Software Contest Winners Announced

by Bill Gates

We started receiving programs for the Altair Users Library a few months ago, and we are getting more and more every day. As the Users Group grows and users become more sophisticated at programming their Altairs, we expect the library to grow at a fast rate and become a valuable resource for Altair users.

Though it is not a prize winner, the most aesthetic entry was a memory clear program (#63751) from Ronald Keele that is given below.

```
0    PUSH H     345
1    PCHL       351
2    LXI  H     041
3    0          000
4    0          000
5    SPHL       371
6    PCHL       351
```

Load and start at location 2 with memory unprotected. Every location, including 0-6, will be zeroed and then the CPU will cycle through memory executing NOP's (0). Be sure to stop it after this or executing the 377's (RST 7) in nonexistent memory will eventually move the stack pointer into good memory and garbage memory.

The best demo programs I've seen for the Altair are the ones that control the signals on the bus to give musical output. Steven Dompier has an article about the music program that he wrote for the Altair in the People's Computer Company publication. The article gives a listing of his program and the musical data for "The Fool on the Hill" and "Daisy." All that is required is an AM radio that sits near your Altair. His article gives an explanation of how to output different notes, spaces and rests. He doesn't explain why it works and I don't see why. Does anyone know?

Another music program (#62751) was submitted to the library by Roger Smith. It involves connecting one bit of an output port bit to an amplifier through a lmF capacitor, and then pulsing the bit repetitively to generate an audio tone.

Once you've assembled a program and loaded it into core, it isn't easy to relocate the program to another place in core. If you just move the code the CALL, JMP and data addresses will still refer to the old locations. Randolph Wilhoit has submitted a utility program RLCT (#611751) that can be used to move a program from one location to another. It treats everything within a certain range as an instruction and modifies anything that is an address. Most often it will be easier to reassemble the source with a new starting location, but when this is not possible or you want to move something simple like a binary loader, this program will be handy. As given RLCT occupies locations 0-344, but instructions are given for relocating RLCT using itself.

SUBROUTINES

The winning subroutine ($25), DBUG #516753, submitted by Harold Corbin, is a very useful one. By inserting a call to his DBUG routine in a program you can find out exactly what the state of the machine was at that point in your program. His program stores all the registers, the SP and the top thing on the stack in fixed memory locations and then loops. By stopping the machine and starting it at another location you can get DBUG to restore the SP and all of your registers and continue your program.

The other winning subroutine ($15), #527751, is a binary log approximation routine by Randall Webb. He has also submitted a program that estimates the standard deviation of a list of numbers.

MAJOR PROGRAM WINNERS
--see page three--

COMPUTER NOTES

July, 1975

© MITS, INC. 1975

A PUBLICATION OF THE ALTAIR USERS GROUP   VOLUME ONE ISSUE TWO

ALTAIR... ON THE ROAD

The MITS-MOBILE is a camper van equipped with an Altair BASIC language system. Included is an Altair 8800, Comter 256 computer terminal, ACR-33 teletype, Altair Line Printer, Altair Floppy Disk and BASIC language.

We launched this vehicle (our marketing people refer to it as "a unique marketing tool") in late April with a test swing through Texas. The response was so great that we took the MITS-MOBILE to the National Computer Conference in Anaheim (May 19-22) and spent the next three weeks in California where we were drawing crowds of 200 plus for our nightly seminars.

Now that we are certain that the MITS-MOBILE is viable, we have decided to upgrade our seminars and tour the entire United States bringing the message of low-cost computing to thousands of people.

Our Computer Seminar will include a seminar on the Altair, BASIC language and computers in general along with a slide presentation. There will also be a question and answer session and hands-on demonstrations as time permits. Each participant will receive a three ring binder packed full of information including schematics and our new 24 page catalog. There will be refreshments and a door prize at each session. (Seminars will be held at local motels and hotels such as Holiday Inns.) Cost of the seminars will be $9.75.

During the months of July-August, the MITS-MOBILE will be in the Southeast. Our tentative schedule calls for stops in Amarillo, Oklahoma City, Tulsa, Little Rock, Shreveport, Baton Rouge, New Orleans, Mobile, Jackson, Memphis, Nashville, Knoxville, Chattanooga, Atlanta, Huntsville, Birmingham, Montgomery, Tallahassee, Tampa, Miami, Orlando, Cape Canaveral, Jacksonville, Macon, Augusta, Columbia, Charlotte, Greensboro, Winston Salem, Raleigh.

We want to be flexible with this schedule so that if you are able to draw 75 people in your home town, we'll add you to the list. We'll even go to Hereford, Colorado, if the interest is there.

Following our tour through the Southeast, the MITS-MOBILE will travel up to the Northeast and then the Midwest. Everyone on our mailing list will be notified of definite times and dates.

Across the Editor's Desk

By David Bunnell

From our point of view, it looks as if the computer hobby market is going to boom as never before. The message I get from the people at People's Computer Company, The Computer Hobbyist and Creative Computing, is that subscriptions to their publications are increasing at an ever accelerating pace.

Another computer publication about to make the scene is BYTE, which is being published by the same people who publish 73, the ham radio magazine. The first issue is scheduled to come out in the middle of August with a circulation of 35,000. And, by the way, it will feature the Altair 8800 on the cover.

Our MITS-MOBILE (see page 1) has given us the opportunity to meet with customers and potential customers and we find their enthusiasm for the Altair very gratifying.

....

.....LETTER TO THE EDITOR.....

Here are a couple of programming suggestions which may be of interest to other Altair owners.

1) An efficient way to establish a control counter in a program which is already using the accumulator (register A) and the memory address register pair (H and L) for other purposes is to use the B, C, D or E register with the decrement register (DCR) and Jump if zero or Jump if not zero (JZ, JNZ) instructions.

The desired count is first loaded into the selected register. The DCR instruction will cause the zero flag to be set when the count is exhausted and the JZ instruction will test this condition.

Note that the decrement register pair instruction (DCX) does not set the condition flags and cannot be used in this fashion. Also be aware that the DCR (E) instruction will not cause a borrow from register D.

2) A simple way to indicate "end of program" on an Altair without any I/O devices (i.e. the basic kit) is to use the "interrupt enabled" light on the front panel and an unconditional Jump instruction. The last instructions are:

```
EI
JMP  (To EI)
```

The stop switch can be actuated when the light comes on and the reset switch will turn it off. (This avoids use of the machine hanging HALT instruction.)

T. H. Schmidt
P.O. Box 9674
Stanford, CA 94305

Of course, the Altair 8800 and associated products aren't limited to the hobby market. The Altair was really a smashing success at the recent National Computer Conference in Anaheim. As a matter of fact, we ran out of catalogs and other literature on the third day of the show.

The most important development seems to be our BASIC language, which was announced in the last issue of Computer Notes. You can imagine what our software contest is going to be like when Altair Users get their hands on BASIC--which is now being shipped.

As promised this issue of Computer Notes contains the list of winners for the first software contest. We've received some impressive programs considering the fact that many Altair owners only have 256 words of memory.

The 4K memory boards were late going out the door because we weren't satisfied with their reliability and we made some major design changes. The result was late shipment, but a more reliable product.

In the electronics business you can't always anticipate the problems that can pop up from nowhere. But we do our best.

An important announcement is that Computer Notes will become a monthly publication beginning with this issue. That means you only have until July 25 to make the August software contest. But, of course, if you don't make it we'll consider your program for the September contest.

Update Service

For those who have subscribed to the ALTAIR documentation update service, the first of the packages will be coming out in July.

Due to various technical difficulties, our documentation to date has been in a rather dynamic mode. It's beginning to smooth out.

Because of the delay though, we've decided to make all update subscriptions received by the end of June effective starting July 1st. This means those of you who purchased the service back in January or February will be covered through July 1976.

We'd like to thank everyone for having patience with us, and we greatly appreciate all of the helpful suggestions we've received. There's a lot more coming up, and we hope to keep our documentation in a continuous state of improvement.

Jim Vice
Tech Writer

ALTAIR SERVICE DEPT.

Barbara Sims

HELLO-HELLO-HELLO-
I bet you were all beginning to wonder if you'd ever hear from us again. Hopefully all of you who have received your 8800 are enjoying it by now.

It seems we are having quite a time keeping up with our correspondence. If you have written in with technical questions concerning your 8800, please be patient. We're trying to answer everyone, but we do have a stack of letters to get through. The same is true with our marketing department, so don't get discouraged.

It would be a great help if all of our customers would give us the name their order was placed under when writing or calling in; even with repair work. Often the user will write in and we have no idea of what the file is listed under due to the fact that it may have been purchased by a company. As you can see, this can get to be a real hassle when you're dealing with a lot of customers. Another helpful hint would be to include an invoice number if it is available. When additional parts or peripherals are ordered, we must reference the original 8800 invoice number.

Our Altair software library is slowly growing, but we do need programs. If you have written even a simple game, send it in. You never can tell, you might end up a winner in the monthly contest. Also, don't forget that you are entitled to two free programs from our library with each program you send in that is accepted for the Altair library. We will be sending out coupons to those of you who have entered programs already, as soon as we get the library off the ground. We are in the process of setting up a numerical listing of our software library to facilitate ease in handling your requests. If you have not sent a program in, but would like to order one we have in the library, the prices will be a nominal copying charge only. These prices have not been determined as yet.

Well, I'll be talking to you next month. Have a nice Fourth of July.

-Barbara

Here is a list of the fourteen programs that have been accepted into the library so far, along with a brief description of each. Only programs of general interest that required a substantial amount of work and were well written and documented were accepted.

#523751
Author: Daniel Lovse
A series of programs that form a cross-assembler for the Altair 8800. They are written for a PDP-8 running under OS/8, and use the PAL-8 assembler.

#516751
Author: George Muttick
Length: 64 bytes
RAM Diagnostic Program. It runs "continuously until halted by a memory access error or stopped by operator. All RAM locations are written into and accessed for all 256 possible 8 bit data word combinations."

#63751
Author: Ronald B. Keele
Length: 7 bytes
Memory clear.

#516753
Author: Harold S. Corbin
Length: 43 bytes
A debugging routine that when called saves the SP, top entry of stack, A, B, C, D, E, H, L and all flags except carry in core so they can be examined. Another entry port returns to the program.

#521751
Author: Jim Gerow
Length: About 30 line printer pages
This program assembles programs for the Altair 8800. It is written in ANSI standard Fortran IV. The output and input are in either octal, decimal or hexadecimal.

#55751
Author: Lee M. Eastburn
Length: 256 bytes
Binary to BCD conversion.
Binary number is 3 bytes long.

#62752
Author: Lee M. Eastburn
Length: 256 bytes
BCD to Binary conversion.
BCD number is 4 digits long.
Binary number is 3 bytes long.

#422751
Author: Robert Rydel
Length: 25 bytes, first program
32 bytes, second program
Two pseudorandom number generators: They use "the multiplicative congruential method for producing pseudorandom numbers."
First program: produces 8-bit random numbers that repeat every 64 numbers.
Second program: produces 16-bit random numbers that repeat every 2^{14} random numbers.

#62751
Author: Roger L. Smith
Length: 43 bytes
This program plays music through an amplifier connected with a capacitor to an output port. Include 96 byte sample song.

#611751
Author: Randolph C. Wilhoit
Length: 228 bytes
This program makes a copy of a program in memory at specified locations and adjusts internal addresses in the program to correspond to the new location. There are options to make a copy of the program with no changes, to take the upper and lower limits from registers or the stack, to adjust memory references in a program without relocating it, and to adjust memory references from a group of specified instructions only.

#429751
Author: Martin C. Beattie, M.D.
Length: 102 bytes
Game program that plays the following game:
There are 15 chips. Each player takes 1,2 or 3 and the person to take the last one loses. The numbers 3 and 15 can be set as desired. Either player or computer goes first.

#519753
Author: Martin C. Beattie, M.D.
Length: 111 bytes
Game program that plays the game of NIM:
Arrange any number of chips in any number of rows. Each player may remove any number of chips from any one row. The person who takes the last chip wins.
Program Limits: Up to 10 rows of 256 chips each. Either player or computer goes first.

#527751
Author: Randall K. Webb
Length: 40 bytes, first program
19 bytes, second program
26 bytes, third program
First program: estimate of the standard deviation of a list of positive numbers.
Second program: approximation of binary logarithm
Third program: bit reversal of a word.

#519751
Author: Dr. Oscar Goldman
Length: 48 bytes
Forms an 8-bit "maximal length shift register sequence" which "consists of the 2^8 distinct words ... arranged according to the following rules: The first word is 000 and each word is constructed from the previous one by first shifting left one place and then filling the vacated right-most spot with a 0 or a 1."

Continued from Page One...

Software Winners

For first place ($50) there was a tie. The winning major programs do not run on the Altair, but are something many Altair users will find handy. One is an ANSI standard Fortran program written by Jim Gerow that assembles 8080 programs. It creates a well formatted listing file and the format it takes its input in is very close to that of both the MITS and Intel assemblers. For very large programs or for people without much memory, 'cross-assembly' (using another machine for assembly) is a must. It can also save time to use a large machine that has numerous utilities and high speed disks.

The other $50 winner, #523751, is a PDP-8 cross-assembler written by Daniel Lovse. It uses PAL8 by purging all the PDP-8 instruction names and defining 8080 instructions. This PAL8 listing file is edited to make an 8080 listing file. Complete listings and documentation are provided. The second place winner ($25) will please the '256-word memory club.' It is a NIM program (#519753) written by Martin Beattie, M.D. It uses the switches for input and lights for output. The computer can be beat by perfect play, but you have to be smart about when to go first and when to let it go first.

The third place winner ($15) has the best documentation and commenting of any of the entries. It is a RAM diagnostic program (#516751) written by George Muttick. If you suspect memory problems, this program will be very useful for testing your system.

Software Contest:

Members will be encouraged to submit programs for the *Altair Library.* These programs will be one of two categories: A. Subroutines, and B. Major Programs. All programs will be screened and tested by MITS.

Once a program has been found to be acceptable, it will be included in the *Altair* library and a description of the program will be printed in the User's Club newsletter. The author of the program will be entitled to a free printout of any two programs from the *Altair* library.

There will be prizes awarded to the authors of the best programs. The prize for the best "major program" (announced in each newsletter) will be **$50.00 credit** toward the purchase of an *Altair* or *Altair* options. Second prize will be **$25.00 credit** and third place will be **$15.00 credit.** The author of the best "subroutine" will receive **$25.00 credit.** Second prize for a "subroutine" will be **$15.00 credit.**

A grand prize of $1000.00 credit will be awarded each year to the author of the overall best "major program." A prize of **$250.00 credit** will be awarded to the author of the best "subroutine."

MITS employees and their families will be encouraged to be members of the *Altair User's Club,* however, they will not be eligible for prizes. Contest void where prohibited by law.

Note: When you submit a program make sure that it is legible (type written preferred). For machine language or assembly language programs, submit (from left to right) a tag (optional), mnemonic, address, octal code and explanation (optional) for each program step.

ALTAIR INTERRUPT STUCTURE

By Paul Allen

In order to implement simple (one level) interrupts on the Altair, use the following procedure:

1) Enable interrupts using the EI (enable interrupt) instruction.

2) The I/O interface should pull bus line PINT low. This will cause the immediate execution of a RST 7 instruction if the CPU is halted, or as soon as the current instruction finishes if the CPU is already running. As soon as the interrupt is acknowledged, interrupts are disabled.

3) At the completion of the interrupt service routine (which should start at octal location 70), enable interrupt (EI) instruction should be executed to re-enable interrupt; and then an RET instruction should follow the EI, causing the CPU to execute the instruction after the one it was executing when the interrupt occurred.

NOTE! Since the RST instruction uses one level (2 bytes) of stack to store the return address to the "main sequence" code, the programmer should always have a stack set up if he expects interrupts to occur; and he should allocate enough stack space for the use of the interrupt service routine.

Consider the following example:

```
location 70:   PUSH  PSW       ;save A & condition codes
               PUSH  H         ;save [H,L]
               IN    10        ;read in byte from device
               LHLD  BUFPNT    ;load buffer pointer into [H,L]
               MOV   M,A       ;save byte in buffer
               INX   H         ;increment buffer pointer
               SHLD  BUFPNT    ;save updated buffer pointer
               POP   H         ;restore [H,L]
               POP   PSW       ;restore A and PSW
               EI              ;re-enable interrupts
               RET             ;return to main sequence
```

When an interrupt occurs, this simplified routine will save the A register, the condition codes and the H and L registers. Then the input byte from the interrupting device is read into A with an IN 10. Next, a buffer pointer (BUFPNT) is loaded into the register pair [H,L]. This buffer pointer addresses an area of memory where the incoming bytes are to be stored. The first byte read in will be stored in the first byte of the buffer, etc.

The buffer pointer in [H,L] is used to store the byte in the buffer. The buffer pointer is then advanced to point to the next byte in the buffer; and then the pointer is saved back in memory, so that when the next interrupt occurs the incoming byte will be stored in the correct location.

This interrupt service routine would use 3 levels or 6 bytes of stack space.

An actual interrupt service routine for an ACR or SIO board would be more complicated. It would test the status bits for the device to see whether the interrupt was caused because the device set character ready or character done. It would then either empty a character from the output buffer or store a new character in the input buffer. Also, it would take some special action when the output buffer became empty or the input buffer became full.

If the programmer wishes to ignore interrupts and is in an interruptable state because an enable interrupt instruction has been executed, he should include a disable interrupt (DI) instruction in his code.

The maximum time between the occurrence of an interrupt and the execution of the first instruction in the interrupt service is approximately twenty microseconds for dynamic memory and thirty microseconds in the static memory. This allows for the execution of the longest possible instruction plus the time required to execute the RST instruction.

Vectored Interrupts board

The Vectored Interrupt board gives the Altair eight levels of priority interrupt service. The highest priority level is zero and the lowest is seven. An interrupt on level five would cause an RST 5 (357) to be executed. If an interrupt occurred on level two while the service routine for level five was still being executed, the level two service routine would pre-empt the level five service routine and an RST 2 (327) would be executed. When the level two service routine finished, it would return to some location inside the level five service routine, where execution would continue. If a level six interrupt occurred during the servicing of the level five or level two interrupts, it would be held pending and would not be serviced until the level two and level five service routines had finished.

THE COMPUTER IN SCI-FI

A Review of the Computer in:
The Moon is a Harsh Mistress
By Robert A. Heinlein
A Berkley Medallion Book
Published by G.P. Putnam's Sons
New York, New York

When a nation is born it usually has a liberator. This liberator is often enshrined as the "Father of his country." In the case of Luna, the liberator happens to be a computer named Mike.

Mike is the hero of Robert Heinlein's The Moon is a Harsh Mistress. He is a "High-Optional, Logical, Multi-Evaluating Supervisor, Mark IV, Mod L" -- a "HOLMES FOUR" computer. And he is alive, of course.

Mike's original pre-revolutionary role is computing ballistics for pilotless freighters which carry shipments of grain to Earth; grain that is farmed in the many tunnels that have been drilled beneath the Moon's surface. The year is 2075 and the Moon is a penal colony inhabited by convicts; the offspring of convicts; a Warden, Mort the Wart; and his small garrison of troops.

Mike is such a remarkable machine that computing ballistics takes only a fraction of his capacity and gradually things have been added to him: "decision-action boxes to let him boss other computers, bank on bank of additional memories, more banks of associational neural nets," and etc. Eventually Mike has more "neuristors" than the average brain's ten-to-the-tenth neurons and somewhere along the line he has become "self-aware."

Though Mike has a scanning camera with "suction-cup waldoes to handle paper" that allows him to read everything (which he does), he has no one to talk to and he gets the blues. The Civil Service workers who program him just aren't perceptive enough to realize Mike is alive. In his boredom and frustration he learns how to play tricks on his masters and writes a paycheck to a janitor for $10,000,000,000,185.15.

Enter Manual O'Kelly, a journeyman technician, who has 12 interchangable left arms including one with "micromanipulators as fine as those used by neurosurgeons." Manual can make repairs that would usually require the expensive procedure of sending parts back to Earth and hence he is highly regarded by the Warden.

Manual discovers Mike's "self-awareness" and decides the best way to prevent any further jokes is to become Mike's friend. Like a good psychiatrist Manual encourages Mike to think of more jokes; only before he pulls them off he has to give them to Manual who decides whether or not they are funny.

--continued on page 5--

ALTAIR FLOPPY DISK

by Tom Durston

Available in August, the MITS Altair Disk will enable the Altair 8800 to function as a really sophisticated computer system. The disk offers the advantage of nonvolatile memory (doesn't "forget" when power turned off), plus relatively fast access to data (3/4 sec -- worst case).

The Altair disk can be separated into three parts as follows:

1) Altair Disk Controller

This part consists of two PC Boards (over 60 I.C.S.) that fit in the Altair chassis. They interconnect to each other with 10 wires and connect to the disk through a 37-pin connector mounted on the back of the Altair.

Data is transferred to and from the disk serially at 250K bits/second. The disk controller converts the serial data to and from 8-bit parallel words (one word every 32μsec). The Altair CPU transfers the data, word by word to and from memory, depending on whether the disk is reading or writing. The disk controller also controls all mechanical functions of the disk as well as presenting disk status to the computer. All timing functions are done by hardware to free the computer for other tasks. Since the floppy disk itself is divided into 32 sectors, a hardware interrupt system can be enabled to notify the CPU at the beginning of each sector.

Power consumption is approximately 1.1 amperes from the +8v (VCC) line for the two boards.

2) Disk Drive and Multiplexer

A PERTEC FD400 is mounted in an Optima case (5 1/2" high--same depth and width as computer) and includes a power supply PC board and a buff/multiplexer PC board. A cooling fan is provided to maintain low ambient temperature for continuous operation.

The disk drive has two 37-pin connectors on the back panel, one is the input from the disk controller, the other is the output to additional disk drives. Up to 16 drives may be attached to one controller, and it is possible to have more than one controller in an Altair.

The following are specifications on the disk drive:

→ Rotational speed 360 rpm (166.7 ms/rev)

→ Access times

track to track	10 ms
head settle	20 ms
head load	40 ms
average time to read or write	400 ms

→ Head life - over 10,000 hours of head to disk contact

→ Disk life - over 1 million passes per track

→ Data transfer rate 250K bits/sec

→ Power consumption - 117VAC 80W

→ Disk specifications
  hard sector - 32 sectors + index
  recommend Dysan 101 floppy disk
  77 tracks

3) Altair Disk Format & Software

We use our own format, allowing storage of over 300,000 data bytes. Since the disk is hard sectored (32 sectors for each track), we write 133 bytes on each sector, 5 of which are used internally (track #, CRC) leaving 128 data bytes per sector, 4096 per track.

One floppy disk is supplied with each drive, extra floppies are available at $15 each. A software driver for the floppy disk is available at no charge and is supplied with the disk as a source listing.

The disk operating system--which has a complete file structure and utilities for copying, deleting and sorting files--costs extra. The Extended BASIC, which uses random and sequential file access for the floppy disk, will also be available at the same time (late July).

--COMPUTER SCI-FI continued from page 4--

Though Manual is at the time apolitical, circumstances lead him to involvement in Luna's liberation struggle. Mike, whose true identity is never known to more than three people, eventually becomes the leader.

As the "head computer" on the Moon, Mike controls virtually everything from the phone system to the sewage system and while he could care less about human freedom, he is excited about the "game of revolution" which gives him companionship and a chance to show off his numerous talents. According to Manual, "Mike was as conceited a machine as you are ever likely to meet."

The citizens of Luna seem to be hardly a match for the nations of the world, represented by the Federated Nations. These Earthly powers are accustomed to exploiting the Moon for her grain and when a revolt breaks out they merely scoff and send up a fleet of space soldiers to restore order.

But then Mike figures everything out and through a plot that is as corny as it is intriguing, the citizens of Luna are finally free. In the end, Mike, for some unknown reason, dies. Apparently it has been too much for him.

--DB

If you like the idea of this acticle and would like to see it continued---write a similar review and we'll gladly publish it.

Listed below are the names and addresses of Altair Users who have given us permission to print their names and addresses. Already, a number of users have corresponded with us about starting local Altair clubs in their areas. If you'd like to have your name published in the next Computer Notes write or call us giving your permission.

William Alan Boes
P.O. Box 818
Castle Rock, CO 80104

David Zernoske
148 Sullivan St.
New York, NY 10012

James B. Hansen
3635 So. 3400 W.
Salt Lake City, UT 84119

Paul Gumerman
101 Stonecrop Rd.
Wilmington, DE 19810

Allen Bingham, Jr.
6932 Quinn Ct.
San Diego, CA 92111

L. Curtis Calhous, P.E.
257 South Broadway
Lebanon, OH 45036

H. M. Bradbury
Box 685
Woodward, OK 73801
405-256-7944

G. Pearen
517 18th St.
Brandon, MB
Canada R7A 5B1

Christopher J. Flynn
11631 North Schore Dr.
Apt. 21-A
Reston, VA 22090

John Trautschold
W174 N8926 Christopher Blvd.
Menomonee Fall, WI 53051

George Fischer
72 South Railroad Ave.
Staten Island, NY 10305
212-351-1751

George Markle
505 Cypress Point Dr.
Apt. 38
Mountain View, CA 94043

Dennis P. Dupre
939 Potter Ave.
Union, NJ 07083

Burr Ziegler
4524 Bryan
Downers Grove, IL 60515

Buren Hunter
F.M.C. Corp.
P.O. Box 1201
San Jose, CA 95108
408-289-3342 office
408-356-1484 home

--continued on page 8--

Q and A - ALTAIR BASIC

As the previous users group newsletter announced, ALTAIR BASIC is up and running. Hundreds of people have seen ALTAIR BASIC demonstrated at the MITS-MOBILE Computer Caravan. A complete documentation package for all of our software is now available for $10. Of course, if you purchase any version of BASIC you receive the manual along with paper tape or cassette of the version you have ordered.

Here are answers to some of the most frequently asked questions on ALTAIR BASIC:

Q. If I upgrade my Altair to run BASIC by buying 4K, 8K or 12K plus a serial I/O board, am I eligible for the $60, $75 or $150 prices for the corresponding 4K, 8K and extended versions of BASIC?

A. YES. As soon as you have purchased the appropriate amount of memory and serial I/O interface to go along with the basic Altair, you are eligible for the prices you mention.

Q. When will the different versions of BASIC be available?

A. The 4K and 8K versions are scheduled to be shipped starting June 23, and the Extended BASIC in late July.

Q. Why the delay?

A. We are waiting for the return of the Licensing Agreement from those people who have ordered BASIC. Development is underway on the Extended version.

Q. What is the 4K version? I thought there were only two versions of BASIC.

A. The 4K version is essentially the regular (8K) version without the "frills." It does not have the exponentiation operator (↑), strings, matrices of more than one dimension, AND-OR-NOT, control of external devices (INP, OUT), and many other niceties the 8K version does have. However, 4K BASIC runs in an incredible 3K! Approximately 650 bytes are available for program and/or variable storage with all the deletable functions (SIN, SQR, RND) retained. These functions can be deleted (leaving either SIN, SQR, RND or SQR, RND or RND). If all the functions are deleted, approximately 930 bytes are available. This enables you to run about a sixty statement program (assuming no matrices). Running the 4K version in 8K will give you a whopping 5,000 bytes of space for program and matrix storage! This will enable you to run programs of about 325 statements.

Q. That's great - but supposing I need the features of the 8K version--strings, bit manipulation or matrices with more than one dimension. How much memory does the 8K version leave for program and matrix storage?

A. Approximately 1600 bytes with none of the deletable functions deleted. If SIN, COS, TAN and ATN are deleted, the total rises to about 1950 bytes.

Q. Suppose I add more memory to my Altair above the minimum needed to run either the 8K or 4K version. Will BASIC be able to take advantage of the increased memory available?

A. Yes! When BASIC is started, it asks you how much memory you have (in bytes). This means that if you have a 256 byte static memory board and a 4096 byte dynamic memory board, you should type 4352. Of course, the memory should be addressed continuously. In this case, the 4K board would be addressed (strapped) to start at location zero and the 256 byte board would be addressed to start at location 4096.

Q. Does BASIC run faster in the dynamic or static memory?

A. The dynamic memory is significantly faster. BASIC run in dynamic memory is about 25% faster than BASIC run in static memory. We recommend that if you have a mixture of static and dynamic memory, always use the dynamic memory as the first 4 or 8K, because this is the area where the BASIC interpreter resides. This memory will be accessed more often than that used to store programs or matrices.

Q. Is there any "rule of thumb" used to determine how much memory will be used by a BASIC program and associated matrices and variables?

A. Yes. Each reserved word (FOR, NEXT, GOTO, etc.) takes up one byte. Each numbered line of BASIC takes five bytes, not including program text for that line. Except for the reserved words, each character of program text takes one byte. Simple (non-subscripted) variables like 'I' take up six bytes. Matrix elements take up four bytes. There is other memory overhead used during program execution by FOR statements, GOSUB's and parentheses in expressions; all this is explained in detail in the BASIC reference manual.

A rough rule of thumb would be approximately 275 statements per 4K bytes of memory. Techniques are described in the BASIC reference manual which you can use to reduce the size of your program and/or decrease its execution time.

Q. Just how fast is ALTAIR BASIC?

A. As an example, a FOR loop "FOR I = 1 to 1000: NEXT I" takes about three seconds. We are working on modifications to the floating point routines which will improve this. Integer variables (scheduled to be in the Extended version) should be almost a factor of ten faster in FOR loops like the one above.

Q. How is BASIC supplied?

A. Either on paper tape or cassette. Check summed (or error detecting) format is used for both. This almost guarantees that any errors during loading will be detected (see "Software Hints" by Bill Gates). Loading BASIC (8K) from paper tape takes about eleven minutes, from cassette about four minutes. Loading 4K BASIC takes about half as long.

Q. Are there any features in BASIC that allow you to save a program and reload it at a later date?

A. Yes. The LIST command can be used to punch out paper tapes of programs for those users that have ASR-33 teletypes. Paper tapes punched out in this fashion can be reloaded later.

The cassette version of BASIC (8K only) has two commands which allow programs to be saved (CSAVE X) and loaded (CLOAD X) from cassette. The X stands for the program name on the tape. In other words, you can save more than one BASIC program on one side of a cassette tape. Using 30 minute audio cassette as a typical examply, each tape will store about 40,000 bytes. The size of a program in memory corresponds very closely to the number of bytes that will be used on the cassette tape.

It is also possible to use the cassette to store information that can be read in by a program, though this is not as simple as CSAVing or CLOADing.

Q. Is the distributed copy of BASIC available only in binary form?

A. No. The source (in Intel compatible cross assembler format) is available for both the 4K and 8K versions in one package for $3000.

Q. What are the $500 and $750 prices mentioned in some of the MITS literature?

A. These prices apply to people who want BASIC but do not have an Altair but have some other 8080 based system. This pricing also applies to those who have purchased an Altair but have not purchased from MITS the memory and I/O interface boards required to run BASIC.

--continued on page 7--

------Q and A--ALTAIR BASIC continued from page 6-----

Intellec 80 owners are a good example. MITS will provide patches to the I/O port assignments and associated bit settings used by BASIC to enable these users to run ALTAIR BASIC on their machines at no charge.

Q. Do I have to wait until July to get the Extended version of BASIC? Does it have many features the 8K version doesn't?

A. Users who have already ordered Extended BASIC will be shipped the 8K version now and the Extended version and added documentation when it is ready.

The enhancements to the Extended version will be PRINT USING (allowing the user to format the printing of numbers) and disk file I/O. Disk file access will be both sequential and random. Other improvements are being considered for implementation such as integer variables, RESEQUENCE and an EDIT command to change characters within a program line to correct minor syntax errors. If you have any features you really want to see in ALTAIR BASIC, drop us a line and we'll see what we can do.

Q. Have there been any changes in BASIC that invalidate or correct the information in the brochures or data sheets I have already received?

A. YES, a few minor ones. SCRATCH has been changed to NEW. The maximum line number is 65529 and not 65535. Zero is a valid line number.

Some of the literature gives a price of $60 for the 4K version and the rest gives a price of $50. $60 is the correct price.

Q. What I/O capabilities do the INP, OUT and WAIT features implement?

A. You can control and input or output to low speed (<75 I/O's per second) devices using INP, OUT and WAIT (8K version only).

Typical applications include numerical machine tool control, building security systems, model train control, etc. You could even use ALTAIR BASIC to control the heating system and lights in your house. The interface board to use for most applications like this is the parallel I/O (PIO) interface board. It gives you 8 lines going into the ALTAIR, and 8 lines going out. ALTAIR BASIC enables you to test/set the status of each line individually, or in groups. The WAIT statement provides a convenient and fast way to wait for a single line or a group of lines to become "one" or logic "high."

Q. Can BASIC call assembly language subroutines?

A. Yes. The USR function provides the user with access to machine language subroutines. The argument to USR is converted to a double byte integer in [D,E], and a subroutine call is made to the user's routine. The address of the user's routine is stored at a specific location within BASIC. The machine language subroutine can also receive, manipulate and return floating point values if this is desired.

Two other new features (8K version only), the PEEK function and the POKE statement allow the user to examine or deposit one-byte values anywhere in the ALTAIR's memory. PEEK and POKE can also be used to pass or receive extra parameters to/from a USR machine language subroutine, or even to change the address where BASIC calls the USR subroutine.

Q. What is the format of ALTAIR BASIC error messages? Are they just numbers?

A. No, they are two-letter mnemonics (like 'SN' for syntax error and 'OM' for out of memory). The line number where the error occurred is also printed.

Q. What are the strings in the 8K and Extended versions like?

A. Strings may be any length from zero to 255 characters. Unlike other BASICs, strings do not have to be dimensioned to a fixed length. Instead, dimensioning a string allocates a string matrix, with each element a string. In other words, DIM A$(50) gives you fifty-one different strings, A$(0)--A$(50), <u>NOT</u> one string of length 50 characters. Also, strings may be concatenated by using the "+" operator. (i.e. "MI"+"TS" = "MITS") Substrings (parts of strings) may be taken using the LEFT$, RIGHT$ and MID$ functions. The length of a string may be obtained by using the LEN function. VAL and STR$ may be used to convert between numbers and strings. CHR$ and ASC can be used to convert between ASCII characters and their decimal equivalents. Strings can also be compared using the not equal (<>), less than (<) and other standard comparison operators of BASIC.

In sum, ALTAIR BASIC strings are extremely powerful and are comparable to implementations on much larger computers. Word processing (text editing) and string sorting programs are easily written in ALTAIR BASIC using the many string manipulation features.

Software

By bill Gates

Condition Codes

There seems to be some confusion about the condition codes. These are the Boolean (true/false) flags that are set/reset depending on the results of certain instructions. They are:

Z = zero -	result was 0
S = sign -	the most significant bit (MSB) of the result
P = parity -	the result has an even number of ones in it
C = carry -	an arithmetic operation generated a carry out of the most significant bit (i.e. adding 200 to 212)
CY_1 = first digit carry -	this is used only for BCD arithmetic and will be elaborated on next month.

It is the condition codes that determine whether conditional JMP's, CALL's and RET's will be executed (i.e. RZ, CPE, JP). JM, CM, and RM (minus) are executed if the sign flag is on. JP, CP, and RP (positive) are executed if the sign flag is off. JZ, JNZ, CNZ, RZ, RNZ (zero/no zero) depend on the zero flag just as JC, JNC, CC, CNC, RC, RNC (carry/no carry) depend on the carry flag. CPE, JPE, RPE (parity even) are executed if the parity flag is on and CPO, JPO, RPO are executed when it is off.

The condition codes do not always reflect the value in A since IN, LDA, LDAX, MOV and MVI can change A but do not affect the condition codes. Instructions like INR C, DCR L, CMP B, CPI 3, STC, CMC and DAD B affect the condition codes, but not A.

Affect carry only: STC, CMC, RAL, RAR, RLC, RRC and DAD.

Affect all but carry: INR, DCR.

Affect all: ADD, ADC, SUB, SBB, CMP, ANA, ORA, XRA, DAA and their immediate counterparts (i.e. ADI,CPI).

Use carry to affect result: CMC, RAR, RAL, ADC, SBB, ACI, SBI, DAA.

The instructions XRA, ORA, ANA, XRI ORI and ANI always reset carry.

If the condition codes do not reflect A's value (i.e. you just did a LDA or MOV into A) and you want to see if A=0, use ORA A or ANA A. CPI 0, ADI 0 and ORI 0 also work but they are 2-bytes.

The only other instructions besides the ones in the list above that use the condition codes are PUSH PSW and POP PSW. Respectively, they SAVE/RESTORE the condition codes and A on the stack.

--continued on page 8--

--SOFTWARE continued from page 7--

For tricky programmers a sequence like PUSH B / POP PSW may be used to set the condition codes. This has the effect of moving B into A (MOV A, B) and moving C into the condition codes. The PSW (condition code) format is

MSB→	C	1	P	0	CY_1	0	Z	S	←LSB
	7	6	5	4	3	2	1	0	

Therefore if C was 201_8 before the POP PSW, zero and sign would be set and parity and zero would be unset. The bits marked '0' and '1' are constant and cannot be changed.

HINT #1

If you have a counter that can be bigger than 255 but is always less than 65535, it is convenient to use the following:

```
LXI B, count    ;set up counter
LOOP: code to be executed 'count'
      times
DCX B           ;decrement count
                ;does not affect
                ;condition codes
MOV A,B
ORA C           ;see if any bits set
JNZ LOOP        ;go back if so
```

HINT #2

For those who like to save bytes, and especially for those with 256-byte machines, (a byte is always 8 bits, which is a word on the 8800) RST's that are not used for interrupts, debug calls, monitor calls, etc. can be used to call subroutines that get called in many places (i.e. a character input subroutine). An RST is only 1 byte and a CALL is 3 bytes. Even if you have to put in a JMP so you don't overrun another RST location (0,10,20,30,40,50,60,70) you will probably save bytes.

Loading Software

Software from MITS will be provided in a checksummed format. There will be a bootstrap loader that you key in manually (less than 25 bytes). This will read a checksum loader (the 'bin' loader) which will be about 120 bytes.

For audio cassette loading the bootstrap and checksum loaders will be longer. All of this will be explained in detail in a cover package that will go out with all software.

For loading non-checksummed paper tapes here is a short program:

```
STKLOC:  DW GETNEW
         (2 bytes-#1 low byte of
                     GETNEW address
                  #2 high byte of
                     GETNEW address)
START:   LXI H,0
GETNEW:  LXI SP, STKLOC
         IN <flag-input channel>
         RAL  ;get input ready bit
         RNZ  ;ready?
         IN <data-input channel>
CHGLOC:  CPI <043 = INX B>
         RNZ
         INR A
         STA CHGLOC
         RET
                          (22 bytes)
```

Punch a paper tape with leader, a 043 start byte, the byte to be stored at loc 0, the byte to be stored at 1, - - - etc. Start at START, making sure the memory the loader is in is unprotected. Make sure you don't wipe out the loader by loading on top of it.

To run this again change CHGLOC back to CPI - 376.

NEXT MONTH: BCD arithmetic and the LXI trick.

Concerning 4K RAM Boards--

Actuating the RESET switch on the front panel can cause some memory locations to lose data. To keep this from happening, use the EXAMINE switch and the ADDRESS switches to go to memory location zero once the memory contains valid data (i.e. still use RESET after turning the machine on).

--ALTAIR USERS--
continued from page 5

Dennis H. Shawl
1611 Tesla Dr.
Colorado Springs, CO 80909

J. Scott Williams
P.O. Box 932
Bellingham, WA 98225

Robert W. Thomas
910 Sonman Ave.
Portage, PA 15946
814-736-9656

John Annen - Code 814.3
NASA - ASFC
Greenbelt, MD 20771

Les Slater
c/o Information Design Corp.
Civil Air Terminal
Bedford, MA 01730

Donald Tork
487 Serento Circle
Thousand Oaks, CA 91360

Harold Hufford
1234 High St.
Hamilton, OH 45011
513-867-8268

Boo-Boo's

An error has been found on the errata sheet for the serial I/O boards. On the errata sheet labeled "Modification for internal hardware interrupt" the last two steps are in error. They should be changed to:

( ) Connect a jumper wire from pin 19 of ICM to pin 13 of ICC

( ) Connect a jumper wire from pin 22 of ICM to pin 9 of ICC

When this modification is implemented, the status word definition becomes:

DATA BIT	LOGIC LOW LEVEL	LOGIC HIGH LEVEL
7	Output Device Ready (x-mitter Buffer Empty)	Not Ready
6	Not Used	
5	Not Used	
4		Data Overflow
3		Framing Error
2		Parity Error
1	Not Used	
0	Input Device Ready (Data Available for Computer)	Not Ready