SPECIAL SOFTWARE ISSUE

COMPUTER NOTES
© mits 1976
Vol. 2 Issue 6
November, 1976

Software Library

The Software Library is in the process of being transferred to the same facility that houses the new Altair Software Distribution Company.

Effective immediately, the Software Department will no longer accept orders for library programs.

Please refer all orders to:

Altair Software Library
3330 Peachtree Road
Suite 343
Atlanta, Georgia 30326

~~corrept~~ correction

In the October issue there was an error in the 4PIO Operation article (page 5). In B Section Control Register, Bit 3 should read zero rather than 1.

B Section Control Register			CB2	
Bit 5	Bit 4	Bit 3	Cleared	Set
1	0	0	LOW on positive going transitions of first E pulse following write on B Data Channel after Control/Status bit 7 is cleared by read of B Data Channel.	HIGH when C/S bit 7 is set HIGH by CB1 active transition.

Bits and Pieces

By Sondra Koppenheffer

Projected Shipping Dates:

Remember that MITS offers a limited rather than a full warranty. Our 90-day warranty on assembled items covers manufacturing defects plus any labor incurred as a result of the defect. We also have a 90-day kit warranty which covers only parts. If you have any specific questions about your warranty, please feel free to contact us.

Imagine this: You've just come home from a long day at work. Looking through your mail, you see a letter from MITS, Inc. With excitement, you rip it open and find your order acknowledgement inside. As you glance over the rest of the letter, you're suddenly horrified and angered by the projected shipping dates. "Two months! But they assured me that they had those items in stock!"

I'm sure this is a familiar scene to many of you. No, MITS did not mislead you about the availability of the items which you ordered. But our computer is set up to automatically print out a two month projected shipping date. Like our customers, we also buy parts from various companies. On occasion, they are unable to meet a deadline, which causes a delay in our own production. This two month shipping date allows for such delays.

We routinely ship out items on schedule in accordance with our promise to you. But if any problems should arise, feel free to contact us and we'll promptly check into the matter for you.

Helpful Hints on Speeding up Address Changes:

Print or type clearly, both your new and old address. Include the MITS order number from your original purchase of an 8800 or 680. If you no longer have your original invoice, include a label from a current mailing of Computer Notes (your order number will be printed in the upper right hand corner). State whether or not you have any items currently on order.

Reminder:

The only way in which we will accept a phone order is if payment is through either BankAmericard or MasterCharge. If a mail order is placed and the payment is through the use of a personal or company check, we require a three-week holding period.

Time Payments:

For those of you on our Kit-a-Month Package, let me remind you that one time payment is not a definite limit on your monthly orders. If on occasion you wish to order more than one installment during a one month period, that's perfectly OK with MITS. Simply state which time payments you want, and they will be shipped out to you as soon as possible.

Defective Parts?

To insure that you receive the correct part in exchange for your defective parts, we ask that you include the MITS part number as stated in each manual, plus the name of the part. When this information is included, your new parts will get shipped out to you much more quickly.

Those of you who have purchased units from any of our dealers and discover that you need replacement parts, must again go through that dealer. We will no longer accept orders for replacement, defective or missing parts from any customer who purchased the item in question through one of our dealers.

Warranty:

Recently we have had several people who learned a lesson the hard way. Any item which is purchased from MITS has a certain warranty time on it. If ever you purchase something from MITS and find that it is defective, please inform us as soon as possible. If you wait, hoping that the problem will disappear, the warranty on the item might run out, and you will end up paying a sum of money that would not have been necessary. We're on your side, but the line must be drawn somewhere. Don't end up paying for this lesson out of your pocket.

COMPUTER NOTES

Editor Andrea Lewis
Asst. Editor Linda Blocki
Production Tom Antreasian
Al McCahon
Steve Wedeen
Grace Brown
Contributors Sondra Koppenheffer
Paul Allen
Mark Chamberlin
Gary Runyan
David Le Jeune
Wayne Cronin
H. Craig Wiles
O. E. Dial
John Hayes

MITS ANNOUNCES FORMATION OF ALTAIR SOFTWARE DISTRIBUTION COMPANY

By John Hayes

MITS President, Ed Roberts, has announced the formation of a subsidiary company designed to provide Altair computer users with the finest applications software available for microcomputers. Announcing the formation of the Altair Software Distribution Company, Roberts said, "The ASDC will help MITS provide the total support for the Altair family of computers. MITS is now producing the widest line of microcomputer systems and peripherals available from any manufacturer. The formation of the ASDC will provide a mechanism to make certain that Altair computer users have applications software that meets their needs and will be fully supported for years to come."

Roberts briefly explained how ASDC will run. "The Altair Software Distribution Company will acquire sophisticated applications software from persons around the country. The software will be carefully evaluated, thoroughly checked for errors, fully documented and distributed through all Altair computer centers. The ASDC will contract for applications software on either a direct purchase arrangement or a continuing royalty fee." Roberts said that people who have written software will be very well compensated for their efforts. "We are trying to encourage creativity and make it financially attractive for people to produce quality software for the Altair family of computers," he added.

Ron Roberts (no relation) has been named president of the Altair Software Distribution Company. He said ASDC will be looking for only first quality software. "By first quality software, we mean software which has been carefully designed, properly coded, and thoroughly debugged," he explained. "We have exacting standards for the type of documentation which must be produced. Well-written software is only half the problem--the other half is the completion of thorough documentation." Ron Roberts also noted that the ASDC has published a booklet titled ASDC Software Submittal Packet, which describes the standards for applications software and the documentation to accompany the software. The packet is available to anyone who is interested in submitting software to the ASDC for possible inclusion in the ASDC library. Packets can be obtained from Altair computer centers in each major city or directly from the Altair Software Distribution Company.

This month,MITS President, Ed Roberts, also announced the release of the Altair Business System by the ASDC. (See article, page 4). This system includes packages for accounting, inventory management and word processing. The accounting package includes a comprehensive general ledger system and will include packages for invoicing, receivables, payables and payroll. All the applications software supported by the ASDC will be available through local Altair computer centers in each major city. The software will be sold only through Altair dealers rather than directly to the public by ASDC.

The Altair Business System carries a guarantee of three years of software maintenance included in the initial licensing fee in addition to the continuing support that the ASDC and the dealers will provide for the software.

For more information about submitting software to the ASDC or the Altair Business System, contact your local Altair dealer (see list, page18) or write the Altair Software Distribution Company at Suite 343, 3330 Peachtree Road N.E., Altanta, Georgia 30326, phone (404) 231-2308.

ALTAIR 8800 AIDS IN CELL RESEARCH

By H. Craig Wiles

Mr. Wiles is a biomedical research engineer at Duke University in North Carolina.

The Department of Biomedical Engineering at Duke University places a strong emphasis on research as well as the academic program and so is experiencing an increasing requirement for the microprocessor as a research support tool. This evolution began with the decision to introduce the microprocessor into the teaching lab program which stresses biomedical instrumentation and systems design. Two Altair 8800's were acquired for the teaching labs. After observing their capabilities, the decision was made to utilize one of them as a prototype for use in research.

The research project involved extensive exploration of the mechanochemical properties of red blood cell (RBC) membranes. The structural properties of these bilayer membranes have only recently received serious consideration. It's these structural characteristics which provide security for the interior cytoplasm and nucleus as well as a means of control for ion transport from the external surrounding plasma to the cell. For example, these structural properties determine the resistance to deformation which the red cell membrane experiences during microcirculation through the capillaries.

Any method of study involving RBC's obviously requires extreme microscopic levels of observation and instrumentation. One of the research methods used involves single cell micropipette aspiration to produce nanogram forces on the cell membrane surface. The recorded data include temperature of the RBC environment, negative pressure of aspiration, magnitude of deformation and length of time for deformation. It's in this analysis of the micropipette aspiration method that the Altair 8800 is used.

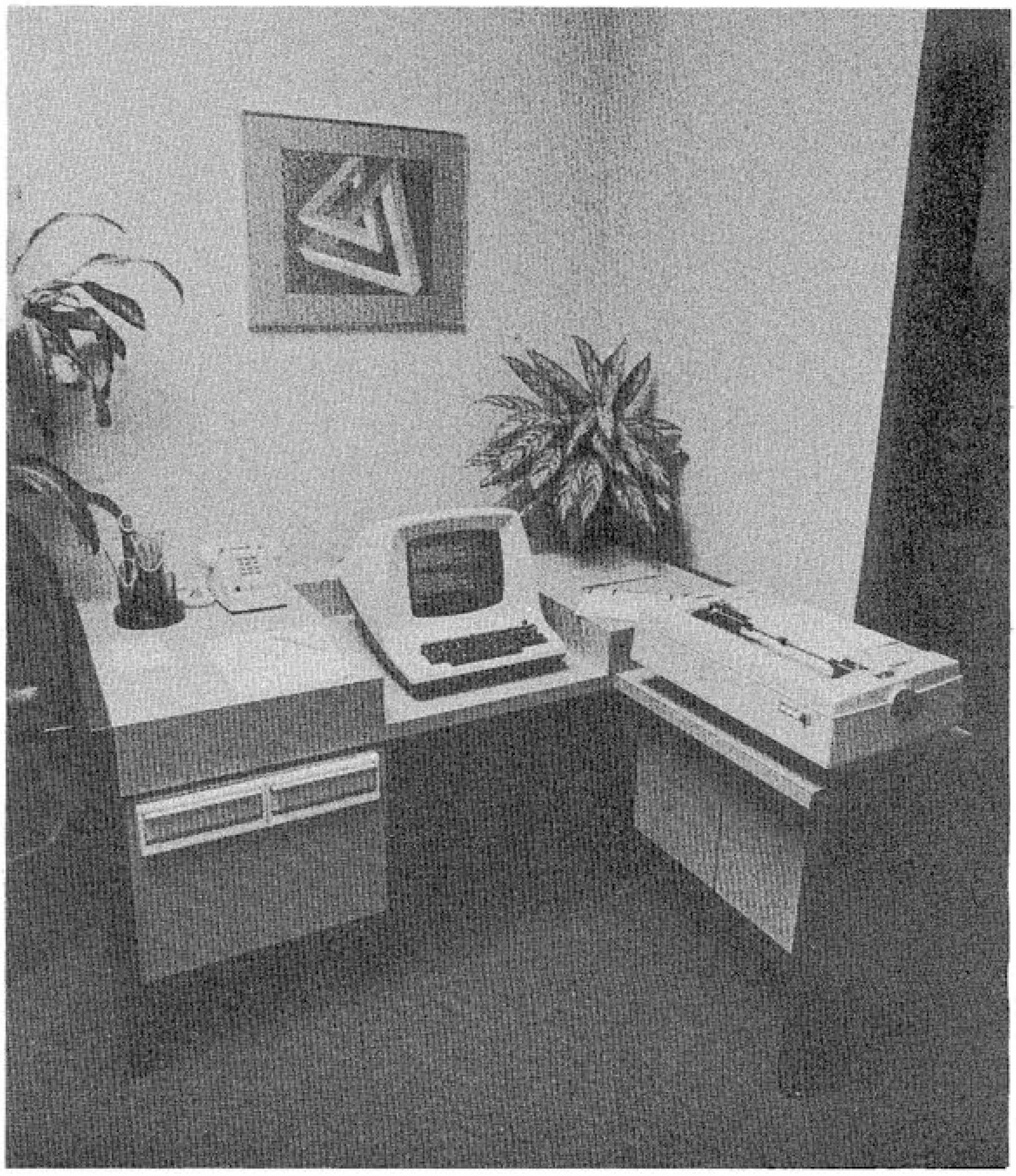
Continued on Page Twelve

ASDC INTRODUCES ALTAIR BUSINESS SYSTEM

By John Hayes

The Altair Software Distribution Company has announced the introduction of a comprehensive set of software packages designed for the small business system market. The Altair Business System includes complete software packages for accounting, word processing and inventory management. It also includes a timekeeping package. The software may be licensed for use in individual packages to accomodate the needs of retail stores, small wholesale distribution centers, industrial users, professional firms and other business offices.

The new Altair Business System is designed around the Altair 8800 Computer. The system hardware may be individually configured for each installation and typically includes a CRT terminal, a typewriter-quality precision printer and one or more floppy diskettes. The ASDC software will be available only through local Altair dealers.



Altair Business System on display at the Atlanta Systemcenter.

The Altair Business System software is packaged in modules to allow a purchaser to select the components of a system that will most closely fit his needs. The accounting package is comprised of four modules--general ledger, receivables, payables and payroll. The GENERAL LEDGER package is the heart of a financial reporting system for a small business. It allows a firm to keep a detailed monthly general ledger of all its transactions by generating a monthly balance sheet and income statement to provide timely information on the financial status of the company. The general ledger package is already available. The PAYROLL package allows a company to prepare its periodic payroll for hourly or salaried employees while accumulating the necessary information for tax reporting. It automatically generates the monthly, quarterly and annual returns to be filed with local, state and federal governments. It also prepares employee W-2s and maintains an up-to-date information reference for each employee. The RECEIVABLES package is a complete invoicing and monthly statement generating system that keeps track of the current and aged accounts receivable. The PAYABLES package keeps track of current and aged accounts payable and incorporates a check writing feature. Each of the three subsidiary systems--receivables, payables and payroll--provides input directly to the general ledger package. The subsidiary packages will be available during the first quarter of 1977.

The WORD PROCESSING package is a flexible text editor system that allows large volume text material, such as contracts or other lengthy documents, to be stored, easily edited and printed. In addition, documents can call for inserts from other files, thereby making repetitive letters and complicated documents easy to produce. The text material is stored in a file without regard to pages or margins. In this way additional text material may be inserted conveniently while page heading, numbering, margins, spacing and other formats may be specified at the time of printing. A draft copy may be corrected and a final printed with different margins. A single document may contain up to 128,000 characters (about 35 single spaced pages), and documents may be linked for longer text. The text editor allows simple in-line corrections and extremely powerful global editing to be easily accomplished. As with the other components of the Altair Business System, the Word Processing package contains a complete set of prompts and other helping messages that allow even an inexperienced operator to make full use of the system with minimum instructions. The Word Processing package will be available during the first quarter of 1977.

The INVENTORY MANAGEMENT package is a flexible data base management system which allows a business to keep complete inventory records "on line". It's designed to allow a user to structure the fields in an inventory file so that the file and its reports contain the information needed by the particular business. In its off-the-shelf form, the Inventory Management package is structured for a typical retail store whose inventory reorder policy is based on minimum reorder

Continued on Page Five

Altair Business System (cont.)

points. Also available will be a POINT OF SALE option for the Inventory Management system. The Point of Sale option prepares a sales receipt while automatically updating the inventory file and providing a direct sales entry for the general ledger. The Inventory Management system is currently available and the Point of Sale option will be available during the first quarter of 1977.

A TIMEKEEPING system is also available for law or accounting offices or other professionals who bill clients for time and expenses. The Timekeeping system keeps track of time and expenses separately and provides automatic billing and statement generation. The Timekeeping system will directly interface with the general ledger and will be available during the second quarter of 1977.

The component packages of the Altair Business System are available under a one-time fee licensing arrangement. The licensing includes three years of software maintenance. Each package is accompanied by a comprehensive set of documentation including operator guides and systems guides as well as training aids.

The Altair Business System may be seen at your local Altair Computer Center. For additional information see your local Altair Computer dealer or contact the Altair Software Distribution Company, Suite 343, 3330 Peachtree Rd., N.E., Atlanta, Georgia, 30326, phone (404) 231-2308.

Program Progress

How about a program on how to do an ascending and/or descending bubble sort of complex numbers in the form (a,b) and/or a≠b; using BASIC?

Suggested By:

Roger Mann
248 Beacon Hill Drive
Ft. McMurray
Alberta, Canada
R9H 2R1

Altair BASIC 4.0

By Paul Allen

Changes to 4K and 8K versions in 4.Ø

A number of new features have been added to BASIC Version 4.Ø. They include all the non-disk features added to the extended version since 3.2--long program lines, the substring assignment MID$ function, octal and hex constants, etc.

Only minor changes have been made to the 4K and 8K versions for release 4.Ø. There are no user visible changes in the 4K except for the change in sense switch encoding (not described in this article). The main changes to the 8K version are the new editing characters (↑U-Line delete, DEL-Rubout), the improved random number generator, and saving matrices on cassette (CSAVE.A and CLOAD.A).

The material below describes only those features added between 3.4 and 4.Ø to the Disk version.

Editing Input - Control/A

The control-A character may be used to edit a line as it is being typed in. As soon as the control-A is typed, a carriage return line feed exclamation mark space sequence is printed, and the user may edit the input using any of the features of the EDIT command before hitting carriage return.

Example:

```
FOR I=1 TO 1ØØ:?A(J),:NEXT I^A
!  (SJCI<return>)FOR I=1 TO 1ØØ:?A(I),:NEXT I

<FOR loop is executed>
```

If an error is discovered during typing and before carriage return has been typed, control-A may also be used to edit the data at an INPUT Statement.

Dot - The Current Line Number

The dot character (.) may be used when a line number is expected in an EDIT, DELETE, LIST, or DELETE command. Dot is set to the current line when an error occurs, a line is listed, edited or inserted.

Examples:

```
LIST 1ØØ
1ØØ X=X+1
OK
EDIT.
1ØØ

5ØØØ THIS IS A NEW LINE
EDIT.
5ØØØ
```

Automatic Line Insertion - The AUTO Command

When a program is created, program line numbers are usually entered in sequential fashion with a standard increment between the lines. The AUTO command provides for automatic generation of line numbers when entering program lines. The format of the AUTO command is:

```
AUTO [<initial line>[,[<increment>]]]
```

Example:

```
AUTO 1ØØ,1Ø
1ØØ INPUT X,Y
11Ø PRINT SQR(X^2+Y^2)
12Ø ^C
OK
```

Control-C is used to terminate an AUTO command. If the <initial line> is omitted, an initial line of 1Ø and an increment of 1Ø is assumed. If the <initial line> is followed by a comma but no increment follows, the last increment used in an AUTO command will be used.

Continued on Page Six

SPACE$ and STRING$ Functions

Two functions are available for generating a string containing a character repeated N times. This is useful for formatting output and for blank filling fields. The STRING$ function returns a string of its string argument repeated N times. If N is zero, the null string is returned. The string argument must be non-null. Only the first character of the string argument is used to form the result string:

```
STRING$(<numeric formula>,<string formula>)
```

Example:

```
PRINT STRING$(1Ø,"A")
AAAAAAAAAA
OK
```

Optionally, a second <numeric formula> may be substituted for the <string formula>. This causes the ASCII value of the <numeric formula> to be used to generate the character to be repeated:

```
PRINT STRING$(1Ø,65)
AAAAAAAAAA
```

The most common case for STRING$ would be to generate a string of spaces. A SPACE$ function is provided to make this convenient:

```
PRINT SPACE$(10);"*"
          *
OK
```

XOR, EQV, IMP

Three new operators have been added to extended BASIC. These operators function in exactly the same way as the AND and OR operators, that is, they force their arguments to integer and then return a sixteen bit integer result. The precedence of these operators is shown in relation to AND and OR:

Highest:

AND
OR
XOR
EQV
IMP

The truth tables below describe how the bits of the result are formed from bits of the left-hand side (LHS) and right-hand side (RHS) operands.

XOR:

LHS	RHS	Result
1	Ø	1
Ø	1	1
1	1	Ø
Ø	Ø	Ø

EQV:

LHS	RHS	Result
1	Ø	Ø
Ø	1	Ø
1	1	1
Ø	Ø	1

IMP:

LHS	RHS	Result
1	Ø	Ø
Ø	1	1
1	1	1
Ø	Ø	1

The FIX Function

The FIX function takes a numeric argument and returns the truncated integer part of the argument. FIX is equivalent to SGN(X)*ABS (INT(X)).

```
PRINT FIX(-1.1),INT(-1.1)
-1             -2
```

The major difference between FIX and INT is that FIX does not return the next lowest number for negative numbers.

Continued on Page Seven

New Club Missouri-Illinois

The first meeting of the St. Louis Area Computer Club was held October 29 on the campus of Washington University. Approximately thirty people attended the organizational meeting, representing various vocations, ages, and geographic areas (including the Illinois side of the Mississippi river). Jon Elson, president pro tem, remarked that the club may be unique in having a number of obscure computers, making communication among members difficult at any level below the flowchart. Committees on hardware, software, and applications were formed, and questionnaires were circulated to aid in planning tutorials and facilities needed. Members stayed late discussing their favorite topics, with hardware reliability a commonly-voiced concern. Meetings are expected on a monthly basis. Contact:

Lou Elkins
314-427-6116, or
Box 1143,
St. Louis, MO 63188

New Club British Colombia

Anyone having questions about the newly formed British Columbia Computer Society should contact one of the members listed below.

1. Officers:

President: Karl Brackhaus
203-1625 W 13th Ave.
Vancouver, BC, Canada
V6J 2G9
(604) 738-9341

Treasurer: Ken Browning
605 Spender Drive
Richmond, BC, Canada
(604) 271-2637

Secretary: George Bowden
1250 Nicola St.
Vancouver, BC, Canada
(604) 681-0688

2. Time and Place of meetings:

At 8:00 PM on the first Wednesday of every month in Room 129 of the British Columbia Institute of Technology (BCIT).

Altair BASIC 4.0 (cont.)

LINE INPUT With a String Argument

The LINE INPUT function is now available in the Disk version. It can now also take an Optional string argument which is printed as a prompt:

LINE INPUT [<quoted string literal>;] <string variable>

(Note: LINE INPUT destroys the input buffer.)

Direct INPUT Allowed/Input of Double Precision Numbers

The INPUT statement may now be given in direct mode in the extended version only without causing an "ILLEGAL DIRECT" error.

In BASIC versions prior to 4.Ø, the Double Precision number input routine was not called when double precision values were INPUT, causing imprecise values to be returned. This has since been corrected.

The amount of CPU time required to output a double precision value has also been improved. If the number is less than 1ØØØØ, the number of additions required to scale the number properly has been reduced.

Speed Improvements in the Disk Version

A number of speed improvements have been made to the Disk version. All constants in programs are now converted to a one, two, three, five or nine byte token. All GOTO, GOSUB, THEN and ERL line numbers are now converted to pointers during program execution. This means that the program does not have to be searched for GOTO references.

Changing Program Line Numbers - RENUMber

The RENUMber command allows program lines to be "spread out" to permit the insertion of a new line or sequence of lines. The format of RENUMber is:

RENUM [NN[,MM[,II]]]

NN is the new line number of the first line to be resequenced. If omitted, NN is 1Ø. MM is used to specify which lines in the program will be renumbered. Lines less than MM will not be renumbered. If MM is omitted, the whole program will be resequenced. II is the increment between the lines resequenced. If II is omitted, 1Ø is used.

To RENUMber the whole program to start at line ten with an increment of ten type: RENUM

To RENUMber the whole program to start at line 1ØØ with an increment of 1ØØ, type: RENUM 1ØØ, 1ØØ

To RENUMber lines 5ØØØ and up so they start at line 6ØØØ with an increment of 1ØØØ, type: RENUM 6ØØØ, 5ØØØ,1ØØØ

(Note: Any attempt to RENUMber parts of the program on top of other parts of the program or to create line numbers greater than 65529 will cause a "FUNCTION CALL" error.

All line numbers appearing after a GOTO, GOSUB, THEN ON . . . GOTO, ON . . . GOSUB and ERL <relational operator> will be properly changed by RENUMber to reference the new line number. ON ERROR GOTO Ø statements are not affected by RENUMber.

If a line number appears after one of the statements above but does not exist in the program, the message "UNDEFINED LINE XXXXXX IN YYYYY" will be printed. This line reference (XXXXX) will not be changed by RENUMber.

(Note: The line number YYYYYY may later be changed by RENUM to a different line number.)

PEEKing and POKEing Above 32K

It is now possible to PEEK and POKE addresses above 32767 using a positive argument. Negative arguments may still be used as before.

Continued on Page Eight

NOVEMBER SOFTWARE CONTEST

By Stan Webb

This month Henry Lacy wins both first and second place in the major program category with two useful, well-documented entries. He receives first place for a combination of two programs, a Decimal Support Package which extends the capabilities of a program already in the Altair User Software Library, and a Decimal Output Routine to print the data run through the package.

He takes second place for his Self-Incrementing Hand Loader. This program is an octal (or binary, depending on your viewpoint) loader which requires minimal hardware, that is, only an 8800 (no terminal).

Third place goes to Lee Wilkinson for his simple, useful Accounts Receivable program designed for small businesses.

Darrel Van Buer takes first place in the subroutine for his Inverse Normal Distribution Function subroutine. This routine generates random numbers with a normal distribution.

First Place Major Program

#10-15-761
Author: Henry E. Lacy
Length: 153 bytes/136 bytes
Title: Decimal Support Package (requires #8-18-752)/ Decimal Output Routine

Second Place Major Program

#10-21-762
Author: Henry E. Lacy
Length: 74 bytes
Title: Self-Incrementing Hand Loader

Third Place Major Program

#10-19-761
Author: Lee Wilkinson
Length: 60 lines Altair BASIC
Title: Accounts Receivable

First Place Subroutine

#10-12-761
Author: Darrel Van Buer
Length: 16 lines Altair BASIC
Title: Inverse Normal Distribution Function

Entries Accepted into Library

#11-4-761
Author: Gordon Berry
Length: 32 lines Altair BASIC
Title: Standardized and Weighted Scores

Continued on Page Eight

The HEX$ and OCT$ Functions

Two new functions have been provided to make conversion between numbers and their hex or octal representations easy.

Functions:

```
HEX$(<integer formula>)
OCT$(<integer formula>)
```

Example:

```
PRINT HEX$(255),OCT$(255)
FF             377
```

Both HEX$ and OCT$ return a string value whose characters represent the hexadecimal or octal value of their argument. The resulting string does not have any leading or trailing spaces.

Changes in Error Messages

A new Error message has been added and an old error message changed.

A MISSING OPERAND (Code 29) Error message occurs if an operator is encountered during formula evaluation, but no operand succeeds it:

```
PRINT 2+
MISSING OPERAND
```

The UNDEFINED STATEMENT error message has been changed to print UNDEFINED LINE (UL in the 8K version), which is more appropriate.

Control-I and the Line Printer

Tab (Control-I) now works properly with the line printer and moves the printer carriage to the next eight character field.

New Features and Corrections to the Disk Version

Random Disk data files may no longer be LOADED as a program accidentally, causing unpredictable results. A BAD FILE MODE error will occur.

A new function has been provided in the Disk version to allow retrieval of error parameters when a DISK I/O ERROR occurs. ERR(Ø) returns the disk number of the disk. ERR(1) returns the track number (0-76) and ERR(2) returns the sector number.

The disk number is no longer included in the I/O error message.

If a LINK ERROR occurs while a file is being KILLed, the file name is now deleted from the directory.

The VARPTR Function

The VARPTR(<variable>) function returns the address of the variable given as the argument. If the variable has not been assigned a value during the execution of the program, a FUNCTION call error will occur.

The main use of the VARPTR function is to obtain the address of variable or array so it may be passed to an assembly language subroutine. Arrays are usually passed by specifying VARPTR(A[0]) so that the lowest addressed element of the matrix is returned.

(Note: All simple variables should be assigned values in a program before calling VARPTR of an array. Allocation of a new simple variable will cause the addresses of all arrays to change.)

#10-21-761
Author: Philip Romanik
Length: 30 lines HP BASIC
Title: Random
Random Number Generator

#10-18-761
Author: Jay Lucas
Length: 100 bytes
Title: Memory Test
Assembler memory test, a very thorough one.

#11-4-762
Author: Gordon Berry
Length: 300 bytes
Title: Print Registers

#10-25-761
Author: Byron Johnson
Length: 2 lines BASIC
Title: Extended Precision Square Roots

#10-25-762
Author: Byron Johnson
Length: 7 lines BASIC
Title: BASIC Line Renumbering
Renumbering program for 3.2 Extended BASIC.

#10-27-761
Author: Steven Armbruster
Length: 210 bytes 680 Assembler
Title: Political Influence

CN /Subscriptions

In the October issue of Computer Notes we stated that original subscriptions would expire at the end of December, 1976. Due to unforeseen problems in the development of an entirely new computer mailing system, we have extended all subscriptions to January, 1977. All those customers whose subscriptions will, at this time, expire should receive a notice approximately one month prior to the expiration date. The notice will describe both the terms and cost for an additional one year subscription to Computer Notes. Those persons who have already sent in their renewal orders will have these entered into the computer during the month of December, and they will become effective starting with the month of February. If you have any additional questions, please do not hesitate to contact us.

Mailbox

By David Le Jeune

David Le Jeune is Lieutenant Colonel in the Army Signal Corps.

MAILBOX is a combination amateur radio store and forward system which uses the Altair 8800 system and Altair Disk Extended BASIC to create a fully automatic message storage and retrievable system. For the first time MAILBOX is now available to amateur radio operators worldwide.

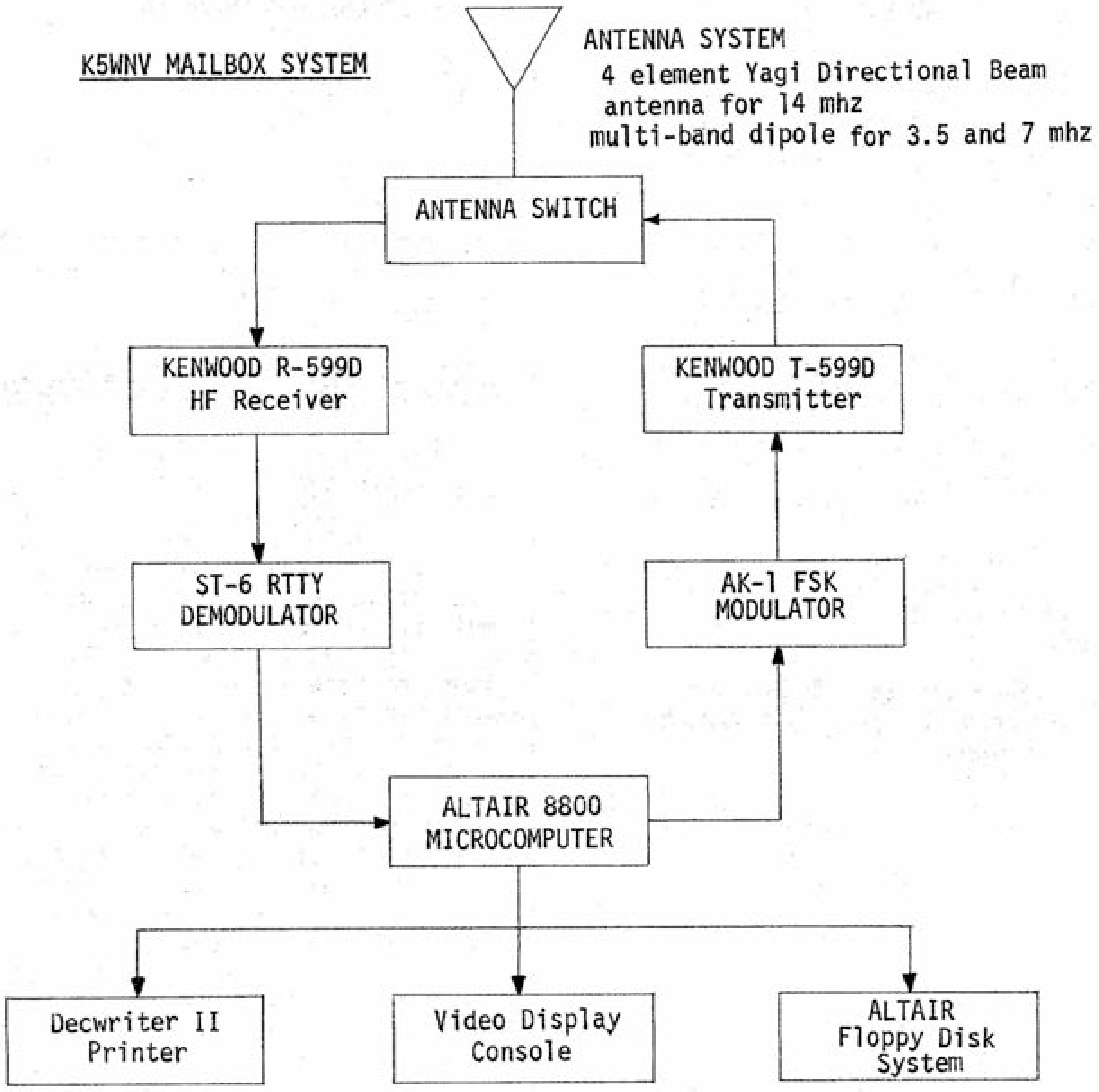
This system gives users remote access to my computer via amateur radio teletype and the ability to store messages on the system disk for later retrieval by the message addressee or anyone else. It also provides a near real time repeater capability, something that has been seen on ham radio teletype (RTTY), but never with the features provided by MAILBOX.

With previous systems, usually implemented with punched paper tape, the message to be relayed or repeated was entered and then the tape was transmitted. Unless there was an operator present at the relay station, it was impossible to back up the tape and retransmit the message. MAILBOX allows the user to enter a message into the system for relay and then ask for multiple transmission of that message.

There is also an editing capability not usually found on other repeater/relay systems. The random access store and forward system has never, to my knowledge, been implemented on the amateur radio HF bands.

MAILBOX consists of an amateur RTTY station interfaced to an Altair 8800 microcomputer system. (See Figure 1 for a block diagram of the system.) Signals are received on a Kenwood R-599D high frequency (HF) receiver and demodulated by an ST-6 RTTY frequency shift keying (FSK) demodulator. These signals are interfaced to the Altair 8800 via a serial I/O port. This same serial port keys an AK-1 FSK modulator driving a Kenwood T-599D HF transmitter with a 60 watt output. A directional antenna for 14 mhz and a dipole antenna for 3.5 mhz and 7 mhz are used for both transmitting and receiving. Other peripheral devices include an Altair Floppy Disk system, an LA36 Decwriter II printer and a video console.

The simplest RTTY station consists of a teleprinter, a modulator which converts received FSK signals to on-off keying for the teleprinter, an HF receiver, a transmitter, a modulator which converts the teleprinter keyboard on-off signals to FSK signals and an antenna system. Although the receiver must be of relatively good quality, the transmitter can be as simple as the transmitters used for morse code (CW) transmission. This simple station is all that is required to use the MAILBOX system. MAILBOX operates Monday through Friday on 14.075 mhz during the day and on 3.6125 mhz in the evening. Both the transmitter and receiver are crystal controlled for frequency stability.


K5WNV MAILBOX SYSTEM
ANTENNA SYSTEM
4 element Yagi Directional Beam antenna for 14 mhz
multi-band dipole for 3.5 and 7 mhz
ANTENNA SWITCH
KENWOOD R-599D HF Receiver
KENWOOD T-599D Transmitter
ST-6 RTTY DEMODULATOR
AK-1 FSK MODULATOR
ALTAIR 8800 MICROCOMPUTER
Decwriter II Printer
Video Display Console
ALTAIR Floppy Disk System

How the System is Used

Eight commands, which call up various features of MAILBOX are available to the user. Each command consists of the three-letter prefix WNV (the last three letters of my amateur radio call sign) followed by a three to six letter command descriptor. Since the system works in half duplex (i.e. neither station can transmit and receive simultaneously), the user follows each command with a carriage return and then turns off his transmitter to await acknowledgement or execution of his command.

The most commonly used command is WNVWRU which causes MAILBOX to respond with the following message:

```
DE K5WNV FORT HUACHUCA ARIZONA
MAILBOXES ARE OPEN
FOR INSTRUCTIONS SEND 'WNVINS'
FOLLOWED BY A CARRIAGE RETURN
```

The WNVWRU command is used to determine if MAILBOX is up and running and to check propagation conditions between my station and the user. The response will often also contain "system bulletins" providing information on new features, scheduling (mine or the computer's) or other system information.

The WNVINS command causes a detailed set of instructions on how to use the system to be transmitted. WNVDIR returns a list of MAILBOX users. The command WNVXXXI, where XXX represents the last three letters of a MAILBOX user's call sign, is used to enter a message into XXX's mailbox. The MAILBOX system will respond to that request with the following message:

```
"THE K4XXX MAILBOX IS OPEN"
```

(Here I have assumed XXX's full call is K4XXX.) All data received by the MAILBOX system subsequent to this message, and until a line appears containing a string of four or more "n's", is entered into a special file on disk. Upon receipt of the NNNN sequence, this file is closed and the system responds with a message verifying the file entry.

To check whether he has any messages in the system, the user types the WNVXXC command with his call substituted for XX. MAILBOX returns a value indicating the number of messages saved under his call sign. To retrieve a message, the command WNVXXXOYY, where YY represents the message number to be retrieved, is given.

MAILBOX (cont.)

If YY is a valid message number, the message is retrieved off disk and transmitted. It is preceded by the phrase "NEXT VIA K5WNV" and followed by the phrase "PREVIOUS VIA K5WNV" and a morse code indentification message.

In many instances the user may simply want to use the MAILBOX system as a near real time relay or repeater. Because of propagation conditions, a station in Florida and Virginia may not be able to hear each other, but both may be able to hear and work my station. The command WNVREL causes MAILBOX to respond with

"SEND YOUR MESSAGE TO BE RELAYED"

and saves everything subsequently received in a special "RELAY" disk file until the NNNN sequence is received. Either station can then cause the message to be retransmitted using the WNVRPT command. This command always causes the last message entered on disk via the WNVREL command to be re-transmitted.

The I/O routine also includes an ASCII to Baudot and Baudot to ASCII conversion routine, which is called whenever the Baudot ports are active.

The main program is written in Altair Disk Extended BASIC and makes extensive use of the disk file handling system. All "canned" messages, such as the WNVWRU and the WNVINS messages, are stored in sequential files on disk, as are messages stored by the users. The current version of Altair BASIC is 3.4, which has a neat error trapping capability. This has proven invaluable in debugging and system operation, since it allows the program to first turn off the transmitter if it happens to be on when an error occurs in the program prior to exiting to BASIC command level. In fact, if the error is of the non-fatal variety, the computer can be set to ignore it or print "ERROR MESSAGES" on the air to inform users of bad disk sectors or other software (and sometimes hardware) problems.

The entire program consists of about 175 lines of BASIC and an I/O routine which requires about 750 bytes. A future article will provide a step-by-step description of the program.

Conclusion

The MAILBOX system has shown the fantastic real-time capability of Altair BASIC. Admittedly, the system is operating with an I/O port at 45.45 baud, and this allows a lot of things to be done between letters input or output to this slow port. However, the PEEK and POKE, as well as IN and OUT commands make BASIC an unusually well-suited substitute for Assembly Language programming. I doubt that I would have attempted this if I had had to do it in Assembly Language. The Altair 8800 has proven exceptionally reliable operating in a somewhat "dirty" RFI environment. The only problems experienced have been with the Altair Disk Drive and associated electronics (not, however, the Disk Controller, which has worked beautifully). Considering that the drive has operated day in and day out, often with the drive motor running 12-18 hours a day, one really can't complain.

Maintenance turn around time between Albuquerque and Ft. Huachuca, AZ has been unbelievably fast, due in large part to the efforts of Wayne Cronin, MITS' resident disk repair genius. Writing the program and putting it on the air to watch the reaction of regulars on the 14.075 mhz autostart frequency, who are rarely turned on by "new fangled computers", was a lot of fun. So, if you have an amateur radio RTTY station or know someone who does, listen on 14.075 mhz or 3612.5 khz, or better yet, give the system a try by sending one of the commands described in this article.

MAILBOX Program

```
LIST

1010 WIDTH 73
1020 CLEAR 350
1030 S$="
1040 POKE &57773,1
1050 FORI=0TO5:READP$(I):NEXTI
1060 DATA BAD1, BAD2, BAD3, BAD4, BAD5, BAD6
1070 DIM C$(17),ID$(14),NA$(11),L$(8),I$(49)
1080 DEFINT A-Z
1090 BI=&57776:BO=&57777:DO=&57772:BF=&57756
1100 POKE BF,1
1110 CW=45
1120 DD=CW*.66:DU=CW*.88:DA=CW*2.09
1130 FORK=0TO49
1140 READI$(K)
1150 NEXTK
1160 DATA-,.,-,/,.,.,.,.,.,/,.,-,-,/,-,., /,.,.,.,-,/,/,/,/
1170 DATA-,.,.,-,.,/,-,-,.,.,.,/,/,/,/,/,/,/
1180 DATA/,/,/,/,/,/
1190 ON ERROR GOTO 2350
1200 FOR I=0 TO 17
1210 READ C$(I)
1220 NEXT
1230 DATA WRU,REL,WNV,QST,EID,FFC,SDZ,UAR,WPX,RAW,HAB,SRG,NSR,DXR
1250 DATA INS,DIR,RPT,CWID
1260 FOR I=0 TO 14
1270 READ ID$(I)
1280 NEXT
1290 DATA DAILY,RELAY,K5WNV,QST,K4EID,W6FFC,W8SDZ,K5UAR,WB6WPX
1300 DATA W6RAW,W1HAB,K6SRG,WB0NSR,K4DXR,INSTRU
1310 L1$="NNNNN"
1320 L2$="NEXT VIA K5WNV/7"
1330 L3$="PREVIOUS VIA K5WNV/7"
1340 L4$="DE DAVE K5WNV/7"
1350 DIM ID(14)
1360 FOR I=2 TO 13
1370 PRINT ID$(I);
1380 INPUT ID(I)
1390 NEXT
1400 POKE BI,1
1410 POKE DO,0
1420 LINEINPUTI$:IF INSTR(I$,"<")>0THEN 2290
1430 IF LEN(I$)<4 THEN 1420
1440 W=INSTR(I$,"WNV")
1450 IF W>0 THEN 1470
1460 GOTO 1420
1470 I$=MID$(I$,W+3,7):J1$=LEFT$(I$,4):FOR J=0 TO 17
1480 IF INSTR(J1$,C$(J))>0 THEN 1500
1490 NEXT J
1500 ON J+1 GOSUB 1540,1590,1810,1810,1810,1810,1810,1810
1510 ON J-7 GOSUB 1810,1810,1810,1810,1810,1810,2160,2090
1520 ON J-15 GOSUB 2080,2280
1530 GOTO 1420
1540 GOSUB 2390
1550 CLOSE 1: OPEN "I",1,"WRU"
1560 IF EOF(1) THEN 2430
1570 LINEINPUT #1,I$:PRINT I$
1580 GOTO 1560
1590 POKE BI,0:POKE BO,1:OUT0,1:PRINTS$
1610 PRINT"SEND MSG TO BE RELAYED":PRINT:OUT0,0:POKE BO,0
1630 POKE BI,1:CLOSE 1
1640 OPEN "O",1,"RELAY"
1650 LINEINPUT I$:IF LEN(I$)<4THEN 1650
1660 PRINT #1,I$
1670 IF INSTR(I$,"NNNN")=0 THEN 1650
1680 CLOSE 1: MB$="RELAY"
1690 GOSUB 2390:PRINT
1700 PRINTL2$
1710 OPEN "I",1,MB$
1720 IF EOF(1) THEN 1780
1730 LINEINPUT #1,O$
1740 GOSUB 2650
1750 IFY>0THEN 1720
1760 PRINTO$
1770 GOTO 1720
1780 CLOSE 1
```

Continued on Page Eleven

MAILBOX Program (cont.)

```
1790 PRINTL3$
1800 GOTO 2430
1810 I2$=MID$(I$,4,1)
1820 IF I2$="I" THEN 1940
1830 IF I2$="C" THEN 2230
1840 IF I2$<>"O" THEN 2250
1850 K=VAL(MID$(I$,5)):IFK>ID(J)THEN 1910
1860 IF K=0 THEN 1910
1870 K=K-1
1880 K$=MID$(STR$(K),2)
1890 MB$=ID$(J)+K$:CLOSE 1
1900 GOTO 1690
1910 GOSUB 2390
1920 PRINT"THERE IS NO MESSAGE"K"FOR "ID$(J)
1930 GOTO 2430
1940 K$=STR$(ID(J)):ID(J)=ID(J)+1
1950 K$=MID$(K$,2)
1960 MB$=ID$(J)+K$:K$=ID$(J)
1970 GOSUB 2390:PRINT"THE "ID$(J)" MAILBOX IS OPEN":OUT0,0
1980 POKE BI,1:POKE BO,0
1990 CLOSE 1
2000 OPEN "O",1,MB$
2010 LINEINPUT I$: IF INSTR(I$,"NNNN")>0 THEN 2050
2020 IF LEN(I$)<4 THEN 2010
2030 PRINT#1,I$
2040 GOTO 2010
2050 CLOSE 1
2060 GOSUB 2390:PRINT"THE MESSAGE FOR "K$" HAS BEEN ENTERED"
2070 GOTO 2430
2080 MB$="RELAY":CLOSE 1:GOTO 1690
2090 GOSUB 2390
2100 PRINT"DIRECTORY"
2110 FOR I=2 TO 13
2120 IF ID(I)=0 THEN 2130ELSE PRINTID$(I);ID(I),
2130 NEXT
2140 PRINT:PRINTL4$
2150 GOTO 2430
2160 MB$="INSTRU"
2170 GOSUB 2390:PRINT
2180 CLOSE 1:OPEN "I",1,MB$
2190 IF EOF(1) THEN 2220
2200 LINEINPUT #1,I$:PRINTI$
2210 GOTO 2190
2220 GOTO 2430
2230 PRINTL$(0):GOSUB 2390:PRINT:PRINTID$(J)" HAS"ID(J)" MESSAGES."
2240 PRINT:GOTO 2430
2250 PRINTL$(0):GOSUB 2610:PRINT:PRINT"COMMAND NOT UNDERSTOOD";
2270 RINT:GOTO 2430
2280 OUT0,1:GOTO 2430
2290 POKE BI,0: POKE BO,1:OUT 0,1
2300 LINEINPUT I$: IF INSTR(I$,"(")>0 THEN 2330
2310 IF INSTR(I$,"BK") THEN 2340
2320 GOTO 2300
2330 GOSUB 2430:POKE BI,1:POKE BO,0:GOTO 1420
2340 POKE BI,1:POKE BO,0:OUT 0,0:GOTO 1420
2350 IF ERR=53 THENRESUME 1010
2360 IF ERR=62 THENRESUME NEXT
2370 OUT0,0
2380 ON ERROR GOTO
2390 POKE BI,0:POKE BO,255
2400 OUT0,1
2410 FOR I=0TO20:WAIT0,128,128:OUT1,31:NEXT
2420 RETURN
2430 PRINTL1$:GOSUB 2470
2440 PRINT
2450 PRINT:POKE BO,0:OUT0,0:POKE BI,255
2460 RETURN
2470 WAIT0,128,128:OUT1,27
2480 FORI=0TO300:NEXTI
2490 FORU=0TO40
2500 IFI$(U)="-"THENI=DAELSEIFI$(U)="."THENI=DIELSEJ=DA:GOTO 2540
2510 OUT0,9
2520 FORK=0TOI
2530 NEXTK
2540 OUT0,1
2550 FORK=0TOJ
2560 NEXTK
2570 J=DU
2580 NEXTU
2590 FORI=0TO100:NEXTI
2600 RETURN
2610 POKE BO,1:POKE BI,0:OUT0,1:RETURN
2620 PRINT:FOR G=1 TO 71:PRINT"-";:NEXTG:PRINT:RETURN
2630 FOR G=1 TO 20:PRINT"-";:NEXTG:RETURN
2640 FORG=1 TO 20:PRINT" ";:NEXT:RETURN
2650 Y=INSTR(O$,"YY"):IFY>0THENRETURN
2660 X=INSTR(O$,"XX"):IFX=0THEN 2780
2670 IFX=1THEN 2720
2680 FORJ=XTO1STEP-1
2690 IF MID$(O$,J,1)=" "THEN 2720
2700 NEXTJ
2710 J=1
2720 FORK=J+1TOLEN(O$)
2730 IFMID$(O$,K,1)=" "THEN 2760
2740 NEXTK
2750 K=LEN(O$)
2760 U$=LEFT$(O$,J)+MID$(O$,K+1)
```

Continued on Page Twelve

HAM on the side

By Wayne Cronin

In this issue of Computer Notes Dave Le Jeune has written an article on his MAILBOX system. Dave has done a fantastic job putting together an advance RTTY store and forward system. His detailed article includes a BASIC source listing of the control program for those of you ambitious enough to set up a similar system. (Hopefully, it will also encourage more hams to write articles for us describing their own projects.)

Speculation on the Computer Repeater

If you would like to get into computers but don't have the time or money to put together a system of your own, maybe you can inspire your local repeater group to add a computer to its machine.

An example of this type of system is the two-meter computer repeater now under construction here in Albuquerque. Five local hams are conspiring to put up a new repeater with 6800-based computer power.

The repeater will operate as a normal voice relay until tone commanded to enter the computer mode. In this mode traffic on the input frequency will not be repeated to the output frequency. Input will consist of commands to the system monitor in Baudot coded FSK. The output frequency will be under control of the computer and will be busy only when the computer generates output data in response to user input. Another tone command will return the system to the standard repeat mode.

With suitable interface devices the computer will be capable of monitoring the repeater and log status information for the use of the key voltages, currents and time of measurement. If any of these parameters are outside a predefined tolerance range, system software will shut down the repeater and alert the control operator. Security of the repeater site will also be monitored. Analog to digital converters monitoring descriminator voltage and limiter current can be read by the computer, and their values relayed to the user give an indication of frequency error and signal strength.

Other system commands will allow users to store messages for other users in reserved memory areas, run standard software packages from a PROM library or enter programs from their own terminals. Since the system will not have access to a mass storage device, users will have to provide their own program storage on cassette or paper tape. (continued over)

Cell Research (cont.)

All of this data is recorded by using a highly sophisticated video system in conjunction with the microscope. A microthermister and related electronics are also used for temperature monitoring and control to within 0.1°C. The video signal is analog and digitally processed to provide an enhanced image. The video signal is recorded on VTR. Mixed with the video signal is the data in digital form for temperature, pressure and time. Later, the video tape is played back, enhanced and sliced, then observed on the monitor. Prior to development of the Altair system, the cell deformation data was hand punched on IBM cards and walked across campus to the Computer Center's IBM 370 for data processing.

Initially, the Altair computer was equipped with just the basics--one 4-slot motherboard, 4K of memory and one SIO board. To interface the Altair to the experimental system required much expansion. The motherboard was replaced with a 16-slot board. A Digital Equipment Corporation Decwriter was purchased and successfully interfaced to provide for programming and limited data acquisition. Some revision was needed on the MITS SIO board to accomodate the Decwriter. An 8K memory board was installed with future plans to include a 16K memory board. A National Multiplex Corporation digital data recorder was used for storage of programs and the assembler. For handling large volumes of data, a double-density dual-drive floppy disk system from Midwest Scientific Instruments was interfaced using the Processor Technology 3P+S I/O board. This provides over 1.2 megabytes of data storage allowing data input to be immediately dumped on the disk and subsequently processed. Investigation is underway to see if the software might be structured to allow for limited processing on a pseudo-time shared basis with the data dumping activity.

Data accuracy was improved by a Video Vector Calculator (VVC), designed and constructed by Vista Electronics in California. It's used to superimpose two cursors on the video picture. Using manual controls, the cursors are positioned at the limits of the deformed membrane and the VVC determines the distance between the cursors scaled in micron units. This data is provided as an LED digital readout as well as an analog signal. Thus, the VVC provides direct and accurate data with less than 1% error. However, the analysis still involved punched cards and use of the IBM 370 in the Computer Center. At this point the Altair was explored for possible use.

Some custom interfacing was done in developing an A/D system to extract maximum resolution from the video-analog signal. A Datel 12-bit A/D converter is the core of the custom interface card. It provides a maximum 40 μsec. conversion time for the 12 bits of straight binary parallel output. The appropriate logic was included for multiplexing two bytes to the data bus via two "hardwired" sequential addresses. The LSB byte contains 4 bits of data information and one flag bit signaling end of conversion. Currently, a floating point arithmetic scheme is used to handle the double-word data.

The software is an 8K Basic with the appropriate subroutines to handle the A/D and floating point. At a later time a more extensive assembly language program will be developed in conjunction with a floating point processor board. This will perform a fifty-fold speed increase in arithmetic operation and a significant reduction in memory requirements. The computer results are hand copied on the Decwriter at a convenient time. In the interim it is stored on disk.

Plans for future expansion include interfacing an incremental plotter to the system for immediate publishable graphs.

The Altair has proven to be a very reliable computer for specialized research application with excellent expansion capabilities. Its compact size and price advantage make it an attractive unit to the individual researcher whose budget limits access to the larger computers. We are seeing increasing interest in the microprocessor as a useful laboratory instrument throughout the university research community. The limitations of speed and word length necessitate some unique and, at times, cumbersome provisions to handle the data load. However, the general concensus is that these constraints will soon be historical, because now the microprocessor is rapidly closing a long existing gap in basic research instrumentation.

HAM (cont.)

When (and if) the system becomes completely operational, I will write a more detailed description for CN and provide a listing of the control routines.

Nets

K5WNV has informed me that there are three active RTTY auto-start nets devoted almost exclusively to computers. Nationwide users can try 14,075. East Cost users 3637.5, and West Coast users meet on 3612.5. All nets use narrow shift.

I'm still looking for information on any other computer nets that may be operating. If you want to start a new net, let me know when and where.

Mailbox Program (cont.)

```
2770 GOTO 2660
2780 FORI=0TO5:J=INSTR(O$,P$(I))
2790 IFJ>0THEN 2820
2800 NEXTI
2810 RETURN
2820 IFJ=1THENO1$=" "ELSEO1$=LEFT$(O$,J-1)
2830 IFLEN(O$)=J+LEN(P$(I))-1THENO2$=" "ELSEO2$=MID$(O$,J+LEN(P$(I)))
2840 O$=O1$+"-CENSORED-"+O2$
2850 GOTO 2780

OK
```

Altair Users

Alfred R. Howes
Box 342 Boyce Rd.
Glenford, NY 12433

Dick Fehriback
5779 Blaine SE
Grand Rapids, MI 49508
(616) 455-3138

Mr. Fehriback is interested in forming a club in the Grand Rapids area.

John J. Herro - W8CW
4419 Bascule Bridge Dr., #1321
Dayton, OH 45440

classified ads

For Sale: Four 4K Dynamic Boards, $100.00 each, or best offer. Contact:

Bill O'Toole
312 West Main St.
Emmitsburg, MD 21727
(301) 447-2690

Case Study:

DATA RECORDING WITH THE ALTAIR

By O. E. Dial

Mr. Dial is a professor in the Graduate School of Public Affairs at the University of Colorado.

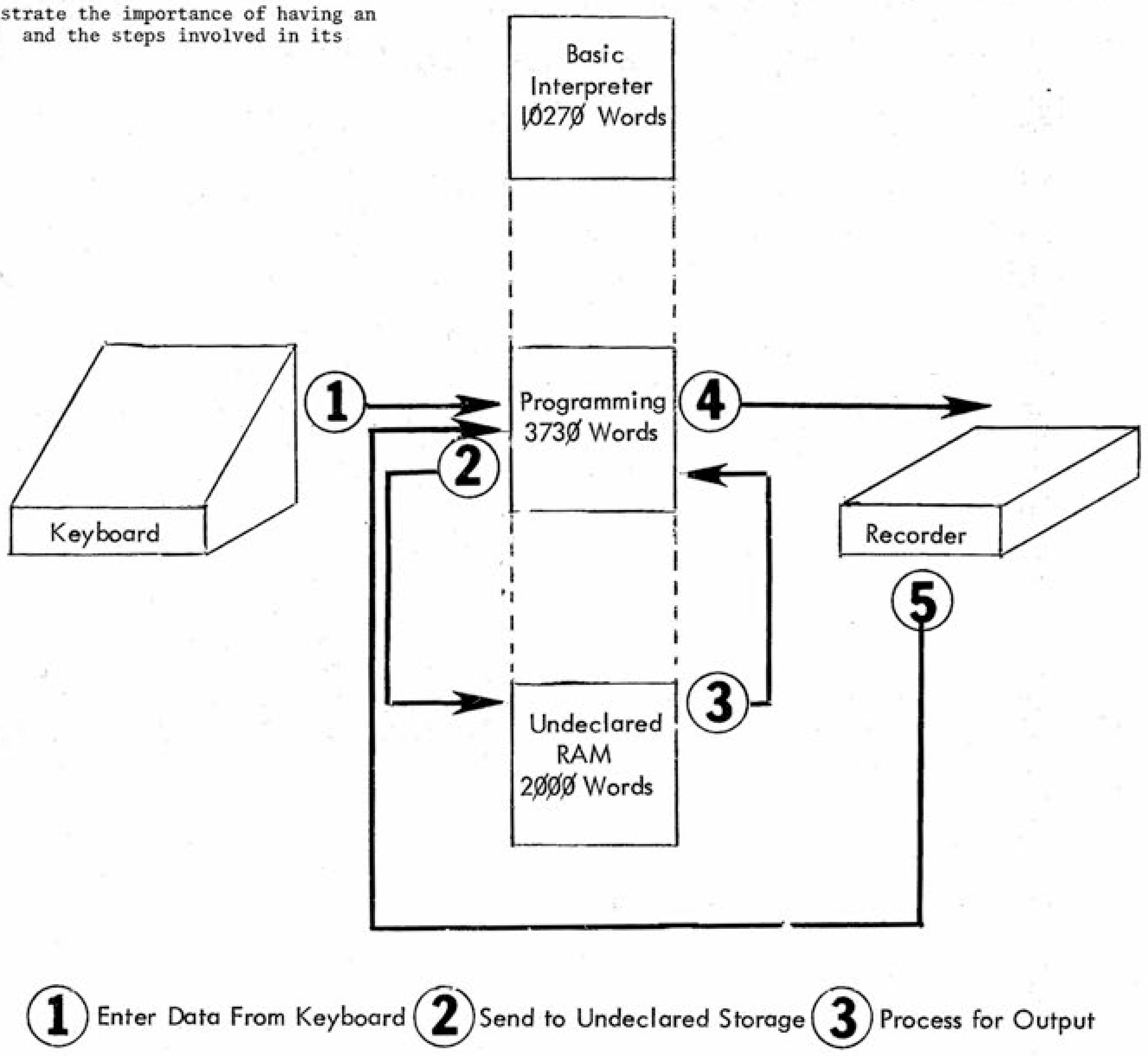
Recording programs from the Altair is quite easy. MITS documentation of the BASIC interpreter provides clear instructions in the use of CSAVE and CLOAD. Recording data is somewhat more complex. It involves the use of "undeclared" memory and the PEEK, POKE, INP, OUT, and VAL instructions, in addition to the concatenation of strings.

Existing documentation, even that contained in the many available textbooks on BASIC, doesn't provide sufficient information for recording data. Entering a lengthy data file from the keyboard is a dull, time-consuming, error-prone experience. To avoid this task of re-entry, we need some way of preserving the file in machine-readable media. We specifically need to know how to:

1. Temporarily store data in the undeclared reaches of memory
2. Record data (output the stored data and record it on a cassette tape)
3. Load a file from tape (input the data from cassette to undeclared memory)
4. Decode the string (input data from undeclared memory to the program in which it is to be used)

To accomplish any of these tasks requires an available region of "undeclared" memory, that is to say, memory which has not been revealed to the interpreter. When the question "MEMORY SIZE?" is asked, the operator should declare less than is available. The amount desired to be reserved for data storage (at one byte per character) should be subtracted from the total available and the difference entered in response to the question. The operator might also keep 2K in reserve.

The following graphic should illustrate the importance of having an undeclared region of memory in reserve and the steps involved in its utilization.



DATA STORAGE ROUTINES

Allocation of 16K Words of RAM

1 Enter Data From Keyboard 2 Send to Undeclared Storage 3 Process for Output

4 Output to Recorder 5 Load Data From Recorder

Continued on Page Fourteen

Data Recording (cont.)

1. Storing Data in Memory

Data must first be stored in undeclared memory before it can be output to the recorder. Two variations of the routines which accomplish this will also be presented. The following routine takes the data from a program statement.

```
1Ø        A$=" - - - not more than the permitted statement length - - -"

              (Any alpha, numeric, or other symbol may be contained
              within the quotes.)

15        Y=14ØØØ+LEN(A$)

              (This assumes 16ØØØ words of memory, of which 2K are
              reserved as undeclared memory. This statement will be
              used in connection with statement 6Ø, below.)

2Ø        FOR X=Ø TO LEN(A$) - 1

              (The "-1" is to compensate for the use of "Ø" in the
              array.)

3Ø        Z=ASC(MID$(A$,X + 1))

              (This picks off a character each time the loop is cycled,
              moving from left to right, defining the ASCII equivalent
              of each in turn. Again, the "+1" compensates for the X=
              Ø cell of the array.)

4Ø        POKE X + 14ØØØ,Z

              (This POKEs the word-length ASCII equivalent of each
              character, in turn, into undeclared memory, beginning at
              14K.)

5Ø        NEXT X

6Ø        ? "YOUR DATA IS STORED FROM 14ØØØ TO " Y "LOCATIONS."
```

Two variations of this program are needed in order to: 1) permit entering data from the keyboard and 2) enter data from READ statements.

a. Keyboard Data Entry. Entering data from the keyboard merely requires that a flag be inserted for END OF DATA and that a cumulative count be maintained of each LEN(A$). This permits the report of memory locations used.

```
5         Y = 14ØØØ

1Ø        INPUT "DATA";A$

              (If other than alphas, don't forget to enter in quotes.)

2Ø        IFA$="OUT OF DATA" THEN 1ØØ

              (This is simply a flag to end the input.)

3Ø        L=LEN(A$)

              (This obtains the character length of each string input.)

4Ø        FOR X=Ø TO L -1

5Ø        Z = ASC(MID$(A$, X + 1))

6Ø        POKE X + Y, Z

7Ø        NEXT X

8Ø        Y = Y + L

              (This is to cumulatively count the memory locations used.)

9Ø        GOTO 1Ø

1ØØ       ? "YOUR DATA IS STORED FROM 14ØØØ TO " Y "LOCATIONS."
```

b. READ Statement Data Entry. Entering data from READ statements is closely similar to the foregoing but does represent some new requirements.

```
5         Y = 14ØØØ

1Ø        READ A$

              (This could be changed to a FOR - NEXT loop if the number of
              strings of data were known, or, with an exit flag from the
              loop.)

2Ø        IF A$ = "OUT OF DATA" THEN 1ØØ

              (This provides a flag to end the input.)

3Ø        L = LEN (A$)

4Ø        FOR X = Ø TO L - 1

5Ø        Z = ASC(MID$(A$, X + 1))

6Ø        POKE X + Y, Z

7Ø        NEXT X

8Ø        Y = Y + L

9Ø        GOTO 1Ø

1ØØ       ?"YOUR DATA IS STORED FROM 14ØØØ TO " Y "LOCATIONS."

11Ø       DATA ABCD, "A.B.C.D.", "1,Ø", "3.4ABCD"

              (Note the selective use of quotes and commas.)
```

2. Recording Data.

First, the CSAVE is not used in recording data. Second, while I have listed the basic program below, for the sake of simplicity, it does not include certain desirable features such as will be found in 2.1.

```
1Ø        FOR X = 14ØØØ TO 16ØØØ

              (Note that this would read every word in undeclared
              memory.)

2Ø        Z = PEEK (X)

3Ø        WAIT 6, 7, 1

              (As explained in the MITS BASIC manual (p. 37), this
              reads the status of port "6", exclusive OR's bit "1" with
              that status, then AND's the result with "7" until a non-
              zero result is obtained.)

4Ø        OUT 7, Z

              (The stated test has been met, and work "Z" is output to
              the device via port "7".)

5Ø        NEXT X

              (Now the program indexes back to get the next character.)
```

a. Recording a Tape Leader. It is a good practice to output a stream of data having the same value as a prefix to the file. Among other things, this provides a check on the beginning of the file of data. Similarly, a stream of values can be appended to identify the end of the file. The value selected for this purpose should not be greater than 255 (the maximum decimal value which can be represented in eight bits) for this purpose. This can be accomplished by prefixing, or appending, the following routine to the program in 2.Ø.

```
1ØØ       FOR X = Ø TO 99

              (Vary the terminal index value consistent with the length
              of leader you desire.)

11Ø       Z = 255: WAIT 6, 7, 1 : OUT 7, Z : NEXT X

              (For additional information on WAIT and OUT, see the MITS
              Theory of Operation manual for the Serial I/O Board.)
```

3. Load a File from Tape.

This section assumes the file has been recorded in the manner stated in Section 2.

```
1Ø        FOR X = 14ØØØ TO 16ØØØ

              (Again, less than a range of 2ØØØ words, if unneeded.)

2Ø        WAIT 6, 1, 1

              (Check the status of the control port.)

3Ø        Z = INP (7)

              ("Z" will have the value of the word input at port 7.)

4Ø        POKE X, Z

              (The "Z" just read is POKEd into the 14ØØØth word location
              or memory in the first pass.)

5Ø        NEXT X
```

4. Entering Data from Memory.

Data is entered into undeclared memory as the binary equivalent of decimal numeric character representations, or, the binary of the ASCII code. To restore alphas, numerics, and other symbols, the data must be translated from its ASCII code. The CHR$ statement is used for this purpose.

The following routine can be incorporated into your program to permit entering data from undeclared memory.

```
1Ø        FOR X = 14ØØØ TO 16ØØØ

2Ø        Z = PEEK (X)

              (At this point we are reading the character in ASCII code.)
```

Continued on Page Fifteen

```
3Ø        ?CHR$ (Z) : NEXT X
```

(Now the character will be printed as originally represented, i.e., before its conversion to ASCII.)

a. Decoding the String. It's important to remember that rendering a string of values from undeclared memory does not ordinarily solve our problem of data re-entry. The string may contain numerous values which the operator may want to deal with individually. If this is true, some symbol must be inserted between each value in the string when it is originally stored in undeclared memory. A comma is useful for this purpose. The string may then be broken up to restore values between commas. Note that a comma must be used at the end of the last value and before the end-quote.

Furthermore, since what is being accomplished is breaking a string up in to many strings with each representing a datum value, it will be necessary to concatenate the pieces of the broken string, restoring each numeric character of the datum and then change its character from a string to a numeric variable. The VAL instruction is used for this purpose.

Since this may seem complex, perhaps an example would be helpful. Let's assume we have a string of data such as the following: "21, 34, 1, Ø, 345, 67.5, 21, .623,". PEEKing undeclared memory will result in calling the string character-by-character from left to right. In the process, we can break up the string containing eight data values into eight strings. But the newly created strings are still strings. We need a method of breaking them up character-by-character, letting each character be a string. For example, in the first datum ("21") when the "2" appears, give it the string name A$; and when the "1" appears, give it the string name B$. When the comma appears, it is regarded as a flag to let us know the value is completely rendered.

Our problem then is simply to concatenate the broken string and to give it a new name. For example, G$ = A$ + B$. The string, G$, would now read "21", and this may in turn be rendered into a numeric variable with the VAL statement. For example, H = VAL(G$). Use of this programming technique means that we will receive a sequence of values for H, at least until the last comma has been received. The program can be designed to process each "H" as it arrives, or each "H" can sequentially be placed in a matrix. The program below calculates a running average and so processes each value as it arrives.

```
200       Y = Ø : W = Ø : G = Ø
```

(The first two variables will be used as counters, and the last as an accumulator. This statement is used for initialization.)

```
21Ø       FOR X = 22ØØØ TO 22Ø38
```

(At the time the file was recorded, a note was made of the exact locations which were used, thus saving time and storage.)

```
22Ø       Z = PEEK (X): IF Z = 44 THEN 3ØØ
```

(The ASCII code for a comma is 44, thus if a comma is PEEKed, the program will branch to statement 3ØØ.)

```
23Ø       Y = Y + 1
```

(This will count the number of characters between commas.)

```
24Ø       ONLY GOTO 260, 270, 280, 290
```

(Here is the real value of the "Y" counter. The first character read will be directed to statement 260 where it will be given a unique string name, because "Y" has a value of "1". The second character will be directed to statement 270 because "Y" has a value of "2" and so on. Note that you must anticipate in your program the maximum number of digits and the decimal point, if any, which will be encountered in the data file. In this program the maximum value is 3 digits and one decimal point; hence, the "ON Y" statement provides four branches.)

```
26Ø       A$ = CHR$ (Z) : E$ = A$ : GOTO 4ØØ
```

(Here the ASCII code is translated into the symbol it represents and the string character is named "A$".)

```
27Ø       B$ = CHR$ (Z) : E$ = A$ + B$ : GOTO 4ØØ
```

(The second character in the string is now broken off and given the name, "B$". It is then concatenated with A$ and given the name E$. The program then branches to statement 4ØØ which indexes the loop and causes the next character to be PEEKed. Should that character have the ASCII value of "44", the full length of the datum will have been formed. It has two characters, A$ and B$, and the two concatenated are equal to E$.)

```
28Ø       C$ = CHR$ (Z) : E$ = A$ + B$ + C$ : GOTO 4ØØ
29Ø       D$ = CHR$ (Z) : E$ = A$ + B$ + C$ + D$ : GOTO 4ØØ
```

(Even though this is the last possible branch, we must still go back and PEEK for the comma. Hence, the branch, instead of merely continuing to the VAL statement.)

```
3ØØ       F = VAL (E$): Y=Ø:W=W+1
```

(Here we change E$ from a string variable and find its value. We also initialize the Y-counter for its use in connection with statement number 25Ø. With "W" we are beginning to count each datum in the set.)

```
31Ø       G = G + F : H = G / W
```

(At this point, we are processing each datum, accumulating it to prior values and then finding the average at that point.)

```
32Ø       ? W ; F ; G ; H ; : ?
```

(This statement will print each of the values in which we are interested and then cause the printer to index to the next line.)

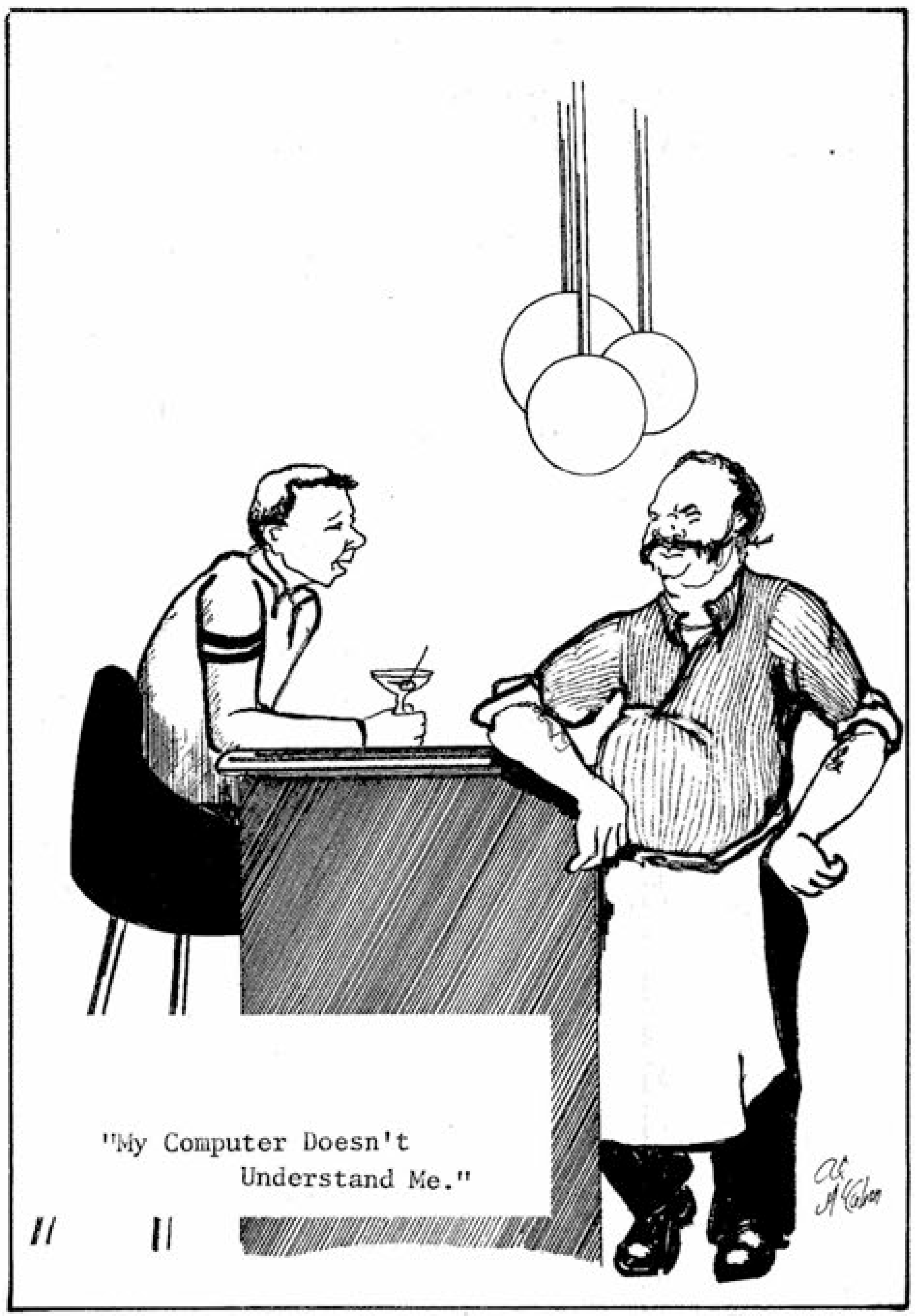
```
4ØØ       NEXT X
```

Finally, remember that PEEKing does not erase, so the file continues to be available so long as the machine is on, regardless of the mode of operation. This feature enables us to PEEK the same file as often as we may need to re-enter it into our programs.

* * *

The use of PEEK, CHR$, VAL, and concatenation, complete the loop in the movement of a file from keyboard to undeclared memory, then to tape, then re-entry from tape to undeclared memory, and finally, to a program where it is to be used. The routines represent valuable extensions to the use of ALTAIR BASIC. They should provide relatively fast, inexpensive, durable, dependable and accurate storage for as many files as you may wish to save.

Editor's Note: Altair BASIC Version 4.0 includes the capability to save and load data from audio cassette. For example, CSAVE.A saves the array A on cassette. The array can be recalled with the command CLOAD.A.



"My Computer Doesn't Understand Me."

Case Study:

ALTAIR Implementation in the Graphic Arts

by **Barry J. Yarkon**
Vice President, PhotoSystems
Graphicomposition, Inc.

Through this article I would like to share with you an interesting Altair implementation in a commercial typesetting company in New York City. While my specific application in type production might not apply directly to you (or in your company), some unique aspects of this system that "got-the-job-done" in manufacturing and administration could prove useful. So, regardless of our specific businesses, many "computer novice/ businessmen" face similar obstacles in learning to apply small computers (under small budgets) in our companies.

Before going into system specifics I'll first present a quick look at recent developments in typesetting technology from which to form a perspective:

The Graphic Arts industry is comprised of newspaper and periodical publishers, printing firms, commercial typographers and in-plant operations. Although it is the tenth largest industry nationally, most of the Industry's businesses are small owner-operated companies. Printing and typesetting have historically been craft-oriented trades utilizing complex mechanical machinery such as: printing presses, litho cameras, plate-making, binding, and linecasting machines.

Setting type used to be a highly skilled specialty involving only one person and a machine. The machine converted an operator's efforts directly into lines of type suitable for reproduction. Therefore, as a very labor-intensive activity, the typesetting industry was ripe for technological automation —particularly large volume operations such as the Government Printing Office and large newspapers. The first important invention was an automated version of the linecasting machine invented in the 1920s. It was driven by paper control tapes with perforations indicating which letters were to appear in a line of type—similarly to player piano rolls. Next came small hardwired computers which determined where and how a paragraph of text was divided into lines of equal length to form even margins. Tapes were then perforated to drive linecasters indirectly, replacing the operator who formerly sat at each machine.

It wasn't until the late 1960s that small programmable computers (digital controllers) became sufficiently cost effective to find any application in the average typesetting business. At this time, too, a technique for producing lines of type photographically on film (photocomposition) was maturing. Freed from the mechanical limitations inherent in molten metal linecasters, these photocomposing machines operated at high speeds—too fast for human interaction but just right for those small computers. So, by 1970 several companies were offering standalone photocomposing machines with built-in digital computers controlling all machine functions and having the intelligence to make many decisions formerly accomplished only by hand.

At present, the Graphic Arts industry is on an accelerated acceptance curve which closely approximates the general learning curve of microprocessor application by its vendors. Discrete-logic and microprocessor-based computers are now commonly found in many production tools from phototypesetters to text editing/input terminals. Ironically, application of small computers toward "office" procedures (i.e., record keeping, estimating, production planning, traffic control, etc.) has lagged far behind production uses in most Graphic Arts firms. And, far too often, the typical manager purchasing machinery with a small computer buried in it seldom has access to its computational power.

Introduction to Altair

As many of you did, I first learned of the Altair microcomputer revolution in *Popular Electronics* (January 1975). At that point I had already learned how to patch and how to add new subroutines to the control program resident in our company's phototypesetters. This was in direct, hand-compiled machine code (16-bit single-word instructions) into a vendor written main program—obviously not too productive. Many hours of home study using textbooks and articles had alerted me to some tantalizing possibilities minicomputers seemed to offer for our company: efficient text editing; text correction; file maintenance and data processing. But, how to proceed? And, at what cost? Although the time had seemed correct for computer utilization this concept was new to us and was possibly beyond our capabilities as a small company.

Introduction of the Altair 8800 kit brought the project into managable proportions in learning and cost investment. We no longer had an excuse to procrastinate!

That original 1/4-K kit, which seemed so imposing then, has grown into the system configuration shown in Figure 1. Central to our system is the 32K Altair 8800 with dual Altair floppy disc drives and several interface cards. An original ASR-33 Teletype had been replaced with an upper & lower case ASCII display terminal, a used 30 cps 80-column impact printer and an 8-level 80 cps paper tape reader. The tape reader is interfaced with a parallel card while the terminal and the printer simultaneously share one serial card. For printing, the system is switched (on the I/O card) for 300 baud; when no printout is needed, we switch the card and the terminal back to 4800 baud and turn off the printer's motor. By the way, the terminal is leased for 12 months at a cost of only $72 per month—we plan to upgrade at term for another with internal editing buffers.

Special Interface Hardware

Although I felt capable of learning to tackle the software components of the system, we required a custom interface design to provide two major functions: allow the Altair to transmit "finished" text to our phototypesetter's parallel port; and, secondly, to enable existing text tapes to be read by their special, bar-coded tape scanner into the Altair and eventually onto disc. Fortunately I had met another Altair user, Ronald Boley (Baron Technical Products) with the skill to design and install the interface hardware. Ron's "box" consists of a single serial card in the Altair 8800 communicating with a transmit/receive RS232C device over 100 feet of cable at 1200 baud. Figure 2 shows the setup transmitting from the Altair through the box to the phototypesetter. The handshaking scheme allows the phototypesetter to demand a character at a time (typically 16-18 mS) from the Altair which merely treats it as a rather slow line printer. I "send" whole strings by CONSOLEing to the serial card's address, PRINTing a string of text and CONSOLEing back to terminal address, echoing on the display, and so on.

When the switch (in Figure 2) is thrown to Dual Image® reader, the system now has control over the bar-coded tape scanner allowing old job tapes in 6-level parallel code to be received at the box and sent in serial discipline to the Altair—and onto disc.

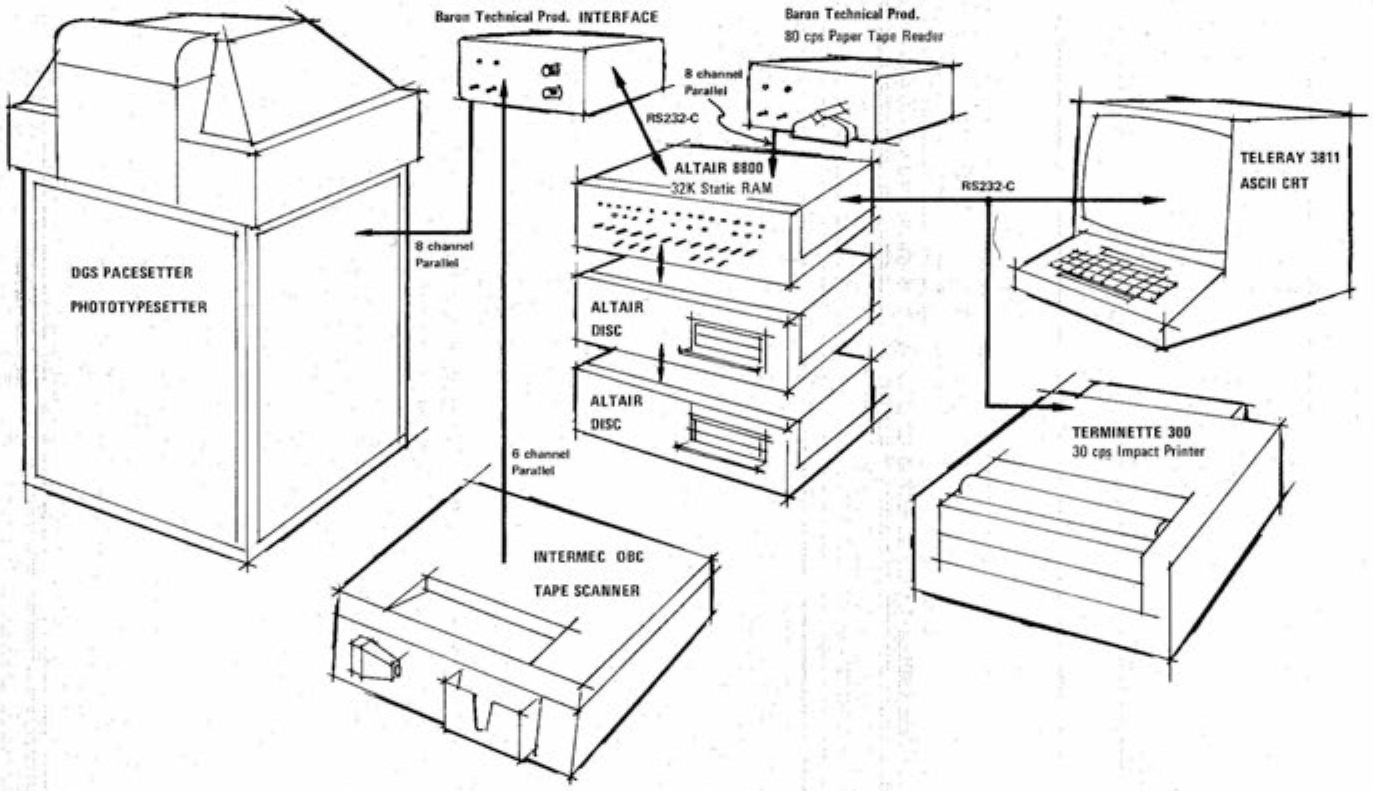


Figure 1. System configuration at Graphicomposition, Inc.

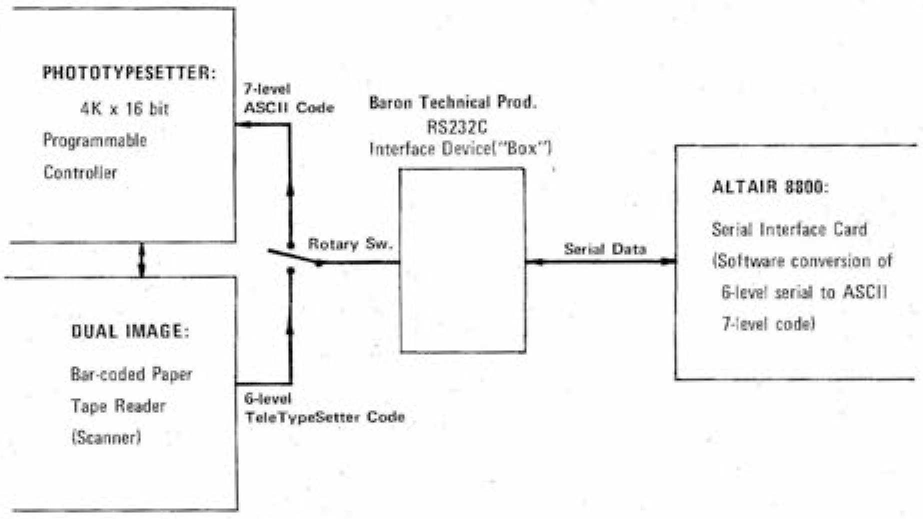


Figure 2. Detail of Altair/phototypesetter interface.

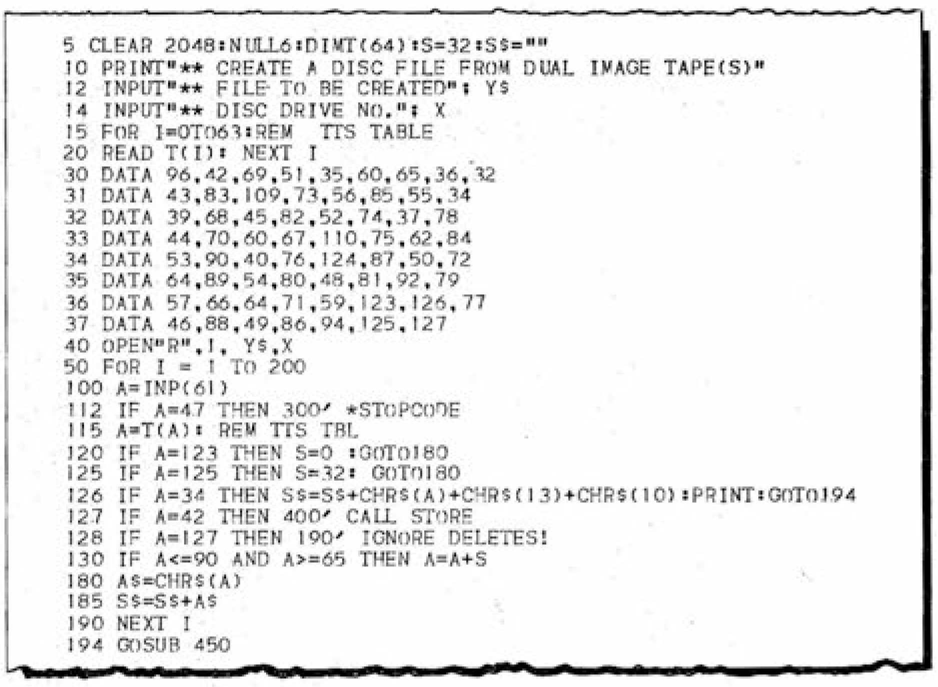
One interesting problem arose here due to the 6-level TeleTypeSetter code used on existing paper tapes. As 7-level ASCII is the standard for the Altair, terminal and printer each frame received from the tape scanner is first converted to an octal ASCII equivalent by the routine shown in Figure 3. Notice the routine must keep track of shift/unshift codes and add 32octal to uppercase ASCII letters to make the lowercase ASCII codes!

Disc Extended BASIC (ver. 3.4) has proven to be a very powerful tool in this application. The ability to control I/O ports and board status allowed a minimum of interface software. Intrinsic string manipulation commands (particularly INSTR) and file management functions have enabled me, self-taught remember, to write custom text editing and file manipulation programs that give my company a competitive advantage in the typesetting marketplace. Why, there are still functions in BASIC I haven't learned to use yet awaiting future application programs.

The Products

Graphicomposition services industrial and direct publishing accounts with art, design, typesetting and mechanical assembly. Figure 4 shows specimens from typical recent projects dealing with book publishers' promotional literature, i.e., catalogs, price lists, order forms, direct mail advertising. Our competitors can purchase (relatively expensive) equipment to do "electronic editing" of projects like these—our advantage is the ability to maintain small databases and to massage existing data into new products. Without this capability the client assembles each product from scratch and pays to typeset each product even though much of the information is the same among them. The largest single cost for typesetting services is the input labor cost—is it the same in your business?

To dramatize the levels of benefit using a flexible, small computer system refer to the specimens. The "master" book list is a product just as each of the two others. Previously our client prepared seperate manuscripts for each of these three and then sent them out for typesetting. Of course, the client paid for three jobs. Yet, looking more closely into the production cycle, you'll appreciate that the client also *proofread* each one, *corrected* each, sent each back to the typographer who then corrected each job and ran a second final proof—which the client then *reread*, etc. Our point is all of the information for the second two jobs already exists on the first list! We sort each of the other two from the first without incurring additional input labor cost. And, when we make corrections, deletions or additions to the master job *we actually correct each of the two others automatically*.



```
5 CLEAR 2048:NULL6:DIMT(64):S=32:S$=""
10 PRINT"** CREATE A DISC FILE FROM DUAL IMAGE TAPE(S)"
12 INPUT"** FILE TO BE CREATED": Y$
14 INPUT"** DISC DRIVE NO.": X
15 FOR I=0TO63:REM  TTS TABLE
20 READ T(I): NEXT I
30 DATA 96,42,69,51,35,60,65,36,32
31 DATA 43,83,109,73,56,85,55,34
32 DATA 39,68,45,82,52,74,37,78
33 DATA 44,70,60,67,110,75,62,84
34 DATA 53,90,40,76,124,87,50,72
35 DATA 64,89,54,80,48,81,92,79
36 DATA 57,66,64,71,59,123,126,77
37 DATA 46,88,49,86,94,125,127
40 OPEN"R",1, Y$,X
50 FOR I = 1 TO 200
100 A=INP(61)
112 IF A=47 THEN 300' *STOPCODE
115 A=T(A): REM TTS TBL
120 IF A=123 THEN S=0 :GOTO180
125 IF A=125 THEN S=32: GOTO180
126 IF A=34 THEN S$=S$+CHR$(A)+CHR$(13)+CHR$(10):PRINT:GOTO194
127 IF A=42 THEN 400' CALL STORE
128 IF A=127 THEN 190' IGNORE DELETES!
130 IF A<=90 AND A>=65 THEN A=A+S
180 A$=CHR$(A)
185 S$=S$+A$
190 NEXT I
194 GOSUB 450
```

Figure 3. Disc Extended BASIC (ver. 3.4) listing of the conversion routine.

Our client now prepares only one manuscript, proofreads just one set of proofs and receives all three lists for a substantially lower cost with less effort. This is a marketing advantage that might apply to your business too. It is especially valuable in repeatable or periodic applications.

Future Goals

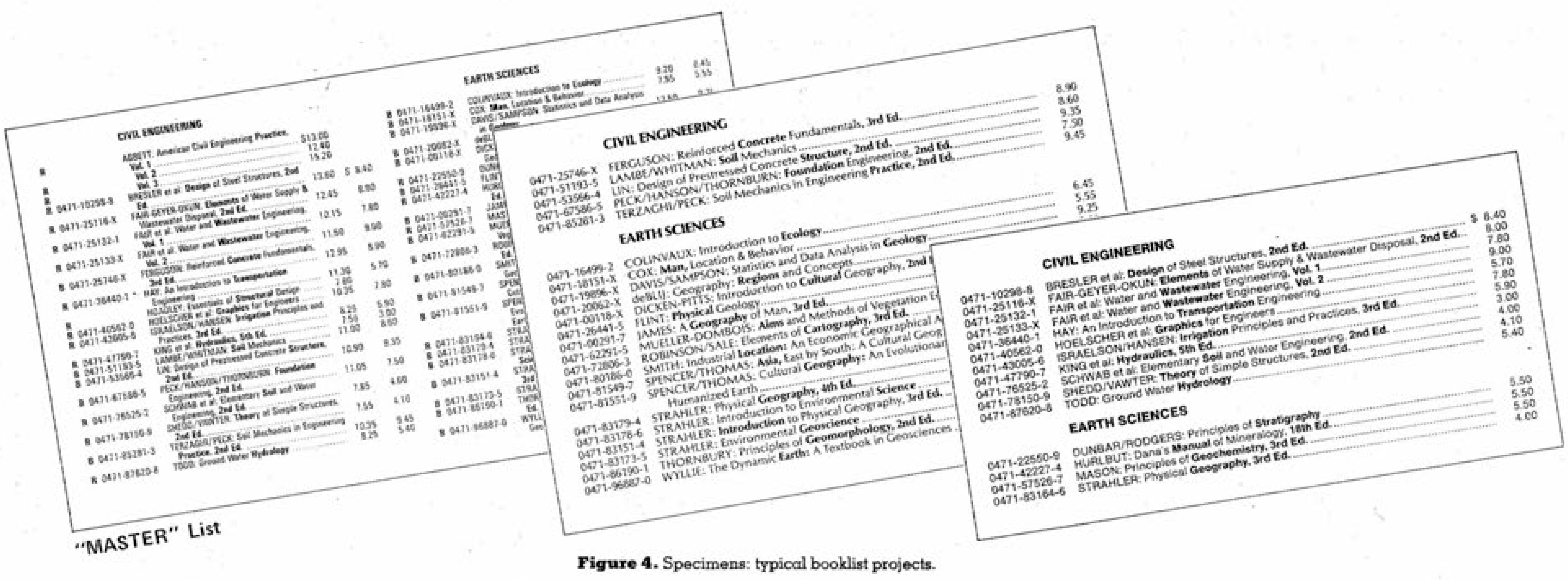
Production goals center on refining and expanding our library of job oriented file manipulation programs. We have found the ease of including prompt-lines on the terminal allows a program to guide input operators through each task. I would like to create a training package for our company, on diskette, to drill new employees in typesetting terminology and input procedures.

There has been little that we have thought to do that could not be handled by the system. For example, I have a model retail pharmacy dispensing/record keeping program that works well on this configuration. It requires a multi-user BASIC to be practical though. In response to continuing "can-we-do-this?" we now have a composition estimating program which steps sales people through the many steps in analysis and cost estimating of typesetting projects up for bidding. Naturally, this leads to other "can-we's" in management tools such as: billable time keeping, productivity analysis, production traffic and scheduling, accounting, etc.

I expect an administrative Altair will be our next short term goal. It seems, at this stage in our development, the limits are only programming time and imagination!

BIOGRAPHY: Barry J. Yarkon, vice president of Graphicompsoition, Inc., a New York City based commercial typesetting firm, is active in educational projects as well. A frequent writer and speaker in the Graphic arts industry, Mr. Yarkon teaches *Photocomposition for Managers* at the Evening School of Printing Industries NY and gives *Minicomputer Programming for Composition*, a two-day seminar series, in major cities around the country.

His well-travelled Altair 8800 had "logged almost 30,000 miles in airplane bellys" before being supplanted by a more portable 17K Altair 680b running 8K BASIC. "The Altair/BASIC combination has been an exceptionally effective tool in motivating newspaper and typesetting audiences to 'get their feet wet' in computer usage."


"MASTER" List

CIVIL ENGINEERING

0471-25746-X	FERGUSON: Reinforced **Concrete** Fundamentals, 3rd Ed.	8.90
0471-51193-5	LAMBE/WHITMAN: **Soil** Mechanics	8.60
0471-53566-4	LIN: Design of Prestressed Concrete **Structure, 2nd Ed.**	9.35
0471-67586-5	PECK/HANSON/THORNBURN: **Foundation** Engineering, **2nd Ed.**	7.50
0471-85281-3	TERZAGHI/PECK: Soil Mechanics in Engineering **Practice, 2nd Ed.**	9.45

EARTH SCIENCES

0471-16499-2	COLINVAUX: Introduction to **Ecology**	6.45
0471-18151-X	COX: **Man,** Location & Behavior	5.55
0471-19896-X	DAVIS/SAMPSON: Statistics and Data Analysis in **Geology**	9.25
0471-20062-X	deBLIJ: Geography: **Regions** and Concepts	
0471-00118-X	DICKEN-PITTS: Introduction to **Cultural** Geography, 2nd	
0471-26441-5	FLINT: **Physical** Geology	
0471-00291-7	JAMES: A **Geography** of Man, **3rd Ed.**	
0471-62291-5	MUELLER-DOMBOIS: **Aims** and Methods of Vegetation E	
0471-72806-3	ROBINSON/SALE: Elements of **Cartography, 3rd Ed.**	
0471-80186-0	SMITH: Industrial **Location:** An Economic Geographical A	
0471-81549-7	SPENCER/THOMAS: **Asia,** East by South: A Cultural Geog	
0471-81551-9	SPENCER/THOMAS: Cultural **Geography:** An Evolutionar Humanized Earth	
0471-83179-4	STRAHLER: Physical **Geography, 4th Ed.**	
0471-83178-6	STRAHLER: Introduction to Environmental **Science**	
0471-83151-4	STRAHLER: **Introduction** to Physical Geography, **3rd Ed.**	
0471-83173-5	STRAHLER: Environmental **Geoscience**	
0471-86190-1	THORNBURY: Principles of **Geomorphology, 2nd Ed.**	
0471-96887-0	WYLLIE: The Dynamic **Earth:** A Textbook in Geosciences	

CIVIL ENGINEERING

0471-10298-8	BRESLER et al: **Design** of Steel Structures, **2nd Ed.**	$ 8.40
0471-25116-X	FAIR-GEYER-OKUN: **Elements** of Water Supply & Wastewater Disposal, **2nd Ed.**	8.00
0471-25132-1	FAIR et al: Water and **Wastewater** Engineering, **Vol. 1**	7.80
0471-25133-X	FAIR et al: Water and **Wastewater** Engineering, **Vol. 2**	9.00
0471-36440-1	HAY: An Introduction to **Transportation** Engineering	5.70
0471-40562-0	HOELSCHER et al: **Graphics** for Engineers	7.80
0471-43005-6	ISRAELSON/HANSEN: **Irrigation** Principles and Practices, **3rd Ed.**	5.90
0471-47790-7	KING et al: **Hydraulics, 5th Ed.**	3.00
0471-76525-2	SCHWAB et al: Elementary **Soil** and Water Engineering, **2nd Ed.**	4.00
0471-78150-9	SHEDD/VAWTER: **Theory** of Simple Structures, **2nd Ed.**	4.10
0471-87620-8	TODD: Ground Water **Hydrology**	5.40

EARTH SCIENCES

0471-22550-9	DUNBAR/RODGERS: Principles of **Stratigraphy**	5.50
0471-42227-4	HURLBUT: Dana's **Manual** of Mineralogy, **18th Ed.**	5.50
0471-57526-7	MASON: Principles of **Geochemistry, 3rd Ed.**	5.50
0471-83164-6	STRAHLER: Physical **Geography, 3rd Ed.**	4.00

Figure 4. Specimens: typical booklist projects.

Ramblings from Ed Roberts

Thanks--every other month this year has broken all previous sales records at MITS. This is particularly remarkable when you realize sales for 1975 were three times the level of 1974. This rapid growth is attributable to a number of things, but the most important factor is the support and loyalty of our customers. Our customers have been vocal and unhesitating in their criticism whenever we erred, but have been equally supportive in positive advice and in purchasing MITS products. All of us here at MITS and at the Altair dealerships are deeply appreciative of your support and are pledged to continue to improve both the quantity and quality of our products and service.

Charlatans, Rip-Off Artists and Other Crooks

It may be a sign of the maturing of our business but there seems to be a proliferation of crooks and other devious operators in the small computer business. Let me address several of the "funny" deals individually.

1. Mail Order Fraud-recently an advertisement appeared in one of the trade journals advertising 13 used Altairs for sale. Based on information we have received this is a sham and the postal authorities have been notified. There have been a number of similar frauds called to our attention. The solution is don't send money to any group, company or individual whose reputation you can't verify in detail. The local Better Business Bureau is a useful starting place for this verification.
2. S-100 Bus - There is now an active attempt by a small group to steal the Altair bus. They are attempting to do this by changing the name so that they will not have to give recognition to MITS for its pioneering efforts in the small computer field. It is clear that the Altair bus is well established and the changing of the name to S-100 does not clarify or improve any situation for the user. It only helps the advertising copy of our competitors.

The Altair bus was designed at least a year prior to the appearance of any of these competitors. The correct name for the Altair bus is simply the Altair bus, it is not the Altair/IMSAI or the Altair/Polymorphic or the S-100, etc. Your help in stopping this sham will be appreciated, I hope you will identify the use of any name other than the Altair bus for what it is.

3. Phony Altair Dealerships: An Altair dealership is unquestionably the most desirable dealership in the small computer field. As many of you know these dealerships are given out very selectively and only to groups which have made major commitments to the support of Altair users. Any Altair dealer who doesn't support his customers is cancelled. I feel that whenever you purchase a MITS product from a dealer that you should have a warm feeling that if you have any problems your dealer will solve them for you.

Which brings me to the major point: there are a number of dealers which claim to be Altair dealers, but in reality are not. A few even have some obsolete "bootlegged" MITS products. In order to assist you in distinguishing the real Altair Computer Centers from the phonies I am including a list of our current Altair dealers. If a dealer is not on this list and claims to be an Altair dealer, caveat emptor. If you ever have any problems with a product purchased from one of the following dealers you can feel confident of having continued dealer as well as factory support (if not, let me know personally).

Altair Computer Centers

ALTAIR COMPUTER CENTER
4941 East 29th Street
Tucson, Arizona 85711
602/748-7363
Mr. Armand Sperduti

COMPUTER PRODUCTS UNLIMITED
2412 Broadway
Little Rock, Arkansas 72204
501/371-0449
Mr. Harry Mohrmann

COMPUTER KITS
1044 University Avenue
Berkeley, California 94710
415/845-5300
Mr. Pete Roberts

THE COMPUTER STORE
820 Broadway
Santa Monica, California 90401
213/451-0713
Mr. Dick Heiser

GATEWAY ELECTRONICS
2829 West 44th Avenue
Denver, Colorado 80211
303/458-5444
Mr. George Mensik

THE COMPUTER STORE, INC.
63 South Main Street
Windsor Locks, Connecticut 06096
203/871-1783
Mr. George Gilpatrick

MARSH DATA SYSTEMS
5404B Southern Comfort Blvd.
Tampa, Florida 33614
813/886-9890
Mr. Don Marsh

THE COMPUTER SYSTEMCENTER
3330 Piedmont Road
Atlanta, Georgia 30305
404/231-1691
Mr. Jim Dunion

ALTAIR SOFTWARE DISTRIBUTION CENTER
3330 Peachtree, N.E., Suite 343
Atlanta, Georgia 30305
404/231-2308
Mr. John Hayes

CHICAGO COMPUTER STORE
517 Talcott Road
Park Ridge, Illinois 60068
312/823-2388
Mr. Lou Van Eperin

THE COMPUTER STORE, INC.
120 Cambridge Street
Burlington, Massachusetts 01803
617/272-8770
Mr. Sidney Halligan

THE COMPUTER STORE OF ANN ARBOR
310 East Washington Street
Ann Arbor, Michigan 48104
313/995-7616
Mr. Peter Blond

THE COMPUTER ROOM
3938 Beau D'Rue Drive
Eagan, Minnesota 55122
612/452-2567
Mr. Dale Hagert

GATEWAY ELECTRONICS
8123-25 Page Blvd.
St. Louis, Missouri 63130
314/427-6116
Mr. Al Elkins

ALTAIR COMPUTER CENTER
2801 Cornhusker Highway
Lincoln, Nebraska 68504
402/466-1853
Mr. Gary Green

THE COMPUTER SHACK
3120 San Mateo, N.E.
Albuquerque, New Mexico 87110
505/883-8282
Mr. Pete Conner

THE COMPUTER STORE
269 Osborne Road
Albany, New York 12211
518/459-6140
Mr. Charles Olds

THE COMPUTER STORE OF NEW YORK
55 West 39th Street
New York, New York 10018
212/221-1404
Mr. Robert Osband

ALTAIR COMPUTER CENTER
Opening soon
Dayton, Ohio
513/252-6785
Mr. John Potter

ALTAIR COMPUTER CENTER
110 The Annex
5345 East 41st Street
Tulsa, Oklahoma 74135
918/664-4564
Mr. Ray Coons

ALTAIR COMPUTER CENTER
8105 S. W. Nimbus Avenue
Beaverton, Oregon 97005
503/644-2314
Mr. Richard W. Landon

BYTE'TRONICS
Suite 103, 1600 Hayes St.
Nashville, Tennessee 37203
615/329-1979
John & Stan Morrow

ALTAIR COMPUTER CENTER
Suite 206
5750 Bintliff Drive
Houston, Texas 77036
713/780-8981
Mr. Robert Burnett

ALTAIR COMPUTER CENTER
Computers-To-Go
6223 West Broad Street Rd.
Richmond, Virginia 23230
804/358-2171
Mr. Walter Witschey

MICROSYSTEMS
6605A Backlick Road
Springfield, Virginia 22150
703/569-1110
Mr. Russell Banks

ALTAIR COMPUTER CENTER
(The Computer Store)
Suite 5
Municipal Parking Bldg.
Charleston, West Virginia 25301
304/343-1360
Mr. Stephen Payne

8800 Software TidBITS

By Mark Chamberlin

In order to provide better support to the full line of Altair I/O boards and devices, the device address defaults and sense switch settings have been changed. Version 4.0 of Altair BASIC, the new Multi-Boot PROM and all future releases of 8800 software will support the new conventions.

Device Address Defaults

Device	Channels (octal)
SIO (A,B,& C) (Rev 1)	Ø,1
LINE PRINTER	2,3
88PIO	4,5
ACR	6,7
DISK	10,11,12,13
2SIO	20,21
4PIO	40,41,42,43
High Speed Reader	44,45,46,47

Sense Switch Settings

Sense switches A8 through A11 are encoded to indicate the load device, and switches A12 through A15 are used to indicate the terminal device. The codes are shown in the table below.

Examples:

1) To load BASIC from an ACR and come up talking to an SIOC, switches A8, A9, and A12 would be raised.
2) To load BASIC from a Teletype^{TM} connected to a 2SIO and come up talking to the same Teletype^{TM}, all switches would be down.

NOTE: 4.0 BASIC recognizes octal 16 as an indication of a non-standard board address. (See the 4.0 BASIC Manual for further details.) Non-standard board addresses are not supported by the Multi-Boot Loader or the new checksum loader.

The New Checksum Loader

The checksum loader has been changed to support the new device address defaults and sense switch settings.

If loading is proceeding properly, the Interrupt Enable light will remain off. Should an error occur, the Interrupt Enable light is turned on, the ASCII code for the error is stored in location Ø, and the error code is sent to all standard terminal devices. The error codes are:

C-checksum error--the computed checksum and the checksum on the tape are not the same.

I-Invalid load device--sense switches A8-A11 do not indicate a standard load device.

M-Memory error--a bad memory location or ROM has been encountered. The address of the "bad" location is stored in 1 and 2.

O-Overlay error--an attempt was made to load into the memory page on which the checksum loader resides.

The Multi-Boot PROM (MBL)

A preprogrammed 1702A PROM which facilitates the loading of all MITS software on paper tape and cassette is available for $45. When used in conjunction with an 88-PMC, the MBL PROM Memory Card eliminates the need to toggle in a bootstrap loader prior to loading a cassette on paper tape.

Device Type	Octal Code	Terminal SS Up	Load Device SS Up
2SIO (2 stop bits)	Ø	none	none
2SIO (1 stop bit)	1	A11	A8
SIOA,B,C (Rev 1)	2	A12	A9
ACR	3	A11,A12	A8,A9
4PIO	4	A13	A1Ø
88PIO	5	A11,A13	A8,A1Ø
High Speed Reader	6	A12,A13	A9,A1Ø
Terminal at Non-Standard Address	16	A12,A13,A14	Not Supported

The MBL PROM supports the new device address defaults and sense switch settings and is therefore best suited for loading 4.0 BASIC. However, it will also load 3.2 BASIC and 3.Ø Package II. In order to load these, it is necessary to first set the sense switches according to the new standards and then after the board is in progress to reset the switches according to the old standards.

The MBL PROM resides at 177000 octal, which is the next to last 256 byte block in memory. To load a tape, one simply examines 177000, sets the sense switches and activates the run switch. (If the I/O board being used requires software initialization, the loader should be run first and then the tape started. Otherwise, the tape should be started first.)

To order a copy of the MBL PROM, contact the Marketing Department at MITS.

BASIC Paper Tapes

Have you ever encountered the problem of dropping the first digit of a line number while loading a paper tape of a BASIC program? This happens because BASIC "crunches" each line of the program as it is read in. The crunching process may require more than 1/10 of a second for a long or complex line, so a character may be lost from the next line read.

This problem can be alleviated by punching some nulls (ASCII Ø) between the program lines. This can be accomplished by using the NULL command in all versions of Altair BASIC except 4K. In the 4K version, the location NULCNT must be patched. (See the Altair BASIC Manual for details.)

However, if you have tapes that have no nulls between lines, problems can still occur. The following machine language program for the 8800 reads a paper tape into memory and then punches a new one with nulls between the lines.

The BASIC paper tape program was written to work with a 2SIO board, but it can be easily modified to work with any Altair I/O board by changing the initialization and the I/O routines.

The Checksum Loader Listing is shown on pages 20-23. The BASIC Paper Tape Program listing is shown on page 25.

4K Checksum Loader Listing

```
                            00100          TITLE   4.0 CHECKSUM LOADER /MLC/FTC
                            00200
                            00300          SEARCH  MCS808
                            00400          SALL
000000'                     00500          MCSSIM(START)
                            00600
                   000004   00800  SIZE==↑D4                              ;HOW MANY K
                            00900
                            01300
007400'                     01500          RELOC   SIZE*2000-400          ;ONE PAGE BELOW SIZE
                            01700
007400' 001000     000363   01800  START:  DI                             ;TURN OFF INT ENABLE LIGHT
                            04000
                            04100
007401' 001000     000257   04200          XRA     A                      ;INIT 4PIO AND HIGH SPEED READER
                            04300
007402' 001000     000323   04800          OUT     42                     ;NON PROM VERSION ONLY INITS
007403' 000000     000042
                            05400
007404' 001000     000057   05500          CMA                            ;OUTPUT SIDE OF 4PIO
007405' 001000     000323   05600          OUT     43
007406' 000000     000043
                            05700
                            06100
007407' 001000     000076   06200          MVI     A,54
007410' 000000     000054
                            06600
007411' 001000     000323   06700          OUT     42
007412' 000000     000042
                            07500
                            07600  ; INIT THE 2SIO BOARD
                            07700
                            08100
007413' 001000     000076   08300          MVI     A,3
007414' 000000     000003
007415' 001000     000323   08500          OUT     20
007416' 000000     000020
007417' 001000     000333   08600          IN      377                    ;READ THE SENSE SWITCHES
007420' 000000     000377
007421' 001000     000346   08700          ANI     20                     ;SEE IF ONE OR TWO STOP BITS
007422' 000000     000020
007423' 001000     000017   08800          RRC
007424' 001000     000017   08900          RRC
007425' 001000     000306   09000          ADI     21
007426' 000000     000021
007427' 001000     000323   09100          OUT     20
007430' 000000     000020
                            09200
                            10000
007431' 001000     000061   10200          LXI     SP,SIZE*2000+$CODE
007432' 000000     010000'
007433' 000000     000000
                            10400
007434' 001000     000333   10500          IN      377                    ;READ SENSE SWITCHES
007435' 000000     000377
007436' 001000     000346   10600          ANI     17                     ;MASK OFF LOAD DEVICE BITS
007437' 000000     000017
                            10700
007440' 001000     000376   10800          CPI     7                      ;MAKE SURE LOAD DEVICE IS VALID
007441' 000000     000007
007442' 001000     000362   10900          JP      IERR                   ;IT ISN'T VALID
007443' 000000     007612'
007444' 000000     007432'
                            11000
007445' 001000     000041   11100          LXI     H,TABLE                ;MAKE H&L POINT TO LOAD
007446' 000000     007654'
007447' 000000     007443'
                            11105                                         ;DEVICE TABLE
                            11200
007450' 001000     000006   11400          MVI     B,0
```

```
007451' 000000  000000
                          11600
                          12110
007452' 001000  000117    12200             MOV    C,A                   ;MULT INDEX BY THREE
007453' 001000  000207    12300             ADD    A
007454' 001000  000201    12400             ADD    C
007455' 001000  000117    12500             MOV    C,A
007456' 001000  000011    12600             DAD    B                     ;AND ADD TO TABLE START ADDRESS
                          12700
                          12900
                          13000    ; MODIFY INPUT ROUTINE
                          13100
007457' 001000  000176    13200             MOV    A,M                   ;DATA CHANNEL ADDRESS
007460' 001000  000062    13300             STA    GET+10
007461' 000000  007646'
007462' 000000  007446'
007463' 001000  000075    13400             DCR    A                     ;STATUS CHANNEL ADDRESS
007464' 001000  000062    13500             STA    GET+1
007465' 000000  007637'
007466' 000000  007461'
007467' 001000  000043    13600             INX    H
007470' 001000  000176    13700             MOV    A,M                   ;JZ OR JNZ
007471' 001000  000062    13800             STA    GET+4
007472' 000000  007642'
007473' 000000  007465'
007474' 001000  000043    13900             INX    H
007475' 001000  000176    14000             MOV    A,M                   ;MASK
007476' 001000  000062    14100             STA    GET+3
007477' 000000  007641'
007500' 000000  007472'
                          14300
                          18800
                          18900    ; NOW SCAN FOR KEY BYTE ON TAPE
                          19000
007501' 001000  000315    19100    SCAN:    CALL   GET
007502' 000000  007636'
007503' 000000  007477'
007504' 001000  000376    19200             CPI    74                    ;START OF BLOCK?
007505' 000000  000074
007506' 001000  000312    19300             JZ     READ                  ;YES - GO READ THE BLOCK
007507' 000000  007530'
007510' 000000  007502'
007511' 001000  000376    19400             CPI    170                   ;START ADDRESS?
007512' 000000  000170
007513' 001000  000302    19500             JNZ    SCAN                  ;NO - SKIP IT
007514' 000000  007501'
007515' 000000  007507'
007516' 001000  000315    19600             CALL   GET                   ;BUILD PROGRAM START ADDRESS
007517' 000000  007636'
007520' 000000  007514'
007521' 001000  000117    19700             MOV    C,A                   ;AND JUMP TO IT
007522' 001000  000315    19800             CALL   GET
007523' 000000  007636'
007524' 000000  007517'
007525' 001000  000151    19900             MOV    L,C
007526' 001000  000147    20000             MOV    H,A
007527' 001000  000351    20400             PCHL
                          20500
                          20600    ; READ AND LOAD DATA BLOCK
                          20700
                          20800
                          21200
007530' 001000  000315    21300    READ:    CALL   GET                   ;GET BYTE COUNT
007531' 000000  007636'
007532' 000000  007523'
007533' 001000  000117    21400             MOV    C,A                   ;SAVE IT IN C
007534' 001000  000006    21500             MVI    B,0                   ;CLEAR CHECKSUM
007535' 000000  000000
007536' 001000  000315    21600             CALL   GET                   ;PUT LOAD ADDRESS IN D&E
007537' 000000  007636'
007540' 000000  007531'
```

4K Checksum Loader Listing (cont.)

```
007541' 001000  000137      21700            MOV    E,A
007542' 001000  000315      21800            CALL   GET
007543' 000000  007636'
007544' 000000  007537'
007545' 001000  000127      21900            MOV    D,A
                            22000
007546' 001000  000172      22100   DATA:    MOV    A,D                      ;ARE THEY TRYING TO LOAD
                            22200                                            ;INTO OUR PAGE? IF SO SEND
                            22300                                            ;THEM AN OVERLAY MESSAGE
                            22320
007547' 001000  000376      22330            CPI    START/400
007550' 000000  000017
                            22340
007551' 001000  000076      22345            MVI    A,"O"
007552' 000000  000117
007553' 001000  000312      22400            JZ     ERR
007554' 000000  007614'
007555' 000000  007543'
                            22500
007556' 001000  000315'     22600            CALL   GET                      ;GET A DATA BYTE
007557' 000000  007636'
007560' 000000  007554'
007561' 001000  000353      22700            XCHG                            ;LOAD ADDRESS TO H&L
007562' 001000  000167      22800            MOV    M,A                      ;STORE THE DATA BYTE
007563' 001000  000276      22900            CMP    M                        ;DID DATA STORE OK?
007564' 001000  000076      23000   MERR:    MVI    A,"M"                    ;JUST IN CASE IT DIDN'T
007565' 000000  000115
007566' 001000  000302      23100            JNZ    ERR                      ;NO - WE HAVE MEMORY ERROR
007567' 000000  007614'
007570' 000000  007557'
007571' 001000  000043      23200            INX    H                        ;YES - BUMP THE LOAD ADDRESS
007572' 001000  000353      23300            XCHG                            ;MOVE IT BACK TO D&E
007573' 001000  000015      23400            DCR    C                        ;DECREMENT BYTE COUNT
007574' 001000  000302      23500            JNZ    DATA                     ;MORE TO READ THEN DO SO
007575' 000000  007546'
007576' 000000  007567'
007577' 001000  000110      23600            MOV    C,B                      ;SAVE CALCULATED CHECKSUM
007600' 001000  000315      23700            CALL   GET                      ;READ CHECKSUM OFF TAPE
007601' 000000  007636'
007602' 000000  007575'
007603' 001000  000271      23800            CMP    C                        ;ARE THEY THE SAME?
007604' 001000  000312      23900            JZ     SCAN                     ;YES - LOOK FOR NEXT BLOCK
007605' 000000  007501'
007606' 000000  007601'
007607' 001000  000076      24000            MVI    A,"C"                    ;NO - WE HAVE CHECKSUM ERROR
007610' 000000  000103
007611' 000000  000001      24100            DB     1                        ;SKIP TRICK
007612' 001000  000076      24200   IERR:    MVI    A,"I"                    ;INVALID LOAD DEVICE
007613' 000000  000111
007614' 001000  000062      24300   ERR:     STA    $CODE                    ;STORE ERROR CODE
007615' 000000  000000'
007616' 000000  007605'
007617' 001000  000042      24400            SHLD   1+$CODE                  ;STORE BAD MEMORY ADDRESS
007620' 000000  000001'
007621' 000000  007615'
007622' 001000  000373      24500            EI                              ;TURN ON INT ENABLE LIGHT
007623' 001000  000323      24600   ERRMES:  OUT    1                        ;HANDLE SIO BOARD
007624' 000000  000001
007625' 001000  000323      24700            OUT    21                       ;HANDLE 2SIO BOARD
007626' 000000  000021
007627' 001000  000323      24800            OUT    5                        ;HANDLE 88PIO BOARD
007630' 000000  000005
007631' 001000  000323      24900            OUT    43                       ;HANDLE 4PIO BOARD
007632' 000000  000043
007633' 001000  000303      25000            JMP    ERRMES
007634' 000000  007623'
007635' 000000  007620'
                            25100
007636' 001000  000333      25300   GET:     IN     0                        ;READ STATUS CHANNEL
007637' 000000  000000
007640' 001000  000346      25400            ANI    0                        ;MASK OFF INPUT STATUS BIT
```

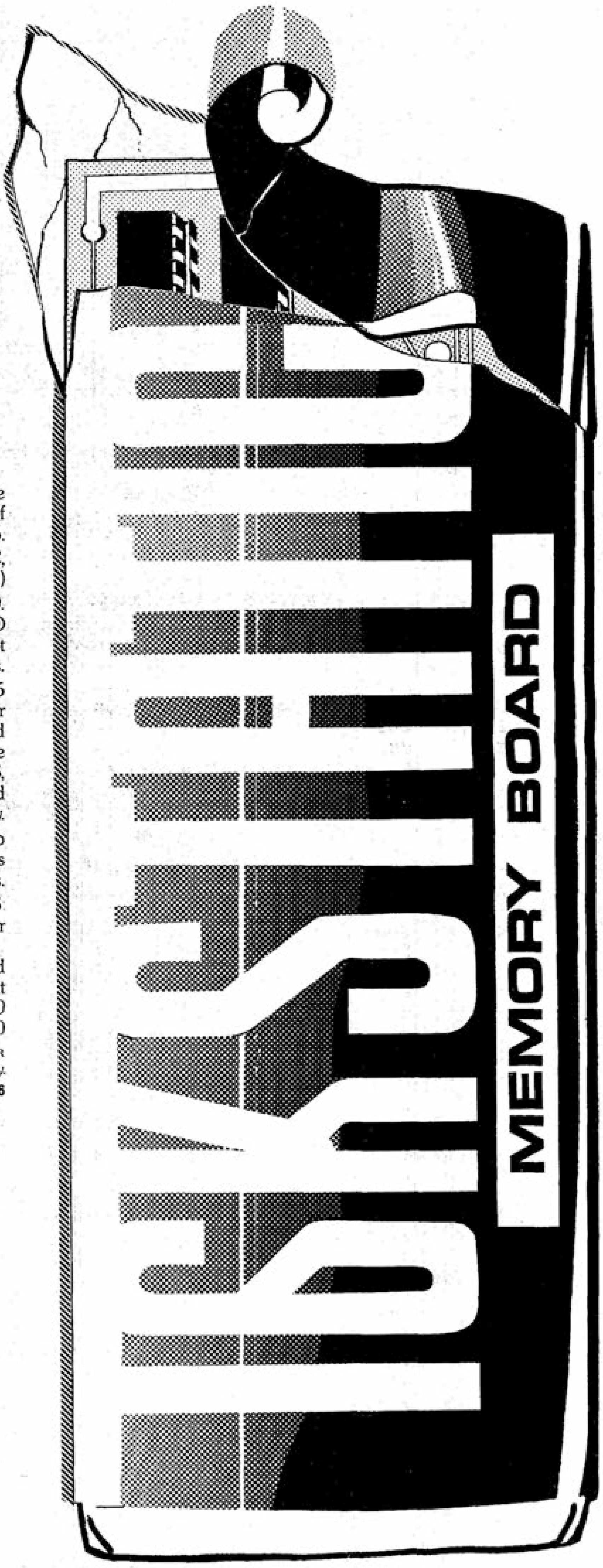
```
007641´ 000000  000000
007642´ 001000  000312       25500            JZ      GET                 ;NOT READY THEN CHECK AGAIN
007643´ 000000  007636´
007644´ 000000  007634´
007645´ 001000  000333       25600            IN      0                   ;READ THE DATA CHANNEL
007646´ 000000  000000
007647´ 001000  000365       25700            PUSH    PSW                 ;SAVE THE CHAR
007650´ 001000  000200       25800            ADD     B                   ;ADD IT TO CHECKSUM
007651´ 001000  000107       25900            MOV     B,A                 ;MOVE CHECKSUM BACK TO B
007652´ 001000  000361       26000            POP     PSW                 ;RESTORE CHARACTER
007653´ 001000  000311       26100            RET                         ;RETURN
                             26300   ; HERE ARE THE LOAD DEVICE TABLES
                             26400   ; BYTE 1 = DATA CHANNEL ADDRESS
                             26500   ; BYTE 2 = ACTIVE HIGH OR LOW
                             26600   ; BYTE 3 = INPUT STATUS BIT MASK
                             26700   ;
007654´ 000000  000021       26800   TABLE:   DB      21                  ;2SIO WITH 2 STOP BITS
007655´ 000000  000312       26900            DB      312                 ;ACTIVE HIGH
007656´ 000000  000001       27000            DB      1                   ;BIT 0
                             27100
007657´ 000000  000021       27200            DB      21                  ;2SIO WITH 1 STOP BIT
007660´ 000000  000312       27300            DB      312                 ;ACTIVE HIGH
007661´ 000000  000001       27400            DB      1                   ;BIT 0
                             27500
007662´ 000000  000001       27600            DB      1                   ;SIO BOARD
007663´ 000000  000302       27700            DB      302                 ;ACTIVE LOW
007664´ 000000  000001       27800            DB      1                   ;BIT 0
                             27900
007665´ 000000  000007       28000            DB      7                   ;ACR
007666´ 000000  000302       28100            DB      302                 ;ACTIVE LOW
007667´ 000000  000001       28200            DB      1                   ;BIT 0
                             28300
007670´ 000000  000041       28400            DB      41                  ;4PIO BOARD
007671´ 000000  000312       28500            DB      312                 ;ACTIVE HIGH
007672´ 000000  000200       28600            DB      200                 ;BIT 7
                             28700
007673´ 000000  000005       28800            DB      5                   ;88PIO
007674´ 000000  000312       28900            DB      312                 ;ACTIVE HIGH
007675´ 000000  000002       29000            DB      2                   ;BIT 1
                             29100
007676´ 000000  000045       29200            DB      45                  ;HIGH SPEED READER
007677´ 000000  000312       29300            DB      312                 ;ACTIVE HIGH
007700´ 000000  000200       29400            DB      200                 ;BIT 7
                             29500
007701´                      29600   LASTWR::
                007643´      29700   .C1==:.P
007701´ 000000  000000       29800   .C2::0
007702´ 000000  000000       29900   .C3::0
                             30000            END


NO ERRORS DETECTED

PROGRAM BREAK IS 007703
CPU TIME USED 00:04.298

4K CORE USED
```

Something Sweet for your altair T.M. 680-b



MITS is pleased to announce the development of a 16K static card for the Altair 680b. With an access time of 215 nanoseconds and low power consumption of 5 watts, we feel that this is an excellent addition to the Altair 680b.

To sweeten the pot even more, we are including a free copy of Altair 680 BASIC, assembler, and text editor on paper tape. ($275 value)

Altair 680 BASIC is identical to the 8K BASIC developed for the Altair 8800. Features include Boolean operators, the ability to read or write a byte from any I/O port or memory location, multiple statements per line, and the ability to interrupt program execution and then continue after the examination of variable values.

Other features of Altair 680 BASIC include variable length strings (up to 255 characters), with LEFT$, RIGHT$ and MID$ functions, a concatenation operator and VAL and STR$ to convert between strings and numbers. Both string and numeric arrays of up to 30 dimensions can be used. Nesting of loops and subroutine calls is limited only by available memory. Intrinsic functions include: SIN, COS, TAN, LOG, EXP, SQR, SGN, ABS, INT, FRE, RND and POS, in addition to TAB and SPC in PRINT statements. Altair 680 BASIC takes 7K bytes of memory.

MITS has also developed an expander card for the Altair 680b that lets you add up to three boards inside the main case. Read "Computer Notes" for announcements of additional Altair 680b boards.

PRICES

Altair 680-BSM, 16K Static Memory Board, including Altair 680 BASIC, assembler and text editor	$685.00 kit $865.00 assembled
Altair 680-MB Expander Card with one Edge Connector	$24.00 kit
Altair 680 BASIC (purchased separately)	$200.00
Altair 680 assembler and text editor (purchased separately)	$ 75.00

PRICE APPLIES ONLY TO PURCHASERS OF ALTAIR 680b COMPUTER
Prices, specifications subject to change. Allow 30-60 days for delivery.

MITS, Inc. 2450 Alamo S.E. / Albuquerque, New Mexico 87106

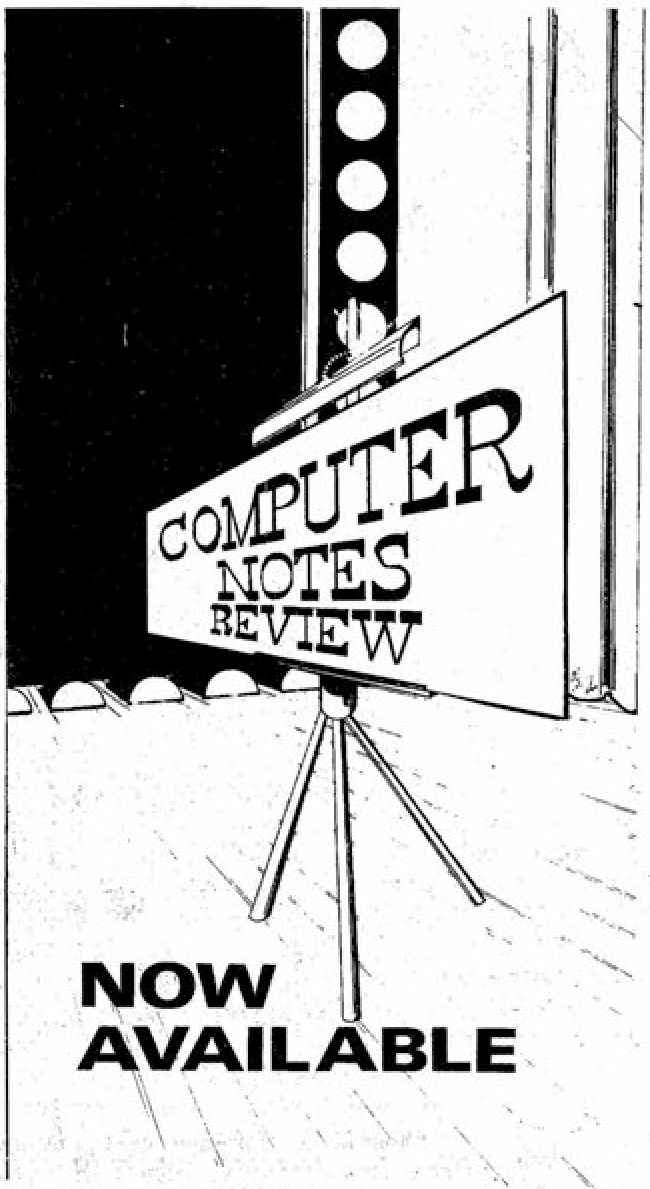
mits

This portion of the program reads a paper tape and stores the bytes in RAM.**

```
ØØØ     ØØ1               LXI     B,NEXT  LOOP ADDRESS
ØØ1     Ø24
ØØ2     ØØØ
ØØ3     Ø21               LXI     D,Ø     BYTE COUNT
ØØ4     ØØØ
ØØ5     ØØØ
ØØ6     Ø41               LXI     H,2ØØ   BUFFER ADDRESS
ØØ7     2ØØ
Ø1Ø     ØØØ
Ø11     Ø61               LXI     SP,2ØØ  TOP OF STACK + 1
Ø12     2ØØ
Ø13     ØØØ
Ø14     Ø76               MVI     A,3     INITIALIZE 2-SIO
Ø15     ØØ3
Ø16     323               OUT     2Ø
Ø17     Ø2Ø
Ø2Ø     Ø76               MVI     A,21    2 STOP BITS
Ø21     Ø21
Ø22     323               OUT     2Ø
Ø23     Ø2Ø
Ø24     3Ø5     NEXT:     PUSH    B       SAVE RETURN ADDRESS
Ø25     333               IN      2Ø      CHECK STATUS BIT
Ø26     Ø2Ø
Ø27     Ø17               RRC             DATA AVAILABLE?
Ø3Ø     32Ø               RNC             NO - CHECK AGAIN
Ø31     333               IN      21      YES - GET BYTE
Ø32     Ø21
Ø33     167               MOV     M,A     SAVE BYTE IN RAM
Ø34     Ø43               INX     H       NEXT BUFFER ADDRESS
Ø35     Ø23               INX     D       BYTE COUNT
Ø36     311               RET             NEXT BYTE
Ø37     ØØØ               NOP
```

This portion of the program punches a paper tape with nulls between the lines.**

```
Ø4Ø     Ø41               LXI     H,2ØØ   BUFFER ADDRESS
Ø41     2ØØ
Ø42     ØØØ
Ø43     ØØ1     PUNCHR:   LXI     B,PUNCHR
Ø44     Ø43
Ø45     ØØØ
Ø46     3Ø5               PUSH    B       SAVE RETURN ADDRESS
Ø47     315               CALL    CHKDON  SEE IF MORE BYTES TO PUNCH
Ø5Ø     1Ø7
Ø51     ØØØ
Ø52     176               MOV     A,M     FETCH BYTE
Ø53     Ø43               INX     H       POINT TO NEXT BYTE
Ø54     315               CALL    OUTTY   PUNCH BYTE
Ø55     Ø73
Ø56     ØØØ
Ø57     326               SUI     12      WAS IT A LINE FEED?
Ø6Ø     Ø12
Ø61     3ØØ               RNZ             NO - PUNCH ANOTHER
Ø62     ØØ6               MVI     B,5     YES - ADD NULLS
Ø63     ØØ5                               NUMBER OF NULLS
Ø64     315     NULLS:    CALL    OUTTY   PUNCH A NULL
Ø65     Ø73
Ø66     ØØØ
Ø67     ØØ5               DCR     B       MORE NULLS TO BE PUNCHED?
Ø7Ø     3Ø2               JNZ     NULLS   YES - PUNCH ANOTHER
Ø71     Ø64                               OTHERWISE...
Ø72     ØØØ                               ...PUNCH LAST NULL
Ø73     365     OUTTY:    PUSH    PSW     SAVE BYTE
Ø74     333     STATUS:   IN      2Ø      CHECK STATUS BIT
Ø75     Ø2Ø
Ø76     Ø17               RRC             TTY READY...
Ø77     Ø17               RRC             ...TO PUNCH A BYTE?
1ØØ     322               JNC     STATUS  NO - CHECK AGAIN
1Ø1     Ø74
1Ø2     ØØØ
1Ø3     361               POP     PSW     FETCH BYTE
1Ø4     323               OUT     21      PUNCH IT
1Ø5     Ø21
1Ø6     311               RET
1Ø7     Ø33     CHKDON:   DCX     D       BYTE COUNT
11Ø     173               MOV     A,E
111     262               ORA     D       MORE BYTES TO PUNCH?
112     3ØØ               RNZ             YES
113     3Ø3     SELF:     JMP     SELF    ALL DONE
114     113
115     ØØØ
```



680 Software News

By Mark L. Chamberlin

A Fix for the CONT Statement in 680 BASIC:

It seems an error slipped past us in 680 BASIC, which prevents the CONT (Continue) feature from operating properly. I'd like to thank Mark E. Becker of Newton Centre, Massachusetts, and Douglas L. Jones of North East, Pennsylvania for bringing this matter to the attention of the MITS software department. The following information will illustrate how to patch 680 BASIC to correct this condition. (All versions of 680 BASIC past 1.Ø will not require this patch.)

1) Load BASIC into your 680b.

2) Use the Monitor's M&N commands to make the following patches:

Address	Contents
ØØDA	8D
ØØDB	36
ØØDC	DE
ØØDD	87
ØØDE	Ø8
ØØDF	27
ØØEØ	Ø3
ØØE1	BD
ØØE2	15
ØØE3	D7
ØØE4	7E
ØØE5	Ø3
ØØE6	3C
Ø64D	ØØ
Ø64E	DA

At this point BASIC can be run and CONT will work fine. However, the program "PUNBAS" can be used to punch the corrected version of BASIC so that the patches don't have to be made each time BASIC is reloaded.

Teletype™ Echo

Unless Teletype™ echo is suppressed while loading object tapes using the Monitor's L command, characters will be dropped. This is caused by a timing problem which occurs when operating at 110 baud. All MITS software for the 680 system suppresses echo automatically as it is loaded. Other object tapes should be loaded only after the echo suppress flag has been set. (See the Altair 680 System Monitor Manual for details.)

It's Your Turn

I'm always glad to hear about any new and exciting 680 applications. Don't forget that CN pays up to $50 a page for good applications articles. So if you're doing something interesting with your 680, share it! Write an article detailing your 680 project and send it to me. Other 680 users will appreciate the benefit of your experience.

```
                    NAM    PUNBAS
                    OPT    NOG,PAGE
             *
             * PUNCH 68Ø BASIC
             * MLC/11-17-76
             *
             * MONITOR ROUTINES IN ACIA MONITOR
             *
        FF81 OUTCH  EQU    $FF81    MONITOR OUT CHAR
        FF6D OUT2H  EQU    $FF6D    MONITOR OUT 2 HEX DIGS
        FFAB CRLF   EQU    $FFAB    MONITOR ENTRY POINT

        1AFF STACK  EQU    $1AFF    JUST BELOW THIS PROGRAM
             *
             * START RIGHT HERE
             *
1BØØ                ORG    $1BØØ
1BØØ 8E 1AFF START  LDS    #STACK   INIT THE STACK POINTER
1BØ3 8D 34          BSR    LEDTRL   PUNCH SOME LEADER
1BØ5 CE 1B97        LDX    #FORM    PUNCH ECHO OFF RECORD
1BØ8 8D 7D          BSR    PMESS
1BØA CE 1BA1        LDX    #ECHOFF
1BØD 8D 78          BSR    PMESS
1BØF CE ØØØØ        LDX    #Ø       START PUNCHING AT Ø
1B12 FF 1B9C        STX    BEGADR
1B15 CE ØØE6        LDX    #$E6     STOP BEFORE MONITOR STACK
1B18 8D 28          BSR    PUN
1B1A CE Ø1ØØ        LDX    #$1ØØ    START ABOVE MONITOR FLAGS
1B1D FF 1B9C        STX    BEGADR
1B2Ø CE 1AB2        LDX    #$1AB2   END AT END OF BASIC
1B23 8D 1D          BSR    PUN
1B25 CE 1B97        LDX    #FORM    PUNCH ECHO ON RECORD
1B28 8D 5D          BSR    PMESS
1B2A CE 1BAC        LDX    #ECHON
1B2D 8D 58          BSR    PMESS
1B2F CE 1BB7        LDX    #EOF     PUNCH EOF RECORD
1B32 8D 53          BSR    PMESS
1B34 8D Ø3          BSR    LEDTRL   PUNCH TRAILER
1B36 7E FFAB        JMP    CRLF     RETURN TO THE MONITOR
             *
             * LEDTRL PUNCHES 256 NULLS
             *
1B39 4F      LEDTRL CLR A
1B3A 5F             CLR B
1B3B BD FF81 LED1   JSR    OUTCH    PUNCH A NULL
1B3E 4A             DEC A
1B3F 26 FA          BNE    LED1     KEEP PUNCHIN THOSE ZEROES
1B41 39             RTS             RETURN TO CALLER
1B42 FF 1B9E PUN    STX    LASADR
1B45 CE 1B97 PUNØ   LDX    #FORM    PUNCH THE FORMAT BYTES
1B48 8D 3D          BSR    PMESS
1B4A B6 1B9F        LDA A  LASADR+1 SUB LOW ORDER BYTES
1B4D BØ 1B9D        SUB A  BEGADR+1
1B5Ø F6 1B9E        LDA B  LASADR   SUB HIGH ORDER BYTES
1B53 F2 1B9C        SBC B  BEGADR
1B56 26 Ø4          BNE    PUN2     LOTS MORE TO PUNCH
1B58 81 1Ø          CMP A  #16      LESS THAN 16 TO PUNCH?
1B5A 25 Ø2          BCS    PUN3     YES
1B5C 86 ØF   PUN2   LDA A  #15      NO, SO PUNCH 16
1B5E B7 1BAØ PUN3   STA A  NUMBYT   STORE #OF BYTES TO PUNCH-1
1B61 8B Ø4          ADD A  #4       ADJUST BYTE COUNT
1B63 BD FF6D        JSR    OUT2H    PUNCH BYTE COUNT
1B66 Ø8             INX             POINT TO BEGADR
1B67 8D 23          BSR    PNCH2    PUNCH ADDRESS
1B69 8D 21          BSR    PNCH2
1B6B FE 1B9C        LDX    BEGADR   POINT TO DATA
1B6E 8D 1C   PUN4   BSR    PNCH2    PUNCH DATA
1B7Ø 7A 1BAØ        DEC    NUMBYT   MORE TO PUNCH THIS RECORD?
1B73 2A F9          BPL    PUN4     YES
1B75 FF 1B9C        STX    BEGADR   STORE NEW START ADDRESS
1B78 43             COM A           ONE'S COMP OF CHECKSUM
1B79 BD FF6D        JSR    OUT2H    PUNCH CHECKSUM
1B7C Ø9             DEX             ADJUST POINTER
1B7D BC 1B9E        CPX    LASADR   ARE WE DONE?
1B8Ø 26 C3          BNE    PUNØ     NO, KEEP ON PUNCHING
1B82 39             RTS             YES, RETURN
             *
             * THIS ROUTINE PUNCHES WHAT X POINTS TO
             * STOPS WHEN IT SEES A CHAR WITH BIT 7 SET
             *
1B83 BD FF81 SENDIT JSR    OUTCH    PUNCH THE CHAR
1B86 Ø8             INX             POINT TO NEXT CHAR
1B87 E6 ØØ   PMESS  LDA B  X        GET THE CHAR
1B89 2A F8          BPL    SENDIT   IF NON NEGATIVE THEN SEND
1B8B 39             RTS             RETURN ON BIT 7 SET
             *
             * PUNCH DATA BYTE POINTED TO BY X
             *
1B8C E6 ØØ   PNCH2  LDA B  X        GET BYTE TO PUNCH
1B8E 1B             ABA             UPDATE CHECKSUM
1B8F 36             PSH A           SAVE CHECKSUM
1B9Ø 17             TBA             COPY BYTE TO A
1B91 BD FF6D        JSR    OUT2H    PUNCH THE BYTE
1B94 32             PUL A           RESTORE CHECKSUM
1B95 Ø8             INX             BUMP BYTE POINTER
1B96 39             RTS             RETURN TO CALLER
1B97 ØD      FORM   FCB    $D,$A,'S,'1,$FF
1B9C ØØØ2    BEGADR RMB    2
1B9E ØØØ2    LASADR RMB    2
1BAØ ØØØ1    NUMBYT RMB    1
1BA1 3Ø      ECHOFF FCC    /Ø4ØØF3FFØ9/
1BAB FF             FCB    $FF
1BAC 3Ø      ECHON  FCC    /Ø4ØØF3ØØØ8/
1BB6 FF             FCB    $FF
1BB7 ØD      EOF    FCB    $D,$A,'S,'9,$FF
                    END
```

Altair BASIC File Structures

By Gary Runyan

Maintaining the extensive inventory required by a computer manufacturer is of primary importance, but also can be a tedious, error-prone job when done manually. To achieve better inventory control, MITS has implemented a computerized system which runs on the Altair 8800 This article will use programs from our system to illustrate how to program for such a typical application. At the end of the article is a list of the hardware in our system and some comments for those who would like to implement on a lower cost system.

The most important part in the design of such a system is how the files are set up. Files that are correctly set up will be easy to use and maintain. Poorly set up files will be a perpetual headache, causing either an eventual rewrite of the system or, more often, abandonment of the system.

The "INVEN" (shown right) listing in this article shows how the central file (a random file) in our system is set up and how it is handled. The "INVEN" listing also shows the use of another random file and a sequential file. The "CALC" listing shows how to read programs as data files. The third listing in this article is an example of a program that will be read as a data file.

The listing of "INVEN" contains modules from the main program in our inventory system. The modules listed were included to show:

a) program startup initialization and comments about the files used by the program (lines 1-35)

b) what the complete program does (lines 60-100)

c) an example of how to modify records in a random file (lines 900-1040)

d) an example of how sequential files are used (lines 1800-1868 and 2700-2820)

e) one approach to the problem of handling a random file that spans more than one disk (lines 2000-2030)

f) two subroutines (lines 300-340 and 9200-9220), which are called by the listed modules.

Continued on
Page Twenty-Nine

"INVEN"

```
1 DEFINT F-N
2 DEFINT R
3 DEFINT Z
5 DEFDBL P
6 DEF FNY#(Q8)=INT((VAL(STR$(Q8)+"D")*A#)+.5D)/A#
7 DEF FNQ#(Q9)=INT((VAL(STR$(Q9)+"D")*1000D)+.5D)/1000D
8 A$=MKD$(0):B$=MKS$(0):A#=100000D
10 DIM Q$(2),P$(2)
11 '

INV1 ON DRIVE 0 HOLDS ITEMS 1-2000
INV2 ON DRIVE 1 HOLDS ITEMS 2001-4000
INV3 ON DRIVE 1 HOLDS SUMS LOGGED IN AND OUT BY DEPARTMENT
12 '
WEKLYRST AND MONTHRST ARE WRITTEN WHILE THE WEEKLY,MONTHLY ACTIVE ITEMS LISTS
ARE PRINTING; CONTAIN THE ITEM #S THAT NEED TO BE RESET; AND ARE READ BY
THE WEEKLY,MONTHLY RESETS.
14 '
Q$() <=> THREE ON HAND QTY FOR: P$() <=> THREE PRICES
[P(0) OLDEST, P(1) NEXT OLDEST, Q(0)<>0 IF Q(1)<>0, Q(1)<>0 IF Q(2)<>0]
D$ <=> DESCRIPTION   LEFT$(D$,3)="$$$" <=> INACTVE ITEM #
15 '
I1$ <=> WEEKLY QTY IN
I2$ <=> MONTHLY QTY IN
O1$ <=> WEEKLY QTY OUT
O2$ <=> MONTHLY QTY OUT
T$ <=> REORDER LEVEL
DI1$ <=> WEEKLY $ IN
ID2$ <=> MONTHLY $ IN
DO1$ <=> WEEKLY $ OUT
OD2$ <=> MONTHLY $ OUT
17 '
DT1$ <=> WEEKLY DEPT $ TAKEN
DX2$ <=> MONTHLY DEPT $ TAKEN
DG1$ <=> WEEKLY DEPT $ GIVEN
DY2$ <=> MONTHLY DEPT $ GIVEN

20 OPEN "R",#1,"INV1"
30 OPEN "R",#2,"INV2",1
32 OPEN "R",#3,"INV3",1
35 FIELD #3,8 AS DT1$,8 AS DX2$,8 AS DG1$,8 AS DY2$
60 PRINT:F=0:INPUT"FUNCTION NUMBER";F:IFF>255THEN63
61 ON F GOTO 210,350,350,1900,600,900,1700,2700,1800,1700,2700,2500,2300
,2400,1880,2900'  2   3    4    5    6    7     8     9    10    11    12    13
  14   15   16
63 PRINT"1 - ENTER NEW ITEM"
64 PRINT"2 - LIST ITEM ON CRT (SHORT FORM)"
65 PRINT"3 - LIST ITEM ON CRT (LONG FORM)"
66 PRINT"4 - PRINT ITEMS ON LINE PRINTER
67 PRINT"5 - ADD TO INVENTORY"
68 PRINT"6 - REMOVE FROM INVENTORY"
69 PRINT"7 - PRINT WEEKLY DEPARTMENT DOLLAR RECORD ON LINE PRINTER
70 PRINT"8 - PRINT WEEKLY ACTIVE ITEMS LIST ON LINE PRINTER
71 PRINT"9 - WEEKLY RESET
72 PRINT"10- PRINT MONTHLY DEPARTMENT DOLLAR RECORD ON LINE PRINTER
73 PRINT"11- PRINT MONTHLY ACTIVE ITEMS LIST ON LINE PRINTER
74 PRINT"12- MONTHLY RESET
75 PRINT"13- RESET ORDER LEVELS
76 PRINT"14- PRINT LISTNG OF ITEMS NEEDING TO BE RE-ORDERED
77 PRINT"15- DELETE OLD ITEM
78 PRINT"16- ERRORS BACKOUT
100 GOTO60
298 '
*
SUB - INPUT PART # & GET RECORD
*
300 PRINT:PRINT:N=0:INPUT"PART NUMBER";N:IFN<1THENRETURN
310 IFN>4000THENPRINT:PRINT"# TOO HIGH":GOTO 300
320 GOSUB2000:GETZ,R1
330 IFLEFT$(D$,3)="$$$"THENPRINT:PRINT"NO INFORMATION ON PART";N:GOTO300
340 RETURN
890 '
*
F=6 - REMOVE FROM INVENTORY
*
900 GOSUB300:IFN=0GOTO63
920 DN=-1:INPUT"NUMBER OF ITEMS REMOVED FROM INVENTORY";DN:IFDN=-1THEN63
950 IFCVS(Q$(0))+CVS(Q$(1))+CVS(Q$(2))<DNTHENPRINT"
    ATTEMPT TO REMOVE MORE THAN ON HAND":PRINT:GOTO63
960 D0=DN:P=0
970 IFD0<CVS(Q$(0))THEN
    P=P+FNQ#(D0)*CVD(P$(0)):LSETQ$(0)=MKS$(CVS(Q$(0))-D0):GOTO1000
980 P=P+FNQ#(CVS(Q$(0)))*CVD(P$(0)):D0=D0-CVS(Q$(0)):
    LSETQ$(0)=Q$(1):LSETQ$(1)=Q$(2):LSETQ$(2)=B$:
    LSETP$(0)=P$(1):LSETP$(1)=P$(2):LSETP$(2)=A$:IFD0THENGOTO970
1000 LSETO1$=MKS$(CVS(O1$)+DN):LSETO2$=MKS$(CVS(O2$)+DN):
     LSETDO1$=MKD$(CVD(DO1$)+P):LSETOD2$=MKD$(CVD(OD2$)+P)
1020 GOSUB9200:IFC%=-1GOTO63
1030 LSETDT1$=MKD$(CVD(DT1$)+P):LSETDX2$=MKD$(CVD(DX2$)+P)
1040 PUT3,C%:PUTZ,R1:GOTO900
1790 '
*
F=9 - WEEKLY RESET
```

The function FNY# (line 6) is used to round dollar amounts to thousandths of a cent. FNQ# (line 7) is used to round quantities to thousandths. These two functions also get around a bus in 3.4 that puts garbage after the sixth digit during certain single to double precision conversions. This bug has been fixed in 4.0.

INV3 is fielded once in the program initialization, but INV1 and INV2 will be repeatedly fielded by calls to the subroutine at line 2000. The IF F>255 (line 60) avoids the program being stopped by an illegal function call at line 61.

PUT statements are the very last statements executed in the Remove from Inventory module, the Add to Inventory module, etc. This prevents updating one file but not the other (as could happen if PUTZ, R1 was at line 1010).

Line 2000 sets Z to 1 and R1 to N if the item wanted, N, is less than 2001. It sets Z to 2 and R1 to N-2000 if the item wanted is greater than 2000. Line 2020 then sets the pointers for the variables in the field statement to point into either the buffer for INV1 or the buffer for INV2, depending on whether the item wanted is less than 2001 or greater than 2000.

The "CALC" listing on page is a partial listing of a program which determines if there are enough parts in inventory to meet projected demands.

Line 26 waits while the disk comes up to speed so "ENABLE DISK 1" will not come up on the terminal. Lines 40-80 input up to fifty different product codes and a number to be built for each product. Line 100 opens a file for each product that contains the parts required for the product. Line 112 builds up a report heading extracting the product description contained in line 10 of each file. Lines 120-150 accumulate the parts required for each product into the matrix Q.

The following is a partial listing of the parts file for the 8800b:

```
5  CODE 1
10 PARTS LIST FOR 8800B
20 OCT 30, 1976
90 REM THIS IS THE START OF DATA
100, 11,1042
110,3,1134
120,4,1040
130,1,1020
140,1,1021
150,1,1024
160,1,1071
170,1,1074
180,1,2105
190,24,348
200,2,326
↑C
```

"INVEN" (cont.)

```
*
1800 PRINT"7 - WEEKLY DEPARTMENT RECORD
1802 PRINT"8 - WEEKLY ACTIVE ITEMS
1804 Z$="":INPUT"HAVE THE ABOVE BEEN LISTED FOR TODAY";Z$
1810 IFRIGHT$(Z$,1)<>"Y"THENPRINT:PRINT"WEEKLY RESET NOT PERFORMED":GOTO63
1843 OPEN"I",4,"WEKLYRST"
1845 IFEOF(4)THENCLOSE4:KILL"WEKLYRST":GOTO1862
1850 INPUT#4,N:IF 1<=NANDN<=4000 THENGOSUB2000:GETZ,R1
     ELSEPRINTN;"OUT OF BOUNDS.  RESET ABORTED. ":END
1855 LSETI1$=B$:LSETO1$=B$:LSETDI1$=A$:LSETDO1$=A$:PUTZ,R1
1860 GOTO1845
1862 FORI=1TO20
1864 GET3,I:LSETDT1$=A$:LSETDG1$=A$:PUT3,I
1866 NEXT
1868 GOTO60
1999 '
*
SUB - GET Z,R1 FOR N AND FIELD TO INV1,2
*
2000 Z=1-(N>2000):R1=N+(Z=2)*2000
2020 FIELD Z,4 AS Q$(0),4 AS Q$(1),4 AS Q$(2), 8 AS P$(0),8 AS P$(1),
    8 AS P$(2),40 AS D$,4 AS I1$,4 AS I2$,4 AS O1$, 4 AS O2$,4 AST$,
    8 AS DI1$,8 AS ID2$,8 AS DO1$,8 AS OD2$
2030 RETURN
2690 '
*
F=8,11 - WEEKLY,MONTHLY ACTIVE ITEMS LIST
*
2700 N=1:GOSUB2000
2703 IFF=8THENOPEN"O",4,"WEKLYRST"ELSEOPEN"O",4,"MONTHRST"
2710 FORI=1TO2000
2720 GETZ,I:IFLEFT$(D$,3)="$$$"THEN2800
2723 Q0=CVS(Q$(0)):Q1=CVS(Q$(1)):Q2=CVS(Q$(2))
2725 IFF=8THENI!=CVS(I1$):O!=CVS(O1$):I#=CVD(DI1$):O#=CVD(DO1$)
     ELSEI!=CVS(I2$):O!=CVS(O2$):I#=CVD(ID2$):O#=CVD(OD2$)
2730 IFI!+O!=0THEN2800
2733 PRINT#4,N+I-1
2740 REM PRINT PART ON LINE PRINTER CODE NOT SHOWN
2800 NEXT
2810 IFN=1THENN=2001:GOSUB2000:GOTO2710
2811 CLOSE4
2820 REM PRINT TOTALS ON LINE PRINTER CODE NOT SHOWN
9190 '
*
INPUT DEPARTMENT # AND GET TOTALS
*
9200 C%=-1:INPUT"ENTER DEPARTMENT CODE";C%:IFC%=-1THENRETURN
9210 IF1<=C%ANDC%<=20THENGET3,C%:RETURN
9220 PRINT"INVALID CODE":GOTO9200
```

"CALC"

```
5 CLEAR 500
10 DEFINT A-Z
20 DIM CN(49),NU(49),Q(4000)
22 CLOSE:UNLOAD1:OUT8,255
24 INPUT"PLACE DISK WITH PARTS LISTS IN DRIVE 1.  HIT RETURN";G$
26 FORI=1TO5000:NEXTI
28 MOUNT1
32 PRINT"TODAY'S MO/DA/YR ";:LINE INPUTDT$:H$=DT$+" PARTS AVAILABLE FOR:"
40 INPUT"CODE NUMBER(0 WHEN FINISHED)";CN(I)
50 IF CN(I)=0 THEN 90
60 IF CN(I)<1 OR 99<CN(I) THEN PRINT"INVALID CODE NUMBER":GOTO 40
70 INPUT"NUMBER OF UNITS TO BE MADE";NU(I)
80 I=I+1:IF I<50 THEN 40
90 FOR K=0 TO I-1
100 OPEN"I",#1,"CODE"+MID$(STR$(CN(K)),2),1
104 LINEINPUT#1,A$:IFA$=""THEN104
106 IFLEFT$(A$,3)="90 "THEN120
108 IFLEFT$(A$,3)<>"10 "THEN104
110 IFKTHENH$=H$+","
112 H$=H$+STR$(NU(K))+STR$(CN(K))+"=("+MID$(A$,20)+")":GOTO104
120 IF EOF(1) THEN 160
130 INPUT #1,A,QN,PN
140 Q(PN)=Q(PN)+NU(K)*QN
150 GOTO 120
160 CLOSE 1:NEXT K
200 REM PROMPT OPERATOR TO RELOAD SECOND HALF OF INVENTORY FILES IN DRIVE
210 REM ONE.  OPEN INVENTORY FILES.  PRINT REPORT OF QTY NEEDED VS ON HAND.
```

BASIC File Structures (cont.)

The parts lists for a product are programs saved with the A option. Since they are programs, their maintenance is very easy. For example: Let's assume part 1071 in the 8800b is too marginal and that from now on they should be built with part 1173 instead of part 1071. With the parts lists disk mounted on drive Ø, the following sequence will update the 8800b file.

```
LOAD "CODE1"
160, 1, 1173
SAVE "CODE1",Ø,A
```

The programmer who is cramped for memory will find that he can still adequately document programs if he sets up comments as separate files. The memory used for variables when a program runs can be utilized for comments if the comments are merged in when the program is to be listed. Additional memory can be obtained by bringing BASIC up without optional functions and with no files. Another alternative would be to list the program a half at a time.

The main inventory program is set up so that if one types the return key with no input in reply to any prompt, the program dumps the function descriptions on the CRT and returns to the prompt FUNCTION NUMBER. If the program was to be run on a printing terminal, instead of a 9600 baud CRT, it would not be set up to print the descriptions every time the operator wanted to get back to the prompt FUNCTION NUMBER. The list of function descriptions would be taped up next to the terminal.

The system consists of an Altair 8800b, two disk drives, a 24-line LEAR SIEGLER CRT, 2SIO board, line printer, PROM memory board with the disk bootstrap loader on PROM and two 16K static memory boards. The software currently being used is 3.4 Disk Extended BASIC. When the main inventory program (with comments) is running, there are 4,126 bytes free.

101 BASIC Computer Games

Book Review

By Linda Blocki

101 BASIC Computer Games, edited by David H. Ahl, is not only the first collection of games all in BASIC, but a uniquely educational book which provides both a complete listing and description of every game along with a sample program for each.

As Ahl points out in his book, educators generally agree that games are highly motivational and promote learning by discovery. What better way is there to learn about Newton's second law than by simulating an Apollo lunar landing in ROCKET? Or to learn about logic by playing BAGLES? You can even increase your vocabulary while playing SYNONM or improve your writing skills in BUZZWD by learning how to compose computer speeches with the latest buzz words.

For those interested in more exotic games, there's CHEMST, in which the player tries to dilute the fictitious kryptocyanic acid; CHOMP, which involves eating a cookie while trying to avoid the poison piece and HELLO, in which the computer dispenses advice on such problems as sex, health, money, or a job.

Computer enthusiasts with a sense of humor will find many entertaining games in the book with such challenging objectives as delivering pizzas successfully (PIZZA), doing a silly profile plot of an ugly woman (UGLY) and finding the happy hurkle beast hiding in a 10 x 10 matrix (HURKLE).

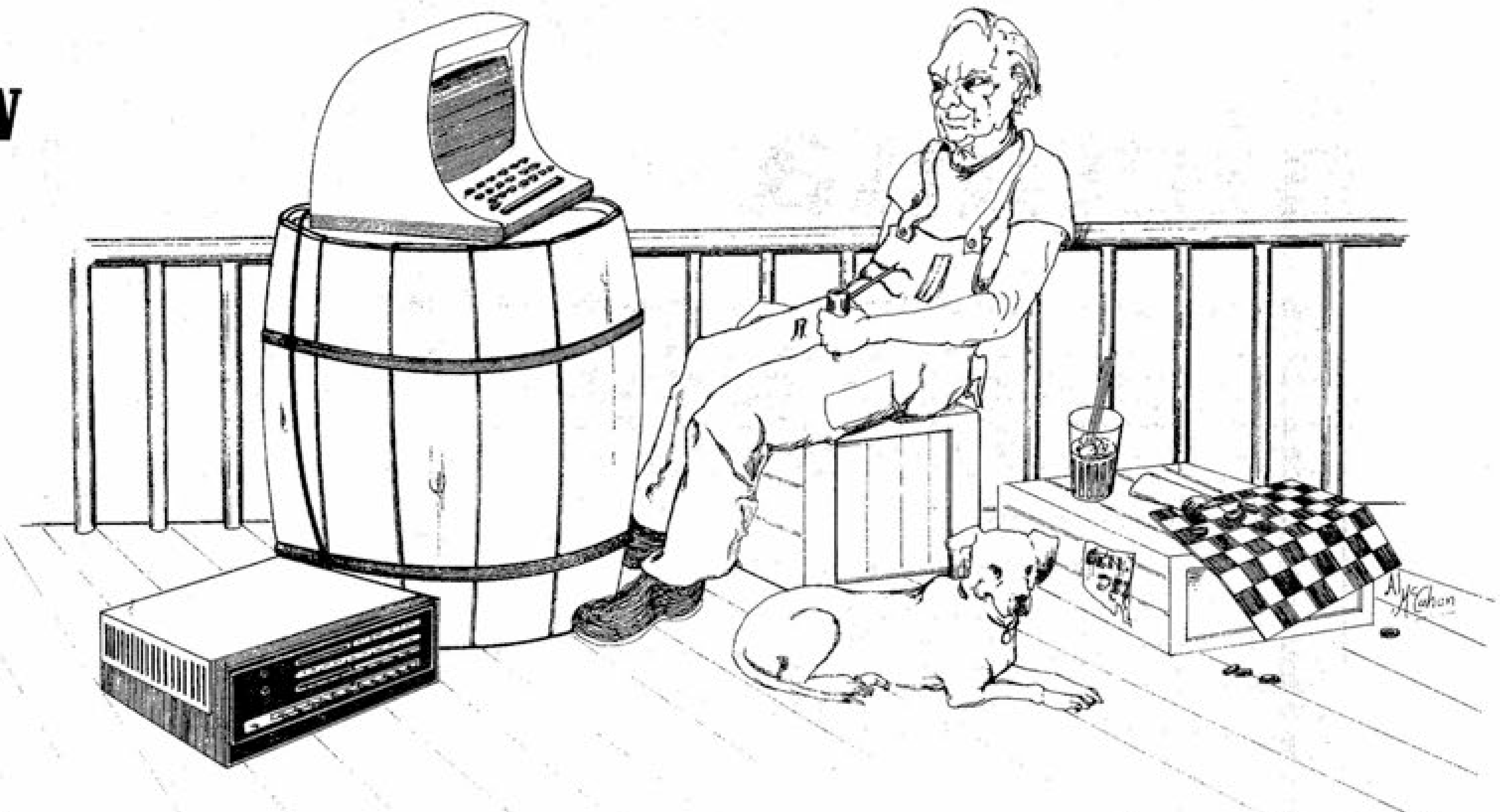
The names alone of many games are intriguing enough to invite further investigation. FIPFOP, SPLAT and ZOOP are sure to send a hobbyist running to his computer. FIPFOP is a solitaire logic game dealing with changing a row of Xs to Os. SPLAT involves opening a parachute at the last possible moment. ZOOP, otherwise known as the BASIC programmer's nightmare, is designed to imitate the system commands of a BASIC compiler, except that it gives totally meaningless and frustrating results.

Ahl spent considerable time collecting his potpourri of games on his travels to various schools as well as from submittals in response to an advertisement. Game authors range from seventh graders in California to PhDs in England.

The games run the gamut from extremely simple to more complex, but most require no special knowledge. To solve the game categorization dilemma, Ahl has simply listed the games in alphabetical order. But in the appendices, he has outlined some "family" groupings, such as logic, plotting and matrix games.

A BASIC-speaking computer is the only equipment needed to play any of the games. However, Ahl suggests that a grid or quadrille paper be used to play four of the matrix games and one of the supplemental diagrams included in the appendices be used when playing QUBIC.

Continued on Page Thirty-One



Book Review (cont.)

Most of the games also run in "standard" BASIC with any exceptions noted under the game title.

Due to the addictive nature of the games, computer enthusiasts should be reminded not to neglect eating and sleeping in favor of playing ANIMAL or FOOTBL.

<u>101 BASIC Computer Games</u> is available in only a softbound edition (248 pages) for $7.50 plus 75¢ postage from:

Creative Computing
P.O. Box 789-M
Morristown, NJ 07960

Games

"Bulcow" and "Nicoma" are just two of the many interesting games from David Ahl's, <u>101 BASIC Computer Games</u>.

BULCOW

```
5 GOSUB5ØØ
1Ø DIM D(1Ø,4),B(1Ø),C(1Ø),G(1Ø)
15 PRINT:PRINT:PRINT
2Ø  A=Ø:GOTO2ØØ
3Ø PRINT:PRINT:PRINT:J=Ø
35 INPUT"YOUR GUESS";N:N=(N+.1)/1ØØØØØ
4Ø FOR I=Ø TO 4:G(I)=INT(1Ø*N):N=1Ø*N-INT(1Ø*N)
41 FOR K=Ø TO I-1:IF I=Ø GOTO 44
42 IFG(I)=G(K) GOTO 17Ø
43 NEXT K
44 NEXT I
45 P=4:A=Ø:GOSUB 3ØØ
5Ø IF V=1 THEN B$="BULL" ELSE B$="BULLS"
51 IF W-V=1 THEN C$="COW" ELSE C$="COWS"
6Ø IF V=5 THEN PRINT"5 BULLS, YOU WIN!":GOTO 2Ø
    ELSE PRINTV;B$;W-V;C$
65 IF J=Ø THEN A=1:GOTO 2ØØ
68 GOSUB 4ØØ
7Ø PRINT"MY GUESS IS ";
75 FOR I=Ø TO 4:PRINT CHR$(D(J,I)+48);:NEXT
8Ø INPUT" MY SCORE";B(J),C(J):C(J)=C(J)+B(J)
81 IF B(J)>-1 THEN IF B(J)<6 THEN IF C(J)<6
    THEN IF C(J)-B(J)>-1 THEN GOTO 83
82 PRINT"RIDICULOUS!!":GOTO 7Ø
83 IF B(J)=4 THEN IF C(J)=5 THEN GOTO 82
85 IF B(J)=5 THEN PRINT " - I WIN - MY NUMBER WAS":GOTO 1ØØ
9Ø GOTO 35
1ØØ FOR I=Ø TO 4: PRINT CHR$(D(1,I)+48);:NEXT
1Ø2 PRINTN$
11Ø GOTO 2Ø
15Ø PRINT:PRINT "YOU HAVE GIVEN ME IMPOSSIBLE SCORES - GAME SPOILED":
      GOTO 2Ø
17Ø PRINT "REPEATED DIGITS NOT ALLOWED":GOTO 35
2ØØ FOR P=Ø TO 4
21Ø D(A,P)=INT(1Ø*RND(1))
22Ø FOR I=Ø TO P-1:IF P=Ø GOTO 23Ø
222 IF D(A,I)=D(A,P) THEN GOTO 21Ø
23Ø NEXT I:NEXT P
25Ø IF A=Ø THEN GOTO 3Ø
26Ø J=1:GOTO 7Ø
3ØØ V=Ø:W=Ø
31Ø FOR I=Ø TO P:IF D(A,I)=G(I) THEN V=V+1
32Ø FOR K=Ø TO 4:IF D(A,K)=G(I) THEN W=W+1
322 NEXT K
33Ø NEXT I
35Ø RETURN
4ØØ P=Ø
4Ø5 G(P)=D(J,P)
41Ø FOR I=Ø TO P-1:IF P=Ø GOTO42Ø
412 IF G(I)=G(P) GOTO 43Ø
415 NEXT
42Ø FOR A=1 TO J:GOSUB 3ØØ
425 IF V<=B(A) THEN IF W<=C(A) THEN IF 4-P>=C(A)-W
    THEN IF 4-P>=B(A)-V GOTO 448
43Ø G(P)=G(P)+3:IF G(P)>9 THEN G(P)=G(P)-1Ø
432 IF P=Ø THEN IF G(P)=D(1,Ø) GOTO 15Ø
435 IF G(P)<>D(J,P) GOTO 41Ø
44Ø P=P-1:IF P<Ø GOTO 15Ø
445 GOTO 43Ø
448 NEXT A
45Ø P=P+1:IF P<5 GOTO 4Ø5
455 J=J+1
46Ø FOR I=Ø TO 4:D(J,I)=G(I):NEXT
465 RETURN
5ØØ PRINT:PRINT:PRINT"       BRADFORD UNIVERSITY BULLS AND COWS GAME"
51Ø GOTO 1Ø
999 END
```

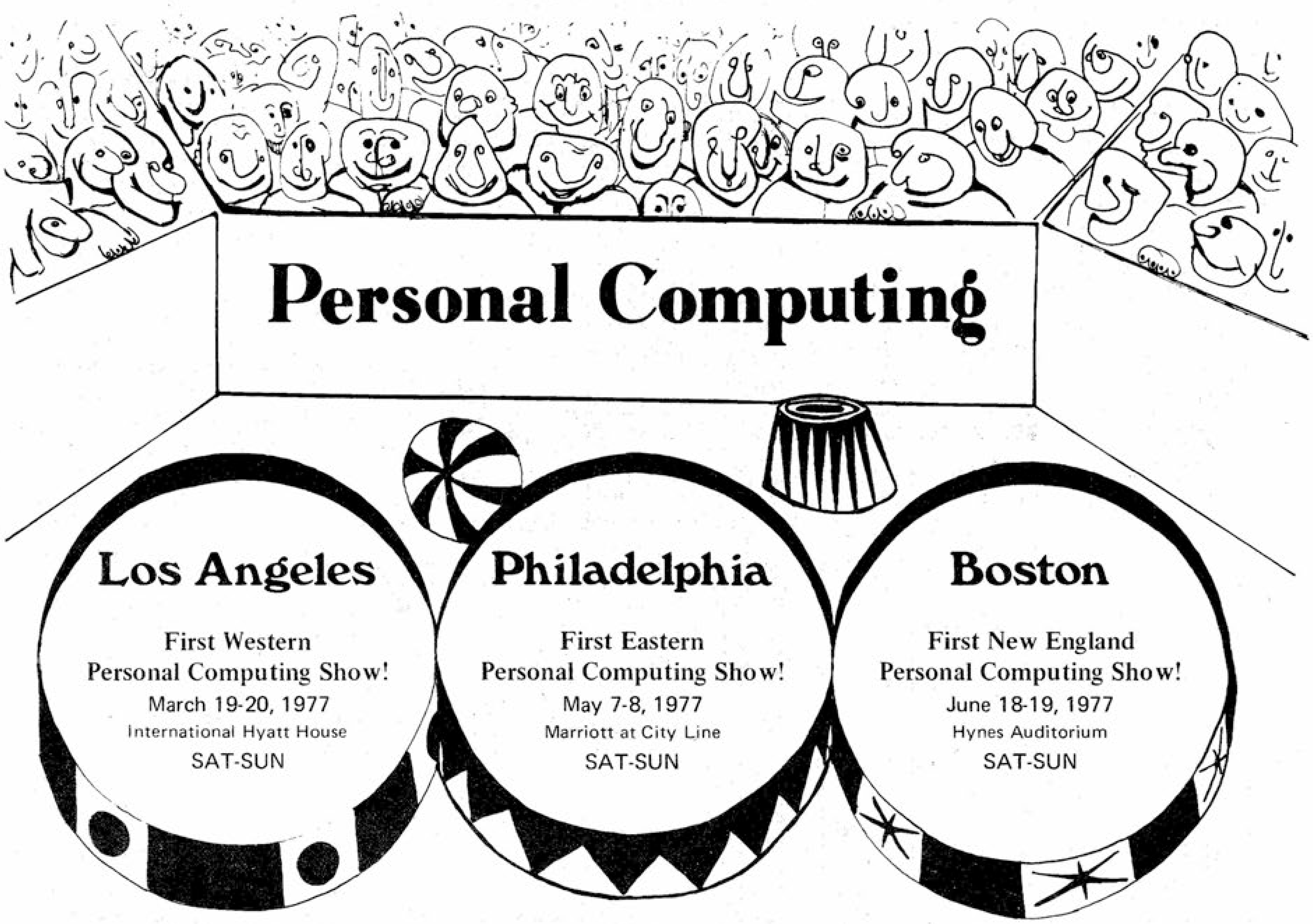
NOTE: <u>Correction to "Bulcow"</u>

NB: Change "GOTO 20" to "GOTO 5" in lines 60 and 110. Otherwise, the computer cheats!

nicoma

```
1Ø PRINT"BOOMERANG PUZZLE FROM ARITHMETICA OF NICOMACHUS -- A.D. 9Ø!"
2Ø PRINT
3Ø PRINT"PLEASE THINK OF A NUMBER BETWEEN 1 & 1ØØ"
4Ø INPUT"YOUR NUMBER DIVIDED BY 3 HAS A REMAINDER OF";A
5Ø INPUT"YOUR NUMBER DIVIDED BY 5 HAS A REMAINDER OF";B
6Ø INPUT"YOUR NUMBER DIVIDED BY 7 HAS A REMAINDER OF";C
8Ø PRINT:PRINT"LET ME THINK A MOMENT......"
9Ø FOR I=1 TO 1ØØØØ:NEXT:REM SLOW IT DOWN A LITTLE
1ØØ D=7Ø*A+21*B+15*C
11Ø IF D<1Ø5 THEN GOTO 14Ø
12Ø D=D-1Ø5:GOTO 11Ø
14Ø PRINT:PRINT"YOUR NUMBER WAS";D;"RIGHT?"
16Ø INPUT A$
17Ø IF LEFT$(A$,1)="Y" THEN GOTO 22Ø
18Ø IF LEFT$(A$,1)="N" THEN GOTO 24Ø
19Ø PRINT"EH?  I DON'T UNDERSTAND '";A$;"' TRY 'YES' OR 'NO'":GOTO 16Ø
22Ø PRINT"HOW ABOUT THAT!!":GOTO 25Ø
24Ø PRINT"I FEAR YOUR ARITHMETIC IS IN ERROR."
25Ø PRINT:PRINT"LET'S TRY ANOTHER."
26Ø GOTO 2Ø

OK
```



Personal Computing

Los Angeles

First Western
Personal Computing Show!
March 19-20, 1977
International Hyatt House
SAT-SUN

Philadelphia

First Eastern
Personal Computing Show!
May 7-8, 1977
Marriott at City Line
SAT-SUN

Boston

First New England
Personal Computing Show!
June 18-19, 1977
Hynes Auditorium
SAT-SUN

Greatest Computer Shows Ever!

Personal Computing magazine is proud to announce that it is sponsoring the first series of regional Personal Computing Shows.

Beginning with the *Western Personal Computing Show* in Los Angeles, and followed by the *Eastern Personal Computing Show* in Philadelphia and the *New England Personal Computing Show* in Boston, Personal Computing magazine intends to make everyone aware of low-cost computing.

Other shows are now being planned for the South, Southwest, Canada, and Europe!

Already, invitations have been sent to all the manufacturers in the personal computing field, computer stores, computer clubs and well-known computer experts.

Special areas of the exhibition halls will be set aside for Personal Computing in Education, in the Home, in HAM Radio, and in Small Businesses. These are all first for a computer show.

Seminars and special presentations include: Computer Synthesized Music, HAM Applications, Trends in Microcomputers, Mass Storage Systems, Lemonade Computer Service Companies, The Kitchen Computer, Computers on the Farm, The Small Business System, Software for Fun and Practical Applications, Computer Club Organization, Standards for the Hobbyists, Computer Art, The House Robot, Computer Crime, Software Protection and Future Computing.

In addition, *special tutorial workshops* will cover all aspects of computer hardware, programming in both machine language and higher-level language and applications. Workshops are designed for both beginners and advanced students in the art of personal computing.

We anticipate 150 different exhibits and crowds of up to 10,000 people at each of these shows. Arrangements for the shows are being handled by a professional management company to ensure that everything runs smoothly.

Cost of Registration:
At the door:
$10 per show (two days)
$ 6 per One Day Pass
Special Pre-Registration Rates:
$ 7.50 per show (two days)
$ 4.00 per One Day Pass
Note: Show tickets and one day passes entitle you to attend all seminars, workshops, exhibits and other events.

Register Now and Save!

Yes, I would like to take advantage of your special, pre-registration rates. I plan to attend the following regional Personal Computing Show(s):

□**Los Angeles**	□**Philadelphia**	□**Boston**
□Show (two days)	□Show (two days)	□Show (two days)
□One Day Pass Only	□One day pass	□One day pass

Enclosed is a check for ______________________

Name ______________________

Address ______________________

City ______________ State & Zip ______________

Send to: Personal Computing, Conference & Exposition Management Co., Box 844, Greenwich, CT 06830.