computer notes

JUNE '77

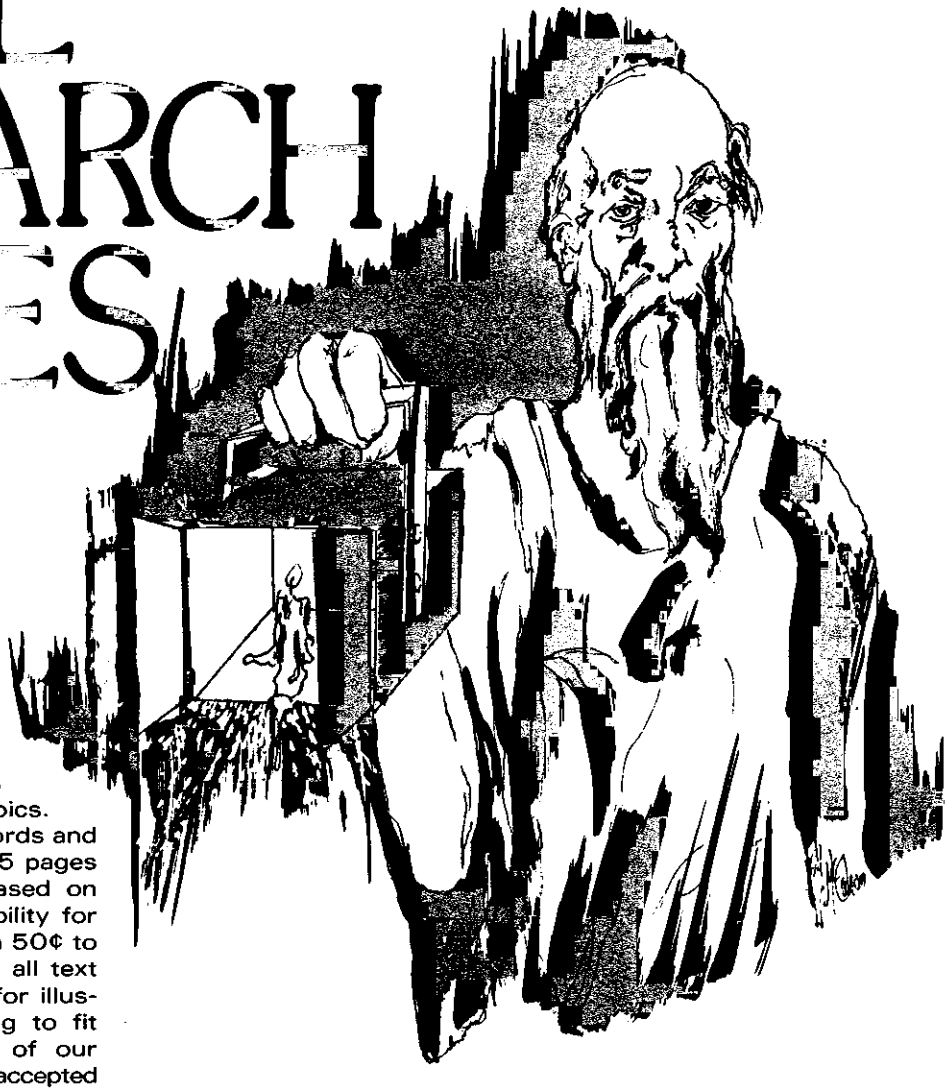
50¢

Altair Timesharing BASIC

NEW PRODUCTS ISSUE

mits

THE SEARCH GOES ON



Computer Notes is continually seeking quality manuscripts on applications, troubleshooting, interfacing, software, book reviews, fiction, cartoons and a variety of other computer-related topics.

Articles should be a minimum of 800 words and a maximum of 3600 words long (about 15 pages typed double-space). Honorariums are based on an article's technical quality and its suitability for **C.N.'s** readership. Payment will range from 50¢ to $1 per typeset magazine column inch for all text and programs. No payment will be made for illustrations. All articles are subject to editing to fit space requirements and content needs of our readership. Payment for articles which are accepted will be sent upon publication.

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

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

Whenever possible, art should be done to finished size. Complicated drawings should be submitted oversize for reduction to format by **C.N.**

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

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

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

Program listings should be recorded with the darkest ribbon possible on blank white paper.

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

 (Volume 3, Issue 1, June)
2450 Alamo S.E., Albuquerque, New Mexico 87106

ALTAIR TIMESHARING BASIC CHALLENGES LARGE COMPUTER SYSTEMS

By Susan B. Dixon

Altair Timesharing BASIC for microcomputers is a unique and dynamic package with powerful capabilities that challenge a field dominated by larger, more costly computers.

Altair Timesharing BASIC and Altair Timesharing Disk BASIC are magnified versions of the powerful and efficient Altair Extended BASIC. Each version includes increased capabilities to accommodate as many as eight different programs running simultaneously and independently within the system.

Typical Altair Timesharing BASIC system, including Altair B-100 CRTs, Altair Floppy Disks and Altair 8800b Turnkey model. Altair Timesharing BASIC supports up to eight users.

Instantaneous Keyboard Response

Input and output are interrupt driven and fully buffered to provide virtually instantaneous keyboard response even when the system supports the maximum number of users. The output buffers empty more quickly than they are filled, so it will appear the CPU is dedicated to each individual terminal.

High Speed Systematic Job Rotation

Operating within a highly efficient round-robin system, the CPU suspends operation of a job currently being executed, stores the address of the next instruction and moves to the next job. Each job is served a hundred millisecond slice of its program.

Partitioned Memory Locations

Established as a Fixed Partition System, each job is confined to a unique area of memory. Users may then access only their individual jobs, not the system or other jobs. This protects jobs from alteration or destruction. The size of the memory area must be established with a minimum of 1024 bytes during initialization. Memory areas may be of different sizes, depending on need. Each program area contains:

- —BASIC program text
- —Variable and string space
- —Work space
- —Approximately 300 bytes of the timesharing system

I/O Device Support

A variety of input devices can be linked to Altair Timesharing BASIC and Altair Timesharing Disk BASIC. This flexibility permits the use of CRT's for high-speed data manipulations and Teletypes™ and hard copy terminals when hardcopy output is required.

Altair Timesharing Disk BASIC facilitates the listing of programs on a line printer. Programs await listing in a queue to prevent mixing different jobs. Users receive a message if the line printer is unavailable.

Versatile Program Storage and Loading Capabilities

Altair Timesharing Disk BASIC provides rapid loading and program retrieval since all programs reside on a floppy disk. Read and modify passwords may be specified for program fixes to limit access by other users.

Altair Timesharing BASIC can be loaded from paper tape or audio cassette. Programs may be stored for later use on paper tape.

Other Features

- —Control of a specific job may be transferred from one terminal to another with a single command.
- —Various control characters allow suspension and resumption of each job without loss of data.
- —Extensive diagnostics for program debugging.
- —Automatic line numbering.
- —Both versions of Altair Time Sharing BASIC furnish line-oriented text editor with line and character manipulation capabilities.

Educational Applications

A single Altair 8800b loaded with either version of Timesharing BASIC can be utilized by several students performing independent operations. One student practices program development, another makes use of Timesharing BASIC's extensive diagnostics to debug a program while several other students calculate complicated equations. All program activity occurs simultaneously with no discernable response delay.

Computer-oriented education need not be limited to programming classes. An Altair Timesharing System is a valuable visual tool in science, math and engineering classes and as an introduction to the various aspects of computer technology to solve real world problems.

An Altair Timesharing System stimulates interest and provides a vehicle for discussion for the younger

CONTINUED

New Turnkey Version Features AUTO-START

By Dar Scott

Those of you who have been asking for a Turnkey version of the Altair 8800b computer may now stand up and cheer, because the Turnkey version of "the mainframe of the 70's" is now available.

The Turnkey version incorporates all the good quality construction and good looks of the Altair 8800b computer. But just because the Turnkey doesn't have a front panel or a front panel interface board doesn't mean it's merely a stripped-down version of the Altair 8800b computer. It has some unique features, including the new Turnkey Module board. With this board all the functional units of the computer — the CPU, RAM and PROM memory sense switches and serial I/O — can be contained on just two circuit boards, which are supplied in the standard Turnkey version package.

But the most important advantage of the Turnkey Module is that it contains AUTO-START, which allows the computer to begin executing a program in PROM as soon as the power is turned on or the START is actuated.

The Turnkey Module includes the following functional parts:

- AUTO-START
- 1K bytes each of RAM and PROM memory
- Serial I/O Channel
- Control for the front panel
- Miscellaneous control and housekeeping logic

The AUTO-START feature is the key to the Turnkey version's ease of use. When the power is turned on or when the START switch is actuated, the computer is forced to begin executing the instruction at the memory location specified by switches on the Turnkey Module. This means that when the power goes on, the computer can be made to start a loader program or a monitor automatically without keying in a bootstrap loader from the front panel. Alternately, the computer can be made to start a custom applications program at start-up. This should appeal to users who want to build the computing power of an Altair 8800b into a lab instrument, machine tool or some other dedicated application.

The serial input/output channel is essentially half of a 2SIO board. With the addition of a jumper selection, the channel interface is compatible with TTY, RS-232 or TTL signals. This means that the Turnkey version can operate with almost any terminal or I/O arrangement with no interface equipment necessary. The I/O ports associated with the serial input/output channel can be assigned to any 128 I/O port pairs by switch selection.

The 1K RAM and the 1K PROM can each be assigned to any 1K block within the full Altair 8800b address space by switch selection. The PROM used is the 1702A-a256-byte by 8-bit PROM. A disk bootstrap loader PROM, a general purpose multi-boot loader PROM and a small monitor PROM are also available from MITS for use with the Turnkey version.

The switches on the Turnkey version front panel are the POWER switch, which has a key lock for system security; START, which initializes the CPU and initiates the AUTO-START sequence; and RUN/STOP, which allows execution of a program to be stopped and started again. The indicators show that an I/O operation is in progress, a HALT state has been entered, interrupts are enabled or an interrupt is in progress. The lights and switches provide the minimum facilities to monitor and control system operation.

CPU Board

The CPU board is the standard Altair 8800b CPU board. It consists of five major functional blocks:

- 8080A CPU Chip
- 8224 Clock Generator Chip
- 8212 Status Latch
- Drivers and Receivers
- Power-On-Clear Circuit

The CPU board is, in most respects, identical to the 8800b CPU board described in a previous **Computer Notes** article (see p. 1, 4-5 in July C/N). However, the Power-On-Clear Circuit is especially important in the Turnkey version because it generates a pulse to reset the system. The circuit is designed so that even short power outages generate a reliable Power-On-Clear pulse. The Turnkey Module uses this pulse to reset the CPU by pulling PRESET low. The CPU board extends this pulse to prevent the CPU from starting before the Power-On-Clear pulse

ALTAIR TIMESHARING BASIC

CONTINUED

student. Children find it particularly fascinating to use a computer to solve arithemetic problems.

Scientific and Engineering Uses

Engineering firms and scientific labs currently using programmable calculators will benefit from the multi-purpose capabilities of a microcomputer for complex math routines and statistical manipulation. An Altair 8800B operating with Timesharing BASIC provides a number of individuals with access to graphic or tabular output in addition to the following mathematical functions: SIN, COS, TAN, LOG, SQR, SGN, ABS, INT and RND.

Required Hardware

The following hardware is needed to support both versions of Altair Timesharing BASIC.

- —Altair 8800 series mainframe and CPU
- —A minimum of 32K RAM
- —Vector Interrupt/Real Time Clock board
- —Up to 4 2SIO boards to interface terminals (no other types of I/O boards can be supported).
- —Line printer (optional for Altair Timesharing Disk BASIC).

The following products are optional for Altair Timesharing BASIC loading and required for Altair Timesharing Disk BASIC.

- —An Altair floppy disk drive and controller.
- —88-PROM board.

is completely removed from all I/O and memory boards.

The mother board for the Turnkey version of the Altair 8800b computer has 18 usable slots.

Power Supply

The power supply for the Turnkey version is the same as that for the standard Altair 8800b computer. This power supply furnishes the following voltages at the indicated full load currents:

8 volts at 18 amps
+18 volts at 2 amps
-18 volts at 2 amps

NEW ALTAIR MINIDISK STORES OVER 71K

By Thomas Durston

The miniaturization of mass storage is just one of the exciting features of the new Altair Minidisk System. Designed to work with the Altair 8800b, the Minidisk has a storage capacity of over 71K bytes per diskette with an access time of less than three seconds.

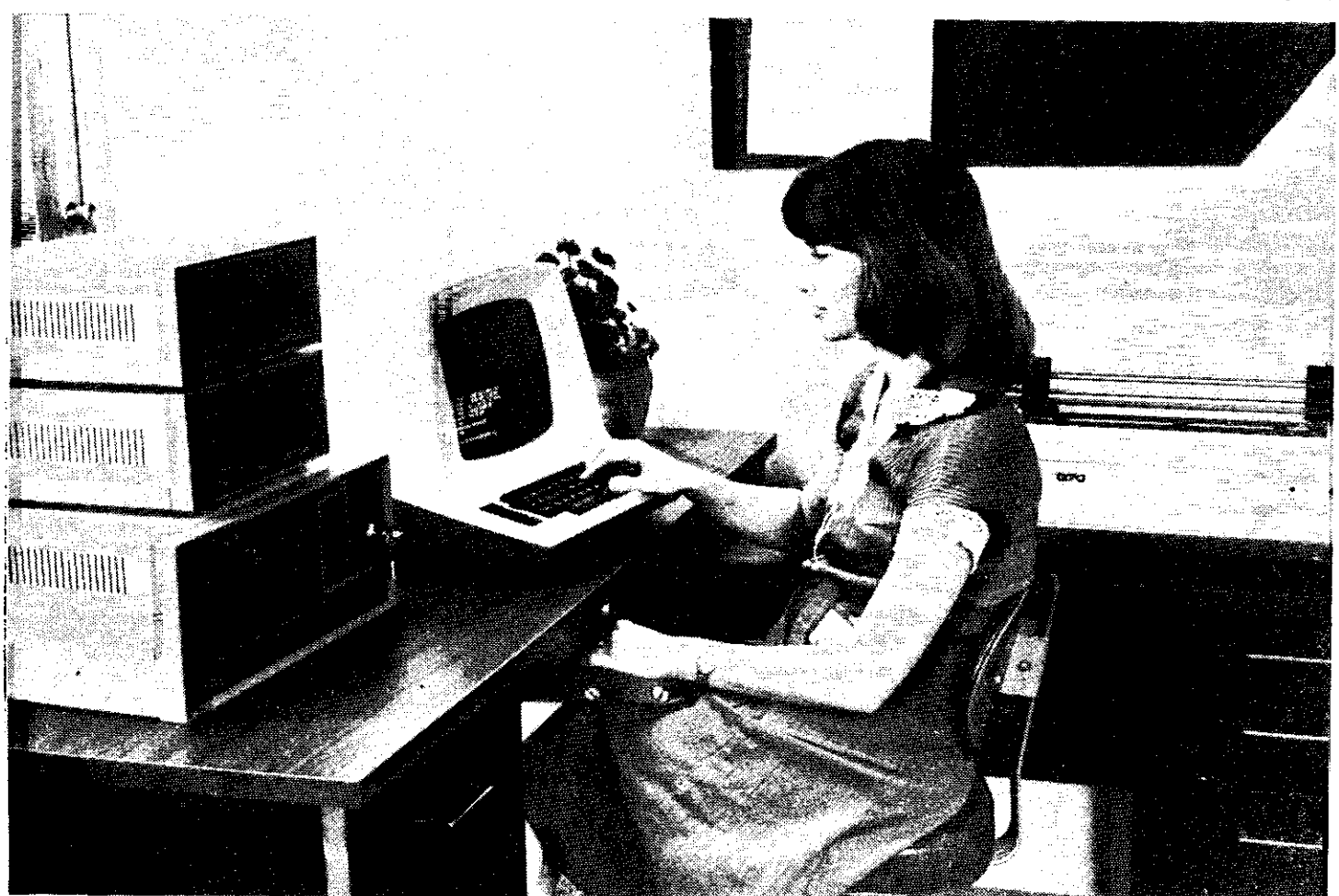
Altair Minidisk BASIC, resides in the lower 20K of memory and provides the disk utilization routines. Altair Minidisk BASIC includes the standard functions of BASIC, plus many extra file maintenance procedures that significantly increase programming power. The software driver for the Minidisk Read/Write functions is based on the hard sectoring format, which simplifies system configuration.

The Altair 8800b interacts with the Minidisk Drive through two Minidisk Controller Cards that plug into the Altair Bus. All control, status and data I/O signals are handled through three I/O ports dedicated to the Minidisk Controller. To insure maximum life of the drive motor, a timer in the Controller turns the system off if the Minidisk is not accessed for five seconds.

The Minidisk Drive Case contains a disk drive, power supply, line buffers and addressing circuitry. The Drive address is switch selectable. The selected address is displayed on the front panel for easy identification. Write protect is also a standard feature on the drive.

Check with MITS or your local dealer for prices and availability.

CONTINUED



Altair system featuring 8800b Turnkey Model, dual Altair Minidisks, LSI ADM-3 CRT and Q70 Line Printer.

NEW ALTAIR MINIDISK STORES OVER 71K
CONTINUED

88-MDS Altair Minidisk Controller

Description Includes a set of 2 Controller Boards and Interconnect Cables. Can control 1 to 4 Minidisk Drives. Similar in circuitry and operation to the Altair Floppy Disk Controller. First drive is included in MDS. Additional Drives are available as 88-MDDR'S.

Specifications

Number of slots required in 8800 bus: 2

Number of ICs:
- TTL Logic: 57
- CMOS: 1
- Voltage Regulators: 2

Interconnect
Wiring: Insulation displacement cables and connectors
I/O Addresses
(Octal): 010, 011, 012

Data Transfer
Rate: 1 byte every 64 us
Data Format: Hard Sectored (16 sectors)
Interrupt
System: Interrupt at beginning of sector (Optional - not used for Altair Minidisk BASIC)
Power
Requirements: 1.4A @ 8V

88-MDDR Altair Minidisk Drive

Description Disk drive in case with power supply, Buffer and Address electronics. Includes Interconnect Cable and one blank Minidiskette.

Specifications

1. Data Capacity — Hard Sectored Format
   - a) Per Minidiskette: 71,680 Data Bytes
   - b) Per Track: 2,048 Data Bytes
   - c) Per Sector: 128 Data Bytes
2. Data Transfer Rate: 125,000 Bits per second
3. Access Time
   - a) Disk Enable to READ or WRITE (Function of motor start-up time): 1 sec (min)
   - b) Track to track: 50 ms
   - c) Average Access Time (including motor start-up time): 1.85 sec
   - 3) Worst Case Access Time: 2.9 sec
   - e) Worst Case Latency: 200 ms

Functional Specifications

1. Rotational Speed: 300 rpm (200 ms/Rev.)
2. Track Density: 48 tracks per inch
3. Number of Tracks: 35
4. Number of Sectors: 16
5. Time Per Sector: 12.5 ms

Software: Altair Minidisk Extended BASIC

Software on a Write Protected Minidiskette is virtually identical to Altair Disk BASIC in operation and features. The manual includes Bootstrap Listing and READ/WRITE Drive Code. Be sure to specify the cassette tape or paper tape for Bootstrap Loader, if required.

Programmed PROMS Available ($45 each):

1. MDBL PROM. Minidisk Bootstrap Loader on programmable read only memory IC to be used with 88-PMC PROM Memory Card at highest 256-byte block address.
2. DRWT PROM. Floppy and Minidisk READ/WRITE Test PROM has the fundamental diagnostic tests for checking hardware operation. To be used with the 88-PMC at 3rd highest 256-byte block address.

ADD HARD COPY CAPABILITIES

TO YOUR SYSTEM

Two New Altair Line Printers Available

The Altair C700 calculates the most rapid way to print each line, eliminating unnecessary carriage returns and reducing wear.

MITS introduces two new peripheral products-the C700 high-speed printer and the Q70 printer-to interface with the Altair 8800 series microcomputers.

The Altair C700 is a high-speed serial character printer which prints up to 60 characters per second. The printhead is a 5X7 dot matrix which prints the 64 character subset of the ASCII font.

The Altair Q70 is an upper and lower case/electric typewriter-quality printer specially designed and modified to interface with the Altair 8800 series. It prints 96 upper and lower case characters and symbols at 45 characters per second.

The Q70 is ideal for business systems, word processing systems or any other application which requires high quality printout for letters, labels, documents, etc.

Each printer operates in a highly efficient yet unique manner. The C700 calculates the most rapid way to print each line so that unnecessary carriage returns are eliminated.

The Q70 operates with a 'daisy wheel' system to rapidly print upper and lower case characters. The wheel spins for character location, and a hammer then forces the wheel against the ribbon, paper and platen.

Both the C700 and Q70 provide other superior features for easy operation with minimal maintenance. Each printer is furnished with form tractors to accommodate forms up to 15 inches in width. An electronic Top of Form in the Q70 provides rapid positioning of each sheet. A manual Top of Form switch is also included. With the Q70, ribbon and/or wheel replacement is not an exasperating, messy task. The print wheel is easy to replace, and the ribbon is conveniently enclosed in a cartridge.

CONTINUED

New Products

ADD HARD COPY CAPABILITIES TO YOUR SYSTEM

CONTINUED

When set to print less than 132 centered character lines, the C700 has increased throughput, since the print head returns to the right or left margin. This special feature results in minimal wear since parts motion is reduced to an absolute minimum. A circuit which allows power to be decreased when the printer is inactive further reduces wear.

Both printers are easily integrated into any Altair 8800 computer system. Each comes with its own special interface card which is plugged into the Altair bus. Ribbon cables and connectors complete the interface by connecting the printer to the computer via the interface board.

Check with MITS or your local dealer for prices and availability of the Q70 and C700.

Altair 88-Q70 Specificiations

Print speed and format: 45 characters per second; Full characters of electric typewriter quality.

Forms: Single sheets or continuous multipart forms with or without sprocket holes. Maximum width of 15 inches.

Font: 96 character positions on 'daisy' print wheel; wide variety of fonts available in 10 and 12 pitch. Prestige Elite comes with each unit.

Format: Horizontally — 132 characters and proportional spacing in increments of 1/120".
Vertically — spacing in increments of 1/48" up or down.
Slew rate at 5" per second.

Altair C700 Specifications

Printing method: Character serial, impact, bidirectional.

Printing rate: 60 characters per second, maximum; 26 132-column lines per minute.

Transmission rate: 75,000 characters per second (bit-parallel, character-serial).

Indicators/Switches: On/Off, Select/Deselect, Paper/Out Indicator.

Internal controls: Automatic motor control, paper runaway inhibitor, auto line feed after carriage return.

Paper feed: Tractor (pin-feed) up to 439 mm (17.3"). Up to 5-part forms can be handled. Any sprocket-fed continuous forms may be handled.

Dimensions: 178 mm high, 457 mm deep, 622 mm wide. (7"x28"x24.5")

Weight: 27 kg (60 lbs.)

Electrical requirements: 115/230 VAC ±10%, 50/60 Hz

MITS OFFERS TWO CRTs

By Bennett Inkeles

MITS now offers the LSI ADM-3 and the Altair B-100 CRT terminals for reliable communication in any Altair computer system. Such user benefits as fast, quiet operation and fewer mechanical problems are just some of the reasons to consider CRT implementation.

Both terminals display 24 lines of 80 characters on a non-glare screen and interface at RS-232 and 20 mA current loop levels. Each CRT includes the standard 64 ASCII character set and switch selectable transmission rates from 75 to 19,200 baud.

The LSI ADM-3 and the Altair B-100 also include many unique features suitable for a variety of system applications. The ADM-3's RS-232 extension connector permits interfacing to asynchronous serial ASCII printers so that permanent copies can be easily retained.

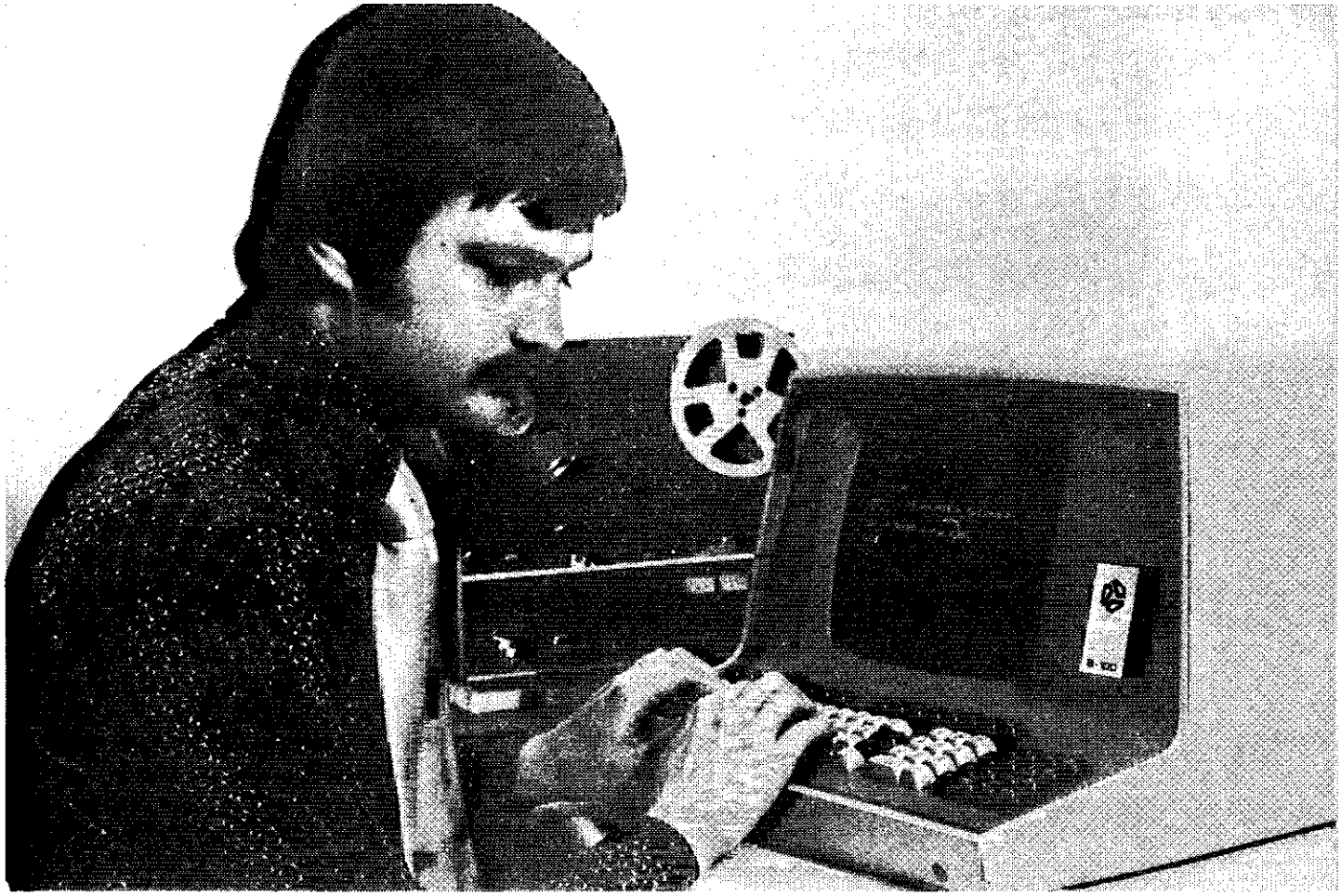
The Altair B-100 includes quad-directional cursor control with carriage return and line feed, an addressable cursor for direct positioning by line and column, 11-key numeric pad with decimal point for simple entry of numeric data, erase mode for cursor to end of line and cursor to end of memory erase.

Altair 16K Memory Board Now Available

The new Altair memory module provides 16K bytes of dynamic Random Access Memory. By implementing low power and fast access dynamic memory ICs, the Altair 88-16MCD runs at a maximum power dissipation of three watts and a maximum time of 350 nanoseconds.

Crystal-controlled logic timing eliminates the need for on-board one-shot multivibrator circuitry. This results in continuous operation without wait states for greater reliability.

Bus strips provide isolation between power and signal lines for maximum noise suppression. Address selection is switch selectable in 4K blocks. Each board requires one slot on the Altair 8800 bus.



The Altair B-100 CRT provides fast, quiet and efficient operation for any Altair computer system.

Publications Offer Variety of Approaches

The April issue of COMPUTER NOTES featured an article entitled, "Publications Provide Novice With Essential Information," (Page 10 May CN), surveyed some of the current computer literature. We regret that the following publications from People's Computer Company were not included.

PEOPLE'S COMPUTER COMPANY is a hardware/software applications-oriented tabloid for the novice as well as the experienced computer user. Information is presented in an informal style with many useful annotations. PCC also acts as a sounding board for novel ideas proposing changes in computer technology.

The format of Dr. DOBB'S JOURNAL OF COMPUTER CALISTHENICS & ORTHODONTIA is similar to that of PCC, but information is aimed at advanced computer users. Software articles, such as games, modifications and unique software systems, make up the bulk of DDJ. Product releases and reader responses round out this informative magazine.

COMPUTER MUSIC JOURNAL offers a comprehensive and highly technical approach to high quality musical applications of digital electronics. This journal is best suited for those who utilize theoretical information for designing computer music systems.

Subscriptions and additional information about these publications can be obtained from:

People's Computer Company
P.O. Box E
Menlo Park, CA 94025

Graphics Display Adds Versatility to Altair System

By Dave Antreasian

The graphics capability of a CRT plays an important role in every computer system. The graphics mode is especially useful for displaying graphs or curves quickly in real-time and is a "must" for such computer games as Ping-Pong.

The following program was written for these applications. Although a hardware restriction limits its versatility, the program is a helpful model for any system display. The hardware restriction refers to cursor control incompatibility with terminals that include this feature.

This program was written for a Beehive 100 terminal, in which the cursor is positioned by sending the following four-byte code to the terminal:

	(ASCII) Octal Code	(ASCII) Decimal Code
Byte 1. Escape Sequence Code	33	27
2. Cursor Address Function Code	50	40
3. Line Number	40-64	*32-54
4. Column Number	40-157	*32-111

*32 Represents line or column 1;
33 Represents line or column 2, etc.

This four byte sequence positions the cursor, and the next byte received prints the desired character. Notice that positioning the cursor beyond either field limit causes the cursor to disappear or scroll the display. If the display is scrolled, the program cannot blank the screen properly, because the first "home" reference point will be lost.

The program is written in Altair BASIC with a user-called Machine Language display routine at line 5000. Although it would be more efficient in machine language, its versatility would be greatly reduced in the translation.

The program accesses an Analog-to-Digital Converter card (88-ADC) which reads eight channels of information and thus defines four points on the display. A fixed background "field" is also displayed.

By substituting another subroutine for the one at line 7100, Altair BASIC can plot any number of desired points simply by writing each point into the "Data Block," which starts at location 8064. (Be sure to limit memory size to 8000 when initializing BASIC.) Each point in the data block is specified by three parameters:

1. the first byte is the line number (ASCII)
2. The second byte is the column number (ASCII)
3. The third byte is the desired character (ASCII)

Notice that the data block is composed of two sections. The first section (bytes 8064 to 8075) is reserved for "moving" data points, and the second section contains "fixed" points. The sections are separated by a data byte = 255. Upon recognition of the (255) code, the display program jumps back to the BASIC program, so the "fixed" field is displayed only once. Then only the "moving" field is refreshed to the display. This significantly reduces the display writing time.

Machine Code Portion of Display Routine

```
037, 100   41  LXIM
     101 [200]
     102 [037]
     103  315  Call Status CK.
     104 [166]
     105 [037]
     106  176
     107  133
     110  323  *
     111  023  Call Status CK.   } Cursor Address
     112  315
     113 [166]
     114 [037]
     115  076
     116  106
     117  323
     120  023
     121  001  LXI (B,C)  0
     122 [000]
     123 [000]
     124  315  Call Status CK.
     125 [166]
     126 [037]
     127  176  ACC MEM
     130  376  CPI } test byte
     131  377     }
     132  310  RET if Zero
     133  323     } Output  data
     134  023     }
     135  043  INCR H.L.
     136  003  INCR B.C.
     137  076  } (ACC)  3
     140  003  }
     141  271  CMP ACC to C
     142  302  JNZ
     143 [124]
     144 [037]
     145  315  Call Status
     146 [166]
     147 [037]
     150  076
     151  033
     152  323
     153  023
     154  315  Call Status   } Cursor Home
     155 [166]
     156 [037]
     157 [076]
     160 [110]
     161  323
     162  023
     163  303  Next Data point
     164 [103]
     165 [037]
     166  333
     167  022  *
     170  037
     171  037                } Status Check
     172  322
     173 [166]
     174 [037]
     175  311
     176  0
     177  0
     200  1st Data Loc.
```

* Port 2/RS232

```
LIST

2 PRINT"NOW DELETE LINES 2-60 AND RE-RUN"
10 POKE 8000 , 33 :POKE 8001 , 128 :POKE 8002 , 31
12 POKE 8003 , 205 :POKE 8004 , 118 :POKE 8005 , 31
14 POKE 8006 , 68 :POKE 8007 , 27 :POKE 8008 , 211
16 POKE 8009 , 19 :POKE 8010 , 205 :POKE 8011 , 118
18 POKE 8012 , 31 :POKE 8013 , 62 :POKE 8014 , 70
20 POKE 8015 , 211 :POKE 8016 , 19 :POKE 8017 , 1
22 POKE 8018 , 0 :POKE 8019 , 0 :POKE 8020 , 205
24 POKE 8021 , 118 :POKE 8022 , 31 :POKE 8023 , 186
26 POKE 8024 , 254 :POKE 8025 , 255 :POKE 8026 , 200
28 POKE 8027 , 211 :POKE 8028 , 19 :POKE 8029 , 35
30 POKE 8030 , 3 :POKE 8031 , 62 :POKE 8032 , 3
32 POKE 8033 , 185 :POKE 8034 , 194 :POKE 8035 , 64
34 POKE 8036 , 31 :POKE 8037 , 205 :POKE 8038 , 118
36 POKE 8039 , 31 :POKE 8040 , 62 :POKE 8041 , 27
38 POKE 8042 , 211 :POKE 8043 , 19 :POKE 8044 , 205
40 POKE 8045 , 118 :POKE 8046 , 31 :POKE 8047 , 62
42 POKE 8048 , 72 :POKE 8049 , 211 :POKE 8050 , 19
44 POKE 8051 , 195 :POKE 8052 , 67 :POKE 8053 , 31
46 POKE 8054 , 219 :POKE 8055 , 18 :POKE 8056 , 31
48 POKE 8057 , 31 :POKE 8058 , 210 :POKE 8059 , 118
50 POKE 8060,31:POKE8061,201:POKE8062,0:POKE8063,0
60 STOP
70 VI=10:BIAS=0:B1=65:L2=66:L3=67:L4=68:ST=8064
80 OUT130,0:OUT131,255:OUT130,4:OUT128,0:OUT129,255:OUT128,44
90 OUT134,0:OUT135,0:OUT134,4:OUT132,0:OUT133,0:OUT132,4
100 OUT19,27:OUT19,69:REM BLANK DISPLAY
150 FOR I=8064TO9267:POKEI,32:NEXT I
200 POKE73,64:POKE74,31:REM SET USR=8000
1000 FOR J=8079 TO 9269:GOSUB8000:POKEJ,Y:NEXT J
5000 Y=USR(Y)
5500 OUT19,32:OUT19,32
5700 POKE8076,255:POKE8077,255:POKE8078,255
6100 IFPEEK(8066)=32 THEN 6500
6200 POKE8066,32:POKE8069,32:POKE8072,32:POKE8075,32
6250 GOTO5000
6500 GOSUB7100
6510 POKE8064,L1+32:POKE8067,L2+32:POKE8070,L3+32:POKE8073,L4+32
6530 POKE8065,C1+32:POKE8068,C2+32:POKE8071,C3+32:POKE8074,C4+32
6540 POKE8066,B1:POKE8069,B2:POKE8072,B3:POKE8075,B4
7000 GOTO 5000
7100 FOR CH=0 TO1:OUT131,CH:OUT129,CH
7200 MS=INP(133):LS=INP(135)
7300 V=((16*MS+((LS/16)AND15))*VI/4095)-BIAS
7350 V=10*(V-4)
7400 IF CH=0 THEN L1=INT(V)
7410 IF CH=1 THEN C1=INT(4*V)
7420 IF CH=2 THEN L2=INT(V)
7430 IF CH=3 THEN C2=INT(4*V)
7440 IF CH=4 THEN L3=INT(V)
7450 IF CH=5 THEN C3=INT(4*V)
7460 IF CH=6 THEN L4=INT(V)
7470 IF CH=7 THEN C4=INT(4*V)
7650 NEXT CH:RETURN
8000 READ Y:IF Y=255 THEN POKEJ,Y:GOTO 5000
8002 RETURN
8500 DATA 35,35,43,36,36,43,37,37,43,38,38,43,39,39,43
8501 DATA 40,40,43,41,41,43,42,42,43,43,43,43,44,44,43
8502 DATA 45,45,43,46,46,43
8590 DATA 255:REM END OF BLOCK
OK
```

SOFTWARE

SOFTWARE NOTES

By John Hayes
General Manager, ASDC

Program Improvement

The Altair User Group is currently reviewing and improving the 300 programs in the User Group Library. Many of the programs will be put in machine readable form and grouped for a particular size computer and language. Single programs will still be distributed in whatever form they are received. Current software listings (printed every month in COMPUTER NOTES and supplied as a separate update sheet to all CN subscribers) will show hardware and language requirements for each program.

When submitting a program, users should indicate the computer (8800 or 680b), the language used and the program's memory requirements in bytes.

Price Increase

Rising costs of program duplication have resulted in a slight increase in prices for some programs. Program and subroutine listings are now $4 for up to 10 pages, $5 for 11-20 pages and $6 for 20-40 pages. These new prices are indicated on the listings in this month's CN insert. Price changes for all previous programs will be issued as an appendix to the AUG Software Catalogue.

Machine-readable code

The User Group is interested in purchasing machine-readable code for software programs already in the library. Anyone who can provide the User Group with either paper tape or cassette forms of earlier programs should contact the Altair User Group Library, Suite 343, 3330 Peachtree, Atlanta, Georgia 30326.

User's Program Handles Complex Numbers

By Chuck Vertrees

Complex numbers often appear in scientific and advanced mathematical calculation. In order to use complex numbers, the reader must first understand the concept of imaginary numbers.

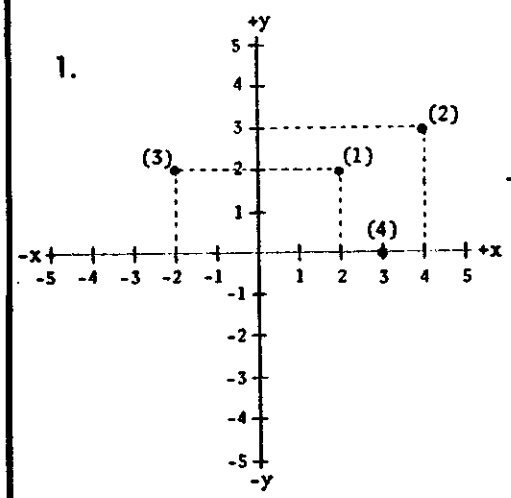
An imaginary number is a real number multiplied with an imaginary operator. (The imaginary operator is usually represented by an "i" or "j.") This operator is defined so that the square root of minus one is equal to the operation, i.e. $\sqrt{-1} = i$, or $(i)^2 = -1$ or $i \times i = -1$. Remember that the square root of a negative number is not usually defined. With the above definitions, the square roots of negative numbers can be represented. For example, $\sqrt{9} = 3$ and $\sqrt{-9} = i3$.

Notice that $(3)^2 = 9$ and $(-3)^2 = 9$. But $(i3)^2 = -9$ and $(-i3)^2 = -9$.

Graph 1 represents imaginary numbers should clarify the difference between imaginary numbers and real numbers. Remember that it is possible to graphically represent a function in two dimension by drawing an "x" and "y" axis and then plotting values of the function.

(1) (X, Y) = (2, 2)
(2) (X, Y) = (4, 3)
(3) (X, Y) = (-2, 2)
(4) (X, Y) = (3, 0)

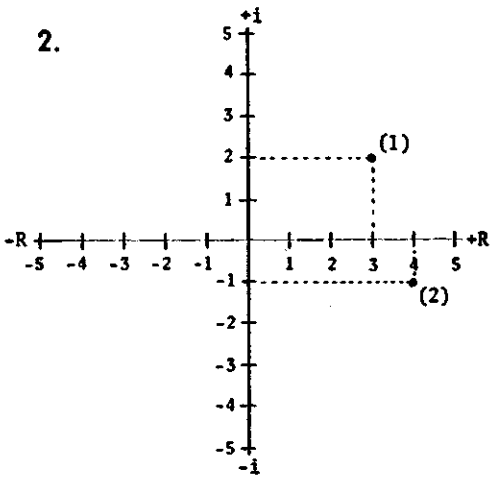
1.



In this case both the X and Y axis are real number lines. Now, let's replace the real Y axis with an imaginary "i" axis so that the Y axis represents imaginary numbers.

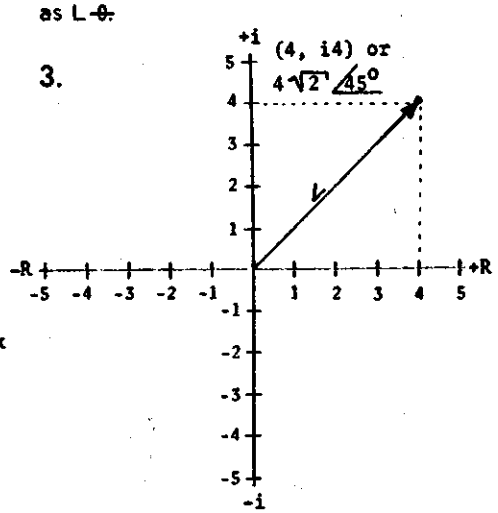
A complex number is defined as a number that has both real and imaginary parts. It is usually represented in rectangular form as A + iB where A is the real part and B is the imaginary part. It is considered as one quantity when doing calculations. For example, the numbers (1) 3 + j2 and (2) 4 - j1 are rerpesented in graph 2.

2.



Graph 3 shows polar form as an alternative method of representing a complex number. Notice that the point 4 + j4 can also be represented as a vector (magnitude and angle), which is written as L θ.

3.



In graph 3 the point 4 + i4 (rectangular) is represented as the length L from the origin at angle θ where $L = \sqrt{X^2 + Y^2} = \sqrt{4^2 + 4^2} = \sqrt{32} = 4\sqrt{2}$ and $\tan \theta = 4/4$ or $\theta = \arctan 1 = 45^0$, which is represented in Standard Polar Form as $4\sqrt{2}\angle 45^0$.

The following complex number flow chart and program written in Altair BASIC is available through the Altair User Group Software Library.
(AUG Library number: 7014)

Complex Number Intrepreter for BASIC
(Used as a Program or Subroutine)

By Dr. John J. Herro

Annotated Program Listing

```
main prgm.    49990 LZ=0: INPUT S$: GOSUB 50010: GOTO 49990
subroutine    50000 LZ=1
              50010 SZ$="( ?+-*/ %!DRPEL ": IF S$=""THEN RETURN
              50020 FOR JZ=1 TO LEN (S$): GOSUB 50030: NEXT JZ:
              RETURN
              50030 FOR IZ=1 TO 16: IF MID$ (S$, JZ, 1)=MID$ (SZ$,
              IZ, 1) THEN 50050
bad char.     50040 NEXT IZ: PRINT "UNRECOGNIZED CHARACTER IN":
              GOTO 50170
good char.    50050 ON IZ GOTO 50070, 50290, 50240, 50220, 50210, 50270, 50260,
                  50300
              50060 ON IZ-8 GOTO 50200, 50190, 50280, 50330, 50120, 50320, 50150:
              RETURN
(             50070 FOR IZ=JZ TO LEN (S$): IF MID$ (S$, IZ, 1)=")" THEN
                  50090
missing )     50080 NEXT IZ: PRINT "MISSING ) IN": GOTO 50170
)             50090 AZ=VAL(MID$(SS$, JZ+1)): EZ=0
;             50100 FOR KZ=JZ TO IZ: IF MID$ (S$, KZ, 1) =";" THEN
              EZ=VAL (MID$ (S$, KZ+1))
              50110 NEXT KZ: JZ=IZ: RETURN
P             50120 IZ=1.570796*SGN(EZ): IF AZ THEN IZ=ATN(EZ/AZ)
              50130 IF AZ 0 THEN IZ=IZ+6.283185*((EZ 0)+.5)
              50140 AZ=SQR(AZ 2+EZ 2): EZ=IZ: RETURN
L             50150 GOSUB 50120: IF AZ THEN AZ=LOG(AZ): RETURN
arith err.    50160 PRINT "LOG(0) OR /0 OR 0 (X;Y),X =0 IN"
error         50170 PRINTS$: PRINT TAB(JZ-1)" "
              50180 IF LZ THEN END ELSE JZ=LEN(S$): RETURN
!             50190 IZ=AZ: AZ=BZ: KZ=EZ: EZ=FZ: GOSUB 50230:
              DZ=IZ: HZ=KZ: RETURN
%             50200 IZ=AZ: AZ=BZ: BZ=IZ: IZ=EZ: EZ=FZ: FZ=IZ: RETURN
-             50210 AZ=-AZ: EZ=-EZ
+             50220 AZ=BZ+AZ: EZ=FZ+EZ
drop stack    50230 BZ=CZ: CZ=DZ: FZ=GZ: GZ=HZ: RETURN
?             50240 PRINT AZ;CHR$(43-(EZ 0)*2) "I" ABS (EZ): GOSUB 50120
              50250 PRINT "=" AZ "AT" EZ*57.29578 "DEG.": GOTO 50330
/             50260 GOSUB 50150: AZ=-AZ: EZ=-EZ: GOSUB 50320
*             50270 IZ=AZ*BZ-EZ*FZ: EZ=EZ*BZ+AZ*FZ: AZ=IZ:
              GOTO 50230
D             50280 EZ=EZ/57.29578: GOTO 50330
              50290 DZ=CZ: CZ=BZ: BZ=AZ: HZ=GZ: GZ=FZ:
              FZ=EZ: RETURN
              50300 IF BZ=0 AND FZ=0 AND AZ 0 THEN AZ=0: EZ=0:
              GOTO 50230
              50310 GOSUB 50200: GOSUB 50150: GOSUB 50270
E             50320 AZ=EXP (AZ)
R             50330 IZ=AZ*COS(EZ): EZ=AZ*SIN(EZ): AZ=IZ: RETURN
```

NOTE: Spaces were inserted in this program listing to improve readability. Deleting all unnecessary spaces will save 149 bytes. This program was written in Extended BASIC. It will run in 8K BASIC if line 50180 is replaced with:

```
50180 IF LZ THEN END
50185 JZ=LEN(S$): RETURN
```

A 30-page user manual is available for $6 from the Altair User Group Software

CONTINUED

Resequencer Program Keeps Lines Evenly Incremented

By Doug Jones

Whenever I write a program in BASIC, the first few line numbers are always nicely incremented — 10, 20, 30 etc. The subroutines also start at easy-to-remember line numbers, such as GOSUB 1000. But things don't seem to stay that nice. I forget to put in a remark explaining what the routine is about and have to squeeze in 999REM. By the time the program is complete, the line numbers look anything but well planned: 10, 16, 17, 33, 49 . . .

The following resequencer program gives Altair 680 BASIC users professional-looking, evenly incremented lines. The user may choose the starting line number and the step of increment.

The BASIC resequencer program is only 33 lines long. I intentionally wrote it in unusual line numbers. To avoid being a 'space' gobbler, the program lines are incremented by one. The last line, 63999, is the largest possible line number that 680 BASIC will accept without displaying an error message.

How to Use the Resequencer.

Write or load your program containing the odd line numbers. Then load the resequencer and type in the starting line — RUN63967. After a few seconds, the computer will ask:

OLD LINE, NEW, STEP, NULLS?

Your answer might be 7,100,10, 5. This indicates that (1) your old program began at line 7, (2) you want to renumber it starting with line 100, (3) in increments of 10 and (4) set NULLS to 5.

Resequencer does not rewrite a program in memory. Instead, it produces a typed listing properly sequenced to the users specifications. With this listing and a punched tape, the user can type NEW and then re-enter the newly sequenced program.

Sometimes the resequencer might be half-way through a program and suddenly show an error message. For example, the newly sequenced listing might print the following message.

240 IF A=B THEN GOSUB
ERROR ON LINE 473
OK

The line number referenced to in the error message is in the original program. This can occur if the resequencer is asked to resequence a line that does not exist.

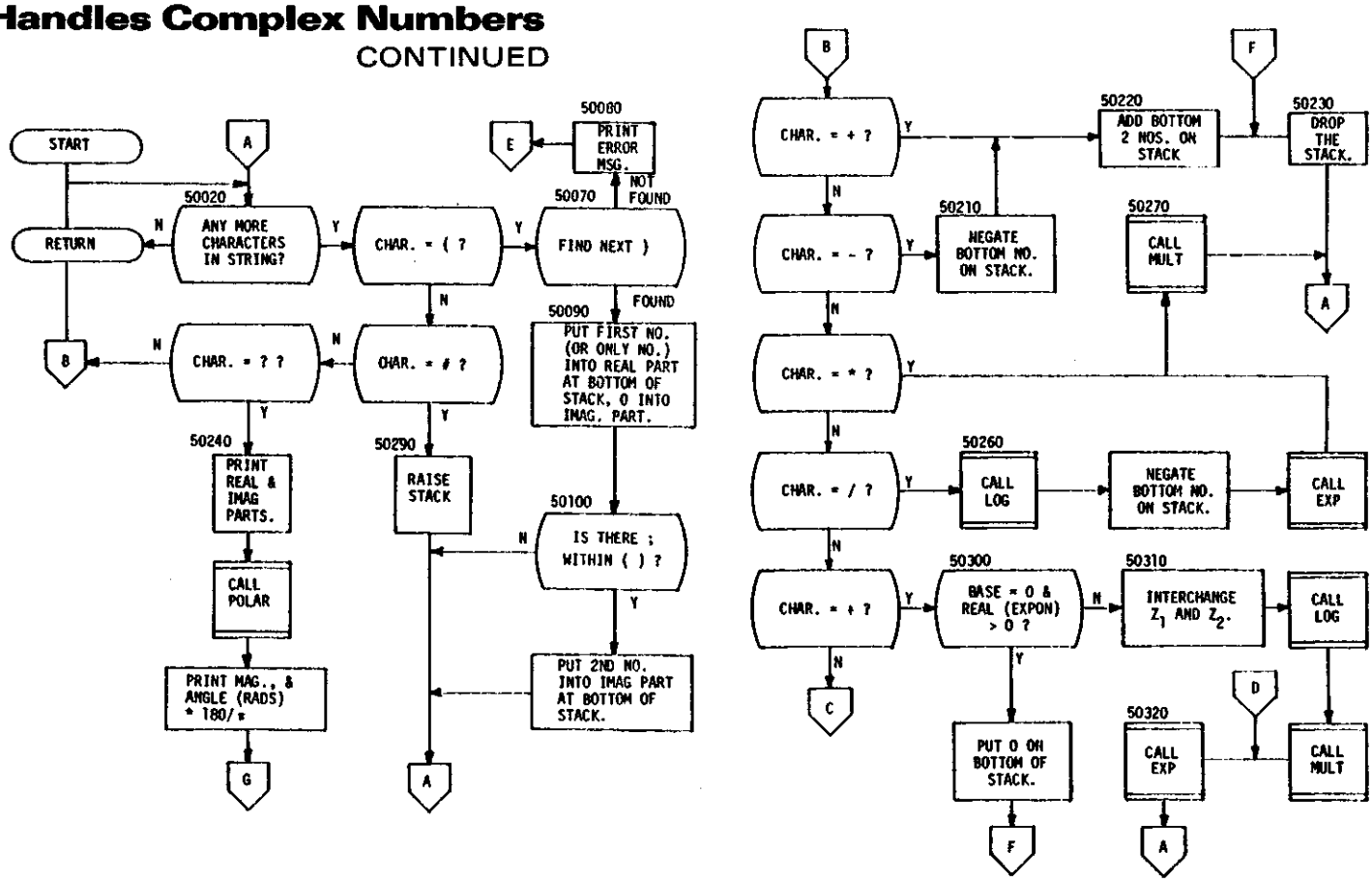
How Does the Resequencer Work?

Resequencer runs through a program three times. The first time it looks for the first OLD line . When it is found, the system examines the program line by line, counting lines as it goes. It stops when it reaches its own first line number.

On the second pass it gathers existing line numbers into an array for

User's Program Handles Complex Numbers

CONTINUED



future GOTO and GOSUB references.

Resequencer displays the number of lines the program contains. It tells the user to READY PUNCH and then pauses while the operation is performed. Starting leader and control-0 are punched automatically to prevent echo.

On the third pass, the program begins listing. Each initial line number is the incremental step from the previous line number. Referenced line numbers are then a function of the starting parameters and the position of a line number in the array.

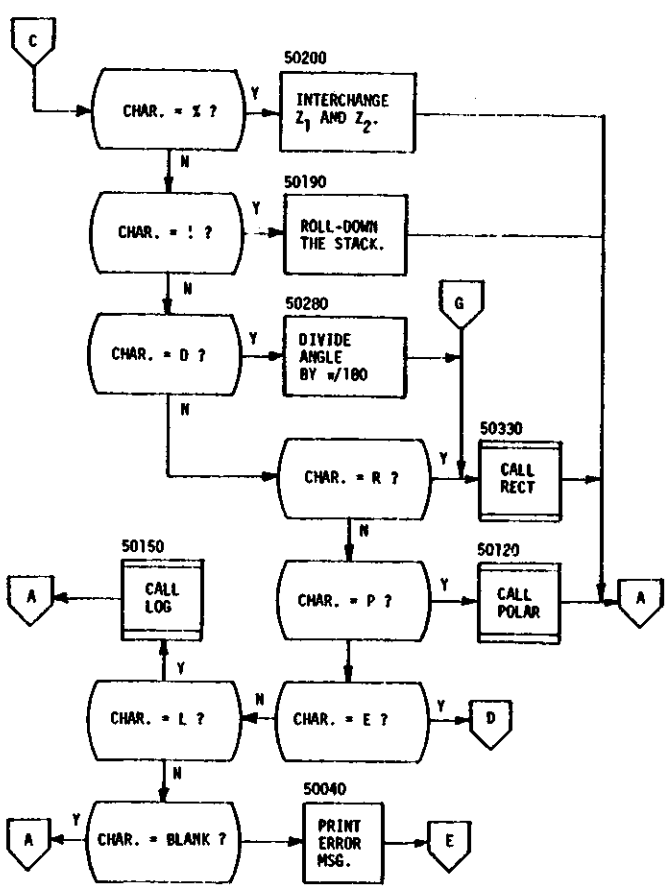
The resequencing and listing stop when the program reaches its starting line number. It then resets nulls to zero, prints an END to undo the control-0 echo suppression and spills out some finishing leader.

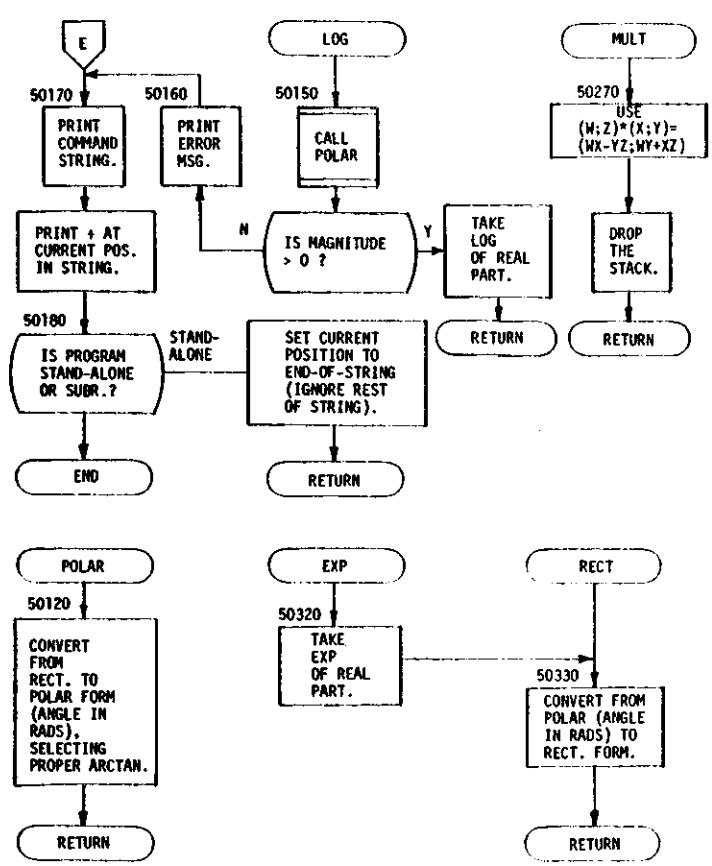
Since resequencer starts looking for the old first line in the top part of BASIC, there is a slight chance that it

```
LIST

 REM ***** DEMONSTRATION OF RESEQUENCER PROGRAM *****
 INPUT A
 IF A=B THEN 493
 IF A=C THEN GOSUB 1:
 ON A GOTO 1,37,49
 GOSUB 1:GOTO 77
 GOTO 1
 DATAEND,FOR,NEXT,DATA,INPUT,DIM,READ,LET,GOTO,RUN,IF,RESTORE
BREAK
OK

RUN63967
OLD LINE #, NEW #, STEP, NULLS? 0,10,10,5
LINES = 7
READY PUNCH
```

```
10 REM ***** DEMONSTRATION OF RESEQUENCER PROGRAM *****
20 INPUT A
30 IF A=B THEN 70
40 IF A=C THEN GOSUB 20:
50 ON A GOTO 20,30,40
60 GOSUB 20:GOTO 60
70 GOTO 20
END

OK
```

CONTINUED





New Policy Adopted

Due to rising costs, MITS will now only accept orders for $25 or more. Please place all other orders with your local dealer.

Resequencer Program Keeps Lines Evenly Incremented

CONTINUED

might find some assembled instruction with the same numeric value as the starting line number. However, I haven't seen it happen yet, and I've used the program to resequence everything I can find on a paper tape.

It usually takes only a few seconds to find the starting line number — just a bit longer if the trigonometric functions are initialized in BASIC. Even if the old first line number doesn't exist, it will try its darndest to find it anyway. I've seen those lights blink for a half-hour, and it still wouldn't give up.

An optional old starting line number further into a program will permit resequencing of a latter portion of the program. This works as long as there is no previous reference to the earlier part of the program that is not being resequenced.

The user can increment the line numbers by any step, even 3.2 or -9.7. All GOTOs and GOSUBs are referenced correctly to line 437.13. Unfortunately, Altair BASIC will not run a program with line numbers like that. What can be done with such a listing? If you have a time-share computer access at work, give it to the time-share coordinator. Tell him or her: "Look what your time-share has done to all my computer files . . . now none of them run!"

```
NULL5:FORN=11010:?CHR$(0)::NEXT::?CHR$(15):LIST

63967 DATAEND,FOR,NEXT,DATA,INPUT,DIM,READ,LET,GOTO,RUN,IF,RESTORE
63968 DATAGOSUB,RETURN,REM,STOP,ON,NULL,WAIT,DEF,POKE,PRINT,CONT
63969 DATALIST,CLEAR,NEW,TAB(,TO,FN,SPC(,THEN,NOT,STEP,+,-,*,/,^
63970 DATAAND,OR,>,=,<,SGN,INT,ABS,USR,FRE,POS,SQR,RND,LOG,EXP,COS
63971 DATASIN,TAN,ATN,PEEK,LEN,STR$,VAL,ASC,CHR$,LEFT$,RIGHT$,MID$
63972 READC$:IFC$<>"END"THEN63972
63973 DIMA$(65):FORN=11065:READA$(N):NEXT:C=0:P=6142:Z=127:Y=63967
63974 INPUT"OLD LINE #, NEW #, STEP, NULLS";OL,NW,IB,M:A$(0)="END"
63975 P1=P:IFOL<>256*(PEEK(P+2))+PEEK(P+3)THENP=P+1:GOTO63975
63976 IF256*(PEEK(P+2))+PEEK(P+3)=YTHENPRINT"LINES ";C:GOTO63978
63977 P=((256*(PEEK(P)))+PEEK(P+1)):C=C+1:GOTO63976
63978 PRINT"READY PUNCH":FORK=1TO5:PRINT:NEXT:FORK=11TO1000:NEXT
63979 NULL(M):FORK=1TO100:PRINTCHR$(0)::NEXT:PRINTCHR$(15):PRINT
63980 DIMA(C):P=P1:FORN=1TOC:A(N)=((256*(PEEK(P+2)))+PEEK(P+3))
63981 P=((256*(PEEK(P)))+PEEK(P+1)):NEXTN:LL=C:C=0:C$="":P=P1
63982 IFC=LLTHENPRINT"END":NULL0:FORN=1TO50:PRINTCHR$(0)::NEXT:END
63983 P2=((256*(PEEK(P)))+PEEK(P+1)):P=P+2:LN=NW+(IB*C)
63984 PRINTRIGHT$(STR$(LN),LEN(STR$(LN))-1);" ";:C=C+1:P=P+2:GOTO63986
63985 P=P+1:IFP=P2THEN63982
63986 F=PEEK(P):IFF>ZTHENPRINTA$(F-128)::IFF=136ORF=140ORF=158THEN63990
63987 IFF>ZTHEN63985
63988 IFF=0THENPRINT:PRINTCHR$(F)::P=P+1:GOTO63982
63989 PRINTCHR$(F)::GOTO63985
63990 P=P+1:G=PEEK(P):IFG>=48ANDG<=57THENC$=C$+CHR$(G):FL=1:GOTO63990
63991 IFG=32ANDFL=0THEN63999
63992 IFFL=1ANDG<ZTHEN63996
63993 IFF<>15ANDG<>0THEN63995
63994 F=G:GOTO63986
63995 PRINT:PRINT"ERROR ON LINE #";A(C):NULL0:END
63996 FORK=1TOLL:IFA(K)=VAL(C$)THENX$=STR$(NW+(IB*(K-1))):GOTO63998
63997 NEXTK:GOTO63995
63998 PRINTRIGHT$(X$,LEN(X$)-1)::FL=0:C$="":IFG<>32ANDG<>44THEN63994
63999 PRINTCHR$(G)::GOTO63990
OK
```

Circuit Analysis Applications Expanded to Run with Altair BASIC

By Tom Simpson

A circuit is a group of components with fixed characteristics interacting in some prescribed manner. An electrical engineer's job is to develop mathematical relationships between the various components of a circuit that describe the total behavior of the network. In order to develop these relationships, electronic circuits must be analyzed.

Computer analysis of electronic circuits is nothing new in itself. However, this program brings automated analysis to the Altair computer running BASIC. It is, therefore, inexpensive to implement and easy to modify for special needs.

The analysis technique and output formats used in this program are based on those in IBM's ECAP (Electronic Circuit Analysis Program), written for the IBM 1620. The input format is slightly different than ECAP's. Although ECAP does three different types of circuit analysis, this program does only steady state D.C. analysis.

The solution technique used is nodal analysis. It's based on Kirchoff's current law, which says the algebraic sum of the currents leaving (or entering) a node is zero, and Ohm's Law, which says voltage equals the product of resistance and current. Applying Kirchoff's current law, Ohm's Law and some matrix alegbra, a computer can find the solution of a circuit after a description of the circuit is in memory.

Circuits are a series of nodes or places where things are connected together and branches or connections between the nodes containing electronic components. Input of the circuit to the

program is in terms of these branches and their nodes.

The arrangement of each branch is standardized and assumed by the solution. (See Fig. 1.) Only the basic electronic components shown in Figure 1 are recognized by the solution. Ohm's Law covers voltage, current and resistance but says nothing about transistors, diodes or 7400s. Since only a steady state solution is found in this program, capacitors are considered open circuits and inductors as short circuits.

However, with the proper combination of the allowed elements shown in Figure 1, the user can "model" transistors, diodes and other non-linear components in a non-steady state and non-D.C. analyzer.

Every branch must contain one and only one non-zero resistance or conductance. According to Ohm's Law, zero resistance requires an infinite current flowing in the branch. Each branch may have a voltage source, current source or dependent current source. Since this program has only a steady state D.C. analysis, the circuit must have at least one independent source, one node other than ground and one branch. Every node must have a path to ground through the branches of the circuit. Without the path to ground, certain sections of the circuit would "float" free of the rest.

Starting with the number one, each node is numbered in increments of one. Ground is always node zero. Branches and dependent current sources are numbered consecutively from one. There is no branch or dependent source zero. Output from the solution is in terms of these numbers.

Input Program

The input program is a compromise between ease of programming and ease of use. The easiest program to write might ask for the number of branches and then quiz the user on the values of it's components and the nodes to which it is connected. This would require a lot of operator and computer time that could be put to more productive use. The easiest program would permit a user to enter and edit any data in any order at any time. But such a program would be rather complicated to write.

```
                ALTAIR DC ELECTRONIC CIRCUIT ANALYSIS PROGRAM
                ADAPTED FROM IBM 1620 ECAP TO ALTAIR 8K BASIC
                ( VERSION 3.2 OR 4.0 ) BY TOM SIMPSON 2/14/77

10 CLEAR99:M0=0:DEFFNI(X)=INT(-(X>0)*X):INPUT"NODES";X:MN=FNI(X-1)
20 INPUT"BRANCHES";X:MB=FNI(X-1):INPUT"DEP SCS";X:MD=FNI(X-1)
30 DIM M(MB,MN),NI(MB),NF(MB),Z(MN+1,MN+1),Y(MB),YL(MB),YH(MB),CU(MB)
40 DIM A(MB),AL(MB),AH(MB),E(MB),EL(MB),EH(MB),D(MD),DL(MD),DH(MD)
50 DIM EQ(MN),IP(MN),NV(MN),BV(MB),IN(MN,1),W(MN),WL(MN),WH(MN)
60 DIM RV(MD),CL(MD),SD(MN):D$=" NBG ":K$=" /%()KM":C$=" BD*GREI ":K1=1
70 PRINT:PRINT"ALTAIR DC ECAP":PRINT:P$="CMBVEVECPLPDWCSDNVBCCVIH":K0=3:
KM=-1:K2=2
80 K3=3:K4=4:K5=5:K6=6:K7=7:K9=9:NN=KM:NB=KM:ND=KM:NY=KM:NE=KM:NA=KM
90 PC=.01:CT=1000:CM=1E6:CI=.001:S6=36
110 PR=K0
120 INPUTX$:L=LEN(X$):I=K0:IFX$<>"RUN"THEN150
130 IFNY=NBAND(NE>KMORNA>KM)ANDNB>KMANDNN>KMTHEN1060
140 PRINT"MIN REQUIREMENTS FOR SOLUTION NOT MET":END
150 IFX$="MODIFY"THENM0=M0+K1:GOTO120
160 IFLEFT$(X$,K1)="R"THEN120
170 IFX$="NEW"THEN10
180 IFLEFT$(X$,K5)="PRINT"THENI=K5:GOTO900
190 IFX$="END"THENEND
200 I=I+K1:IFI>LTHEN120
210 FORK=K1TOK4:IFMID$(X$,I,K1)<>MID$(C$,K,K1)THENNEXT:GOTO1000
220 ONKGOTO230,230,230,470
230 TY=K-K1:GOSUB620:X=T-K1:IFTY=K2THEN280
240 IFX>MBTHENPRINT"BRANCH # TOO GREAT";:GOTO1030
250 IFI0THEN470
260 IFND>KMTHENPRINT"B RECORDS CANNOT FOLLOW D RECORDS";:GOTO1030
270 NB=NB+K1:GOTO310
280 IFX>MDTHENPRINT"DEP SOURCE # TOO GREAT";:GOTO1030
290 IFM0THEN370
300 ND=ND+1
310 I=I+K1:FORK=K1TOK3:IFMID$(X$,I,K1)<>MID$(D$,K,K1)THENNEXT:GOTO1000
320 ONKGOTO310,420,340
340 IFTY=K1THEN1000
350 GOSUB600:TL=TL-K1:TH=TH-K1:IFTL>MBORTH>MBTHEN1010
360 CL(ND)=TL:RV(ND)=TH
370 I=I+K1:FORK=K3TOK5:IFMID$(X$,I,K1)<>MID$(D$,K,K1)THENNEXT:GOTO1000
380 IFK=K5THEN370
390 TT=K-K2:GOSUB610:IFTT=K2THEN410
400 K=CL(ND):T=T*Y(K):TL=TL*YL(K):TH=TH*YH(K)
410 D(ND)=T:DL(ND)=TL:DH(ND)=TH:GOTO120
420 IFTY=K2THEN1000
430 GOSUB600:TL=TL-K1:TH=TH-K1:IFTL>MNORTH>MNTHEN1020
440 IFTL>NNTHENNN=TL
450 IFTH>NNTHENNN=TH
460 NI(X)=TL:NF(X)=TH
470 I=I+K1:IFI>LTHEN120
480 FORK=K5TOK7:IFMID$(X$,I,K1)<>MID$(C$,K,K1)THENNEXT:GOTO1000
490 IFX=K0THEN470
500 TT=K-K4:GOSUB610:IFI=K0AND(TT=K1ORTT=K2)THEN1040
510 T=K1/T:T1=TL:TL=K1/TH:TH=K1/T1
520 Y(X)=T:YL(X)=TL:YH(X)=TH:IFM0=K0THENNY=NY+K1
530 GOTO470
540 E(X)=T:EL(X)=TL:EH(X)=TH:IFM0=K0THENNE=NE+K1
550 GOTO470
560 A(X)=T:AL(X)=TL:AH(X)=TH:IFM0=K0THENNA=NA+K1
570 GOTO470
600 ZZ=K2:GOTO650
610 ZZ=K1:GOTO630
620 ZZ=K0
630 I=I+K1:Z$=MID$(X$,I,K1):IFZ$=" "ORZ$="="THEN630
640 GOSUB790:T=T1*MU:IFZZ=K0THENRETURN
650 I=I+K1:IFMID$(X$,I,K1)=" "THEN650
660 IFMID$(X$,I,K1)<>"("THENTL=T:TH=T:I=I-K1:ONZZGOTO760,1000
670 I=I+K1:GOSUB790:TL=T1*MU
680 I=I+K1:FORK=K1TOK3:IFMID$(X$,I,K1)<>MID$(K$,K,K1)THENNEXT:GOTO1000
690 ONKGOTO680,700,710
700 I=I+K1:GOSUB790:TH=T1*MU:GOTO730
710 IFZZ<>K1THEN1060
720 TL=ABS(T)*TL*PC:TH=T+TL:TL=T-TL
730 I=I+K1:IFI>LORMID$(X$,I,K1)=")"THENRETURN
740 IFMID$(X$,I,K1)=" "THEN730
750 GOTO1000
760 RETURN
790 T1=VAL(MID$(X$,I))
```

CONTINUED

Circuit Analysis Applications

CONTINUED

The format used here falls somewhere in between these two extremes. It's not a particularly simple program, but it's straightforward. Although it doesn't allow a completely free format, it's fairly easy to learn.

Input Format

See the sample runs for the input format. Branches are described with the letter "B" as the first non-space character and followed by the branch number. A node "N" is followed by two node numbers in parenthesis. The first number is the initial node of this branch. The second is the final node. Initial and final nodes are determined by the positive current direction in the branch, which is set arbitrarily by the user. Current flows from the initial node to the final node. Then the component values in the branch are entered in any order. R= defines resistance, G= defines conductance, E= defines voltage source and I= defines independent current source. Only component values included in the branch should be specified. The only thing missing from Fig. 1 is dependent current sources, which sometimes cause modeling difficulties.

A dependent current source has a value which depends on a quantity in another branch of the circuit line. Dependent current sources are carried internally as transconductance values. Another often used value is current gain or beta value. The values are related by the equation: Transconductance = beta / resistance of controlling branch. The problem is that if a beta value is used to describe a dependent source, then the resistance of the controlling branch must be previously defined. Otherwise, a "divide by zero" error occurs when the expression above is evaluated. So, dependent sources are entered separately and only after all other branch data has been entered. Line 260 in the program does not let the user enter any more branch data after one or more dependent sources are entered.

The input line for dependent sources follows the format branch data. The first non-space character is the letter "D", followed by the dependent source number. Next is a branch group with the letter "B", followed by two branch

```
800 FORJ=ITOL:FORK=K1TOK7:IFMID$(X$,J,K1)<>MID$(K$,K,K1)THENNEXTK,J
830 I=J-K1:MU=K1:IFZZ<>K1ORK<K6ORK>K7THENRETURN
840 I=I+K1:IFTT=K2ANDK=K6THENMU=CT:RETURN
850 IFTT=K2ANDK=K7THENMU=CM:RETURN
860 I=I+K1:IFTT=K3ANDMID$(X$,I,K1)="V"THEN890
870 IFTT=K4ANDMID$(X$,I,K1)<>"A"THEN1000
890 MU=CI:RETURN
900 I=I+K1:IFI>LTHEN120
910 IF MID$(X$,I,K1)=" "THEN900
920 IFMID$(X$,I,K3)="ALL"THENPR=4095:GOTO120
930 FORK=K0TO11:IFMID$(X$,I,K2)<>MID$(P$,K2*K+K1,K2)THENNEXT:GOTO1000
940 PR=PROR(K2↑K):I=I+K1:GOTO900
1000 PRINT"SYNTAX ERROR-POS.";I:END
1010 PRINT"BRANCH # TOO GREAT":END
1020 PRINT"NODE # TOO GREAT":END
1030 PRINT" - TRY AGAIN":GOTO120
1040 PRINT"ZERO RESISTANCE OR CONDUCTANCE NOT ALLOWED":END
1060 IFPR=K0THENPRINT"NO OUTPUT SPECIFIED";:GOTO1030
1070 PRINT:PRINT:PRINT:PRINT:IFMOTHENPRINT"** MODIFY";MO;"**":GOTO1090
1080 PRINT"** SOLUTION **"
1090 PRINT:PRINT:PRINT
1100 FORK=K0TONB:FORJ=K0TONN:M(K,J)=K0:NEXTJ,K
1110 FORI=K0TONB:IFNI(I)>KMTHENM(I,NI(I))=K1
1120 IFNF(I)>KMTHENM(I,NF(I))=KM
1130 NEXT
1200 FORI=K0TONN:Z(I,NN+K1)=K0:Z(NN+K1,I)=K0:FORJ=K0TONN:Z(I,J)=K0
1210 FORK=K0TONB:T=M(K,I):T1=M(K,J):IFT=K0ORT1=K0THEN1230
1220 Z(I,J)=Z(I,J)+T*T1*Y(K)
1230 NEXTK,J,I:IFND=KMTHEN1400
1300 FORI=K0TOND:L=RW(I):M=CL(I):FORJ=K0TONN:FORK=K0TONN
1310 T=M(L,J):T1=M(M,K):IFT=K0ORT1=K0THEN1330
1320 Z(J,K)=Z(J,K)+T*T1*D(I)
1330 NEXTK,J,I
1400 FORI=K0TONB:CU(I)=K0:CU(I)=A(I)-Y(I)*E(I):NEXT:IFND=KMTHEN1440
1410 FORI=K0TOND:L=RW(I):L1=CL(I)
1420 IFE(L1)THENCU(L)=CU(L)-D(I)*E(L1)
1430 NEXT
1440 FORN=K0TONN:EQ(N)=K0:FORM=K0TONB:EQ(N)=EQ(N)+M(M,N)*CU(M):NEXTM,N
1450 IFPRANDK1THENGOSUB2000
1500 FORI=K0TONN:IP(I)=K0:NEXT:FORI=K0TONN:T=K0:FORJ=K0TONN
1510 IFIP(J)=K1THEN1560
1520 FORK=0TONN:IFIP(K)>K1THENRETURN
1530 IFIP(K)=K1THEN1550
1540 IFABS(Z(J,K))>ABS(T)THENIR=J:IC=K:T=Z(J,K)
1550 NEXTK
1560 NEXTJ
1570 IP(IC)=IP(IC)+K1:IFIR=ICTHEN1590
1580 FORL=K0TONN:T=Z(IR,L):Z(IR,L)=Z(IC,L):Z(IC,L)=T:NEXTL
1590 IN(I,K0)=IR:IN(I,K1)=IC:PV=Z(IC,IC):Z(IC,IC)=K1
1600 FORL=K0TONN:Z(IC,L)=Z(IC,L)/PV:NEXTL
1610 FORL1=K0TONN:IFL1=ICTHEN1640
1620 T=Z(L1,IC):Z(L1,IC)=K0:FORL=K0TONN
1630 Z(L1,L)=Z(L1,L)-Z(IC,L)*T:NEXTL
1640 NEXTL1
1650 NEXTI
1660 FORI=NNTOK0STEPKM:IFIN(I,K0)=IN(I,K1)THEN1690
1670 IR=IN(I,K0):IC=IN(I,K1):FORK=K0TONN:T=Z(K,IR):Z(K,IR)=Z(K,IC)
1680 Z(K,IC)=T:NEXTK
1690 NEXTI:IFPRAND2048THENGOSUB2600
1700 FORN=K0TONN:NV(N)=K0:FORM=K0TONN:NV(N)=NV(N)+Z(N,M)*EQ(M):NEXTM,N
1710 IFPRAND256THENGOSUB2450
1720 FORM=K0TONB:BV(M)=K0:FORN=K0TONN:BV(M)=BV(M)+M(M,N)*NV(N):NEXTN,M
1730 IFPRANDK2THENGOSUB2700
1740 FORI=K0TONB:BV(I)=BV(I)+E(I):NEXT:IFPRANDK4THENGOSUB2800
1750 FORI=K0TONB:CU(I)=K0:CU(I)=Y(I)*BV(I)-A(I):NEXT:IFND=KMTHEN1790
1760 FORI=K0TOND:L=RW(I):L1=CL(I)
1770 IFBV(L1)THENCU(L)=CU(L)+D(I)*BV(L1)
1780 NEXT
1790 IFPRAND512THENGOSUB2900
1800 FORI=K0TONB:CU(I)=CU(I)+A(I):NEXT:IFPRAND8THENGOSUB3000
1810 FORI=K0TONB:CU(I)=CU(I)*BV(I):NEXT:IFPRAND16THENGOSUB3100
1820 FORI=K0TONB:BV(I)=BV(I)-E(I):NEXT:IF(PRAND224)=K0THEN110
1830 FORJ=K0TONN:T=NV(J):W(J)=T:WL(J)=T:WH(J)=T:SD(J)=K0:NEXT
1840 PV=K1:FORIT=K1TOK4:NU=NB:IFIT<>K2THEN1870
1850 IFND=KMTHEN2360
1860 NU=ND
1870 FORI=K0TONU:ONITGOTO1910,1910,1880,1900
1880 IFE(I)=K0THEN2350
1890 GOTO1910
1900 IFA(I)=K0THEN2350
1910 FORJ=K0TONN:NV(J)=K0:NEXTJ:ONITGOTO1920,1930,1940,1990
```

```
1920 T1=Y(I)*(E(I)+BV(I)):II=I:TR=K1:GOTO2000
1930 II=RW(I):JJ=CL(I):T1=-(E(JJ)+BV(JJ)):TR=K3:GOTO2000
1940 IFND=KMTHEN1980
1950 FORJ=K0TOND:IFI<>CL(J)THEN1970
1960 II=RW(J):T1=-D(J):TR=K4:GOTO2000
1970 NEXTJ
1980 II=I:T1=-Y(I):TR=K3:GOTO2000
1990 II=I:T1=K1:TR=K3
2000 IFNI(II)=KMTHENNI=NN+K1:GOTO2020
2010 NI=NI(II)
2020 IFNF(II)=KMTHENNF=NN+K1:GOTO2040
2030 NF=NF(II)
2040 FORN=K0TONN:NV(N)=NV(N)+(Z(N,NI)-Z(N,NF))*T1:NEXTN
2050 ONTRGOTO2060,2090,2100,1970
2060 IFND=KMTHEN2100
2070 FORJ=K0TOND:IFI<>CL(J)THEN2090
2080 II=RW(J):T1=Y(I)*D(J)*(E(I)+BV(I)):TR=K2:GOTO2000
2090 NEXTJ
2100 IF(PRAND32)=K0THEN2240
2110 IFPVTHENPRINT:PRINT"*PARTIAL DERIVATIVES*":PV=K0
2115 PRINT:PRINT"NODE VOLTAGES WITH RESPECT TO ";
2120 ONITGOTO2130,2140,2150,2160
2130 PRINT"BRS. IN BRANCH";I+K1:GOTO2170
2140 PRINT"VOLTAGE SOURCE IN BRANCH";I+K1:GOTO2170
2150 PRINT"CURRENT SOURCE ACROSS BRANCH";I+K1
2160 PRINT:ONITGOTO2180,2190,2200,2210
2180 T1=PC/Y(I):GOTO2220
2190 T1=PC*D(I):GOTO2220
2200 T1=PC*E(I):GOTO2220
2210 T1=PC*A(I)
2220 PRINT"NODE NO.","PARTIALS","SENSITIVITIES":PRINT:FORN=K0TONN
2230 T=NV(N)*ABS(T1):PRINTN+K1,NV(N),T:NEXTN
2240 ONITGOTO2250,2260,2270,2280
2250 T1=(Y(I)-YL(I))/(Y(I)*YL(I)):T2=(Y(I)-YH(I))/(Y(I)*YH(I)):GOTO2290
2260 T1=DH(I)-D(I):T2=DL(I)-D(I):GOTO2290
2270 T1=EH(I)-E(I):T2=EL(I)-E(I):GOTO2290
2280 T1=AH(I)-A(I):T2=AL(I)-A(I)
2290 T=T1-T2:FORJ=K0TONN:ONSGN(NV(J))+K1GOTO2320,2310
2300 WH(J)=WH(J)+NV(J)*T2:WL(J)=WL(J)+NV(J)*T1:GOTO2320
2310 WH(J)=WH(J)+NV(J)*T1:WL(J)=WL(J)+NV(J)*T2
2320 NEXTJ:T=T*T/S6:FORJ=K0TONN:IFNV(J)=K0THEN2340
2330 T1=NV(J):SD(J)=SD(J)+T1*T1*T
2340 NEXTJ
2350 NEXTI
2360 NEXTIT:IF(PRAND64)=K0THEN2400
2370 PRINT:PRINT"*WORST CASE NODE VOLTAGES*":PRINT
2380 PRINT"NODE NO.","NOMINAL CASE","W.C. MAX","W.C. MIN":PRINT:FORJ=K0T
ONN
2390 PRINTJ+K1,W(J),WH(J),WL(J):NEXT:PRINT:PRINT
2400 IF(PRAND128)=K0THENPRINT:PRINT:GOTO110
2410 PRINT:PRINT"*STANDARD DEVIATION OF NODE VOLTAGES*":PRINT
2420 PRINT"NODE NO.","STD. DEV.":PRINT:FORI=K0TONN
2430 PRINTI+K1,SQR(SD(I)):NEXT:PRINT:PRINT:GOTO110
2450 PRINT:PRINT"*NODE VOLTAGES*":PRINT:PRINT"NODES","VOLTAGES":PRINT:LA
=KM
2460 K=LA+K1:LA=LA+K4:IFLA>NNTHENLA=NN
2470 PRINTK+K1"-"LA+K1,:FORJ=KTOLA:PRINTNV(J),:NEXT:PRINT:IFNN>LATHEN246
0
2480 PRINT:PRINT:RETURN
2500 PRINT:PRINT"*NODAL CONDUCTANCE MATRIX*"
2510 PRINT:PRINT"ROW  COLS.":PRINT:FORI=K0TONN:LA=KM
2520 K=LA+K1:LA=LA+K4:IFLA>NNTHENLA=NN
2530 PRINT:PRINTI+K1" "K+K1"-"LA+K1,:FORJ=KTOLA:PRINTZ(I,J),:NEXTJ:IFNN>
LATHEN2520
2540 PRINT:NEXTI:PRINT:PRINT:PRINT:RETURN
2550 PRINT:PRINT"*EQUIV. CURRENT VECTOR*":PRINT:PRINT"NODE NO.","CURRENT
"
2560 PRINT:FORI=K0TONN:PRINTI+K1,EQ(I):NEXTI:PRINT:RETURN
2600 PRINT:PRINT"*NODAL IMPEDANCE MATRIX*":GOTO2510
2700 PRINT:PRINT"*BRANCH VOLTAGES*"
2710 PRINT:PRINT"BRANCHES","VOLTAGES":LA=KM
2720 K=LA+K1:LA=LA+K4:IFLA>NBTHENLA=NB
2730 PRINT:PRINTK+K1"-"LA+K1,:FORJ=KTOLA:PRINTBV(J),:NEXTJ:IFNB>LATHEN27
20
```

CONTINUED

numbers in parenthesis. The first branch number is the controlling branch. The second is the branch in which the dependent source resides. The next value is the transconductance or beta value: G= defines transconductance and B= defines current gain.

Assumed units are ohm, mho, volt and ampere. The following multipliers are available: K= kilohm, M= megohm, MV= millivolt and MA= milliampere. Numeric values can be in any form acceptable to BASIC's VAL function.

In many cases it's desirable to obtain a "worst case" analysis of a circuit. Circuit values are never absolute and vary in ranges, i.e. ± 5%, ± 10% etc. So the actual value installed in a circuit is unknown unless explicitly measured. A worst case analysis allows components with specified tolerance to vary during evaluation.

To obtain a worst case analysis or standard deviation of node voltages, tolerance values for one or more parameters should be entered. They are entered as minimum/maximum values or a percentage which follows the nominal value, and enclosed in parenthesis. Both types are shown in the sample runs.

By specifying a tolerance on one or more key component values in the circuit, the effect on the possible range can be determined on the desired output.

Numeric values must have some acceptable terminating character. Since the input is into a string and numeric values are extracted using VAL, some means of finding the end of the number is needed to continue the scan of the input line. The first method I used was to compute the number of characters a number used on the input line using LEN (STR$(X)+(X)=Ø). However, a number entered as 4.25E3 causes the expression above to evaluate to LEN ("4250")+(-1) = 4, or two less characters than "4.25E3". This method did not work for exponential numbers. So I decided upon another solution, which allows use of exponential numbers and more flexible input. This final solution uses numeric values immediately followed by some terminating character. The characters are space,/, (,),%,K,M and end-of-input line. They are in K$ and are frequently used in the input string.

Solution and Output

Twelve values are available from the solution. Any combination of output values can be specified by an input line containing the word PRINT, left justified, followed by two-letter abbreviations for each desired value. (See Table 1.) The word ALL following PRINT causes all output to be printed. One variable, PR, is used to store the output information. Each bit of PR is set to a value of one for one specific output type using the OR operator. The bits of PR are tested using the AND operator when different output values are available. Any number of PRINT lines can be used anywhere in the input.

The word RUN, left justified, starts the solution. First a check is made to see if the minimum conditions for a solution have been met. Line 130 checks for a resistance or conductance in every branch. There must be at least one independent source, at least one branch and at least one node. Line 1060 checks for specified output.

After solution and printout is complete, the program returns to the input routine. Several things are now possible. "END" returns to BASIC command level. "NEW" restarts the program from the beginning and resets all circuit values. Since the solution does not alter any input data, additional output can be specified and another RUN made. The print variable, PR, is reset to zero or no output after each solution. Each RUN produces only the output requested for that RUN.

Modify

The word MODIFY, left justified, increments the modify flag, variable MO. Data that changes component values can then be entered. Changing a value to zero effectively removes the component from the circuit. New sources can be entered in any established branch. New branches cannot be entered.

To modify a branch, enter the letter "B" and the branch number. Omit the node group and enter the new values in this branch. Component values that will not be changed can be omitted. To modify a dependent source number, omit the branch group and enter new transconductance or beta values. If beta values are entered, the resistance in the controlling branch should be changed

```
2740 PRINT:PRINT:RETURN
2800 PRINT:PRINT"*ELEMENT VOLTAGES*":GOTO2710
2900 PRINT:PRINT"*BRANCH CURRENTS*":GOTO3010
3000 PRINT:PRINT"*ELEMENT CURRENTS*"
3010 PRINT:PRINT"BRANCHES","CURRENTS"
3020 PRINT:LA=K1
3030 K=LA+K1:LA=LA+K4:IFLA>NBTHENLA=NB
3040 PRINTK+K1"-"LA+K1,:FORJ=KTOLA:PRINTCU(J),:NEXTJ:PRINT:IFNB>LATHEN30
30
3050 PRINT:PRINT:RETURN
3100 PRINT:PRINT"*ELEMENT POWER LOSSES*":PRINT:PRINT"BRANCHES","POWER LO
SSES":GOTO3020
OK
*****SAMPLE RUN OF ALTAIR DC ECAP  3/12/77

*****STARTING WITH THE CIRCUIT IN FIG. 2
RUN
NODES? 2
BRANCHES? 3
DEP SCS? 0

ALTAIR DC ECAP

? R  THE NODES AND BRANCHES HAVE ALREADY BEEN NUMBERED
? R  BRANCH 1 HAS INITIAL NODE 0 (GROUND)
? R  AND FINAL NODE 1
? B1 N(0/1) R=3 E=24
? R  BRANCH 2 HAS INITIAL NODE 2
? R  AND FINAL NODE 1
? B2 N(2/1) R=4
? R  BRANCH 3 HAS INITIAL NODE 2
? R  AND FINAL NODE 0
? B3 N(2/0) R=5
? R  FOR OUTPUT GET NODE VOLTAGES
? R  AND BRANCH CURRENTS
? PRINT NV BC
? R  READY FOR SOLUTION
? RUN

** SOLUTION **
*NODE VOLTAGES*

NODES          VOLTAGES

 1 - 2          18             10

*BRANCH CURRENTS*

BRANCHES       CURRENTS

 1 - 3          2              -2             2


? R  THE CURRENT IN BRANCH 2 IS NEGATIVE OR
? R  OPPOSITE IN DIRECTION TO THAT WHICH
? R  WAS ASSIGNED
? R
? R  NOW MODIFY THE CIRCUIT TO BE FIG. 3
? MODIFY
? R THE VOLTAGE SOURCE IN BRANCH 1 IS NOW 12 VOLTS
? R  THE RESISTANCE IN BRANCH 3 IS NOW 11 OHMS
? B3 R=11
? R  GET THE SAME OUTPUT
? PRINT NV BC
? RUN

 ** MODIFY 1 **
*NODE VOLTAGES*

NODES          VOLTAGES

 1 - 2          10             7.33333

*BRANCH CURRENTS*

BRANCHES       CURRENTS

 1 - 3          .666667        -.666667       .666667
```

```
? R  MODIFY IT AGAIN TO BE FIG. 4
? MODIFY
? B1 R=5 E=-24
? R  THE VOLTAGE SOURCE IN FIG. 4 BRANCH 1
? R  IS OPPOSITE IN POLARITY TO THE
? R  STANDARD BRANCH (FIG. 1)
? B3 E=12 R=6
? R  INCLUDE BRANCH VOLTAGES IN THE OUTPUT
? PRINT BC
? R  FORGOT SOME
? PRINT NV BV
? RUN
** MODIFY 2 **
*NODE VOLTAGES*

NODES          VOLTAGES

 1 - 2         -20            -16.8
*BRANCH VOLTAGES*

BRANCHES       VOLTAGES

 1 - 3          20             3.2          -16.8
BRANCHES       CURRENTS

 1 - 3         -.8             .8           -.8
? R  WANTED RESISTOR CURRENTS TOO
? PRINT EC
? RUN
** MODIFY 2 **
*ELEMENT CURRENTS*

BRANCHES       CURRENTS

 1 - 3         -.8             .8           -.8
? R  NOW ANALYZE THE SINGLE STAGE COMMON EMITTER
? R  AMPLIFIER IN FIG. 5
? R  THE EQUIVALENT CIRCUIT IS SHOWN IN FIG. 6
? R  CAPICATORS HAVE BEEN REPLACED
? R  WITH OPEN CIRCUITS AND THE 2N657
? R  TRANSISTOR WITH IT'S EQUIVALENT.
? R  THERE ARE THREE NODES-SIX BRANCHES-
? R  AND ONE DEPENDENT SOURCE.
? R  HAVE TO START OVER--
? NEW
NODES? 4
BRANCHES? 9
DEP SCS? 3

ALTAIR DC ECAP

? R  THIS TIME INCLUDE TOLERANCE VALUES FOR
? R  WORST CASE ANALYSIS AND STANDARD DEVIATION
? R
? B1 N(0/2) R=2K(7%) E=20(5%)
? B2 N(0/1) R=6K(7%)
? * E=20(19/21)
? B3 N(0/1) R=1000(7%)
? B4 N(1/3) R=350(10%) E=-500MV
? B5 N(3/0) R=500(465/535)
? B6 N(2/3) R=11.1E3(10%)
? D1 B(4/6) B=50(10%)
? PRINT ALL
? RUN
** SOLUTION **
*NODAL CONDUCTANCE MATRIX*

ROW   COLS.
1     1 - 3     4.02381E-03    0              -2.85714E-03

2     1 - 3     .142857        5.9009E-04     -.142947

3     1 - 3     -.145714       -9.00901E-05    .147804
```

CONTINUED

first with the "B" command, if the resistance is changed at all.

Once the word MODIFY is entered, only circuit modifications can be performed. However, they can be performed as many times as necessary. If the ability to add branches to the circuit is necessary, the program must be changed to allow the modify flag, MO, to be reset to zero.

Subroutines are used for printing all output except partial derivatives, worst case and standard deviation, which are computed only if output is requested.

General Program Description

Lines 10 to 110 set up the program. Function FNI minimizes its argument to zero, avoiding a FC error in the DIM statements. Three input statements get maximum nodes, branches and dependent sources to accommodate different memory sizes. To speed execution, variables are used for constants.

Lines 120 to 570 process the input line. "R" as the first character on the line causes the line to be ignored and can be used to insert remarks into the input. Using an asterisk as the first character indicates that this line is a continuation of the previous line and causes a jump to the parameter extraction routine at line 470. The next non-space character of a continuation line must be R, G, E or I. Dependent source input lines cannot be continued.

The processing of an input line is simple. Look for "B" or "D", find its number, check the number's range, find the node or branch group, check the numbers range and then extract parameter values, putting them in the appropriate matrix. Resistance is carried internally as conductance. At line 510 **minimum-maximum resistance** tolerances become **maximum-minimum conductance** tolerances.

The subroutine at line 600 controls extraction of numeric values. It has three entry points at lines 600, 610 and 620 to extract a single value ("B" or "D" numbers), two numbers in parenthesis (node or branch group) or both (nominal and tolerance values).

Subroutine 790 to 890 extracts values, finds their length on the input line and retrieves the multiplier, if present. The value is returned in T1 and the multiplier value in MU.

Lines 900 to 940 handle the PRINT input line. Each two-letter group is compared to allowed commands in P$ by loop variable K. When a command is found, K has the bit number that needs to be set to one. Two raised to the power K has the decimal equivalent needed in BASIC. Using the OR operator to set the bits doesn't change any bits already set. Setting the same bit more than once has no effect either.

Lines 1000 to 1040 are error messages. Error detection is limited and is not recoverable except when an error occurs at the beginning of the input line in a "B" or "D" number.

Lines 1100 to 1130 set up matrix M — the nodal connectivity matrix. It is set to ± 1 or zero, indicating which branches are connected with which nodes. The values indicate: 1 = initial node, -1 = final node and 0 = node not connected to branch. If one node of a branch is ground, no information is entered into matrix M. Instead, it is implied by the absence of one node.

Lines 1200 to 1290 set up matrix Z — the nodal conductance matrix. Z is a square matrix with dimensions max. nodes +1 by max. nodes +1. Its values are the conductance between the nodes indicated by the subscripts. When the subscripts are the same, values on the major diagonal give the conductance from the node to ground.

Lines 1300 to 1380 add transconductance values to matrix Z if there are any dependent current sources in the circuit.

Lines 1400 to 1430 compute the current for each branch in matrix CU. Lines 1440 to 1450 produce matrix EQ=M*CU — an equivalent current vector for each node.

Matrix Z is inverted at lines 1500 to 1690, changing Z to a nodal impedance matrix. There are node resistances in matrix Z and equivalent node currents in matrix EQ. The product NV=Z*EQ is taken at line 1700, setting the NV matrix to node voltages. (Ohm's Law)

Line 1720 takes the difference between initial and final node voltages and gives branch voltages in matrix BV.

Resistor (element) voltages are computed at line 1740. The source voltage is added to the branch voltage. Resistor voltage is a voltage drop, and source voltage is a voltage rise or negative

```
*NODAL IMPEDANCE MATRIX*
ROW   COLS.

 1    1 - 3      823.889        2.85272        18.6852

 2    1 - 3     -3163.61        1977.27        1851.14

 3    1 - 3      810.31         4.01752        26.3149
*NODE VOLTAGES*

NODES          VOLTAGES

 1 - 3          2.79423        11.0728        2.26853
*BRANCH VOLTAGES*

BRANCHES       VOLTAGES

 1 - 4         -11.0728        -2.79423       -2.79423       .525696

 5 - 6          2.26853        8.80427
*ELEMENT VOLTAGES*

BRANCHES       VOLTAGES

 1 - 4          8.9272         17.2058        -2.79423       .0256958

 5 - 6          2.26853        8.80427
*BRANCH CURRENTS*

BRANCHES       CURRENTS

 1 - 4          4.4636E-03     2.86763E-03    -2.79423E-03   2.79423E-03
*ELEMENT CURRENTS*

BRANCHES       CURRENTS

 1 - 4          4.4636E-03     2.86763E-03    -2.79423E-03   7.34167E-05

 5 - 6          4.53706E-03    4.46401E-03
*ELEMENT POWER LOSSES*

BRANCHES       POWER LOSSES

 1 - 4          .0398475       .0493398       7.80771E-03    1.8865E-06

 5 - 6          .0102925       .0393023
*PARTIAL DERIVATIVES*

NODE VOLTAGES WITH RESPECT TO RES. IN BRANCH 1

NODE NO.       PARTIALS       SENSITIVITIES

 1             -6.36671E-06   -1.27334E-04
 2             -4.41288E-03   -.0882576
 3             -8.9663E-06    -1.79326E-04

NODE VOLTAGES WITH RESPECT TO RES. IN BRANCH 2

NODE NO.       PARTIALS       SENSITIVITIES

 1             -3.93768E-04   -.0236261
 2              1.51201E-03    .0907207
 3             -3.87278E-04   -.0232367

NODE VOLTAGES WITH RESPECT TO RES. IN BRANCH 3

NODE NO.       PARTIALS       SENSITIVITIES

 1              2.30213E-03    .0230213
 2             -8.83985E-03   -.0883985
 3              2.26419E-03    .0226419

NODE VOLTAGES WITH RESPECT TO RES. IN BRANCH 4
NODE NO.       PARTIALS       SENSITIVITIES

 1              2.84905E-06    9.97166E-06
 2              2.70973E-04    9.48406E-04
 3             -6.94049E-05   -2.42917E-04
```

```
NODE VOLTAGES WITH RESPECT TO RES. IN BRANCH 5

NODE NO.       PARTIALS       SENSITIVITIES

 1              1.69551E-04    8.47757E-04
 2              .0167975       .0839875
 3              2.38785E-04     1.19392E-03
*EQUIV. CURRENT VECTOR*

NODE NO.       CURRENT

 1              4.7619E-03
 2              .0814286
 3             -.0728571
NODE VOLTAGES WITH RESPECT TO RES. IN BRANCH 6

NODE NO.       PARTIALS       SENSITIVITIES

 1             -1.13134E-06   -1.25579E-04
 2              9.01302E-06    1.00045E-03
 3             -1.59331E-06   -1.76858E-04

NODE VOLTAGES WITH RESPECT TO GM. BRANCH 4 TO BRANCH 6

NODE NO.       PARTIALS       SENSITIVITIES

 1              .406827        5.81182E-04
 2             -3.24105       -4.63007E-03
 3              .57295         8.185E-04

NODE VOLTAGES WITH RESPECT TO VOLTAGE SOURCE IN BRANCH 1

NODE NO.       PARTIALS       SENSITIVITIES

 1              1.42636E-03    2.85272E-04
 2              .988637        .197727
 3              2.00876E-03    4.01752E-04

NODE VOLTAGES WITH RESPECT TO VOLTAGE SOURCE IN BRANCH 2

NODE NO.       PARTIALS       SENSITIVITIES

 1              .137315        .027463
 2             -.527269       -.105454
 3              .135052        .0270103

NODE VOLTAGES WITH RESPECT TO VOLTAGE SOURCE IN BRANCH 4

NODE NO.       PARTIALS       SENSITIVITIES

 1             -.0388067      -1.94033E-04
 2             -3.69089       -.0184545
 3              .945356        4.72678E-03

*WORST CASE NODE VOLTAGES*

NODE NO.       NOMINAL CASE   W.C. MAX       W.C. MIN

 1              2.79423        3.2806         2.31021
 2              11.0728        15.1519        6.97497
 3              2.26853        2.75874        1.78163


*STANDARD DEVIATION OF NODE VOLTAGES*

NODE NO.       STD. DEV.

 1              .0896619
 2              .555564
 3              .0883014


? R  NO MORE TO DO
? END
```

voltage drop. This is reflected in Fig. 1 by the source and resistor voltages that are opposite in polarity. The branch voltage is the sum of the voltage drop in the branch or BV = EV + (-E) and EV = BV + E.

Lines 1750 to 1780 compute branch currents in matrix CU by using Ohm's Law (I-E*G) to find the resistor current, adding the dependent current source and subtracting the independent current source. The independent current source is subtracted rather than added, because it is opposite in direction from the branch current.

By adding the independent current source back in at line 1800, resistor currents are computed in matrix CU.

Line 1810 computes the power lost in each resistor as the product of resistor current and voltage.

Line 1820 returns matrix BV to branch voltages.

Lines 1830 to 2240 compute partial derivatives of node voltages with respect to each circuit parameter. They also compute a sensitivity coefficient representing the change in node voltage for a one percent change in the parameter value.

Worst case node voltages are computed at lines 2240 to 2310. Lines 2240 to 2280 produce tolerance values with maximum tolerances being positive and minimum tolerances negative. A worst case maximum value is computed as the sum of the products of partial derivatives and tolerance values when both terms are positive. A minimum value is the sum when both terms are negative. Assuming that the partial derivatives are nearly constant over the tolerance range, the worst case analysis is within about three percent of actual values. In other words, keeping tolerance values small (10% or less) keeps the worst case analysis within three percent of actual values.

If larger tolerances need to be used or more accurate results are desired, use direct substitution of minimum/maximum values according to the sign of the partial derivative in a MODIFY and RUN again. Be sure to repeat the procedure for each parameter. The analysis used in this program has the advantage of giving results directly from the nominal solution.

CONTINUED

Lines 2320 to 2330 compute a standard deviation of node voltages based on partial derivatives and tolerance values.

The sample solutions that follow were run on an Altair 8800b with 16K memory. I used Altair 8K BASIC (versions 3.2 and 4.0) with some memory to spare.

Processing an input line of branch data requires about three to 10 seconds, depending on the number of values. The complete output in the sample run took about 15 minutes to print on a 110 baud terminal. By using 4.0 BASIC, input can easily be saved for large circuits.

Readers who are unfamiliar with electronic circuit analysis should look for a simple electrical engineering or network analysis text.

For more information on applications, read IBM's 1620 ELECTRONIC CIRCUIT ANALYSIS PROGRAM (ECAP; 1620-EE-02X) USER'S MANUAL (publication number H20-0170-1) or IBM ELECTRONIC CIRCUIT ANALYSIS PROGRAM TECHNIQUES AND APPLICATIONS, Jenren and Lieberman, Prentice-Hall, Inc., Englewood Cliffs, N.J.

Table 1
Available Output

NV	Node voltages
BV	Branch voltages
EV	Element voltages
BC	Branch currents
EC	Element currents
PL	Element power losses
PD	Partial derivatives
WC	Worst case analysis
SD	Standard deviation
CM	Nodal conductance matrix
IM	Nodal impedance matrix
CV	Equivalent current vector
ALL	All the above

Variable list

```
MO - Modify flag
MN - Maximum nodes
MB - Maximum branches
MD - Maximum dependent sources
M(MB,MN) - Connectivity matrix
NI(MB) - Initial node of branch MB
NF(MB) - Final node of branch MB
Z(MN+1,MN+1) - Nodal conductance matrix
               Nodal impedance matrix
Y(MB) - Conductance in branch MB
YL(MB) - Minimum conductance in branch MB
YH(MB) - Maximum conductance in branch MB
CU(MB) - Current in branch MB
A(MB) - Nominal current source in branch MB
AL(MB) - Minimum    "       "     "     "    "
AH(MB) - Maximum    "       "     "     "    "
E(MB) - Nominal voltage source in branch MB
EL(MB) - Minimum    "       "     "     "    "
EH(MB) - Maximum    "       "     "     "    "
D(MB) - Nominal transconductance in branch MB
DL(MB) - Minimum          "           "    "    "
DH(MB) - Maximum          "           "    "    "
EQ(MN) - Equivalent current vector for node MN
IP(MN) - Counters for matrix inversion
IN(MN,1) - Subscripts of matrix Z showing Maximum values
NV(MN) - Node voltage of node MN
BV(MB) - Branch voltage of branch MB
W(MN) - Nominal node voltages
WL(MN) - Worst case minimum node voltages
WH(MN) -   "    "   maximum   "       "
RW(MD) - Residing branch of dependent source
CL(MD) - Controlling branch of dependent current source
SD(MN) - Standard deviation of node voltages squared
D$,K$,C$,P$ - comparison characters for input
KM,KØ,K1,K2,K3,K4,K5,K6,K7,K9,PC,CT,CM,CI,S6 - constants
NN - Number of nodes in this curcuit
NB -    "   "  branches  "      "
NY -    "   "  resistances (conductances) in this curcuit
NE -    "   "  voltage sources in this circuit
NA -    "   "  independent current sources in this circuit
PR - output flags
X$ - input line
L - Length of X$
I - Character pointer in X$, Loop variable
J,K,L,M,N,L1,IT,II,JJ - loop variables
TY - Type of current input line
X - present branch or dependent source number
T - Temporary nominal value
TL -    "      minimum   "
TH -    "      maximum   "
TT - Type of current parameter
NI,NF,T1,T2 Temporary variables
ZZ - Return flag for subroutine 600
MU - Multiplier value
LA - Last value to be printed on this line
PV - current pivot
```

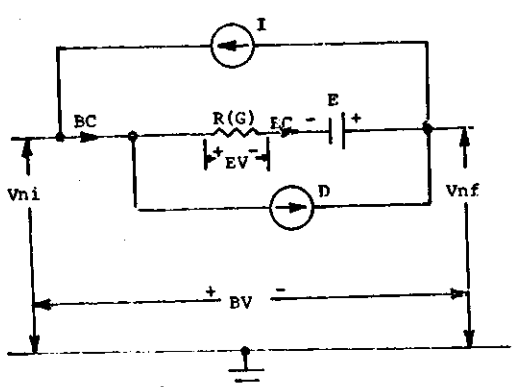


Figure 1
ECAP Standard Branch

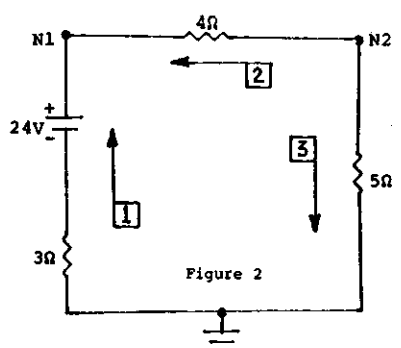


Figure 2

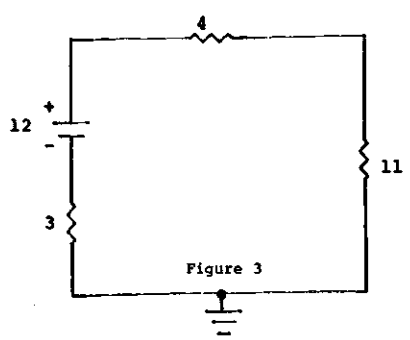


Figure 3

Vni = initial node voltage
Vnf = final node voltage
BV = branch voltage
EV = element(resistor) voltage
BC = branch current
EC = element(resistor)current
D = dependent current source

New Troubleshooting Techniques Defined for Altair 680b

Bruce Fowler

Isolating problems in the Altair 680b is sometimes difficult due to the interaction between hardware and software. The Altair 680b may fail to operate for several reasons. Bit 7 of location F002 might be high and the 6800 would then jump to 0000 instead of outputting to the terminal. Bad memory, particularly in locations 00F3-00FF, can throw the MPU off (the stack pointer is stored in one of these locations). If the 6800 can't access the proper instructions from the PROM monitor, the 680b can go into a wild run condition. Finally, if the ACIA isn't initialized or working, the 680b may not output.

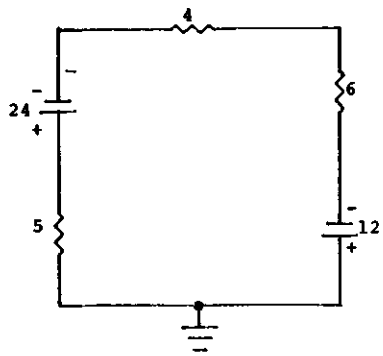


Figure 4

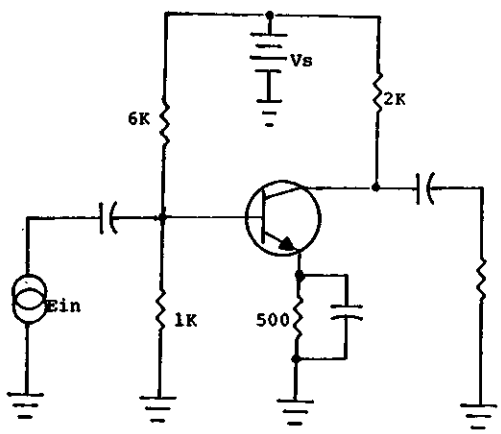


Figure 5

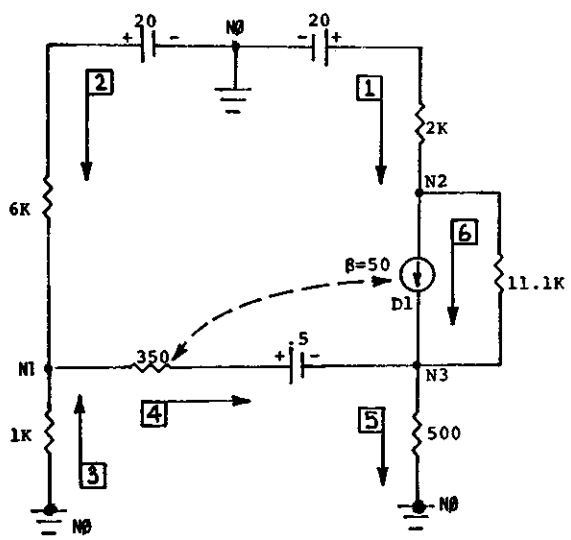


Figure 6

Troubleshooting procedures for these problems are covered in this article, following a brief explanation of the Reset and Monitor operations during Reset. (The Baudot option monitor is not described here. However, those users with the Baudot option should be able to apply this information to their systems.)

Reset

After the Altair 680b is powered up, the 6800 microprocessor can be programmed to start executing at any memory location. This differs significantly from the 8080 microprocessor, in which the 8080 program counter is cleared with reset and execution starts at location 0. The 6800 is reset when RES (reset - pin 40 of the 6800) is low for eight clock periods. A reset operation involves retrieving the contents of memory location FFFE and FFFF and loading this information into the program counter.

The microprocessing unit (MPU) then starts execution at the location indicated by the program counter. Notice that in the Altair 680b, locations FFFE and FFFF are located inside the PROM monitor.

When the 6800 microprocessor is halted and the Reset switch is held, VMA is low, BA and R/W are high (read state), and the address bus contains the restart address FFFE. Address lights A1-A15 are then on, and the A0 light is off when the Reset switch is held. When the Run/Halt switch is moved to Run, the 6800 reads from FFFE and FFFF, storing the byte in FFFE in the upper part of the program counter and the byte in location FFFF in the lower part of the program counter. The 6800 then jumps to the location pointed to by FFFE and FFFF.

Monitor Operation During Reset

When the 6800 starts executing the instructions, the instructions pointed to by locations FFFE and FFFF resets the ACIA and examines location F002. Location F002 is the address of the hardware programmable bits (i.e. terminal/no terminal, Baudot/no Baudot, etc.). The MPU stores this bit pattern from location F002 along with other information into locations 00F3-00FF. If bit 7 (terminal/no terminal) is high, the MPU branches to location 0000 and starts execution. If bit 7 is low, the MPU examines the number of stop bits indicated by location F002 and sends this information to the ACIA. When the ACIA is Ready to output, the MPU then outputs the prompt character.

Troubleshooting

1. Halt the 680b and examine location F002. Data bits 5 and 6 should be high. Data bits 1, 2, 3, 4 and 7 should correspond to the hardware-programmable straps set by the user.
   a. Bit 1 will be low if the no Baudot option is set.
   b. Bit 2 will be low if 2 stop bits are set and high for 1 stop bit.
   c. Bits 3 and 4 are not used and are normally set low.
   d. Bit 7 will be low for terminal option and high for no terminal option.

If the data lights do not match the above list, check for shorts and bad ICs. IC RR should be enabled with a low at pins 1 and 15. The inputs of IC RR correspond to the hardware straps. If IC RR is not enabled IC X is usually at fault.

2. Try the 680b first in no terminal option with the following jump routine.

	Jump	
0000	7E	*jump
1	00	
2	00	

If the 680b fails to execute it properly, examine all the locations in the PROM, particularly locations FFFF, FFFE and FFD8-FFF4. The data in these locations should match the listing given in the System Monitor Manual. If not, do the following.

   a. Check address gating at pin 14 (chip select) for the PROM. Examining addresses from FF00-FFFF should result in a low at pin 14.

* When running, only A0 & 1 should be lit.

CONTINUED

b. Check each address line on the PROM by separately lifting up A0-A7 and monitoring the PROM address pins. If a line fails to go high with the address switch, check for shorts. Shorts between address lines can be found by trying combinations of addresses.

c. Check the power and ground to the PROM. Note that the -9v should be at least -9v at the bottom of the ripple.

d. Check the data from the PROM and the data buffers WW and YY. These buffers should be enabled.

e. Chip select (pin 13 of the 2102 RAMs) should not be low for address FF00-FFFF.

f. The MPU R/W line should be high when the 680b is halted. Check the halt state levels of the 6800 as shown in the following list.

Chip	ID	Pin	Label	Status (MPU Stopped)
MPU	NN	2	HALT	LOW
		3	01 (phase 1 clock)	01
		4	IRQ (interrupt request)	HIGH
		5	VMA (valid memory)	LOW
		6	NMI (non maskable interrupt)	HIGH
		7	BA (bus available)	HIGH
		34	R/W (read/write)	HIGH (pulses LOW during deposit)
		37	02 (phase 2 clock)	02
		40	RES (reset)	HIGH

3. Make sure that working RAM memory exists in location 0000-00FF by depositing into each bit separately. Locations 00F2-00FF are used by the PROM monitor for various software bytes such as breakpoint address, echo/no echo and the stack pointer. Make sure that these locations work and are separate from all other locations by lifting up each address switch one at a time and checking to see if only the corresponding LED lights. If an LED does not light, check for shorts. (For information on deposit problems, see "Troubleshooting the 680b," September CN, pp. 6-7 and 10.)

4. Unfortunately, the ACIA status and data register can't be checked with the 680b in the halt state, because VMA is low when the MPU is halted. If the ACIA is not reset properly or given correct serial information, the 680b may not output the prompt character.

Check the inputs to the ACIA for the halted state as listed in Table 1. Table 2 shows a program which initializes the ACIA and constantly outputs the character whose ASCII code is in location 000B. This program is designed to be run when the 680b is strapped for no terminal so that the inputs which activate the ACIA can be seen. Notice that if 41 is in location 000B, the letter U is output. The ASCII bit pattern for the letter U is 01010101, resulting in an almost symmetrical square wave at pin 6 of the ACIA. Any pulses or levels which do not match Table 1 indicate a problem.

5. The 680b may never output the letter or prompt character if the 6800 microprocessor sees a transmit data register full indication by the ACIA status register. The MPU may be continuously looping, waiting for the ACIA to indicate that it is ready to transmit. The program in Table 3 deposits the status in location 0020 since it cannot be checked when the 680b isn't running.

Run the program for a moment, then stop the 680b and examine location 0020. Data bit 1 should be high, indicating a ready to transmit condition, and data bit 0 should be low if no data has been sent to the 680b. Data bits 2 and 3 should be low. If incorrect, check to see if ACIA pins 23 and 24 are low (which they should be). If bit 1 is low, the ACIA was not reset properly. Check the data lines from the 6800 to the ACIA for shorts or opens (especially DO and D1). Recheck the signals to the ACIA. Highs in bits 4, 5 and 6 indicate transmission errors.

6. If the ACIA will output but not input, run the program in Table 3 and check the status. If the 680b keeps printing periods for proper letters typed to it (like M, N or J), then the ASCII codes for these letters are probably distorted. The program in Table 4 can help isolate this problem.

While running the program, type one character on the terminal. Stop the 680b and examine location 0020. Compare the bit pattern on the data lights to the ASCII code for that character. If they do not match, look for shorts on the data lines. Pin 2 of the ACIA should go low when the character is typed, otherwise check I/O wiring. If pin 2 of the ACIA is always low, the ACIA will think it is always receiving a character. If a Teletype™ is used as the I/O terminal, it will run open. Pin 2 could be low permanently due to shorts, improper placement of components (check R204 if TTY is used), bad I/O components or if the ACIA is not reset with software.

7. Users with 680b-BSM cards who are trying to check the write waveforms can use any program that continuously writes into memory. The program is usually put in the working 1K RAM. Table 5 contains a program that will write whatever is in location 0001 into an address of the 680b-BSM card.

For further repair problems, contact the MITS Repair Department. AN INTRODUCTION TO MICRO-COMPUTERS by Osborne and Associates and M6800 MICROCOMPUTER SYSTEM DESIGN DATA by Motorola are also excellent references on the operation of the 6800 and its family of chips.

Table 1

ACIA pin	Label*	Logic level for 680b in halt state	Logic level for 680b running continuous output program
1	GND	LOW	LOW
2	Rx DATA	HIGH	HIGH
3	RxCIk	symmetrical square wave with period of	
4	TxCIk	568 ms. for 110 baud, and 208 ms. for 300 baud	
6	TxDATA	HIGH	Pulsing
7	IRQ	HIGH	HIGH
8	CS1	HIGH	HIGH
9	CS2	LOW	Pulsing
10	CS1	LOW	Pulsing
11	RS	corresponds to A0	Pulsing
12	VCC	HIGH	HIGH
13	R/W	HIGH	Pulsing
14	E	02	02
23	DCD	LOW	LOW
24	CTS	LOW	LOW

Note: Address F000 or F001 must be examined for CS2 level

Table 2

Location	Op Code	Comments
0000	86	Reset ACIA by
0001	03	sending 03 to
0002	B7	control register
0003	F0	
0004	00	
0005	86	Supply ACIA with
0006	D1	serial format
0007	B7	information
0008	F0	
0009	00	
000A	86	Move to accumulator
000B	41	ASCII code for U
000C	B7	output character
000D	F0	
000E	01	
000F	7E	Loop back and repeat
0010	00	
0011	0A	

Table 3

Location	Op Code	Comments
0000	86	Reset the ACIA
0001	03	by sending 03
0002	B7	to control register
0003	F0	
0004	00	
0005	86	Send ACIA serial
0006	D1	format information
0007	B7	
0008	F0	
0009	00	
000A	B6	Input status
000B	F0	
000C	00	
000D	B7	Deposit status
000E	00	in location 0020
000F	20	
0010	7E	Loop back
0011	00	
0012	0A	

Table 4

Location	Op Code	Comments
0000	86	Reset the ACIA
0001	03	by sending 03
0002	B7	to control register
0003	F0	
0004	00	
0005	86	Send ACIA serial
0006	D1	format information
0007	B7	
0008	F0	
0009	00	
000A	B6	Wait for a character
000B	F0	to be received
000C	00	
000D	47	
000E	24	
000F	FA	
0010	B6	Input character
0011	F0	
0012	01	
0013	B7	Store character
0014	00	in location 0020
0015	20	
0016	7E	Loop back
0017	00	
0018	0A	

Table 5

Location	Op Code	Comments
0000	B6	Move to accumulator
0001	DD	data byte
0002	B7	store in
0003	XX	location XXYY
0004	YY	of 16BSM
0005	7E	
0006	00	
0007	00	

It's a jungle out there,

crawling with publications that deal with every aspect of home and business computing – from spacey games to inventory, accounting and process control. There's a lot you need to know to find your way around.

computer notes™

COMPUTER NOTES offers a monthly survival kit of easy-to-understand features on computer hardware, software and unique applications. CN is published by MITS, Inc., the Altair™ people. Each issue is a combination of articles written by knowledgeable free-lancers and experienced MITS engineers, designers and software specialists.

Whether you're currently a microcomputer "expert" or just taking those first scary steps into the jungle, be sure to take COMPUTER NOTES with you.

You may need it.

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

BERKELEY, CA 94710
1044 University Ave.
(415)-845-5300

SANTA MONICA, CA 90401
820 Broadway
(213)-451-0713

WINDSOR LOCKS, CT 06096
63 South Main Street
(203)-627-0188

DENVER, CO 80211
2839 W. 44th Ave.
(303)-458-5444

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

PARK RIDGE, IL 60068
517 Talcott Road
(312)-823-2388

BURLINGTON, MA 01803
120 Cambridge Street
(617)-272-8770

ANN ARBOR, MI 48104
310 East Washington Street
(313)-995-7616

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

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

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

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

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

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

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

CHARLOTTE, N.C. 28205
1808 E. Independence Blvd.
(704)-334-0242

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

ALBANY, NY 12211
269 Osborne Road
(518)-459-6140

NEW YORK, NY 10018
55 West 39th Street
(212)-221-1404

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

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

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

SPRINGFIELD, VA 22150
6605A Backlick Road
(703)-569-1110

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

altair computer centers

ASDC
3330 Peachtree Road, Suite 343
Atlanta, Georgia 30326

☐ I am a new subscriber.
Please send me one year of **Computer Notes.**
$5.00 ($20.00 overseas) enclosed.

☐ I have moved!
Please note my new address and extend my present subscription.
$5.00 ($20.00 overseas) enclosed.

☐ I have moved.
Please note my new address.
Old mailing label enclosed.
Print or type new address on coupon.

computer notes

mits
2450 Alamo S.E.
Albuquerque, New Mexico 87106

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

NAME:_____

ADDRESS:_____

CITY:_____ STATE:_____ ZIP_____

COMPANY/ORGANIZATION_____

☐ Check Enclosed
☐ Master Charge
☐ BankAmericard/Visa

MC or BAC/Visa #_____

Signature_____

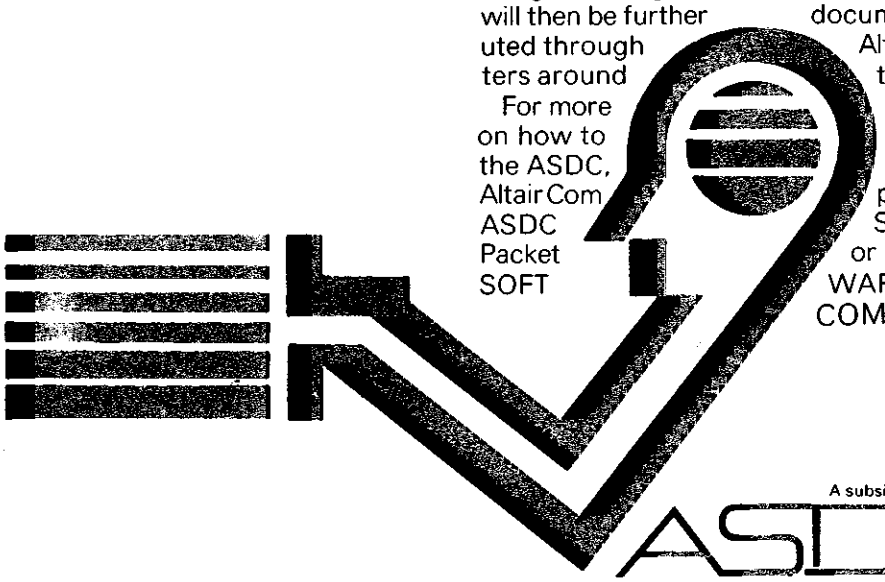
Have you written Software for your Altair™ Computer?

The Altair 8800 computer was the first micro produced for the general public and remains number one in sales, with more than 8,000 mainframes in the field. The wide acceptance of the Altair computer and its rapid adaptation to many diversified applications has truly turned the dream of the affordable computer into a reality.

Yet the machine itself, remarkable as it is, represents only the beginning. The right Software, tailored to meet a user's specific requirements, is a vital part of any computer system. MITS wants to insure that Altair users everywhere have the best applications software available today and in the future. For this reason, a new MITS subsidiary, the ALTAIR SOFTWARE DISTRIBUTION COMPANY, has been formed. Its purpose: to acquire the highest quality software possible and distribute it nationally through Altair Computer Centers.

That's where you come in. The ASDC will pay substantial royalties to the originators of all software accepted into the ASDC library. If you have written business, industrial or commercial use software for the Altair 8800, ASDC wants to hear from you. It is the aim of the ASDC to stimulate and reward creativity in producing useful software that makes those dreams of "computers for everyone" come true. The ASDC will select only software that measures up to its high standards for system design, coding and documentation. The software will then be further documented and distributed through Altair Computer Centers around the country.

For more information on how to submit software to the ASDC, ask your Local Altair Computer Center for an ASDC Software Submittal Packet or contact the ALTAIR SOFTWARE DISTRIBUTION COMPANY.



A subsidiary of MITS

ASDC

ALTAIR SOFTWARE DISTRIBUTION COMPANY

3330 Peachtree Road, Suite 343 Atlanta, Georgia 30326 404-231-2308

see next page for a listing of Altair Computer Centers