMP        REL
ØØ22Ø              *
ØØ221 4189 BD 42DC INHER   JSR        POPØ      FILL OPRND
ØØ222 418C F6 43AØ         LDA B      INST
ØØ223 418F C1 39           CMP B      #$39      RTS
ØØ224 4191 27 16           BEQ        INH1
ØØ225 4193 C1 3B           CMP B      #$3B      RTI
ØØ226 4195 27 ØD           BEQ        INHOUT
ØØ227 4197 C1 3E           CMP B      #$3E      WAI
ØØ228 4199 27 Ø9           BEQ        INHOUT
ØØ229 419B C1 3F           CMP B      #$3F      SWI
ØØ23Ø 419D 27 Ø5           BEQ        INHOUT
ØØ231 419F C6 Ø1           LDA B      #1
ØØ232 41A1 7E 4ØEE         JMP        EXT1A
ØØ233 41A4 86 58   INHOUT  LDA A      #'X       WON'T ALLOW
ØØ234 41A6 7E 4Ø56         JMP        DMP1      PRINT & EXEC
ØØ235 41A9 7D 439A INH1    TST        SUBCNT
ØØ236 41AC 27 F6           BEQ        INHOUT    TOO MANY RTS?
ØØ237 41AE 7A 439A         DEC        SUBCNT
ØØ238 41B1 BF 4398         STS        STKTMP
```

continued

TRACE PROGRAM

Assembled Listing continued

```
41B4 BE 4396          LDS     MYSTK
41B7 32               PUL A
41B8 B7 42CD          STA A   HERE
41BB 32               PUL A
41BC B7 42CE          STA A   HERE+1
41BF FE 42CD          LDX     HERE
41C2 FF 43AB          STX     PCREG
41C5 BF 4396          STS     MYSTK
41C8 BE 4398          LDS     STKTMP
41CB 7E 40F8          JMP     EXT1C
                *
41CE BD 42E1    INDX  JSR     POP1      LOAD OPRND
41D1 FE 43A9          LDX     XREG
41D4 FF 412B          STX     CKADR
41D7 0C               CLC
41D8 5F               CLR B
41D9 B6 43A1          LDA A   INST+1    LOAD INDEX VALUE
41DC 8D 17            BSR     ADDM
41DE B6 43A0    INDX2 LDA A   INST
41E1 81 AD            CMP A   #$AD      JSR?
41E3 27 07            BEQ     INDX4
41E5 81 6E            CMP A   #$6E      JMP
41E7 27 09            BEQ     INDX5
41E9 7E 40CE    INDX3 JMP     DIR3
41EC FE 43AB    INDX4 LDX     PCREG
41EF BD 4148          JSR     SAVLK2
41F2 7E 4104    INDX5 JMP     EXT2
                *
41F5 BB 412C    ADDM  ADD A   CKADR+1   LS BITS
41F8 F9 412B          ADC B   CKADR     MS BITS
41FB B7 412C    ADDM1 STA A   CKADR+1
41FE F7 412B          STA B   CKADR
4201 39               RTS
                *
4202 BB 412C    SUBM  ADD A   CKADR+1
4205 24 05            BCC     SUB1
4207 FB 412B          ADD B   CKADR
420A 20 EF            BRA     ADDM1
420C FB 412B    SUB1  ADD B   CKADR
420F 5A               DEC B
4210 20 E9            BRA     ADDM1
                *
4212 BD 42E1    RL    JSR     POP1      OPRND
4215 FE 43AB          LDX     PCREG
4218 08               INX
4219 08               INX
421A FF 412B          STX     CKADR
421D B6 43A0          LDA A   INST      GET READY FOR JUMP
4220 B7 4227          STA A   PSEUDO
4223 B6 43A6          LDA A   CCREG     LOAD CNDTNS
4226 06               TAP
                **********
4227 00         PSEUDO FCB    0,2
                **********
4229 20 BE            BRA     INDX3     DOES NOT JMP
422B 0C         REL2  CLC               DOES JMP
422C 5F               CLR B
422D B6 43A1          LDA A   INST+1
4230 2A 03            BPL     REL3      IS JMP POS OR NEG
4232 8D CE            BSR     SUBM
4234 8C               FCB     $8C       CPX
4235 8D BE      REL3  BSR     ADDM
4237 20 B9      REL4  BRA     INDX5     MAKE SWAP
                *
4239 FE 439B    RUNVCT LDX    SWIADR    RESTORE INSTR
423C B6 43A0          LDA A   INST
423F A7 00            STA A   X
4241 86 07            LDA A   #7
4243 CE 43A6          LDX     #CCREG
4246 33         SAV1  PUL B
4247 E7 00            STA B   X
4249 08               INX
424A 4A               DEC A
424B 26 F9            BNE     SAV1
424D BF 43A4          STS     STKHI
4250 8D 1A            BSR     CKHUM     CHECK HUMAN
4252 FE 43AB    RUN   LDX     PCREG
4255 09               DEX               DUE TO SWI
4256 FF 43AB    RUN2  STX     PCREG
4259 B6 43A0          LDA A   INST
425C 84 F0            AND A   #$F0      CLEAR JNK
425E 44               LSR A
425F 44               LSR A
4260 44               LSR A
4261 CE 43D3          LDX     #TABLE-1  SET FOR JMP
4264 08         R1    INX
4265 4A               DEC A
```

COMPUTER NOTES IS
MOVING. . .

The main editorial office of Computer Notes will be located at Pertec offices in California.

Due to the change in location and editorial staff the publication of the November and December issues has been delayed.

Manuscripts and letters may still be sent to the MITS address. Watch the upcoming issues of CN for the new mailing address.

continued on page

String Character Editing Routine Runs in BASIC

By Ken Knecht
1240 W. 3rd St.
Space 135
Yuma, Arizona 85364

If you read my article ("Writing Machine Helps Prepare Manuscripts") in the July '77 **Computer Notes**, then you might have noticed that I mentioned plans to write a string character editing routine for my word processor program. I also said that I didn't see how it could be done in BASIC. Well, it can, and the following article explains how to do it.

The heart of the program is lines 65ØØ-651Ø. This subroutine inputs a character from the terminal without echoing it. The routine supports a subset of the MITS SIOA Rev. 1 I/O board. Changes of the port numbers and status flags will enable you to use the 2SIO board.

Essentially, the program supports a subset of the MITS BASIC character editing function. This version recognizes (n)C, (n)D, L, Q, I, H, and X. These are usually ample for most editing requirements. The S would also be useful, so I may add it later. The routine also recognizes the delete (rubout, backarrow, or whatever) command when in the insert mode (or after X or H). Edit commands can be in upper or lower case. As in MITS BASIC, editor command letters and numbers are not echoed.

Line **Description**

6ØØØ ED=1: Set edit flag in my program. The query gets the identifying number of the string to be edited in C. We transpose that to D for the program, set some program flags you don't need to be concerned with, get the length of the string in Z4, and initialize the variable.

6Ø1Ø Here we get the character input without echo in routine 65ØØ.

6Ø2Ø-611Ø Here we get the EDIT command in upper or lower case.

612Ø Error signal (bell); if input is not in edit routine repertoire, then the bell is sounded, and we go back to 6Ø1Ø for a valid input.

613Ø Space input; if LE (length of edited string is greater than Z4 (length of original string), then 612Ø.

614Ø Space input; print next character in string and transfer it to the edited string. Increment edited string character count. Go get next input character.

615Ø Numeric input; Z1$ contains the numeric characters received so far. Put number Z1$ or add to number already there.

616Ø Get next character input.

617Ø C input; if no number prefix (Z1$), then 6174.

6171 C input; set up for (n) changes of C.

6172 C input; get next character. Print it. Add it to edited string.

6173 C input; back to 6171 if more characters to change. When finished, add new characters to edited string count. Put null in Z1$ (numeric input). Get a new command.

6174 C input with no numeric prefix; print new character. Add to edited string character count. Add edited character to edited string. Get new command.

618Ø D input; if no numeric prefix then 622Ø.

619Ø D input with numeric prefix. Print initial "/". Set up character deletion corresponding to numeric input.

62ØØ Print deleted characters as per numeric input.

continued

```
LIST 6000-

6000 ED=1:PRINT"WHAT IS THE LINE NUMBER?":INPUT C:D=C:Z=Z+1:CH(Z,0)=C:
      GOSUB 3010:Z4=LEN(C$):LE=1:D$="":Z1$=""
6010 GOSUB 6500
6020 IF Z$=" "THEN 6130
6030 IF Z$=>"1"AND Z$<="9"THEN 6150
6040 IF Z$="C" OR Z$="c" THEN 6170
6050 IF Z$="D" OR Z$="d"THEN 6180
6060 IF Z$="L" OR Z$="l"THEN 6230
6070 IF Z$="Q" OR Z$="q"THEN 6260
6080 IF Z$="I" OR Z$="i" THEN 6270
6090 IF Z$="X" OR Z$="x" THEN 6290
6100 IF Z$="H" OR Z$="h"THEN 6320
6110 IF Z$=CHR$(13) THEN 6330
6120 PRINT CHR$(7);:GOTO 6010
6130 IF LE>Z4 THEN 6120
6140 PRINT MID$(C$,LE,1);:D$=D$+MID$(C$,LE,1):LE=LE+1:GOTO 6010
6150 IF Z1$<>""THEN Z1$=Z1$+Z$ ELSE Z1$=Z$
6160 GOTO 6010
6170 IF Z1$=""THEN 6174
6171 FOR Z2%=LE TO LE+VAL(Z1$)-1
6172 GOSUB 6500:PRINT Z$;:D$=D$+Z$
6173 NEXT:LE=Z2%:Z1$="":GOTO 6010
6174 GOSUB 6500:PRINT Z$;:LE=LE+1:D$=D$+Z$:GOTO 6010
6180 IF Z1$=""THEN 6220
6190 PRINT"\";:FOR Z2%=LE TO LE+VAL(Z1$)-1
6200 PRINT MID$(C$,Z2%,1);:NEXT
6210 PRINT"\";:LE=Z2%:Z1$="":GOTO 6010
6220 PRINT"\";:PRINT MID$(C$,LE,1);:PRINT"\";:LE=LE+1:GOTO 6010
6230 FOR Z2%=LE TO Z4
6240 PRINT MID$(C$,Z2%,1);:D$=D$+MID$(C$,Z2%,1)
6250 NEXT:C$=D$:D$="":PRINT:Z4=LEN(C$):LE=1:GOTO 6010
6260 PRINT:D$="":GOTO 270
6270 GOSUB 6500
6272 IF Z$=CHR$(127)THEN 6370
6274 IF Z$=CHR$(27)THEN 6010
6275 IF Z$=CHR$(13)THEN 6330
6280 PRINT Z$;:D$=D$+Z$:GOTO 6270
6290 FOR Z2%=LE TO Z4
6300 PRINT MID$(C$,Z2%,1);:D$=D$+MID$(C$,Z2%,1)
6310 NEXT:LE=Z4:GOTO 6270
6320 Z4=LE:GOTO 6270
6330 IF LE=>Z4 THEN PRINT CHR$(13):D$=D$+CHR$(13):C$=D$:GOSUB 3120:GOTO
270
6340 FOR Z2%=LE TO Z4
6350 PRINT MID$(C$,Z2%,1);:D$=D$+MID$(C$,Z2%,1)
6360 NEXT:PRINT CHR$(13):D$=D$+CHR$(13):C$=D$:GOSUB 3120:GOTO 270
6370 PRINT"\";
6380 PRINT MID$(D$,LEN(D$),1);:D$=LEFT$(D$,LEN(D$)-1)
6390 GOSUB 6500:IF Z$=CHR$(127)THEN 6380
6400 PRINT"\";:GOTO 6274
6500 WAIT 0,&O1,&O1
6510 Z2=INP(1)AND&O177:Z$=CHR$(Z2):RETURN
OK
```

621Ø Finished deletion. Print "/". Add deleted character count to pointer for original string. Put null in Z1$. Get next comma or charac-character.

622Ø D input with no numeric prefix. Print initial "/". Print deleted character. Pring final "/". Incremented original string pointer. Get next command.

623Ø L input; set up move to the end of the string.

624Ø Print all characters in the original string to end and add to edited string.

625Ø Transfer edited string to original string variable. Initialize variables to new string. Get next command.

626Ø Q input; put null in edited string. Return to calling program.

627Ø I input; get next command or character.

6272 I input; if rubout, then 637Ø.

6274 I input; if escape, then get next command.

6275 I input; if carriage, return then 633Ø.

628Ø I input; if none of above, then print character. Add to edited string. Get next character or command at 627Ø.

629Ø X input; set up loop to print remainder of the line.

63ØØ X input; print next character in original string. Add to edited string.

631Ø X input; loop to get next character. If finished, set last character to end of string. Go to 627Ø and insert mode.

632Ø H input; Make end of edited string end of string. Go to 627Ø and insert mode.

633Ø Carriage return. If at end of original string, add carriage return to edited string. Return to calling program.

634Ø Carriage return. If not at end of original string, set up loop to print remaining character.

635Ø Carriage return. Print next character in original string. Add to edited string.

636Ø Loop back for next character. If finished, print carriage return. Add carriage return to edited string. Return to calling program.

637Ø Rubout mode. Print "/".

638Ø Print last character. Delete last character from edited string.

639Ø Rubout mode. Get next character or command. If rubout, go to 637Ø.

64ØØ Rubout mode. If character input in 638Ø is not a rubout, then print "/". Return to insert mode.

65ØØ Wait for a character input from terminal &01 is octal 1.

651Ø Character received. Mask to 7 bits with octal 177. Change to single character string. Return.

END

TRACE PROGRAM Assembled Listing continued

```
ØØ326 4266 2A FC             BPL      R1
ØØ327 4268 EE ØØ             LDX      X
ØØ328 426A 6E ØØ             JMP      X          TAKE JMP
ØØ329                  *
ØØ33Ø 426C BD FF24 CKHUM     JSR      POLCAT     HUMAN WANT CONTROL?
ØØ331 426F 24 ØA             BCC      CKHUM2     NO
ØØ332 4271 BD FFØ4 CKHUM1    JSR      INCH+4
ØØ333 4274 C1 1B   CKHUM3    CMP B    #$1B       ESCAPE?
ØØ334 4276 26 Ø3             BNE      CKHUM2     NOPE
ØØ335 4278 7E 4ØØ7           JMP      DEBUG      SCRAM
ØØ336 427B 39      CKHUM2    RTS                 BACK YOU GO
ØØ337                  *
ØØ338 427C BC 43B1 EXMDR     CPX      BIADR      INST BKPNT?
ØØ339 427F 27 2E             BEQ      BKPT
ØØ34Ø 4281 B6 43AE           LDA A    TON+1
ØØ341 4284 F6 43AD           LDA B    TON
ØØ342 4287 8Ø Ø1             SUB A    #1         CRRCT FOR CARRY
ØØ343 4289 C2 ØØ             SBC B    #Ø
ØØ344 428B BØ 412C           SUB A    CKADR+1
ØØ345 428E F2 412B           SBC B    CKADR
ØØ346 4291 25 Ø6             BCS      EX2
ØØ347 4293 BC 43B3 EXMOP     CPX      BOADR      OPRND BKPNT?
ØØ348 4296 27 17             BEQ      BKPT
ØØ349 4298 39      EX1       RTS
ØØ35Ø 4299 B6 43BØ EX2       LDA A    TOFF+1
ØØ351 429C F6 43AF           LDA B    TOFF
ØØ352 429F BØ 412C           SUB A    CKADR+1
ØØ353 42A2 F2 412B           SBC B    CKADR
ØØ354 42A5 25 F1             BCS      EX1
ØØ355 42A7 86 54   EX3       LDA A    #'T
ØØ356 42A9 B7 439F           STA A    WHAT
ØØ357 42AC 7E 431A           JMP      PRNTRG     DMP & RTRN
ØØ358                  *
ØØ359 42AF 86 42   BKPT      LDA A    #'B
ØØ36Ø 42B1 7E 4Ø56           JMP      DMP1       PRINT & EXEC
ØØ361                  *
ØØ362 42B4 BE 43A4 REPAK     LDS      STKHI      REPAK STACK
ØØ363 42B7 86 Ø7             LDA A    #7
ØØ364 42B9 CE 43AC           LDX      #PCREG+1
ØØ365 42BC E6 ØØ   REP1      LDA B    X
ØØ366 42BE 37                PSH B
ØØ367 42BF Ø9                DEX
ØØ368 42CØ 4A                DEC A
ØØ369 42C1 26 F9             BNE      REP1
ØØ37Ø 42C3 FE 43AB           LDX      PCREG      ANYTHING GOING ON?
ØØ371 42C6 FF 412B           STX      CKADR
ØØ372 42C9 BD 427C           JSR      EXMDR      GO SEE
ØØ373 42CC CE                FCB      $CE        LDX #
ØØ374 42CD ØØ      HERE      FCB      Ø,Ø
ØØ375 42CF A6 ØØ             LDA A    X
ØØ376 42D1 B7 43AØ           STA A    INST
ØØ377 42D4 86 3F             LDA A    #$3F
ØØ378 42D6 A7 ØØ             STA A    X
ØØ379 42D8 FF 439B           STX      SWIADR
ØØ38Ø 42DB 3B                RTI
ØØ381                  *
ØØ382 42DC 4F      POPØ      CLR A               NO OPRND
```

continued on page 22

TRACE PROGRAM Assembled Listing continued

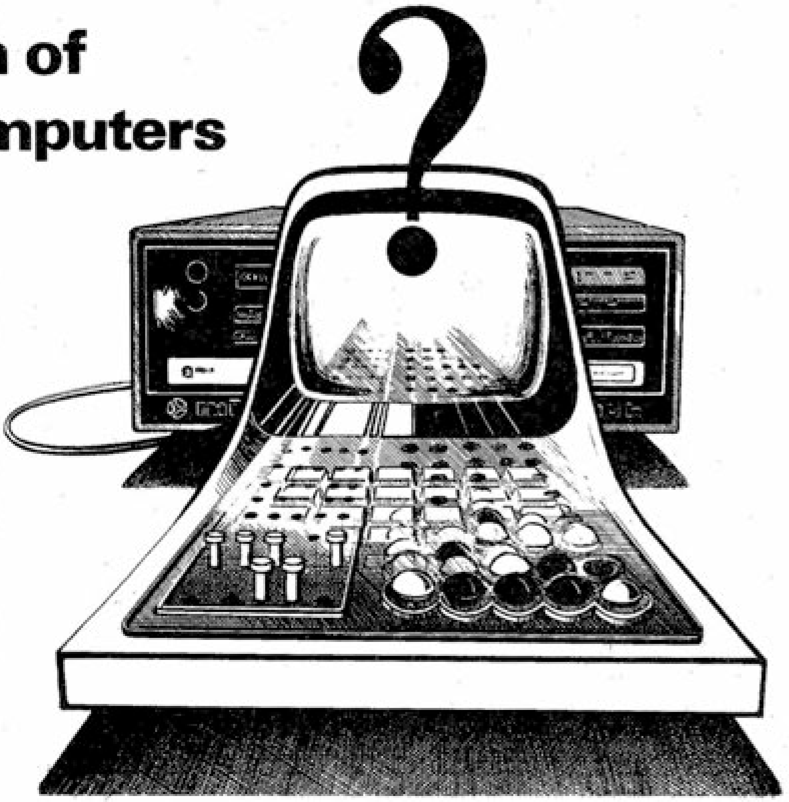
```
ØØ383 42DD B7 43A3         STA A    ASCFG
ØØ384 42EØ 39              RTS
ØØ385 42E1 86 Ø1    POP1   LDA A    #1
ØØ386 42E3 8D F8           BSR      POPØ+1
ØØ387 42E5 FE 43AB         LDX      PCREG
ØØ388 42E8 E6 Ø1           LDA B    1,X
ØØ389 42EA F7 43A1         STA B    INST+1
ØØ39Ø 42ED 39              RTS
ØØ391 42EE 86 Ø2    POP2   LDA A    #2
ØØ392 42FØ 8D F1           BSR      POP1+2
ØØ393 42F2 E6 Ø2           LDA B    2,X
ØØ394 42F4 F7 43A2         STA B    INST+2
ØØ395 42F7 39              RTS
ØØ396               *
ØØ397 42F8 8D Ø5    BAD    BSR      EON       ECHO ON
ØØ398 42FA BD FF62         JSR      BADDR     GET ADDR
ØØ399 42FD 2Ø Ø3           BRA      EOF
ØØ4ØØ               *
ØØ4Ø1 42FF 86 Ø3    EON    LDA A    #$Ø3
ØØ4Ø2 43Ø1 8C              FCB      $8C       CPX
ØØ4Ø3 43Ø2 86 FF    EOF    LDA A    #$FF
ØØ4Ø4 43Ø4 97 F3           STA A    ECHO
ØØ4Ø5 43Ø6 39              RTS
ØØ4Ø6               *
ØØ4Ø7 43Ø7 8D F6    IN     BSR      EON
ØØ4Ø8 43Ø9 BD FFØØ         JSR      INCH
ØØ4Ø9 43ØC F7 439F         STA B    WHAT
ØØ41Ø 43ØF 8D 52           BSR      PNTS
ØØ411 4311 2Ø EF           BRA      EOF
ØØ412               *
ØØ413 4313 8D EA    BY     BSR      EON
ØØ414 4315 BD FF53         JSR      BYTE
ØØ415 4318 2Ø EB           BRA      EOF
ØØ416               *
ØØ417 431A CE 438A  PRNTRG LDX      #MES4
ØØ418 431D BD 4Ø63
                           JSR      MSG
ØØ419 432Ø F6 439F         LDA B    WHAT      WHAT TYPE DMP
ØØ42Ø 4323 8D 3B           BSR      PNT1
ØØ421 4325 B6 43AØ         LDA A    INST      INST
ØØ422 4328 8D 43           BSR      OUT2
ØØ423 432A B6 43A3         LDA A    ASCFG     OPRND?
ØØ424 432D 27 14           BEQ      PRN3      NONE
ØØ425 432F B6 43A1         LDA A    INST*1
ØØ426 4332 BD FF6D         JSR      OUT2H
ØØ427 4335 B6 43A3         LDA A    ASCFG     MORE?
ØØ428 4338 4A              DEC A
ØØ429 4339 27 ØA           BEQ      PRN2      NOPE
ØØ43Ø 433B B6 43A2         LDA A    INST+2
ØØ431 433E BD FF6D         JSR      OUT2H
ØØ432 4341 2Ø Ø4           BRA      PRN1
ØØ433 4343 8D 24    PRN3   BSR      XX
ØØ434 4345 8D 22    PRN2   BSR      XX
ØØ435 4347 8D 2Ø    PRN1   BSR      XX
ØØ436 4349 CE 43A4         LDX      #STKHI
ØØ437               **********
ØØ438 434C C6              FCB      $C6       (LDA B #)
ØØ439 434D Ø9       HMNY   FCB      9
ØØ44Ø               **********
ØØ441 434E 27 ØA    PRNLP  BEQ      PRN4
ØØ442 435Ø A6 ØØ           LDA A    X
ØØ443 4352 37              PSH B
ØØ444 4353 8D 18           BSR      OUT2
ØØ445 4355 33              PUL B
ØØ446 4356 Ø8              INX
ØØ447 4357 5A              DEC B
ØØ448 4358 2Ø F4           BRA      PRNLP
ØØ449 435A 86 Ø9    PRN4   LDA A    #9        FORM RSET
ØØ45Ø 435C B7 434D         STA A    HMNY
ØØ451 435F 39              RTS
ØØ452               *
ØØ453 436Ø BD FF81  PNT1   JSR      OUTCH
ØØ454 4363 BD FF82  PNTS   JSR      OUTS
ØØ455 4366 7E 426C  PNTC   JMP      CKHUM
ØØ456               *
ØØ457 4369 8D F8    XX     BSR      PNTS
ØØ458 436B 2Ø F6           BRA      PNTS
ØØ459               *
ØØ46Ø 436D BD FF6D  OUT2   JSR      OUT2H
ØØ461 437Ø 2Ø F1           BRA      PNTS
ØØ462               *
ØØ463 4372 ØD       PRMPT  FCB      $ØD,$ØA
ØØ464 4374 FF              FCB      $FF
ØØ465 4375 4Ø              FCC      /@ /
```

```
ØØ466 4377 ØØ              FCB      Ø
ØØ467               *
ØØ468 4378 ØD       MES1   FCB      $ØD,$ØA
ØØ469 437A FF              FCB      $FF
ØØ47Ø 437B 44              FCC      /DEBUG/
ØØ471 438Ø ØØ              FCB      Ø
ØØ472               *
ØØ473 4381 2Ø       MES2   FCC      / ADDR ? /
ØØ474 4389 ØØ              FCB      Ø
ØØ475               *
ØØ476 438A ØD       MES4   FCB      $ØD,$ØA
ØØ477 438C FF              FCB      $FF,Ø
ØØ478               *
ØØ479 438E 2A       EM     FCC      /*ERROR*/
ØØ48Ø 4395 ØØ              FCB      Ø
ØØ481               *
ØØ482 4396 3FFF     MYSTK  FDB      START-1
ØØ483 4398 ØØ       STKTMP FCB      Ø,Ø
ØØ484 439A ØØ       SUBCNT FCB      Ø
ØØ485 439B ØØ       SWIADR FCB      Ø,Ø
ØØ486 439D ØØ       STKSV  FCB      Ø,Ø
ØØ487               *
ØØ488 439F ØØ       WHAT   FCB      Ø
ØØ489 43AØ 3F       INST   FCB      $3F,Ø,Ø
ØØ49Ø 43A3 ØØ       ASCFG  FCB      Ø
ØØ491 43A4 ØØ       STKHI  FCB      Ø,Ø
ØØ492 43A6 ØØ       CCREG  FCB      Ø
ØØ493 43A7 ØØ       BREG   FCB      Ø
ØØ494 43A8 ØØ       AREG   FCB      Ø
ØØ495 43A9 ØØ       XREG   FCB      Ø,Ø
ØØ496 43AB ØØ       PCREG  FCB      Ø,Ø
ØØ497 43AD FF       TON    FCB      $FF,$FF
ØØ498 43AF ØØ       TOFF   FCB      Ø,Ø
ØØ499 43B1 ØØ       BIADR  FCB      Ø,Ø
ØØ5ØØ 43B3 ØØ       BOADR  FCB      Ø,Ø
ØØ5Ø1               *
ØØ5Ø2 43B5 4D       JMPTB  FCC      /M/ MONITOR
ØØ5Ø3 43B6 4Ø6E            FDB      MONIT
ØØ5Ø4 43B8 43              FCC      /C/ CREG
ØØ5Ø5 43B9 4Ø99            FDB      STC
ØØ5Ø6 43BB 42              FCC      /B/ BREG
ØØ5Ø7 43BC 4ØA1            FDB      STB
ØØ5Ø8 43BE 41              FCC      /A/ AREG
ØØ5Ø9 43BF 4ØA9            FDB      STA
ØØ51Ø 43C1 58              FCC      /X/ XREG
ØØ511 43C2 4ØB1            FDB      STX
ØØ512 43C4 54              FCC      /T/ TRACE
ØØ513 43C5 4Ø76            FDB      TSET
ØØ514 43C7 4F              FCC      /O/ OPR BKPT
ØØ515 43C8 4Ø89            FDB      BO
ØØ516 43CA 49              FCC      /I/ INST BKPT
ØØ517 43CB 4Ø82            FDB      BI
ØØ518 43CD 4A              FCC      /J/ JMP
ØØ519 43CE 4ØB8            FDB      JMPXX
ØØ52Ø 43DØ 44              FCC      /D/ DMP REG
ØØ521 43D1 4Ø59            FDB      DMP
ØØ522 43D3 ØØ              FCB      Ø
ØØ523               *
ØØ524 43D4 4189     TABLE  FDB      INHER
ØØ525 43D6 4189            FDB      INHER
ØØ526 43D8 4212            FDB      REL
ØØ527 43DA 4189            FDB      INHER
ØØ528 43DC 4189            FDB      INHER
ØØ529 43DE 4189            FDB      INHER
ØØ53Ø 43EØ 41CE            FDB      INDX
ØØ531 43E2 4ØD5            FDB      EXT
ØØ532 43E4 4165            FDB      IMM
ØØ533 43E6 4ØC2            FDB      DIR
ØØ534 43E8 41CE            FDB      INDX
ØØ535 43EA 4ØD5            FDB      EXT
ØØ536 43EC 4165            FDB      IMM
ØØ537 43EE 4ØC2            FDB      DIR
ØØ538 43FØ 41CE            FDB      INDX
ØØ539 43F2 4ØD5            FDB      EXT
ØØ54Ø               *
ØØ541 ØØF3                 ORG      $ØØF3
ØØ542 ØØF3 Ø3              FCB      $Ø3
ØØ543               *
ØØ544                      END

TOTAL ERRORS ØØØØØ

ENTER PASS
```

TRACE PROGRAM continued on page 32

Computer Evaluates Human Logic

A Generalized Version of "Master Mind" for Computers



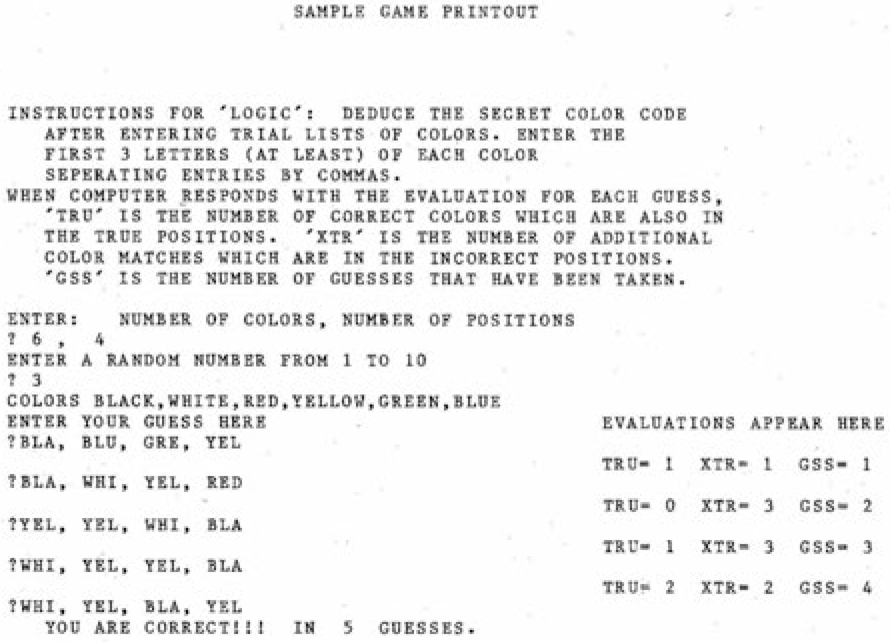
By Doyl Watson
MITS

Master Mind™ is a popular board game marketed by Invicta Plastics LTD. of Leicester England. Based on logic, it involves two players--the code maker and code breaker. Since the Altair™ microcomputer is an ideal code maker which can easily evaluate each play the code breaker makes, I've adapted Master Mind into the following computer program. Because it's more general than the board version, it's even more challenging and fun.

The object of the game is for the code breaker to guess a sequence of colors which has been preset by the code maker. Each time the code breaker tries guessing the ordered list of colors, the code maker responds with the score or evaluation for that guess. The score consists of two numbers: (1) the number of colors that have been guessed correctly and in the correct positions, and (2) the number of additional colors that have been guessed but incorrectly positioned. At the end of each round, the number of guesses taken by the code breaker is tallied and then used as a criterion for how well the player has done. For a given number of positions and colors, two code breakers can compare the number of guesses that they used to break the code.

For example, you've already requested that the computer set up a secret color code using three colors and three positions. Suppose that code is, "RED, BLACK, BLACK." (Notice that repititions are allowed.) Now suppose your first guess is, "BLACK, WHITE, BLACK". The computer would then respond with three numbers. First, the number of correct colors in the right positions =1. (BLACK in the third position of the code matches the BLACK in the third position of the guess.) The second number representing additional correct colors in the wrong places is 1. (BLACK in the second position of the code matches BLACK in the first position of the guess.)

The following program enables the computer to set up a pseudo-random color code when the code breaker enters the number of colors and the number of positions he or she is willing to guess from. (Obviously, difficulty increases with the number of colors or with the number of positions.) The code breaker also must enter a random number from 1 to 10. The computer will then ask "What is your guess." The breaker will respond with a guess, and the computer will then evaluate the guess. The game proceeds accordingly until the code breaker has built up a table of enough guesses and evaluations to deduce the color code.



```
                         SAMPLE GAME PRINTOUT


INSTRUCTIONS FOR 'LOGIC':  DEDUCE THE SECRET COLOR CODE
   AFTER ENTERING TRIAL LISTS OF COLORS. ENTER THE
   FIRST 3 LETTERS (AT LEAST) OF EACH COLOR
   SEPERATING ENTRIES BY COMMAS.
WHEN COMPUTER RESPONDS WITH THE EVALUATION FOR EACH GUESS,
   'TRU' IS THE NUMBER OF CORRECT COLORS WHICH ARE ALSO IN
   THE TRUE POSITIONS.  'XTR' IS THE NUMBER OF ADDITIONAL
   COLOR MATCHES WHICH ARE IN THE INCORRECT POSITIONS.
   'GSS' IS THE NUMBER OF GUESSES THAT HAVE BEEN TAKEN.

ENTER:   NUMBER OF COLORS, NUMBER OF POSITIONS
? 6 ,  4
ENTER A RANDOM NUMBER FROM 1 TO 10
? 3
COLORS BLACK,WHITE,RED,YELLOW,GREEN,BLUE
ENTER YOUR GUESS HERE                         EVALUATIONS APPEAR HERE
?BLA, BLU, GRE, YEL
                                              TRU= 1  XTR= 1  GSS= 1
?BLA, WHI, YEL, RED
                                              TRU= 0  XTR= 3  GSS= 2
?YEL, YEL, WHI, BLA
                                              TRU= 1  XTR= 3  GSS= 3
?WHI, YEL, YEL, BLA
                                              TRU= 2  XTR= 2  GSS= 4
?WHI, YEL, BLA, YEL
   YOU ARE CORRECT!!!  IN  5  GUESSES.
```

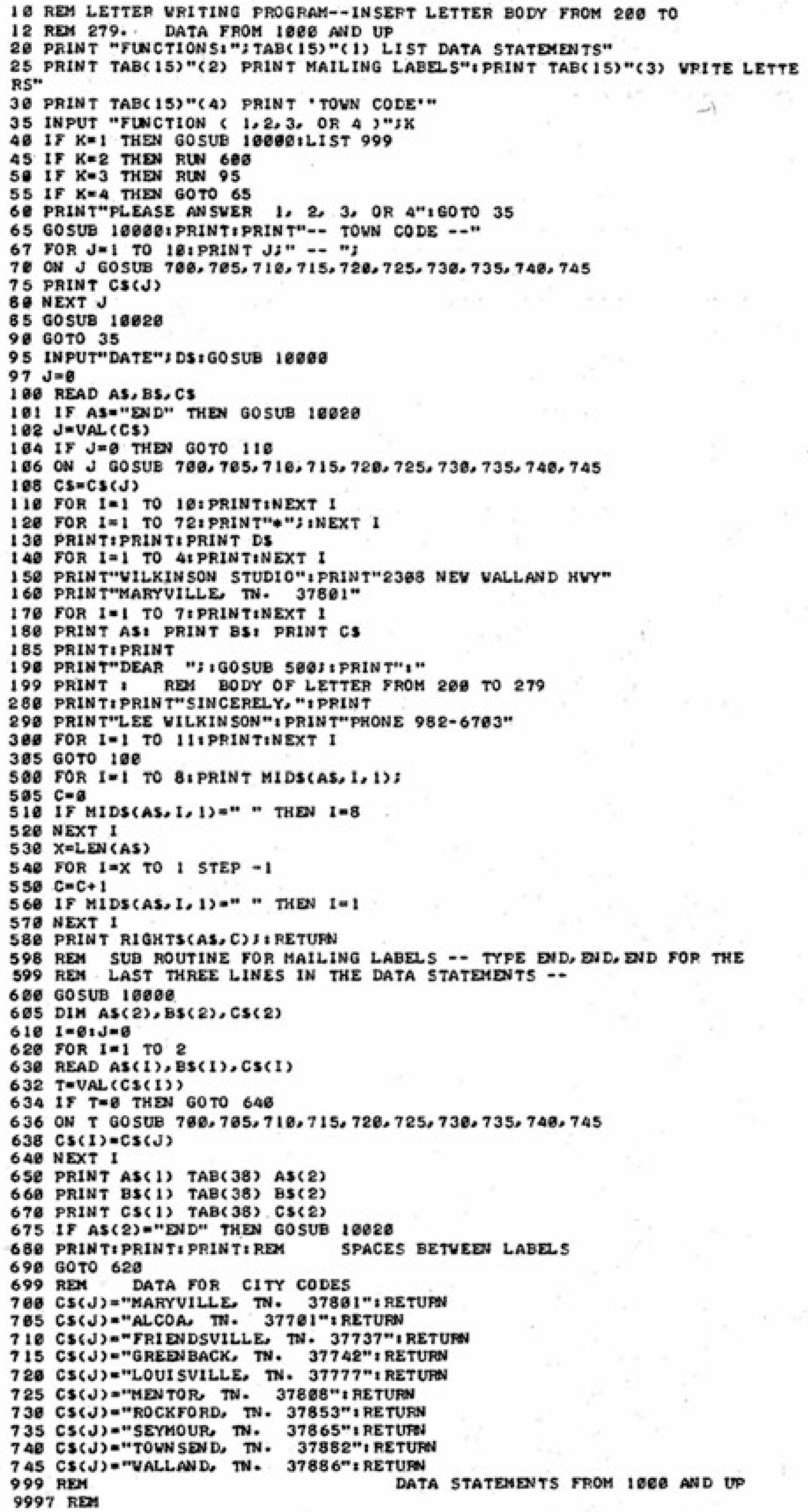
Logic
"Master Mind"

continued

Program

```
10 PRINT"INSTRUCTIONS FOR ´LOGIC´:  DEDUCE THE SECRET COLOR CODE
20 PRINT"   AFTER ENTERING TRIAL LISTS OF COLORS. ENTER THE"
30 PRINT"   FIRST 3 LETTERS (AT LEAST) OF EACH COLOR
40 PRINT"   SEPERATING ENTRIES BY COMMAS."
50 PRINT"WHEN COMPUTER RESPONDS WITH THE EVALUATION FOR EACH GUESS,"
60 PRINT"   ´TRU´ IS THE NUMBER OF CORRECT COLORS WHICH ARE ALSO IN"
70 PRINT"   THE TRUE POSITIONS.  ´XTR´ IS THE NUMBER OF ADDITIONAL"
80 PRINT"   COLOR MATCHES WHICH ARE IN THE INCORRECT POSITIONS."
90 PRINT"   ´GSS´ IS THE NUMBER OF GUESSES THAT HAVE BEEN TAKEN."
95 REM
100 REM         -MAIN PROGRAM-
110 REM
120 PRINT
130 PRINT"ENTER:   NUMBER OF COLORS, NUMBER OF POSITIONS"
140 INPUTC,N
150 IFC=1THENST$="BLACK":GOTO250
160 IFC=2THENST$="BLACK,WHITE":GOTO250
170 IFC=3THENST$="BLACK,WHITE,RED":GOTO250
180 IFC=4THENST$="BLACK,WHITE,RED,YELLOW":GOTO250
190 IFC=5THENST$="BLACK,WHITE,RED,YELLOW,GREEN":GOTO250
200 IFC=6THENST$="BLACK,WHITE,RED,YELLOW,GREEN,BLUE":GOTO250
210 IFC=7THENST$="BLACK,WHITE,RED,YELLOW,GREEN,BLUE,ORANGE":GOTO250
220 IFC=8THENST$="BLACK,WHITE,RED,YELLOW,GREEN,BLUE,ORANGE,PURPLE":GOTO250
230 IFC=9THENST$="BLACK,WHITE,RED,YELLOW,GREEN,BLUE,ORANGE,PURPLE,GOLD"
240 IFC=10THENST$="BLACK,WHITE,RED,YELLOW,GREEN,BLUE,ORANGE,PURPLE,GOLD,GRAY"
250 PRINT"ENTER A RANDOM NUMBER FROM 1 TO 10"
260 INPUTR
270 GOSUB 770:   REM   GET COLOR CODE.
280 PRINT"COLORS ";ST$
290 PRINT"ENTER YOUR GUESS HERE";TAB(48);"EVALUATIONS APPEAR HERE"
300 FORJJ=1TON
310 CC$(JJ)=M$(C,1+ABS(JJ-R))  :REM   CODE GENERATOR
320 NEXTJJ
330 REM   GUESSES ENTERED HORIZONTALLY.. SEPERATED BY COMMAS.
340 IFN=1THENINPUTG$(1):GOTO440
350 IFN=2THENINPUTG$(1),G$(2):GOTO440
360 IFN=3THENINPUTG$(1),G$(2),G$(3):GOTO440
370 IFN=4THENINPUTG$(1),G$(2),G$(3),G$(4):GOTO440
380 IFN=5THENINPUTG$(1),G$(2),G$(3),G$(4),G$(5):GOTO440
390 IFN=6THENINPUTG$(1),G$(2),G$(3),G$(4),G$(5),G$(6):GOTO440
400 IFN=7THENINPUTG$(1),G$(2),G$(3),G$(4),G$(5),G$(6),G$(7):GOTO440
410 IFN=8THENINPUTG$(1),G$(2),G$(3),G$(4),G$(5),G$(6),G$(7),G$(8):GOTO440
420 IFN=9THENINPUTG$(1),G$(2),G$(3),G$(4),G$(5),G$(6),G$(7),G$(8),G$(9)
430 IFN=10THENINPUTG$(1),G$(2),G$(3),G$(4),G$(5),G$(6),G$(7),G$(8),G$(9),G$(10)
440 GOSUB530  :REM    MAKE EVALUATION OF THE GUESS.
450 IFB=NGOTO480:   REM  GUESS IS CORRECT.
460 PRINTTAB(48);"TRU=";B;" XTR=";W;" GSS=";T
470 GOTO300
480 PRINT"   YOU ARE CORRECT!!!  IN ";T;" GUESSES."
490 END
500 REM
510 REM         -GUESS EVALUATION-
520 REM
530 B=0:W=0
540 FORK=1TON
550 REM   FIRST 3 LETTERS OF GUESS COMPARED TO FIRST 3 OF ANSWER.
560 IFCC$(K)<>LEFT$(G$(K),3)THENGOTO620
570 B=B+1
580 REM   POSITIONS ALREADY MATCHED ARE MADE UNIQUE SO THAT-
590 REM    NO ENTRY IS TALLIED TWICE.
600 CC$(K)=CHR$(K+11)
610 G$(K)=CHR$(K+22)
620 NEXTK
630 FORK=1TON
640 FORJ=1TON
650 IFCC$(K)<>LEFT$(G$(J),3)THENGOTO700
660 W=W+1
670 CC$(K)=CHR$(K+11)
680 G$(J)=CHR$(K+22)
690 J=N
700 NEXTJ:NEXTK
710 T=T+1
720 RETURN
730 REM
740 REM         -RANDOM DATA-
750 REM
760 REM   DATA SHOULD BE CHANGED OCCASIONALLY.
770 FORP=1TO10
780 FORQ=1TO10
790 READM$(P,Q)
800 NEXTQ:NEXTP
810 DATABLA,BLA,BLA,BLA,BLA,BLA,BLA,BLA,BLA,BLA
820 DATAWHI,BLA,WHI,BLA,WHI,BLA,BLA,WHI,WHI,BLA
830 DATARED,BLA,RED,WHI,RED,BLA,BLA,WHI,RED,RED
840 DATABLA,RED,BLA,RED,YEL,YEL,WHI,WHI,RED,WHI
850 DATAGRE,YEL,YEL,BLA,RED,WHI,BLA,RED,RED,YEL
860 DATABLA,YEL,WHI,RED,GRE,BLU,GRE,BLA,BLU,BLU
870 DATAORA,YEL,GRE,RED,WHI,BLA,BLA,ORA,RED,YEL
880 DATABLU,BLU,BLU,GRE,ORA,RED,WHI,PUR,RED,BLU
890 DATAYEL,GRE,PUR,ORA,BLA,GOL,WHI,GRE,BLU,WHI
900 DATAGOL,GRA,RED,YEL,PUR,ORA,BLA,GRE,RED,GOL
910 RETURN
```

Letter Writing Program Solves Photographers Mailing Problems



```
10 REM LETTER WRITING PROGRAM--INSERT LETTER BODY FROM 200 TO
12 REM 279.   DATA FROM 1000 AND UP
20 PRINT "FUNCTIONS:";TAB(15)"(1) LIST DATA STATEMENTS"
25 PRINT TAB(15)"(2) PRINT MAILING LABELS":PRINT TAB(15)"(3) WRITE LETTE
RS"
30 PRINT TAB(15)"(4) PRINT 'TOWN CODE'"
35 INPUT "FUNCTION ( 1,2,3, OR 4 )";K
40 IF K=1 THEN GOSUB 10000:LIST 999
45 IF K=2 THEN RUN 600
50 IF K=3 THEN RUN 95
55 IF K=4 THEN GOTO 65
60 PRINT"PLEASE ANSWER  1, 2, 3, OR 4":GOTO 35
65 GOSUB 10000:PRINT:PRINT"-- TOWN CODE --"
67 FOR J=1 TO 10:PRINT J;" -- ";
70 ON J GOSUB 700,705,710,715,720,725,730,735,740,745
75 PRINT C$(J)
80 NEXT J
85 GOSUB 10020
90 GOTO 35
95 INPUT"DATE";D$:GOSUB 10000
97 J=0
100 READ A$,B$,C$
101 IF A$="END" THEN GOSUB 10020
102 J=VAL(C$)
104 IF J=0 THEN GOTO 110
106 ON J GOSUB 700,705,710,715,720,725,730,735,740,745
108 C$=C$(J)
110 FOR I=1 TO 10:PRINT:NEXT I
120 FOR I=1 TO 72:PRINT"*";:NEXT I
130 PRINT:PRINT:PRINT D$
140 FOR I=1 TO 4:PRINT:NEXT I
150 PRINT"WILKINSON STUDIO":PRINT"2308 NEW WALLAND HWY"
160 PRINT"MARYVILLE, TN.  37801"
170 FOR I=1 TO 7:PRINT:NEXT I
180 PRINT A$: PRINT B$: PRINT C$
185 PRINT:PRINT
190 PRINT"DEAR  ";:GOSUB 500;:PRINT":"
199 PRINT :   REM  BODY OF LETTER FROM 200 TO 279
280 PRINT:PRINT"SINCERELY,":PRINT
290 PRINT"LEE WILKINSON":PRINT"PHONE 982-6703"
300 FOR I=1 TO 11:PRINT:NEXT I
305 GOTO 100
500 FOR I=1 TO 8:PRINT MID$(A$,I,1);
505 C=0
510 IF MID$(A$,I,1)=" " THEN I=8
520 NEXT I
530 X=LEN(A$)
540 FOR I=X TO 1 STEP -1
550 C=C+1
560 IF MID$(A$,I,1)=" " THEN I=1
570 NEXT I
580 PRINT RIGHT$(A$,C);:RETURN
598 REM  SUB ROUTINE FOR MAILING LABELS -- TYPE END,END,END FOR THE
 599 REM  LAST THREE LINES IN THE DATA STATEMENTS --
600 GOSUB 10000
605 DIM A$(2),B$(2),C$(2)
610 I=0:J=0
620 FOR I=1 TO 2
630 READ A$(I),B$(I),C$(I)
632 T=VAL(C$(I))
634 IF T=0 THEN GOTO 640
636 ON T GOSUB 700,705,710,715,720,725,730,735,740,745
638 C$(I)=C$(J)
640 NEXT I
650 PRINT A$(1) TAB(38) A$(2)
660 PRINT B$(1) TAB(38) B$(2)
670 PRINT C$(1) TAB(38) C$(2)
675 IF A$(2)="END" THEN GOSUB 10020
680 PRINT:PRINT:PRINT:REM     SPACES BETWEEN LABELS
690 GOTO 620
699 REM    DATA FOR  CITY CODES
700 C$(J)="MARYVILLE, TN.  37801":RETURN
705 C$(J)="ALCOA, TN.  37701":RETURN
710 C$(J)="FRIENDSVILLE, TN. 37737":RETURN
715 C$(J)="GREENBACK, TN.  37742":RETURN
720 C$(J)="LOUISVILLE, TN. 37777":RETURN
725 C$(J)="MENTOR, TN.  37808":RETURN
730 C$(J)="ROCKFORD, TN. 37853":RETURN
735 C$(J)="SEYMOUR, TN.  37865":RETURN
740 C$(J)="TOWNSEND, TN.  37882":RETURN
745 C$(J)="WALLAND, TN.  37886":RETURN
999 REM                          DATA STATEMENTS FROM 1000 AND UP
 9997 REM
```

continued on page 26

Letter Writing Program Solves Photographer's Mailing Problems

continued

```
9998 REM
9999 REM                              SUB-ROUTINES FOR HARD COPY *****
10000 INPUT"WANT HARD COPY";H$
10005 IF LEFT$(H$,1)<>"Y" THEN RETURN
10006 PRINT"TURN ON PRINTER -- PRESS SPACE BAR":WAIT 0,1,1
10010 POKE1352,20:POKE1360,21:POKE1367,20:POKE1374,21:RETURN
10020 POKE1352,0:POKE1360,1:POKE1367,0:POKE1374,1:RETURN
OK
```

Sample Letter

```
***************************************************************

OCTOBER 1 1977



WILKINSON STUDIO
2308 NEW WALLAND HWY
MARYVILLE, TN.  37801







MRS. GEORGE JONES
123 ANYSTREET
MARYVILLE, TN.  37801


DEAR  MRS.  JONES:

          *****  HAPPY BIRTHDAY TO BABY  *****

TO HELP CELEBRATE BABY'S BIRTHDAY WE HAVE A SPECIAL OFFER
FOR YOUR FAMILY.

                    ** 6 MONTH BIRTHDAY SPECIAL  **

1 - 8 X 10 COLOR PORTRAIT FOR YOURSELVES
2 - 5 X 7 COLOR PORTRAITS FOR GRANDPARENTS

ALL FOR ONLY $19.95 *****

AND MRS.  JONES, IF YOU'LL CALL US WITHIN 3 DAYS OF RECEIPT
OF THIS LETTER WE WILL INCLUDE WITH YOUR BIRTHDAY SPECIAL
PACKAGE, ABSOLUTELY FREE, 8 COLOR WALLETS.

REMEMBER MRS.  JONES, TIME FLIES SO CALL US TODAY !

SINCERELY,

LEE WILKINSON
PHONE 982-6703
```

Sample Listing

```
LIST 199

199 PRINT :   REM  BODY OF LETTER FROM 200 TO 279
200 PRINT"          *****  HAPPY BIRTHDAY TO BABY  *****"
210 PRINT:PRINT"TO HELP CELEBRATE BABY'S BIRTHDAY WE HAVE A SPECIAL OFFE
R"
220 PRINT"FOR YOUR FAMILY.":PRINT
230 PRINTTAB(20)"** 6 MONTH BIRTHDAY SPECIAL  **":PRINT
235 PRINT"1 - 8 X 10 COLOR PORTRAIT FOR YOURSELVES"
240 PRINT"2 - 5 X 7 COLOR PORTRAITS FOR GRANDPARENTS":PRINT
245 PRINT"ALL FOR ONLY $19.95 *****":PRINT
250 PRINT"AND ";:GOSUB 500:PRINT", IF YOU'LL CALL US WITHIN 3 DAYS OF RE
CEIPT"
255 PRINT"OF THIS LETTER WE WILL INCLUDE WITH YOUR BIRTHDAY SPECIAL"
260 PRINT"PACKAGE, ABSOLUTELY FREE, 8 COLOR WALLETS."
265 PRINT:PRINT"REMEMBER ";:GOSUB 500:PRINT", TIME FLIES SO CALL US TODA
Y !"
280 PRINT:PRINT"SINCERELY,":PRINT
290 PRINT"LEE WILKINSON":PRINT"PHONE 982-6703"
```

AUDIOSYNCRACIES

Unique Audio Processing Applications of the 88-AD/DA

By Thomas G. Schneider
MITS

AUDIOSYNCRACIES is a three-part series devoted to exploring unconventional applications of the Altair 88-AD/DA board. Hardware and software theory and implementation of the board in the Altair 8800 series mocrocomputers will be covered.

Part I includes: Theory of the audio delay line, a simple audio delay line for producing echo effects, and a description of interface circuitry for this and subsequent audio application articles.

Audio signal processing is one of the more fascinating applications of the Altair 88-AD/DA board. This board's high speed of analog to digital conversion makes it particularly suitable for good quality digitalization of audio information.

One especially interesting application if the creation of audio delays using the 88-AD/DA board. By taking an audio signal, delaying it, and then recombining it with the original signal, a variety of interesting echo and reverberation effects can be produced. In the past, echo effects were produced by a tape loop. A diagram of this method is shown in Figure 1. The audio signal is recorded onto the magnetic tape loop by the record head and then played back off the tape by the multiple playback heads. The distance between the record and playback heads determines the amount of time that passes until an echo is heard. The number of echos that are heard is determined by how many playback heads the tape passes over after it passes the record head. There is a disadvantage to this method: it requires a tape transport, and magnetic tape is one of those mediums that deteriorates with age.

In this first article, we will explore the advantages of using the 88-AD/DA and the Altair computer to implement a solid-state no-moving-parts system which will perform this echo function in addition to producing several other interesting effects.

SOFTWARE

The method for producing the echo effect is shown in flowchart form in Figure 2. After briefly studying the flowchart, you will notice that we are essentially imitating the tape loop echo method, but the medium is the memory of the computer, and the "record" and "playback" head functions are implemented in software. The "record" function is accomplished by using pointer HL to write the digitalized audio information into memory. The "playback" function is accomplished by using pointer DE to retrieve the information from memory. Both pointers are simultaneously stepped through memory, but pointer DE runs behind pointer HL. The time it takes for pointer DE to reach and read data from the same point in memory that pointer HL has written data into, determines the delay time until the echo of the original signal is heard. As each pointer reaches the top limit of memory, it is reset back to the beginning, giving us a continually running loop. The amount of time that passes until the echo of the original signal is heard is determined by the difference in starting points of pointers HL and DE. The offset can be any value you choose, so a wide variety of delay times are possible. The maximum amount of delay is, of course, limited by the amount of memory in the computer. To obtain the maximum delay time, set pointer HL to the middle of the memory space and set pointer DE to the beginning of the memory space. For this first experiment, we will produce only one echo. The machine code program for our delay function is shown in Listing 1.

HARDWARE

To properly interface the 88-AD/DA with real world audio signals, you need to construct one relatively simple circuit. (See Figure 3.) The top half of this circuit takes a real world audio signal and shifts it into the voltage range acceptable by the 88-AD/DA's input. The voltage at the input of the 88-AD/DA must not be lower than ground and higher than 10 volts. Since audio signals usually go both above and below ground, the input conditioning circuit shifts the entire audio signal upwards so that all signals are above ground and below 10 volts. The two diodes at the output of the circuit ensurethat the signal reaching the 88-AD/DA doesn't exceed the 0-10 volt range. The OP-AMP in this circuit can be just about any general purpose OP-AMP, like the 741, for example. The bottom half of the circuit in Figure 3 is used to mix the output of D/A convertor and the original input signal before these signals go out to the real world.

To adjust this interfacing circuitry, use the following procedure. Adjust the original signal gain pot and the delay gain pot to their positions of highest resistance. Adjust the input signal gain pot to its position of least resistance. With no input signal applied, adjust the offset pot so that 5 volts appears at the output of the OP--AMP. Apply an audio signal typical of what you will be running into the system and adjust the input signal gain pot so that the voltage at the output of the OP-AMP swings no more than about seven volts peak-to-peak. After toggling in the program, hit run and adjust the output mixing pots to obtain a pleasant mix of the original and delayed audio signals.

Referring again to the software, you can easily change the delay time by increasing or decreasing the starting address of the HL register. To run this software in your Altair computer, it may be necessary to change a few things in the program, depending on how much memory is available. The contents of the following addresses are important:

41 and 42 contain the starting address of the write pointer.

44 and 45 contain the starting address of the read pointer.

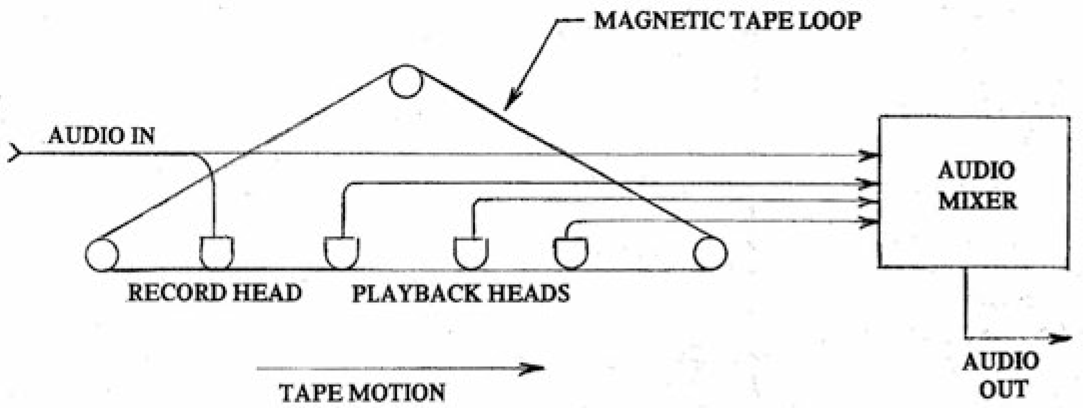
53 and 64 contain the most significant byte of the highest memory address used as storage space.

When modifying this program to suit your memory size, be careful not to write over the program. One thing to remember about audio modification programs...don't be afraid to modify the program itself. You may be surprised with some bizarre and unusual results!

Next month, AUDIOSYNCRACIES will cover a more flexible software routine for the audio delay line and interface circuitry modifications for producing continuously recirculating echo effects.

continued on page 28

FIGURE 1



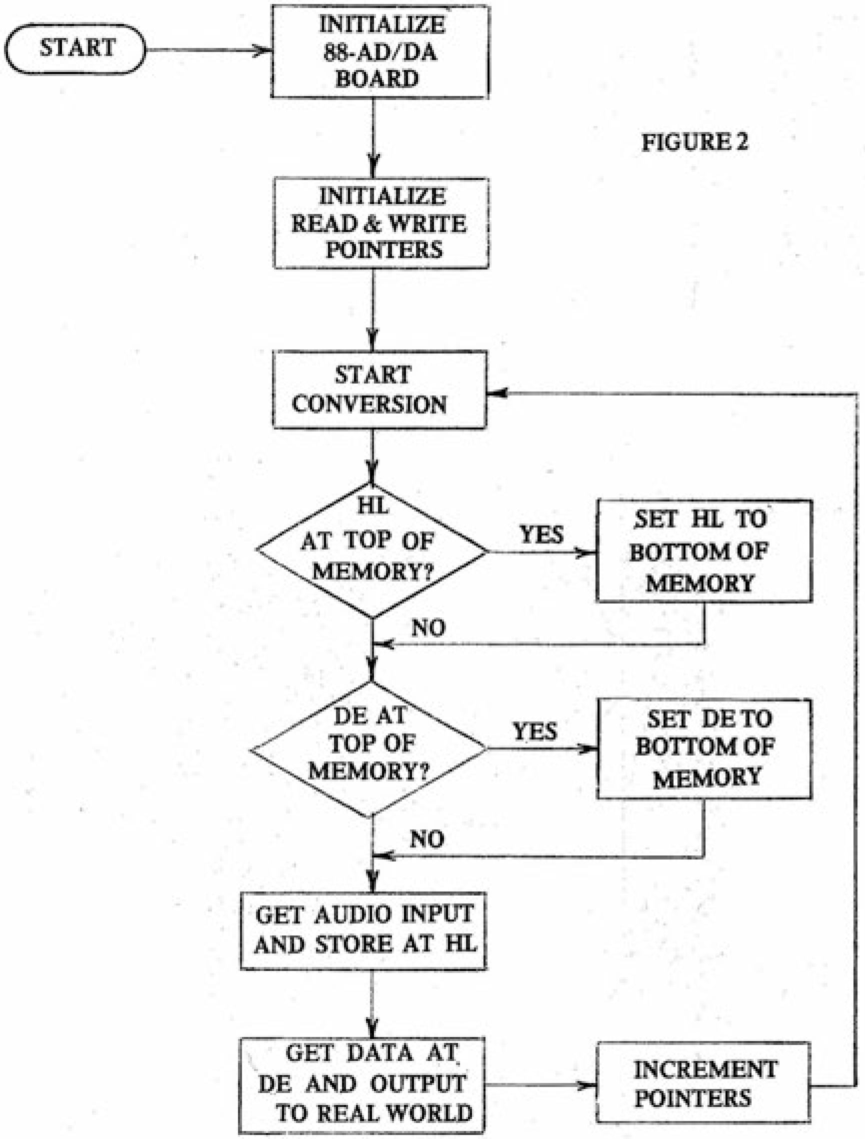
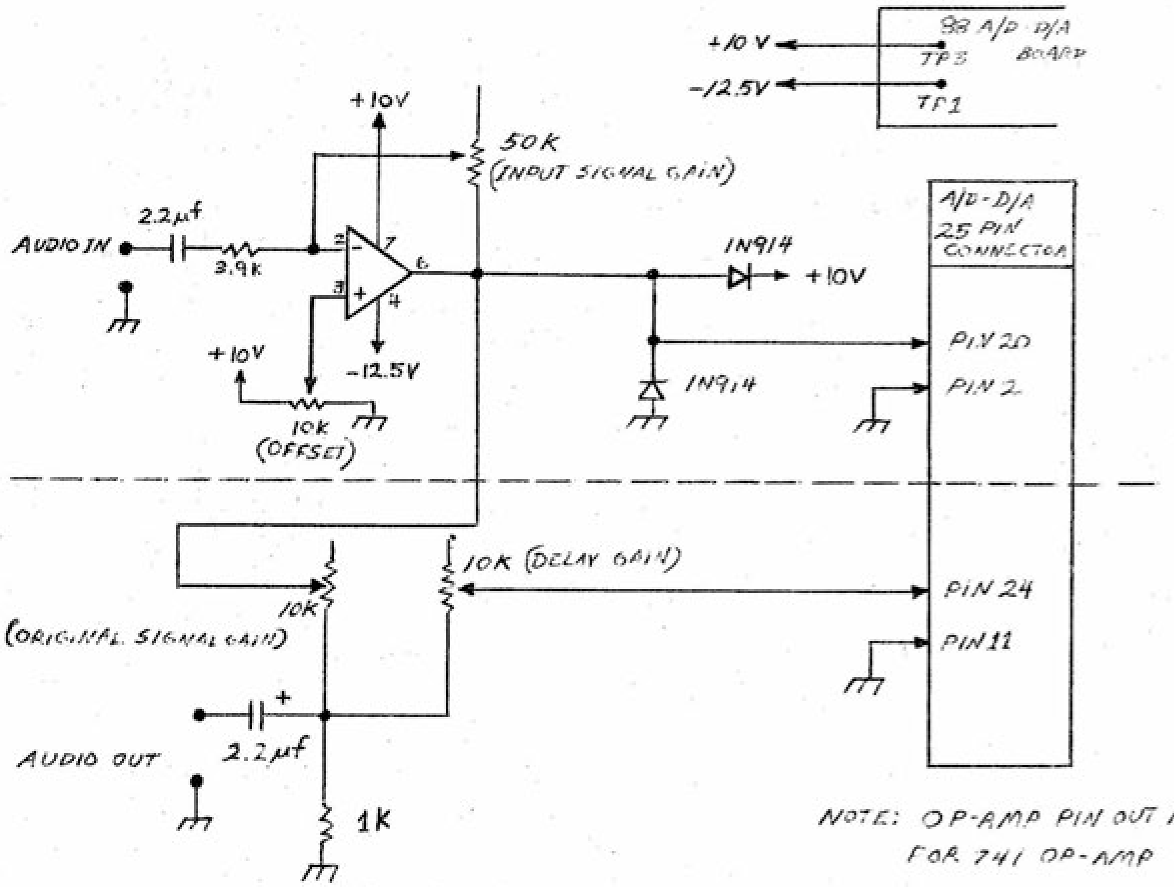

START
INITIALIZE 88-AD/DA BOARD
INITIALIZE READ & WRITE POINTERS
START CONVERSION
HL AT TOP OF MEMORY?
YES
SET HL TO BOTTOM OF MEMORY
NO
DE AT TOP OF MEMORY?
YES
SET DE TO BOTTOM OF MEMORY
NO
GET AUDIO INPUT AND STORE AT HL
GET DATA AT DE AND OUTPUT TO REAL WORLD
INCREMENT POINTERS

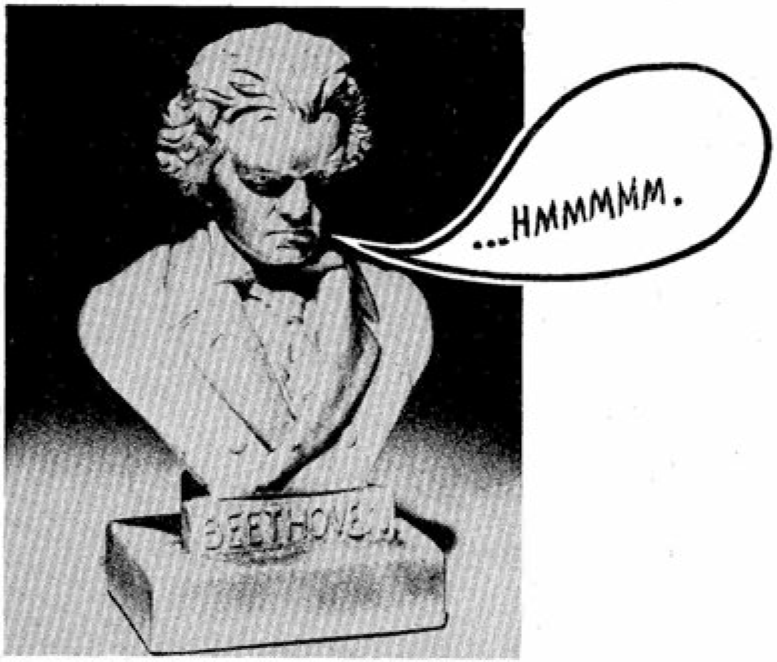
FIGURE 2



NOTE: OP-AMP PIN OUT IS FOR 741 OP-AMP

FIGURE 3

continued on page 30



```
AUDIO DELAY SOFTWARE ( ASSUMES A/D-D/A BOARD IS AT OCTAL ADDRESS 100 )


 0    257    INIT.  XRA A            PROGRAM LINES 0 - 33 INITIALIZE
 1    323           OUT 100          THE A/D-D/A BOARD
 2    100
 3    323           OUT 101
 4    101
 5    323           OUT 102
 6    102
 7    323           OUT 104
10    104
11    323           OUT 106
12    106
13    057           CMA
14    323           OUT 103
15    103
16    323           OUT 105
17    105
20    323           OUT 107
21    107
22    076           MOV A,054
23    054
24    323           OUT 100
25    100
26    323           OUT 102
27    102
30    323           OUT 104
31    104
32    323           OUT 106
33    106
34    000           NOP
35    000           NOP
36    000           NOP
37    000           NOP
40    041    START, LXI H,020/000          LOAD HL WITH WRITE
41    000                                  POINTER STARTING ADDRESS
42    020
43    021           LXI D,001/000          LOAD DE WITH READ
44    000                                  POINTER STARTING ADDRESS
```

continued

```
 45   001
 46   257   CONV,   XRA A           OUTPUT A 0 TO PORT 103
 47   323           OUT 103         TO START CONVERSION
 50   103
 51   174   CHKH,   MOV A,H         SEE IF HL POINTER HAS
 52   376           CPI 200         REACHED THE TOP OF
 53   200                           MEMORY SPACE
 54   302           JNZ CHKD        IF NOT, CHECK THE DE
 48   062                           POINTER
 56   000
 57   076           MVI A,001       LOAD H WITH 1
 60   001
 61   147           MOV H,A
 62   172   CHKD,   MOV A,D         SEE IF DE POINTER
 63   376           CPI 200         REACHED THE TOP OF
 64   200                           MEMORY SPACE
 65   302           JNZ INPT        IF NOT, GET AUDIO INPUT
 66   073
 67   000
 70   076           MVI A,001       PUT 001 IN D
 71   001
 72   127           MOV D,A
 73   333   INPT,   INP 101         GET AUDIO INPUT FROM A/D
 74   101
 75   167           MOV M,A         AND MOVE IT TO MEMORY
 76   353           XCHG            SWAP POINTERS HL & DE
 77   176           MOV A,M         GET DATA FROM MEMORY
100   323           OUT 105         AND OUTPUT IT TO D/A
101   105
102   353           XCHG            SWAP POINTERS BACK
103   043           INX H           INCREMENT HL POINTER
104   023           INX D           INCREMENT DE POINTER
105   303           JMP CONV
106   000
107   000
```

PROGRAM USED TO DEMONSTRATE SAMPLE RUN

```
ØØØØ1                            NAM       SHOWEM
ØØØØ2                            OPT       NOG,M
ØØØØ3 3ØØØ                       ORG       $3ØØØ
ØØØØ4                   *
ØØØØ5                   *SHOWEM - A SAMPLE PROGRAM
ØØØØ6                   *TO SHOW RUNNING FEATURES OF DEBUG
ØØØØ7                   *
ØØØØ8 3ØØØ CE 3ØØE XX      LDX       #TABLE
ØØØØ9 3ØØ3 A6 ØØ   ZZ      LDA A     Ø,X
ØØØ1Ø 3ØØ5 27 FE           BEQ       *
ØØØ11 3ØØ7 BD 3ØØC         JSR       YY
ØØØ12 3ØØA 2Ø F7           BRA       ZZ
ØØØ13                   *
ØØØ14 3ØØC Ø8      YY      INX
ØØØ15 3ØØD 39              RTS
ØØØ16                   *
ØØØ17 3ØØE 41      TABLE   FCC       /ABC/
ØØØ18 3Ø11 ØØ              FCB       Ø
ØØØ19                      END

TOTAL ERRORS ØØØØØ

ENTER PASS X
```

SAMPLE RUN OF DEBUG PROGRAM

```
 J 4ØØØ
DEBUG
@ T   ADDR ? 3ØØØ ADDR ? 3Ø11
@ D
D 3F        ØØ F1 DØ ØØ ØØ ØØ ØØ ØØ ØØ 3Ø ØØ 3Ø 11 ØØ ØØ ØØ ØØ
@
  J   ADDR ? 3ØØC
T Ø8        ØØ F1 DØ ØØ ØØ ØØ ØØ 3Ø ØC
X 39        ØØ F1 DØ ØØ ØØ ØØ Ø1 3Ø ØD 3Ø ØØ 3Ø 11 ØØ ØØ ØØ ØØ
@ J   ADDR ? 3ØØØ
T CE 3ØØE   ØØ F1 DØ ØØ ØØ ØØ Ø1 3Ø ØØ
T A6 ØØ     ØØ F1 DØ ØØ ØØ 3Ø ØE 3Ø Ø3
T 27 FE     ØØ F1 DØ ØØ 41 3Ø ØE 3Ø Ø5
T BD 3ØØC   ØØ F1 DØ ØØ 41 3Ø ØE 3Ø ØC
T Ø8        ØØ F1 DØ ØØ 41 3Ø ØE 3Ø ØC
T 39        ØØ F1 DØ ØØ 41 3Ø ØF 3Ø ØA
T 2Ø F7     ØØ F1 DØ ØØ 41 3Ø ØF 3Ø Ø3
T A6 ØØ     ØØ F1 DØ ØØ 41 3Ø ØF 3Ø Ø3
T 27 FE     ØØ F1 DØ ØØ 42 3Ø ØF 3Ø Ø5
T BD 3ØØC   ØØ F1 DØ ØØ 42 3Ø ØF 3Ø ØC
T Ø8        ØØ F1 DØ ØØ 42 3Ø ØF 3Ø ØC
T 39        ØØ F1 DØ ØØ 42 3Ø 1Ø 3Ø ØA
T 2Ø F7     ØØ F1 DØ ØØ 42 3Ø 1Ø 3Ø Ø3
T A6 ØØ     ØØ F1 DØ ØØ 42 3Ø 1Ø 3Ø Ø3
T 27 FE     ØØ F1 DØ ØØ 43 3Ø 1Ø 3Ø Ø5
T BD 3ØØC   ØØ F1 DØ ØØ 43 3Ø 1Ø 3Ø ØC
T Ø8        ØØ F1 DØ ØØ 43 3Ø 1Ø 3Ø ØC
T 39        ØØ F1 DØ ØØ 43 3Ø 11 3Ø ØA
T 2Ø F7     ØØ F1 DØ ØØ 43 3Ø 11 3Ø Ø3
T A6 ØØ     ØØ F1 DØ ØØ 43 3Ø 11 3Ø Ø3
T 27 FE     ØØ F1 D4 ØØ ØØ 3Ø 11 3Ø Ø5
T 27 FE     ØØ F1 D4 ØØ ØØ 3Ø 11 3Ø Ø5
T 27 FE     ØØ F1 D4 ØØ ØØ 3Ø 11 3Ø Ø5
T 27 FE     ØØ F1 D4 ØØ ØØ 3Ø 11 3Ø Ø5
T 27 FE     ØØ F1 D4 ØØ ØØ 3Ø 11 3Ø Ø5
T 27 FE     ØØ F1 D4
DEBUG
@ C 77
@ B 88
@ A 99
@ X AAAA
@ I   ADDR ? BBBB
@ O   ADDR ? CCCC
@ D
D 27 FE     ØØ F1 77 88 99 AA AA 3Ø Ø5 3Ø ØØ 3Ø 11 BB BB CC CC
@ M
 .
 .
```

A Definition of Terms:

sub-scribe /ˌsəb-ˈscrib/ *vb* **sub-scribed; sub-scrib-ing** [**ME** *subscriber*] **1:** to sign one's name to a document (as a coupon; as the one below) **2:** to enter one's name for a publication (as **CN**-Computer Notes; one year for **$5.00/$20.00** per year overseas) **3:** to feel favorably disposed **syn** ASSENT **ant** boggle—**sub-scrib-er** *n*

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 MC or BAC/Visa #________________
□ Master Charge Exp Date ________
□ BankAmericard/Visa Signature________________