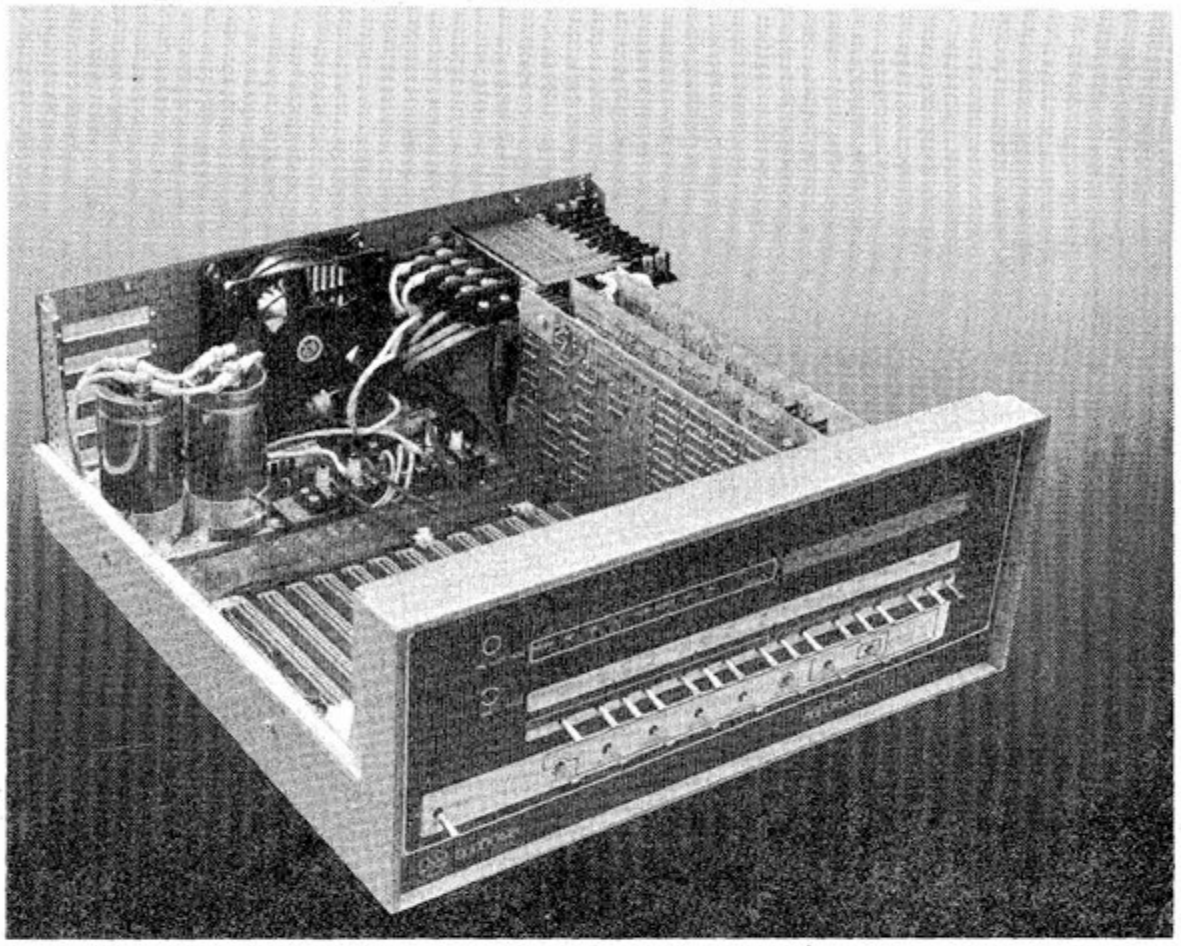
The Mainframe of the Seventies

altair T.M. 8800-b



The Altair 8800b, now in full production, has generated a large number of inquiries, reminiscent of the response to the original Altair 8800. Because of this, we are devoting a large portion of this month's Computer Notes to technical information on the 8800b.

When the original Altair 8800 first went out into the field, it was by far the most advanced design of its kind. By being at the vanguard of the computer movement, MITS has been in a unique position to assimilate feedback and new information from many sources: from hobby customers, from business users, from computer design industries. All of these influences have been percolating at MITS since the first Altair computer came off the line, and the current result is the Altair 8800b. We feel it will be "the mainframe of the 70's."

As anyone associated with microcomputers will tell you, the field is evolving so rapidly that keeping current is almost a day-to-day job. The Altair 8800b incorporates many new electronic and mechanical features including some of the newer integrated circuits for the 8080 family of microprocessors.

The new design features of the Altair 8800b that will be discussed here include: enhanced front panel capabilities, new Display/Control logic, the Front Panel Interface Board, the new CPU Board, added bus lines and heavy duty power supply.

-continued on page 4

altair system to monitor jewelry plant

Arthur Rezac of Applied Computer Research recently announced the development of an Altair system for monitoring production at a large jewelry manufacturing plant in the East. This system includes an Altair 8800a mainframe with 48K of memory, seven teletypes, four CRTs, four floppy disk drives, and a data communication line. In addition, six electronic weighing scales are in turn interfaced to six of the teletypes.

At each stage of production, the jewelry, which is made of gold, will be weighed to determine if there has been any weight loss and if that loss is consistent with the process the jewelry just went through. In addition to this, a worker will enter the production number, where the jewelry came from and what stage it is going to next. This information is transmitted to the Altair to verify that the production number is valid, the job is at the correct production stage, and it is headed in the right direction.

Thus, the manufacturer can control theft and quality of his production as well as validate where any particular job is at any point in time.

At the end of the day, the Altair is connected to a Burroughs 3500 data processing system, and all of the day's production data is transmitted for permanent storage.

Mr. Rezac, who has years of experience in large systems development and sales, reported to Computer Notes that he evaluated every microcomputer on the market and determined that the Altair "had the best price/performance ratio without a doubt".

Mr. Rezac and Applied Computer Research are also at work on a low-cost business system which they plan to market throughout the Northeast. This system, which is also built around the Altair 8800a, includes 32K of memory, dual floppy disk drives, and a DEC writer.

The first unit is scheduled to be installed at a marina on the East Coast, where it will provide inventory control, labor distribution analysis, work-in-process control, payroll, accounts receivable and payable, general ledgers, and a sophisticated sales follow-up system.

According to Rezac, the sales follow-up system is one feature not offered by his nearest competitors, who are nearly $10,000 higher in price. This system is designed to determine whether salesmen have adequately followed up leads and what the results of this work has been as well as analyze the cost effectiveness of advertising and other marketing expenses.

-continued on page 9

COMPUTER NOTES
©mits
JULY 1776
Vol 2 Issue 2

Business Week — Are You Listening?

By David Bunnell

This column started as a tirade against the folks at Business Week, however, before I could complete it, I had to make a complete turn around. The reasons are explained below.

On July 5 or thereabouts, I received a copy of Business Week, and lo and behold there was a microprocessor on the cover. (Or to be more accurate, there was an illustration of a microprocessor sitting on the tongue of some kind of robot.) The headline read, "Smart Machines--the computer-on-a-chip adds decision making and memory to all kinds of products".

Hurriedly I turned to page 38 where the article began. At last, I thought, the real world has discovered the low-cost microcomputer and all the things that go along with it including computer hobbyists, computer clubs, and personal computing conventions.

The article started out well enough with a quote from J. Stanley Webb, a VP at TRW, Inc. According to Mr. Webb, the microprocessor represents the second industrial revolution: "It multiplies man's brain power with the same force that the first industrial revolution multiplied man's muscle power."

The article went on to list some of the new, exciting products that have resulted from the microprocessor, including the smart watch, the smart scale, the smart mobile phone, the smart can making system, and the smart video game.

I found the smart watch particularly attractive. According to the article, you can now buy a watch that will permit you to enter a date such as your wedding anniversary, your kid's birthday, or your appointment with the dentist and the watch will remind you when the day comes (or the day before, if you want). These dates can be set months in advance. Thousands of men could probably save their marriages by purchasing one of these watches.

I read on, thinking that buried somewhere in this was going to be a few paragraphs about microcomputers, computer hobbyists, etc., and how they fit into this big picture. However, there wasn't one word about the microcomputer movement. There was interesting text about a taxi cab meter that could keep track of five fares simultaneously, but not one word about the Altair. Not one word about the SCCS.

Someone from Fairchild was quoted as predicting that computers would someday be in the home, probably as a result of Fairchild's smart video game. That, I thought, was rather amusing.

So I began this column as a tirade against Business Week. The people there must really have their collective heads in the sand. However, before I could complete my hysterics, the next week's copy of Business Week came across my desk and what do you know on page 50 was an article headlined: "Microcomputers Catch on Fast".

This second article included a photograph of Paul Terrell standing at the counter in one of his Byte Shops in California. Behind him was the Altair "Created by Man" poster.

Not only did Business Week accurately detail the phenomenon of the home computer, they said some very nice things about MITS and Altair. According to Business Week, "MITS is the IBM of the home computers".

The results of these kind words reminded me of the early days when the Altair was on the cover of Popular Electronics. For the next few days I received phone calls from people all over the United States and several calls from people in Canada, Europe and such far away places as South Africa. Most of these people were interested in setting up new computer stores, distributorships, or oem type businesses, but some of them called simply to find out more about MITS and the Altair. Other people at MITS and other people associated with MITS found the response to be the same.

I called Lou Fields at the Southern California Computer Society and he informed me that he too had been receiving lots of phone calls. The membership of SCCS is now 6,000 and growing faster than ever.

This publicity and the response it created, reinforces my belief that we are still seeing the mere tip of the iceberg. Practically every day I hear of an Altair application that could conceivably lead to literally thousands of Altair systems. (Two such applications can be found on page one of this issue.)

Now I look forward to publicity in places like Time and CBS. Already one of our VP's has inquired whether or not I can arrange to have him interviewed by Playboy. This particular VP isn't really a publicity hound, he just wants to spend a few days at the mansion.

customer service news

By Gale Schonfeld

I certainly hope you all had a an enjoyable Fourth of July celebration.

This month I'd like to welcome Sandy Koppenheffer to the Customer Service Department. Please call her if you need help with Users Group or "Computer Notes" problems.

VISITING MITS?

If you are planning on visiting us at our new factory, we would very much appreciate it if you would make prior arrangements with our Marketing Department. If you advise us ahead of time, we can be certain to have someone available to answer all your questions.

If, during your visit to Albuquerque, you wish to purchase more equipment for your Altair, we will refer you to the Computer Shack, our new Albuquerque retail store. Pete Conner, owner of Computer Shack, will be happy to help you with decisions on what to purchase for your system. The

-continued on page 5

COMPUTER NOTES

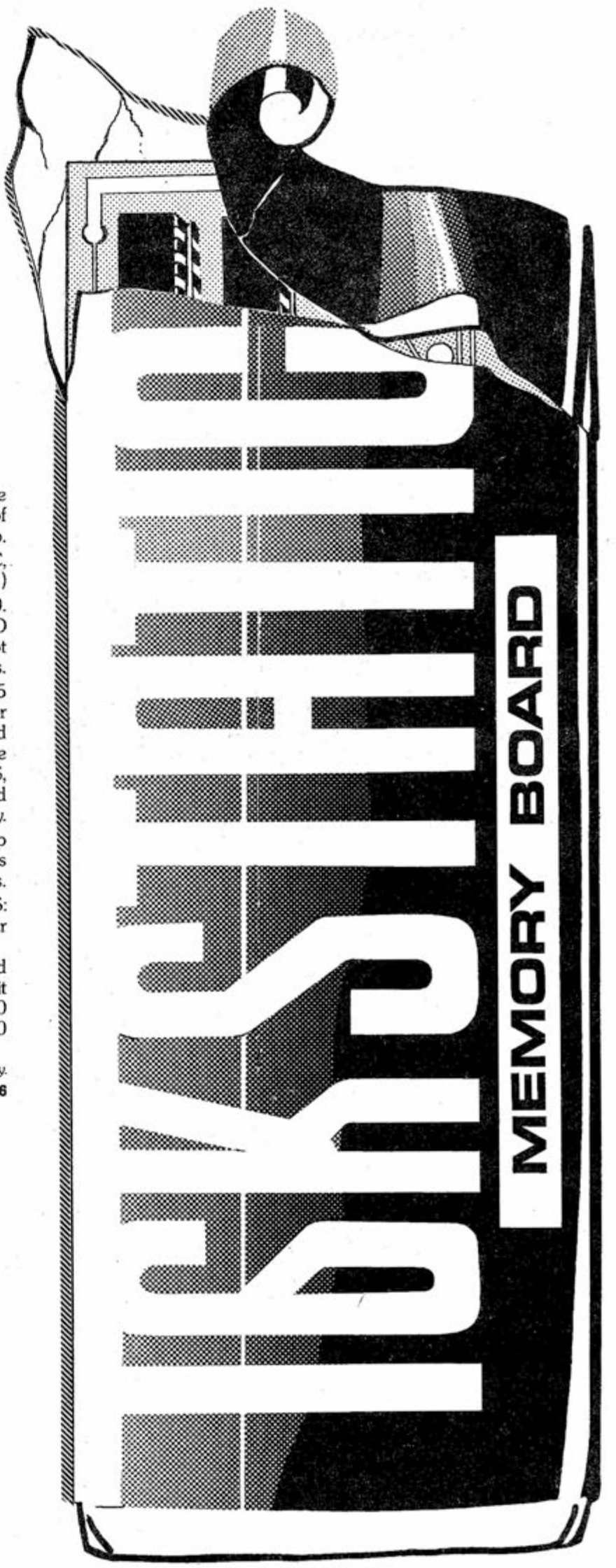
Publisher David Bunnell

Editor Andrea Lewis

Production Tom Antreasian
Al McCahon
Grace Brown

Contributors Ed Roberts
Gale Schonfeld
Paul Allen
Tom Durston
Steve Pollini
Pat Godding
Mark Chamberlin

Something Sweet for your altair™ 680-b



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

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

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

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

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

PRICES:

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

Prices, specifications subject to change. Allow 30-60 days for delivery.

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

mits

MEASURING INTERRUPT ACTIVITY

by NORMAN CROWFOOT

Altair User Devises Interrupt Monitor

"During the course of completing a process-control application utilizing the Altair computer, it became necessary to obtain certain measurements of interrupt activity. I devised and implemented a hardware/software mechanism in which Computer Notes readers may be interested.

"Briefly, a four-chip circuit was constructed on a prototype card and four instructions were added to the general interrupt service routine. This mechanism is further detailed below."

Sincerely,

Norman C. Crowfoot
Lowell Observatory
Flagstaff, AZ 86001

Interrupt Monitor

General Description:

In order to accurately measure certain time-dependent interrupt parameters on a heavily-loaded Altair 8800 system, the following hardware/software additions were made.

The hardware additions consist of four integrated circuit (IC) packages on a prototype board. The function of this circuit is to latch data written by the central processing unit (CPU) to a specific memory address area. This data is further used to gate a 1 MHz pulse stream.

The software additions consist of four additional instructions inserted into a general interrupt routine. These instructions set and reset the hardware data latch.

Together these additions allow the accurate measurement of the following parameters:

- interrupt rate;
- interrupt count;
- microseconds used to process interrupts;
- percentage of total machine cycles used to process interrupts;
- interrupt response latency time

Hardware Description:

Constructed on a prototype card, the four-package circuit is diagrammed in Figure 1. The circuit responds to all memory writes with address bit 15 high; that is, all addresses greater than or equal to X'8000'. The data is latched from data line zero into IC C, a D-type flip-flop. Test point A reflects the status of this latch.

IC D continuously divides the 2 MHz $\emptyset_2$ clock to yield a 1 MHz square wave. This 1 MHz pulse stream is then gated to test point B by the current setting of IC C.

Software Description:

The general interrupt service routine is listed in Figures 2 and 3.

Figure 2 lists the memory locations to which the Vectored Interrupt (VI) board forces the CPU to execute. A typical routine (INT1), first pushes the B/C and D/E register pairs, and then calls the general interrupt service routine (LEVEL). Following the call are three bytes of parameters for LEVEL. First is the new level mask work for the VI board, then two bytes of the address of the specific interrupt handling subroutine for the level.

Figure 3 lists LEVEL, the general interrupt service routine, and BACK, the general return routine. LEVEL completes the saving of registers, pushes the current status of the VI board (CLMASK) and sets the VI board up for the new level. Note that at this time, all previous context is pushed on the stack and that interrupts may now be enabled. The ';;;' notation in the remarks field indicates instructions for which interrupts are masked off.

LEVEL then fakes a call on the specified handler subroutine, by pushing the address of BACK and then pushing the handler address. Finally a return (RET) instruction is executed, causing a branch to the handler subroutine.

The code at location BACK is entered when the handler routine executes a RET instruction, causing the address pushed by LEVEL to be returned to. First BACK restores the VI board and location CLMASK to their previous values. Then all registers are restored and a RET instruction is executed to return control to the interrupted code.

This technique of handling interrupt context changes allows interrupts to be nested to an indefinite level. In fact, interrupts are allowed at all times, except when the VI board is being updated.

In Figure 3, the four additional instructions have been marked. They simply store a '1' at location X'8000' upon entry to LEVEL and set X'8000' to '0' upon exiting BACK.

Measurement Procedure:

Referring to Figure 1, the following procedures are used to measure various timings and counts:

- interrupt rate - attach frequency meter at test point A;
- interrupt count - attach event counter at test point A;
- microseconds used to process interrupts - attach event counter at test point B;
- interrupt response latency time - trigger 'scope on rise of CPU line PINT, measure time to rise of status line SINTA. The lines PINT and SINTA are available on the MITS bus.

Doubtless, there are many other measurements that may be made with this relatively simple setup.

General Suggestions and Comments:

Only half of IC C, the 7474 latch, is used. Another data line may be connected to the unused half for adding further software "hooks" for more complicated measurements. Additionally, more gating logic might be added to combine several latch outputs. For instance, this could be used to determine how often the CPU is interrupted from some specific code or interrupt level.

The Power On Clear (POC), bus line 99, was tied to Reset, line 75, to cause the CPU to automatically start at location zero when power is first applied to the machine. Obviously, some type of involatile memory is required to enable this feature to work properly. And logically, one might wonder why this minor alteration was not included in the original MITS design.

We also found a subtle problem with the Intel 8214 priority interrupt control chip, used on the VI board. While it is described as a latch, it is clocked at 2 MHz and thus appears as a buffer. Too short an interrupt request signal causes the 8214 to develop a level zero interrupt, rather than the one originally requested. The interrupt request line must be held until the CPU grants the interrupt. Beware also of holding the line too long, as multiple interrupts will be generated.

A final point is that the MITS engineers, in their infinite wisdom, have reversed the level numbers on the VI board going into the 8214. This explains the apparent reversal of the VI mask bytes in Figure 2.

This work is an outgrowth of a larger project which has been supported in part by the National Science Foundation grant AST 73-05269 A01 to the Lowell Observatory.

-continued-

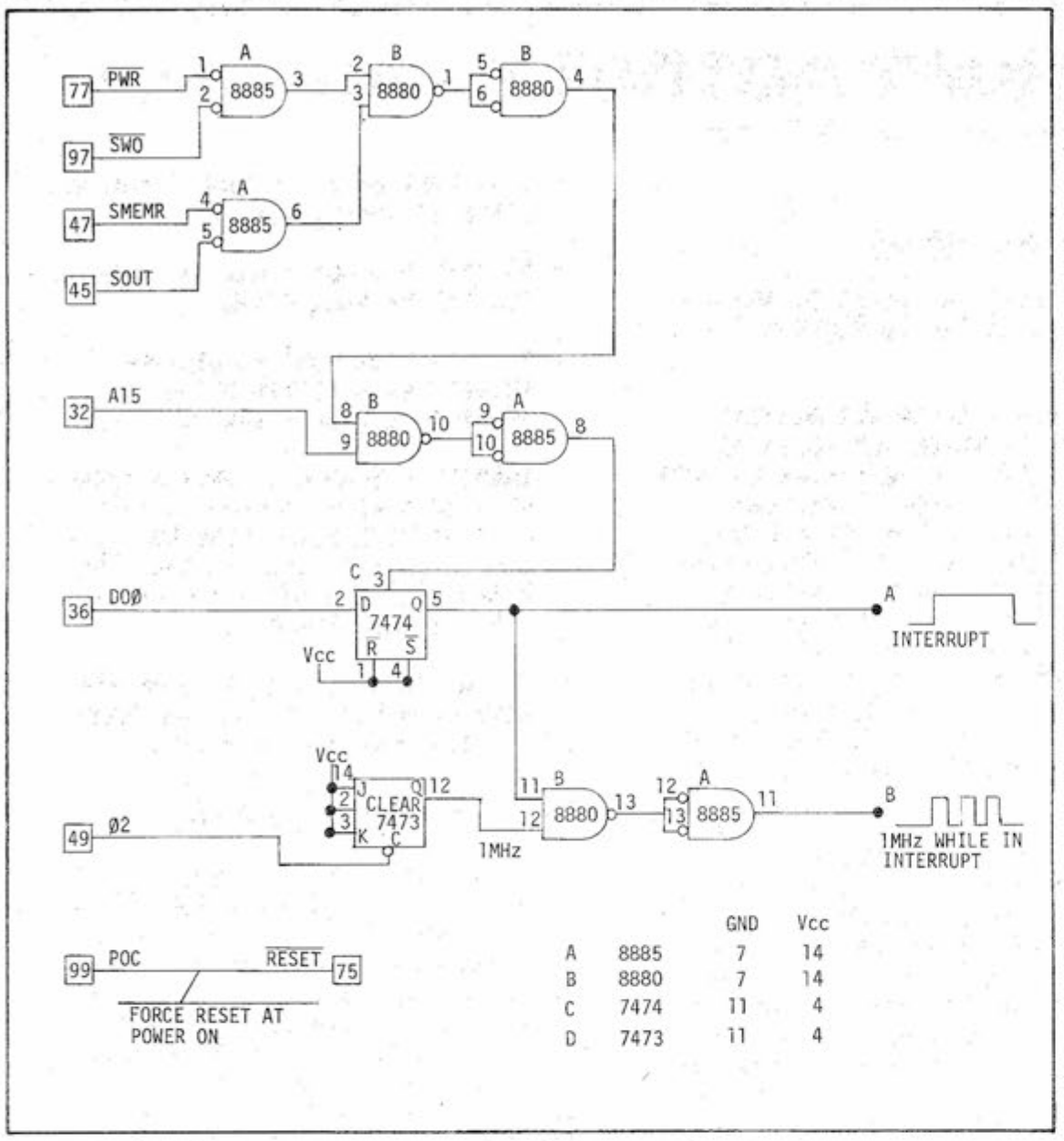


FIGURE 1 - HARDWARE

FIGURE 2: HARDWARE INTERRUPT LEVELS

```
*
*-------INTERRUPT LEVEL 0
        ORG     0               ;;;ALWAYS AT LOC 0
INT0    LXI     SP,STACK        ;;;SETUP STACK POINTER
RESET   EQU     INT0            ;;;RESTART POINT
        JMP     LEVEL0          ;;;CONTINUE WITH STARTUP
*
*-------INTERRUPT LEVEL 1
        ORG     0'10'           ;;;HARDWARE FIXED LOCATION
INT1    PUSH    B               ;;;SAVE B/E REG PAIR
        PUSH    D               ;;;SAVE D/E REG PAIR
        CALL    LEVEL           ;;;CONTINUE WITH COMMON CODE
        OCT     316             ;;;NEW LEVEL MASK WORD
        DEF     LEVEL1          ;;;ADDRESS OF HANDLER SUBROUTINE
*
*-------INTERRUPT LEVEL 2
        ORG     0'20'           ;;;HARDWARE FIXED LOCATION
INT2    PUSH    B               ;;;SAVE B/E REG PAIR
        PUSH    D               ;;;SAVE D/E REG PAIR
        CALL    LEVEL           ;;;CONTINUE WITH COMMON CODE
        OCT     315             ;;;NEW LEVEL MASK WORD
        DEF     LEVEL2          ;;;ADDRESS OF HANDLER SUBROUTINE
*
*-------INTERRUPT LEVEL 3
        ORG     0'30'           ;;;HARDWARE FIXED LOCATION
INT3    PUSH    B               ;;;SAVE B/E REG PAIR
        PUSH    D               ;;;SAVE D/E REG PAIR
        CALL    LEVEL           ;;;CONTINUE WITH COMMON CODE
        OCT     314             ;;;NEW LEVEL MASK WORD
        DEF     LEVEL3          ;;;ADDRESS OF HANDLER SUBROUTINE
*
*-------INTERRUPT LEVEL 4
        ORG     0'40'           ;;;HARDWARE FIXED LOCATION
INT4    PUSH    B               ;;;SAVE B/E REG PAIR
        PUSH    D               ;;;SAVE D/E REG PAIR
        CALL    LEVEL           ;;;CONTINUE WITH COMMON CODE
        OCT     313             ;;;NEW LEVEL MASK WORD
        DEF     LEVEL4          ;;;ADDRESS OF HANDLER SUBROUTINE
*
*-------INTERRUPT LEVEL 5
        ORG     0'50'           ;;;HARDWARE FIXED LOCATION
INT5    PUSH    B               ;;;SAVE B/E REG PAIR
        PUSH    D               ;;;SAVE D/E REG PAIR
        CALL    LEVEL           ;;;CONTINUE WITH COMMON CODE
        OCT     312             ;;;NEW LEVEL MASK WORD
        DEF     LEVEL5          ;;;ADDRESS OF HANDLER SUBROUTINE
*
*-------INTERRUPT LEVEL 6
        ORG     0'60'           ;;;HARDWARE FIXED LOCATION
INT6    PUSH    B               ;;;SAVE B/E REG PAIR
        PUSH    D               ;;;SAVE D/E REG PAIR
        CALL    LEVEL           ;;;CONTINUE WITH COMMON CODE
        OCT     311             ;;;NEW LEVEL MASK WORD
        DEF     LEVEL6          ;;;ADDRESS OF HANDLER SUBROUTINE
*
*-------INTERRUPT LEVEL 7
        ORG     0'70'           ;;;HARDWARE FIXED LOCATION
INT7    PUSH    B               ;;;SAVE B/E REG PAIR
        PUSH    D               ;;;SAVE D/E REG PAIR
        CALL    LEVEL           ;;;CONTINUE WITH COMMON CODE
        OCT     310             ;;;NEW LEVEL MASK WORD
        DEF     LEVEL7          ;;;ADDRESS OF HANDLER SUBROUTINE
*
```

FIGURE 3:

GENERAL INTERRUPT SERVICE ROUTINE, COMPLETE STACK FRAME AND UPDATE CURRENT LEVEL ON VI BOARD

GENERAL FORM OF THE CONTEXT STACK FRAME

```
        ADDRESS OF RETURN HANDLER ('BACK')
        OLD CLMASK
        PSW
        H/L
        D/E
        B/C
        PC
```

```
*
*
LEVEL   XTHL                    ;;;SAVE H/L REG PAIR + GET PARM LIST
        PUSH    PSW             ;;;SAVE PSW/A REG PAIR
*       MVI     A,1             ;;;SIGNAL WE'RE IN AN INTERRUPT
*       STA     IMARK           ;;;TO EXTERNAL COUNTER BOARD
        LDA     CLMASK          ;;;GET CURRENT LEVEL MASK
        PUSH    PSW             ;;;SAVE AS PART OF STACK CONTEXT
        MOV     A,M             ;;;READ NEW LEVEL MASK
        STA     CLMASK          ;;;RETAIN FOR NEXT STACK FRAME
        OUT     VI              ;;;INFORM VI BOARD
        EI                      ;;;OPEN UP FOR MORE INTERRUPTS
*
*-------NOW FAKE SUBROUTINE CALL ON SPECIFIED LEVEL SUBROUTINE
        LXI     B,BACK          ;;;POINT TO RETURN POINT
        PUSH    B               ;FORCE ONTO STACK AS A RETURN POINT
        INX     H               ;BUMP PARM LIST POINTER
        MOV     E,M             ;READ FIRST BYTE OF ADDRESS
        INX     H               ;BUMP TO SECOND BYTE
        MOV     D,M             ;READ SECOND BYTE
        PUSH    D               ;SAVE JUMP ADDRESS
        RET                     ;THEN GO TO THE ROUTINE
*
*-------RETURN FROM INTERRUPT ROUTINE, RESTORE STACK FRAME
BACK    DI                      ;;;KEEP IT QUIET FOR A COUPLE OF CYCLES
        POP     PSW             ;;;PULL BACK OLD LEVEL MASK
        STA     CLMASK          ;;;RETAIN FOR NEXT INTERRUPT
        OUT     VI              ;;;INFORM VI BOARD
        EI                      ;;;OPEN UP NOW, TRICKLY STUFF'S DONE
*       XRA     A               ;;;CLEAR A REG
*       STA     IMARK           ;SIGNAL THAT WE'RE THROUGH
        POP     PSW             ;RESTORE PSW/A REG PAIR
        POP     H               ;RESTORE H/L REG PAIR
        POP     D               ;RESTORE D/E REG PAIR
        POP     B               ;RESTORE B/C REG PAIR
        RET                     ;THEN EXIT THIS INTERRUPT LEVEL
*
```

(additional instructions marked '*')

Customer Service News -continued from page 2

store location is: Computer Shack/3120 San Mateo NE/ Albuquerque, NM/505-883-8282.

BAUDOT HARDWARE OPTION

We have received very little response from our Altair 680b owners regarding the possible marketing of a Baudot level conversion and isolation circuit kit. If you are interested in this kit for your 680b, please drop a post card to my attention. I'll let you know the results of this poll next month.

SOFTWARE LIBRARY

Questions have arisen as to what you receive when you order a program from the software library. For the price listed in the Software Library Update, you will receive a copy of the listing ONLY.

There are four programs in the library which we offer on paper tape. (We do not offer card decks or cassette tapes for these programs.) They are:

#521751	8800 Cross Assembler
#1123751	8800 Simulator
#1203751	8800 Mini-Monitor
#5-24-763	6800 Cross Assembler

The price of the listings for the above four programs is $15.00 each. If you wish paper tape on these programs, the price is also $15.00 each. If you wish both listing and paper tape, the cost is $30.00. Please specify listing or paper tape when ordering the above programs. If you do not specify, you will be shipped a listing.

REMINDERS

To obtain faster delivery on your kit orders, please send money orders or cashiers checks. We do have to allow a 3 week delay for processing a personal check.

If you are among our many foreign customers and would like to expedite your orders by sending in an international money order, please be advised that if the money order is not made payable through a United States bank, then a 3 to 7 week delay occurs while it is processed through the foreign currency exchange clearing house. If your international money orders are made payable through a US bank, then this will definitely expedite delivery time.

Imagine a microcomputer

Imagine a microcomputer with all the design savvy, ruggedness, and sophistication of the best minicomputers.

Imagine a microcomputer supported by dozens of interface, memory, and processor option boards. One that can be interfaced to an indefinite number of peripheral devices including dual floppy discs, CRT's, line printers, cassette recorders, video displays, paper tape readers, teleprinters, plotters, and custom devices.

Imagine a microcomputer supported by extensive software including Extended BASIC, Disk BASIC, DOS and a complete library of business, developmental, and industrial programs.

Imagine a microcomputer that will do everything a mini will do, only at a fraction of the cost.

You are imagining the Altair™ 8800b. The Altair 8800b is here today, and it may very well be the mainframe of the 70's.

The Altair 8800b is a second generation design of the most popular microcomputer in the field, the Altair 8800. Built around the 8800A microprocessor, the Altair 8800b is an open ended machine that is compatible with all Altair 8800 hardware and software. It can be configured to match most any system need.

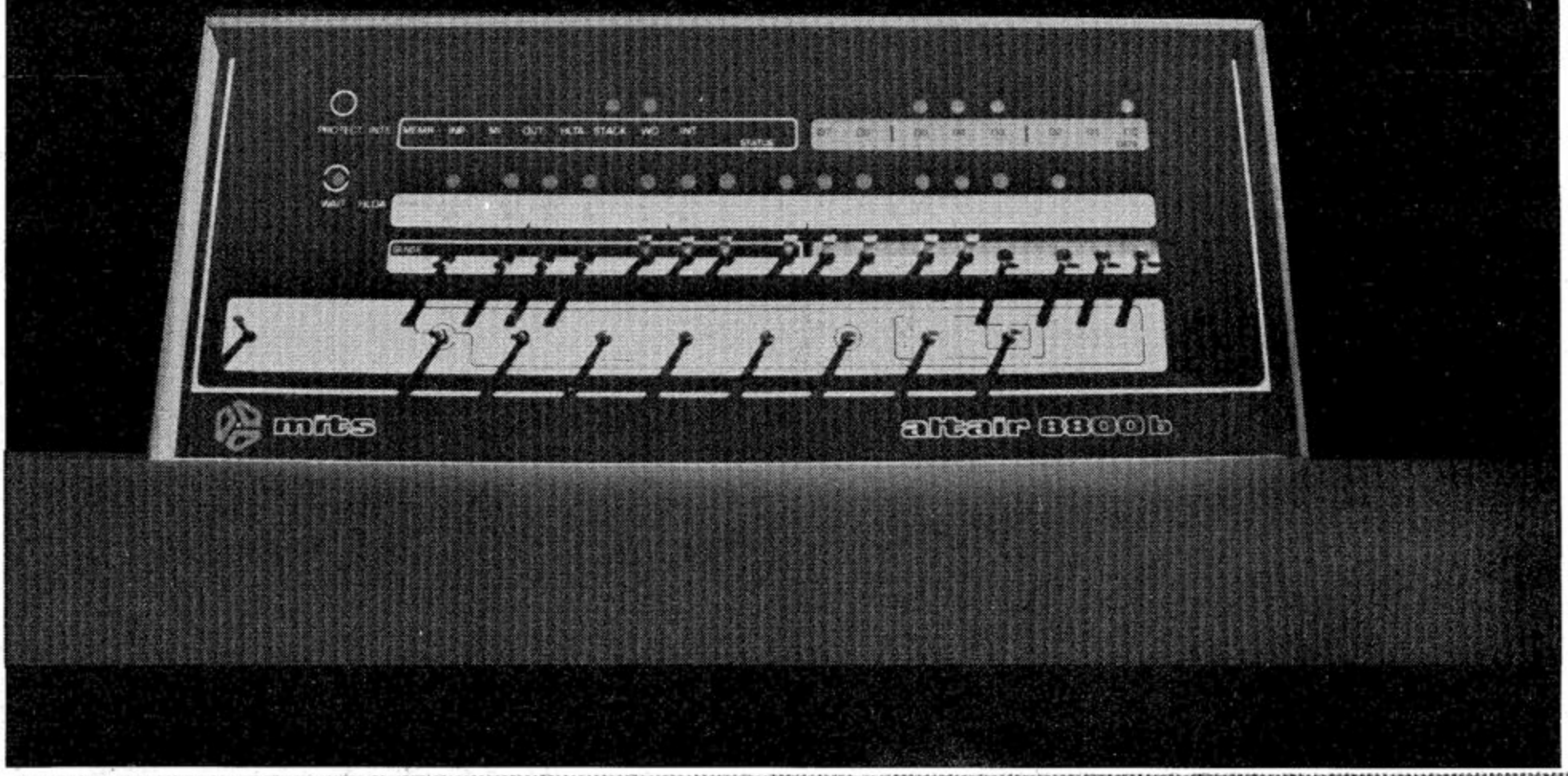
MITS' plug-in compatible boards for the Altair 8800b now include: 4K static memory, 4K dynamic memory, 16K static memory, multi-port serial interface, multi-port parallel interface, audio cassette record interface, vectored interrupt, real time clock, PROM board, multiplexer, A/D convertor, extender card, disc controller, and line printer interface.

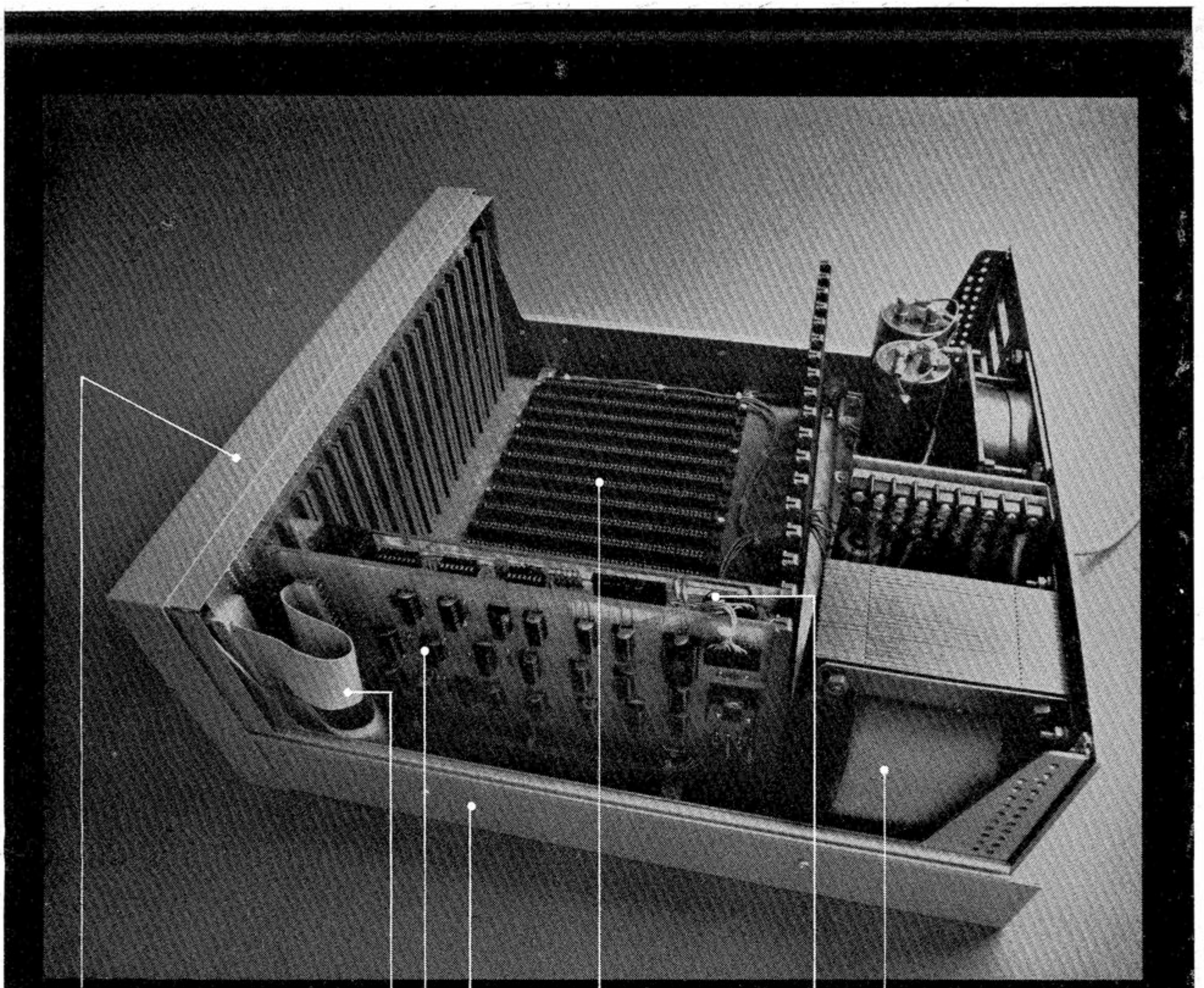
MITS' peripherals for the Altair 8800b include the Altair Floppy Disc, Altair Line Printer, teletypewriters, and the soon-to-be-announced Altair CRT terminal.

Introductory prices for the Altair 8800b are $840 for a kit with complete assembly instructions, and $1100 for an assembled unit. Complete documentation, membership into the Altair Users Club, subscription to "Computer Notes," access to the Altair Software Library, and a copy of Charles J. Sippl's Microcomputer Dictionary are included. BankAmericard or Master Charge accepted for mail order sales. Include $8 for postage and handling.

Shouldn't you know more about the Altair 8800b? Send for our free Altair Information Package, or contact one of our many retail Altair Computer Centers.

MITS, Inc. 1976/2450 Alamo S.E./Albuquerque, New Mexico 87106





Redesigned front panel. Totally synchronous logic design. Same switch and LED arrangement as original Altair 8800. New back-lit Duralith (laminated plastic and mylar, bonded to aluminum) dress panel with multi-color graphics. New longer, flat toggle switches. Five new functions stored on front panel PROM including: DISPLAY ACCUMULATOR (displays contents of accumulator), LOAD ACCUMULATOR (loads contents of the 8 data switches (A7-A0) into accumulator), OUTPUT ACCUMULATOR (Outputs contents of accumulator to I/O device addressed by the upper 8 address switches), INPUT ACCUMULATOR (inputs to the accumulator from the I/O device), and SLOW (causes program execution at a rate of about 5 cycles per second—for program debugging).

Full 18 slot motherboard.

Rugged, commercial grade Optima cabinet.

New front panel interface board buffers all lines to and from 8800b bus.

Two, 34 conductor ribbon cable assemblies. Connects front panel board to front panel interface board. Eliminates need for complicated front panel/bus wiring.

New, heavy duty power supply +8 volts at 18 amps, +18 volts at 2 amps, 18 volts at 2 amps. 110 volt or 220 volt operation (50/60 Hz). Primary tapped for either high or low line operation.

New CPU board with 8080A microprocessor and Intel 8224 clock generator and 8216 bus drivers. Clock pulse widths and phasing as well as frequency are crystal controlled. Compatible with all current Altair 8800 software and hardware.

altair 8800-b

mits

Price, specifications subject to change. Please allow up to 60 days for delivery.

Assembling from the Edit Buffer with Package II

By Mark Chamberlin

The typical program development process usually involves the following steps:

1. Load the Editor.
2. Use the Editor to enter the program into the Edit Buffer.
3. Output the program from the Edit Buffer to an I/O device (e.g. paper tape or cassette tape).
4. Load the Assembler.
5. Assemble the program from the tape.

However, to expedite the program development procedure, Package II allows the user to assemble a program directly from the Edit Buffer. This saves the time spent outputting the program from the Edit Buffer to an I/O device and reading it back into the Assembler.

The steps outlined below constitute a general procedure for creating a program file using the Editor and then using the Assembler to assemble the program directly from the Edit Buffer.

Step 1:

Load the Monitor.

Step 2:

Load the Editor.

Step 3:

Type E to return to the Monitor.

Step 4:

Use the Monitor's DEP command to modify the contents of locations 5124-5125Q and 5530-5531Q. These locations contain the starting and ending addresses of the Edit Buffer, respectively. This step is necessary because the default location of the Edit Buffer is directly above the Editor, and Version 2 of the Assembler (AM2) loads directly above the Editor. In the sample program given here ("ASC"), the Edit Buffer has been moved to start at 12K and end at 16K-1. Note that 12K is 30000Q which is 0011000000000000 in binary.

Split into 8 bit bytes, this address becomes:

00110000 00000000

high order byte — low order byte

Converting the bytes to octal yields:

00 / 110 / 000 00 / 000 / 000

0 6 0 0 0 0

Thus, the high order byte is 060 (octal) and the low order byte is 000 (octal).

The 8080 must always have addresses stored with the low order byte first and the high order byte stored second. Therefore, the command:

DEP 5124
Ø
6Ø
↑Z (control Z - not echoed)

is used to change the starting address of the Edit Buffer. Similarly, the ending address of the buffer is changed (see sample).

Step 5:

Restart the Editor by typing EDT and enter the program into the Edit Buffer. (See the Package II Manual for details on the use of the Editor.)

Step 6:

When the program has been entered, type E to return to the Monitor.

Step 7:

Type the command OPN FIL,EB,A. This opens the symbolic device "FIL" to the Edit Buffer in ASCII mode.

Step 8:

Load and run Version 2 of the Assembler (AM2).

Step 9:

Type FILE to teli the Assembler to read and assemble the program from the symbolic device FIL. (In this case, the contents of the Edit Buffer.)

Note that the last line of the program is a RUN directive which tells the assembler to execute the code that it assembled.

The sample program "ASC" accepts characters from the Teletype and prints the ASCII value of the character in octal. Control is returned to the Monitor when a $ is typed.

Memory Map

When using the Editor and Version 2 of the Assembler in the fashion.outlined above, it is necessary to plan memory use carefully. Below is the memory map for the above example.

377777Q*	Edit Buffer
30000Q*	User Program Area
24000Q*	Assembler Symbol Table
17041Q	Assembler (AM2)
11553Q	Editor (EDT)
5100Q	Monitor

*User defined addresses

Memory Map
for Program Development

Reentering the Editor

Should it ever be necessary to reenter the Editor to modify the text left in the Edit Buffer from the last edit session, the R parameter should be used.

For example, to modify the sample program "ASC" after returning to the Monitor, the command EDT (R) would be used to restart the Editor.

Do not use the command EDT to reenter the Editor or the contents of the Edit Buffer will be lost.

In other words, use the command EDT to create a new program file, and use the command EDT (R) to modify a program that is already in the Edit Buffer.

See next page for Sample Program

JULY SOFTWARE CONTEST

Development Procedure for Sample

Program "ASC"

```
  Development Procedure for Sample Program "ASC"

  ?OPN ABS,AC
  ?EDT
START INPUT
*E
  ?DEP 5124
Ø
60
  ?DEP 5530
377
77
  ?EDT
START INPUT
*I
        ORG     24ØØØQ          ;Set location counter
OUTCH:  DB      1               ;Use LXI trick to get
                                ;around print space entry point
OUTS:   MVI     A," "           ;Load A with a space
        PUSH    PSW             ;Save char to be output
OUTCH1: IN      Ø               ;TTY ready?
        RLC
        JC      OUTCH1          ;No, try again
        POP     PSW             ;Retrieve char to be output
        OUT     1               ;Yes, send the char
        RET                     ;Return to calling program
GETCAR: IN      Ø               ;Anything typed?
        RCC
        JC      GETCAR          ;No, check again
        IN      1               ;Yes, read the char
        CALL    OUTCH           ;Echo the char
        ANI     177Q            ;Strip the parity bit
        CPI     "S"             ;Should we quit?
        JZ      MON             ;If so, return to monitor
        MOV     L,A             ;Copy char into L
        XRA     A               ;Clear A to clear H
        MOV     H,A
        CALL    OUTS            ;Send out a space
        MVI     D,3             ;Initialize digit counter
        JMP     FIRTWO          ;Print digit containing high
                                ;Order two bits
NXTDIG: DAD     H               ;Shift left 1 bit
FIRTWO: DAD     H
        DAD     H
        MOV     A,H             ;Move octal digit to A
        ANI     7               ;Use low order three only
        ORI     60Q             ;Add in ASCII Ø
        CALL    OUTCH           ;Print out the digit
        DCR     D               ;Decrement the digit counter
        JNZ     NXTDIG          ;More digits to go
        CALL    OUTS            ;Send out space and
        JMP     GETCAR          ;go get next character
        BEG     GETCAR          ;Execution begins at GETCAR
        END     ASC
        RUN     ASC
*E
  ?OPN FIL,EB,A
  ?AM4(S)
ALTAIR LOADING ASSEMBLER - REVISION 3.Ø

*ASM*
FILE
  UNDEFINED SYMBOLS

  SYMBOL TABLE

  $ Ø241ØØ
  OUTCH Ø24ØØØ
  OUTS Ø24ØØ1
  OUTCH1 Ø24ØØ4
  GETCAR Ø24Ø16
  FIRTWO Ø24Ø54
  NXTDIG Ø24Ø53

  A 1Ø1 B 1Ø2 C 1Ø3 D 1Ø4 E 1Ø5 F 1Ø6 G 1Ø7 H 11Ø $
```

correction

"Software Initialization of Parallel and Serial I/O Boards", Computer Notes, June 1976

Note at the end of the I/O article it was stated that an interrupt could not occur in the HALT state if the new I/O boards are used. This is true only on the 88-4PIO. The 88-2SIO works normally in the HALT condition.

By Drew Einhorn

This was a slow month for the Software Contest. There were only eleven entries, but what was lacking in quantity was more than made up for with quality.

This month's first place winner was an easy choice. I will even predict that it will be a strong entry in the best of the year contest. You guessed it - Jim Gerow wins again with his 8800 Assembler written in Altair BASIC.

Second place went to Jim Blackstone for an 8080 Debug Package.

Third place was a difficult choice, since there were two people working independently on similar problems. I finally decided to award a tie between James Hansen and Jim Wiggins (see below).

Because there were only two subroutines entered this month and both were written by Alan Miller, I decided the lack of competition in this category did not justify naming subroutine winners this time.

NOTE: The library is already over-stocked with number guessing games, ASCII to Baudot converters and programs to punch and load Baudot paper tapes. Therefore, beginning in August, we will no longer accept any more library programs in these three categories.

FIRST PLACE MAJOR PROGRAM

#7-8-761
Author: Jim Gerow
Length: 300 lines BASIC
Title: 8800 Assembler
Altair 8800 Assembler written in Altair BASIC.

SECOND PLACE MAJOR PROGRAM

#7-7-761
Author: Jim Blackstone
Length: 635 bytes (hex notation)
Title: 8080 Debug Package
Access and modify memory
Copy memory from one block to another
Dump memory to Teletype printer
Fill memory block
Go To program
Print registers and flags

THIRD PLACE MAJOR PROGRAM TIE

#6-18-761
Author: James B. Hansen
Length: 215 bytes
Title: ASCII to Baudot Translate Routine

#6-22-761
Author: Jim Wiggins
Length: 106 bytes
186 bytes
Title: Tape Load-Octal TLQ
Tape Dump-Octal TDQ
for Baudot Teletypes

#6-25-761
Author: Alan R. Miller
Length: 12 lines BASIC
Title: "ERF"
Evaluates the error function.

#6-25-762
Author: Alan R. Miller
Length: 4 lines BASIC
Title: GAMMA
Evaluates the Gamma Function.

#6-28-761
Author: Jim Salem
Length: 71 bytes
Title: Guess 1
Random number guessing game.

#6-28-762
Author: Jim Salem
Length: 88 bytes
Title: Guess 2
Random number guessing game.

#6-29-761
Author: Alan R. Miller
Length: 7 lines BASIC
Title: "DROOT" Double Precision square root.

#7-6-761
Author: Alan R. Miller
Length: 9 lines BASIC
Title: BASIC Subroutine Newton
Newton's method for finding solution to 8(X) = 0.

Jewelry Plant -continued from page 1

The total cost of this Altair business system is approximately $16,000. If you are interested in hearing more, you can contact Mr. Rezac at the following address:

Applied Computer Research Systems
130 Sun Valley Road
Toms Rivers, New Jersey

If you have an unusual or interesting application for your Altair, send us a description of your system, with photos if possible, and we'll publish it in C/N for the benefit of other Altair users.
-Editor

SOFTWARE

USERS GROUPS

New Clubs:

Portland Computer Society
Mike Enkelis
503-246-4616

. . . was listed as nameless in our last issue.

Dwight Instrument Company is sponsoring the formation of a new club: The Northern New Jersey Amateur Computer Group. Membership is open to everyone at $5.00 yearly. Meetings will be held on the second Friday of each month beginning at 6:30 p.m., at Fairleigh Dickinson University, Rutherford Campus, Becton Hall. First meeting is September 10, 1976. For more information contact Beth at:

Dwight Instrument Company
201-438-3334
593 New York Avenue
Lyndhurst, NJ 07071

Interested in starting new clubs:

Dan Schless
#6 Marquette Dr.
Florissant, MO 63031
(St. Louis area)

Dr. Mike Allen
University of North Carolina
Charlotte, NC

Allen Grayson
6908 Foxworth Dr.
Charlotte, NC 28211

Altair Users

Kenneth L. Bowen
Box 1711
Nellis AFB, NV 89191

Larry Simon
94 Tilley Road
Sault Ste. Marie, Ontario
Canada P6B 3Y9

Peter Graulich
1157 Concord Dr.
Haddonfield, NJ 08033

Philip A. Cohlin Jr.
PO Box 8
Sebastopol, CA 95472

Richard P. Brennan
601 South Knight
Park Ridge, IL 60068

Steve Grumette
Omega Products
921 N. La Jolla Ave.
Los Angeles, CA 90046

Steve Agnew
804 104th Ave. S.W.
Calgary

Wirt Atmar
PO Box 4691
University Park, NM 88003

Altair BASIC Biorhythm Program

Reprinted from Personal Systems
(The San Diego Computer Society Newsletter)
June, 1976

I wrote this program in MITS 8K BASIC, and it runs on my 12K ALTAIR 8800.

I don't think that anyone will get rich selling a computerized biorhythm chart (at least at the small users' level), but I think it can be used as a means of showing what our computers can do and might serve as a programming example for the novice programmer (also it's possibly good for a laugh from the more experienced "bit-diddlers").

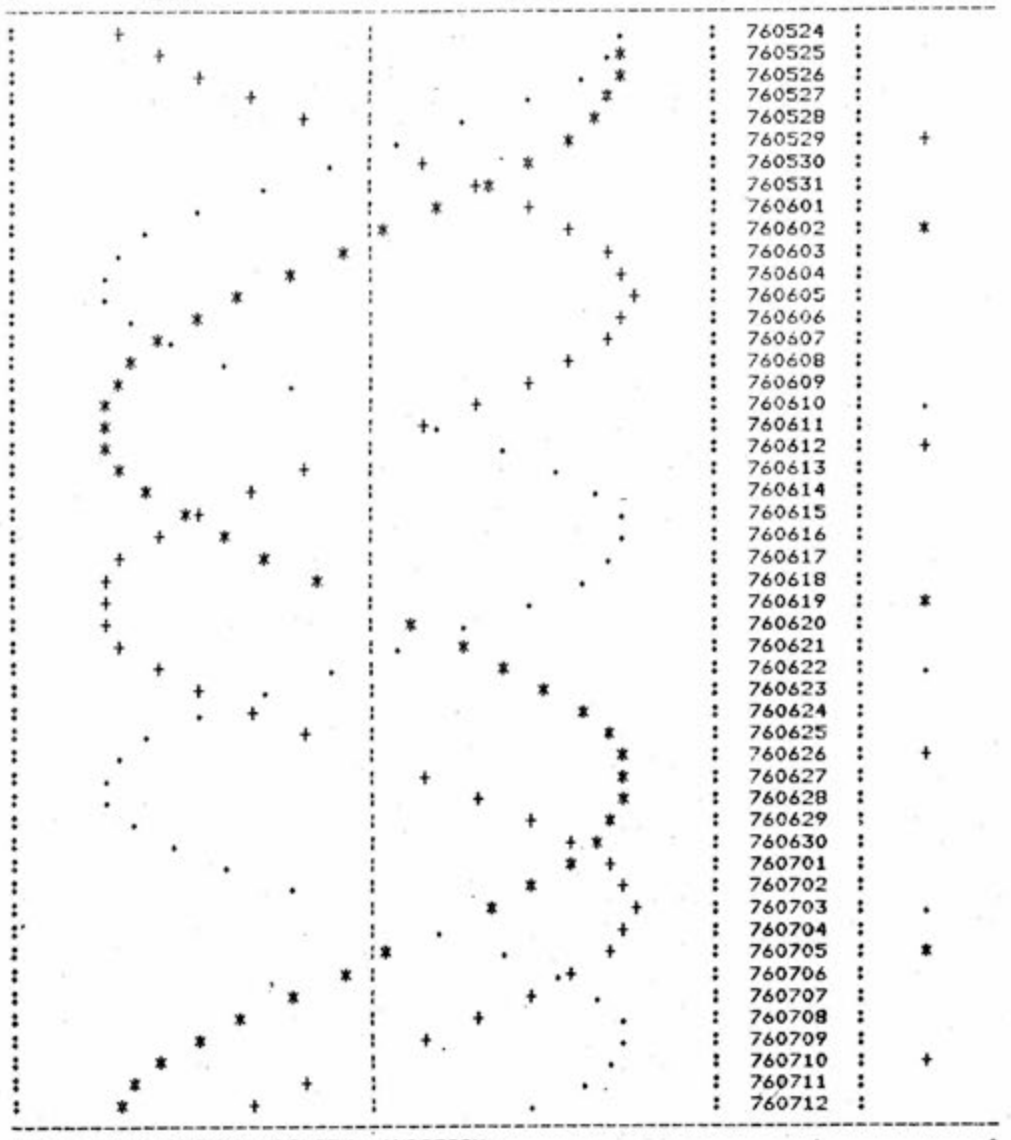
Well, enough rambling! For what it's worth, here's my version of a biorhythm program.

Thank you,
Henry O. Arnold, Jr.

(Editor's note: Great!)

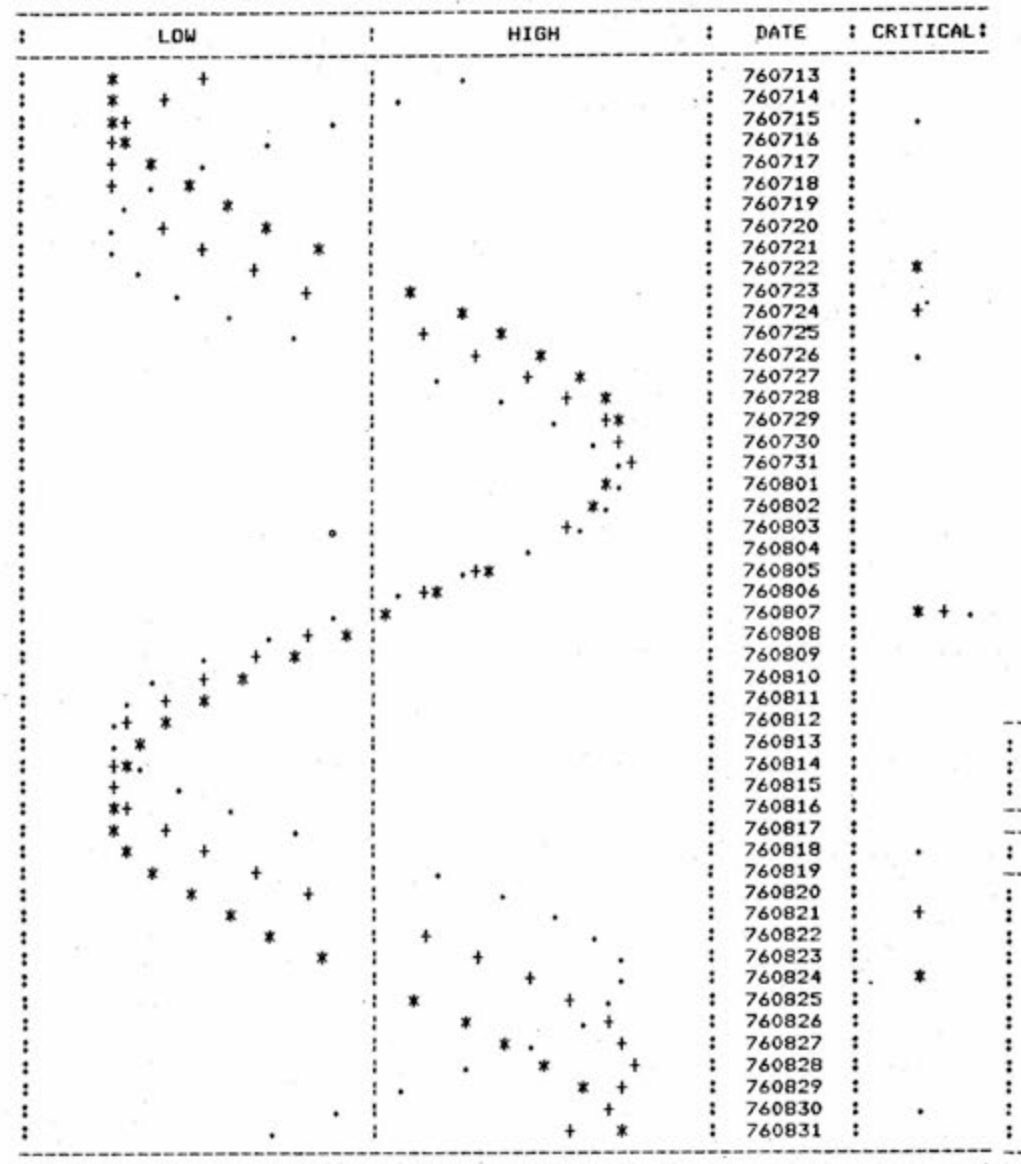
```
LIST

1 LET R1 = (360/33)/57.2958
2 LET R2 = (360/28)/57.2958
3 LET R3 = (360/23)/57.2958
50 DATA 0,31,59,90,120,151,181,212,243,273,304,334
51 DATA 365
60 DIM L$(50)
75 RESTORE
100 PRINT "ENTER BIRTHDATE,CURRENT DATE (YYMMDD)"
125 LET P1 = 0
150 LET J6 = 1
200 INPUT D1,D2
205 LET D9 = D2
206 PRINT "ENTER DURATION"
207 INPUT J5
210 PRINT "ENTER NAME OF SUBJECT"
220 INPUT A$
230 GOSUB 12000
300 IF D1 > D2 THEN PRINT "INVALID DATES":GOTO 200
400 LET X1 = D1
500 GOSUB 1000
550 LET Y1 = X2:LET M1 = X3:LET D1 = X4
600 LET X1 = D2
625 GOSUB 1000
650 LET Y2 = X2:LET M2 = X3:LET D2 = X4
800 GOTO 4000
1000 LET X2 = INT(X1/10000)
1100 LET X3 = INT(X1/100)-(X2*100)
1200 LET X4 = X1-((X3*100)+(X2*10000))
1300 RETURN
4000 LET D4 = (INT((Y2-1)*365.25)-INT((Y1-1)*365.25))
4100 FOR I = 1 TO M1
4200 READ J1
4300 NEXT I
4400 RESTORE
4500 FOR I = 1 TO M2
4600 READ J2
4700 NEXT I
4800 LET J1 = J1+D1
4900 LET J2 = J2+D2
5000 LET L1 = (Y1/4)-(INT(Y1/4))
5100 IF L1 = 0 THEN LET L1 = 1:GOTO 5300
5200 LET L1 = 0
5300 LET L2 = (Y2/4)-(INT(Y2/4))
5400 IF L2 = 0 THEN LET L2 = 1:GOTO 5600
5500 LET L2 = 0
5600 IF M1 > 2 THEN LET J1 = J1+L1
5700 IF M2 > 2 THEN LET J2 = J2+L2
5800 LET D4 = D4+J2-J1
6000 LET D1 = (D4-(INT(D4/33)*33))
6100 LET D2 = (D4-(INT(D4/28)*28))
6200 LET D3 = (D4-(INT(D4/23)*23))
6300 FOR L3 = 1 TO 50
6350 FOR I = 1 TO 50
6360 LET L$(I) = " "
6370 NEXT I
6400 LET X = SIN(R1*D1)
6500 LET Y = SIN(R2*D2)
6600 LET Z = SIN(R3*D3)
6700 LET L$(X*20+25) = "*"
6800 LET L$(Y*20+25) = "+"
6900 LET L$(Z*20+25) = "."
6950 PRINT ":  ";
7000 FOR I = 1 TO 50
7050 LET L$(25) = ":"
7100 PRINT L$(I);
7200 NEXT I
7205 PRINT ": ";
7207 GOSUB 10000:PRINT D5;" :    ";
7210 IF D1 = 0 THEN LET C = 1:PRINT "* ";
7215 IF D1 = 16 THEN LET C = 1:PRINT "* ";
7220 IF D2 = 0 THEN LET C = 1:PRINT "+ ";
7225 IF D2 = 14 THEN LET C = 1:PRINT "+ ";
7230 IF D3 = 0 THEN LET C = 1:PRINT ". ";
7235 IF D3 = 12 THEN LET C = 1:PRINT ". ";
7240 IF C = 1 THEN LET C = 0
7250 PRINT
7300 LET D1 = D1+1
7400 LET D2 = D2+1
7500 LET D3 = D3+1
7600 IF D1 = 33 THEN LET D1 = 0
7700 IF D2 = 28 THEN LET D2 = 0
7800 IF D3 = 23 THEN LET D3 = 0
7900 LET J2 = J2+1
7920 LET J6 = J6+1
7950 IF J5<J6 GOTO 8300
8000 NEXT L3
8050 LET P1 = P1+1
8100 GOSUB 14500
8125 PRINT:PRINT
8150 GOSUB 12000
8200 GOTO 6300
8300 LET P1 = P1+1
8350 GOSUB 14500
8400 FOR I = 1 TO 60:PRINT:NEXT I
8500 GOTO 75
10000 RESTORE
10100 FOR I = 1 TO 13
10150 LET J4 = J3
10200 READ J3
10250 IF J2 > 59 THEN LET J3 = J3+L2
10300 IF J2 <=J3 GOTO 11000
10400 NEXT I
10500 LET Y2 = Y2+1
10510 LET L2 = (Y2/4)-(INT(Y2/4))
10520 IF L2 = 0 THEN LET L2 = 1:GOTO 10600
10530 LET L2 = 0
10600 LET J2 = J2-365
10700 GOTO 10000
11000 LET M2 = I-1
11100 LET D6 = J2-J4
11150 IF J2 = 60 THEN LET D6 = D6+L2
11200 LET D5 = Y2*10000+(M2*100)+D6
11300 RETURN
12000 FOR I = 1 TO 75
12100 PRINT "-";
12200 NEXT I
12250 PRINT
12300 PRINT ": COMPUTERIZED STUDY OF BIORHYTHMIC
12400 GOSUB 13600                              CURVES";
12500 PRINT ": SUBJECT, ";A$;
12600 GOSUB 13600
12700 PRINT ": DATE OF STUDY - ";D9;" - DURATION";J5
          " DAYS";
12800 GOSUB 13600
12810 FOR I = 1 TO 75:PRINT "-";:NEXT I:PRINT
13200 FOR I = 1 TO 75:PRINT "-";:NEXT I
13210 PRINT
13250 PRINT ":          LOW
          HIGH          :";
13260 PRINT "   DATE   : CRITICAL";
13400 PRINT ":"
13500 FOR I = 1 TO 75
13510 PRINT "-";
13520 NEXT I
13530 PRINT
13540 RETURN
13600 LET J = 75-POS(X)
13700 FOR I = 1 TO J-1
13800 PRINT " ";
13900 NEXT I
14000 PRINT ":"
14100 RETURN
14500 FOR I = 1 TO 75:PRINT "-";:NEXT I:PRINT
14600 PRINT ": * = INTELLECTUAL ABILITY, AMBITION.";
14700 GOSUB 13600
14800 PRINT ": + = SENSIBILITY, NERVES, MOOD, CREATIVE ABILITY.";
14900 GOSUB 13600
15000 PRINT ": . = PHYSICAL STRENGTH, ENDURANCE, CONFIDENCE.";
15100 GOSUB 13600
15150 FOR I =  1 TO 75:PRINT "-";:NEXT I:PRINT
15200 PRINT TAB(31);"PAGE ";P1
15400 RETURN
OK
RUN
ENTER BIRTHDATE,CURRENT DATE (YYMMDD)
? 410906,760524
ENTER DURATION
? 112
ENTER NAME OF SUBJECT
? HENRY O. ARNOLD JR.
```

COMPUTERIZED STUDY OF BIORHYTHMIC CURVES
SUBJECT, HENRY O. ARNOLD JR.
DATE OF STUDY - 760524 - DURATION 112 DAYS



* = INTELLECTUAL ABILITY, AMBITION.
\+ = SENSIBILITY, NERVES, MOOD, CREATIVE ABILITY.
. = PHYSICAL STRENGTH, ENDURANCE, CONFIDENCE.

COMPUTERIZED STUDY OF BIORHYTHMIC CURVES
SUBJECT, HENRY O. ARNOLD JR.
DATE OF STUDY - 760524 - DURATION 112 DAYS



* = INTELLECTUAL ABILITY, AMBITION.
\+ = SENSIBILITY, NERVES, MOOD, CREATIVE ABILITY.
. = PHYSICAL STRENGTH, ENDURANCE, CONFIDENCE.

meet Bill Kuhn

One area in which MITS' leadership has always been evident is customer support. In an effort to maintain this leadership and expand our services to you, our customers and potential customers, the new post of Chief Applications Engineer has been created and William E. Kuhn III has been appointed to fill that post.

Bill will be writing regularly in these pages.

APP NOTES

First let me say hello and tell you that I'm happy to have this opportunity to write to you.

Next, for those of you who don't know what an applications engineer is, let me explain.

An applications engineer works out uses for a given product. He helps customers put their equipment to work doing whatever it is that they wish to accomplish.

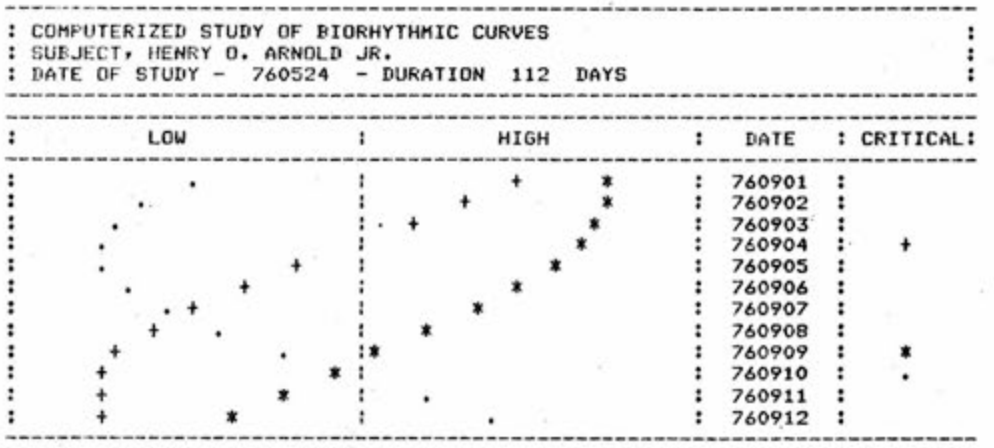
So, I'll ask you: "What would you like your Altair to do?" If there are particular applications you would like to find out about, write to me here at MITS.

Those applications that seem most popular or most interesting I'll try to publish in Computer Notes in the coming months. Also if you have a system up and running doing something that you would like the world to know about, write me. I'll see if I can include it in my column. If you have pictures (black and white) or diagrams that would help explain your system, send them too.

I am anxious to find out what you want your Altairs to do and look forward to helping you in any way I can.

William E. Kuhn III,
Chief Applications Engineer

COMPUTERIZED STUDY OF BIORHYTHMIC CURVES
SUBJECT, HENRY O. ARNOLD JR.
DATE OF STUDY - 760524 - DURATION 112 DAYS



* = INTELLECTUAL ABILITY, AMBITION.
\+ = SENSIBILITY, NERVES, MOOD, CREATIVE ABILITY.
. = PHYSICAL STRENGTH, ENDURANCE, CONFIDENCE.

TROUBLESHOOTING 2-SIO BOARDS

from the MITS Repair Department

by Bruce Fowler

In comparison to MITS' older I/O boards, the 2SIO Board may seem far more complex due to its programing requirement. In the earlier I/O boards, the information containing number of stop bits, type of parity, and number of bits per character was hardwired. Reset was also provided by hardware. In the 2SIO, all of this information is supplied through software. This difference only means that troubleshooting will use both software and hardware.

Special considerations:

If $\overline{\text{DATA CARRY DETECT}}$ and $\overline{\text{CLEAR TO SEND}}$ are not used but are connected through the circuitry to the Molex connectors, they must be set to a high level. This is done by jumpering S1-1 and S1-2 (or S2-1 and S2-2 for second port) of Molex pins to +5v. (Earlier errata sheet on jumping to ground should be ignored.) These jumpers are necessary for RS-232 level interfacing. If these lines are not connected, jumper the D and E pads to ground (D1, D2, E1, E2).

When using a 2SIO to load software, start the bootstrap before starting the loading device. The 2SIO's ACIA must be reset before it will accept any data. When assembling the 2SIO board, IC J is installed only if the 2SIO is to be used for 2 ports. The 2SIO requires the 3.2 version or later of BASIC and must be addressed at Location 20. Switch A-11 must be up for operation of the 2SIO at Location 20. Use the echo routine (page 101 of BASIC Manual) and the bootstrap loader (page 99 of BASIC Manual) with the 2SIO. (The echo routine is given at the end of this article).

Troubleshooting

Check the power supply levels on the voltage regulators and check for solder bridges. An easy way to check the wiring is with an ohmmeter. Use a scope, if available, to check the baud rate. The frequency is 16 times the baud rate. 110 baud should produce a square, symmetrical waveform of roughly .568 millisecond pulse width (1760 hertz).

110 baud .568 milliseconds

300 baud .208 milliseconds

When troubleshooting the 2SIO, use the status register information of the ACIA to indicate the problem. You can single step through the 2SIO echo routine, checking status at appropriate times. (It is necessary to check status since it is possible to echo on a 2SIO which will not respond with BASIC.) The status register indicates the condition of the ACIA at any given moment, and each bit indicates one characteristic. Status register bits appear on the data lights when instruction 333,020 of the echo program has been single stepped. ("HIGH" indicates a lit LED, and "LOW" indicates an unlit LED on the data lights.)

The status register bits are defined as follows.

Bit Ø:

HIGH - Receive data register full. A character has been received from the terminal.

LOW - Receive data register empty. No character received yet from the terminal.

Bit 1:

HIGH - Transmit data register empty. No character is being sent from the CPU to the ACIA.

LOW - Transmit data register full. ACIA has a character stored and is transmitting the character to the terminal.

Bits 2 and 3 are for use with a modem.

Bit 2:

HIGH - No carrier is present. (Pin 23, $\overline{\text{DCD}}$, of ACIA will be HIGH accordingly.) In this state, pin 23 inhibits the receiver section of the ACIA, thus no data can enter the 2SIO.

LOW - Carrier is present. (Pin 23, $\overline{\text{DCD}}$, of ACIA will be LOW accordingly.) In this state, pin 23 activates the ACIA receiver section, and ACIA is free to accept data.

Bit 3:

HIGH - Output device is not ready to receive. In this state, the ACIA's transmitter section is inhibited and the 2SIO cannot output (i.e. transmit) data. (Pin 24 of ACIA will be HIGH.)

LOW - Output device is ready to receive. In this state the ACIA transmitter section is free to output data.

The other bits, which are defined on page 8 of the Theory Manual, are not vital in troubleshooting. Thus, for proper operation, the status register should have all bits LOW except for Bit 1.

Single step the echo routine through to where the status is checked (hit single step 12 times). If you receive proper status, hit a key on the input device. Bit 0 will light up, indicating that the character has been received. If single stepping is continued, the echo routine will output the character.

When the ACIA is neither receiving nor transmitting, pins 2 and 6 of the ACIA must be HIGH. With a Teletype, pin 2 of the ACIA is LOW until the Teletype is ON LINE. If either pin is LOW, the ACIA responds as if data were being transferred.

Bad Status and Areas to Look At

If Bit 3 (or 4) and pin 23 (or 24) of the ACIA are HIGH: This indicates either bad inverters, diodes installed backwards, or S1-1 and S1-2 not tied to +5v for RS-232 levels.

If Bit 0 is HIGH before entering data, then Pin 2 of ACIA is LOW when it should be HIGH, or the ACIA hasn't been reset. The latter could be caused by the data buffers IC A and B not being enabled after hitting single step four times starting at the beginning of the echo routine.

-continued on page 21

HARDWARE

DISK HARDWARE NOTES

By Tom Durston

If you are having difficulties with your 88-DCDD hardware, follow these guidelines for servicing:

A. Controller Boards:

1. On Controller Board #1 be sure the bus strips are soldered on both the top and bottom of the P.C. Board. Do not apply pressure to bus strips after installation.

2. On Controller Board #1 jumper the top end of R16 (VIIB) to the track from pin 7 of IC F2 (on back of card). This ties floating inputs of sector logic high to prevent noise pickup.

3. On Controller Board #1 check the track from Pin 9 of IC H1 where it goes through the board on the plated hole. Some P.C. Cards had shorts to the adjacent track on the back of the card.

4. On Controller Board #1 check jumper wires to be sure there are no shorts to bus strips (insulation on wires melted), and check jumper wires for correct wiring.

5. On both Board 1 and 2 check Stab Connector for shorts on fingers. File at an angle along the length of the Stab Connector and the bevel edge of the card to remove any shorts.

6. Be sure all interconnect cables are wired correctly and the pins are making good contact.

7. Check one shot timing on both boards as follows, using the Disk Test Program that appeared in April '76 Computer Notes, pages 12 and 13.

a) Controller Board #1:

FUNCTION	IC and PIN #	POSITIVE PULSE WIDTH RANGE
Read Clock Mask	IC A1 Pin 13	0.7us to 1.2us
Read Data Window	IC A1 Pin 5	2.6us to 2.9us
Sector Pulse Mask	IC E1 Pin 13	150us to 600us
Index Pulse Window	IC E1 Pin 5	3.3ms to 4.5ms
Read Clear	IC F1 Pin 13	130us to 150us
Index Pulse Verification	IC F1 Pin 5	3.3ms to 4.5ms
Sector True	IC F4 Pin 13	20us to 40us
Write Data Enable	IC F4 Pin 5	250us to 300us

b) Controller Board #2:

FUNCTION	IC and PIN #	POSITIVE PULSE WIDTH RANGE
Repeat Step OK (Status)	IC A1 Pin 13	0.4ms to 0.8ms
Step Inhibit 1 (Status)	IC A1 Pin 5	9.5ms to 11.5ms
Head Settle	IC B1 Pin 13	35ms to 70ms
Step Inhibit 2 (Status)	IC B1 Pin 5	17ms to 30ms
Trim Erase Start Delay	IC B2 Pin 13	180us to 225us
Trim Erase End Delay	IC B2 Pin 5	420us to 520us
Disk Enable Timer	IC B3 Pin 13	1.5us to 4.5us
Disk Power Disable	IC B3 Pin 5	1.5us to 4.5us

c) If the measured time constants are not within the specified tolerance, vary the resistor value for the one shot affected.

d) We have had difficulty using National 74123 ICs for B3 on Board #2. Replace with Signetics or TI ICs if you suspect problems.

8. If you are using 4K Dynamic cards, be sure they are using only one wait state. See May '76 Computer Notes, pages 9 and 10.

9. Check the Power Supply to be sure the negative peaks of the +8V unregulated do not go below +7V.

-continued on page 19

My Computer Just Stares at Me...

By Steve Pollini

If you just bought or built a computer and if you're new to computing, maybe you're wondering why this inscrutable device is giving you the LED stare-down. Why doesn't it DO something? Well, it's probably waiting for you to break the ice. Be assured, however, this is not an uncommon situation. Microcomputers are relatively new devices which carry a stigma of "avant-gardeness," and even many engineers don't quite know how to handle them. So, in this article (and a follow-up article next month) we'll begin to explore just what a microcomputer is, how it works, and what it can do.

An important idea to keep in mind is that a microcomputer computes. It does not think and is not autonomous. For the present, I'll avoid the delicate philosophical argument concerning the definition of the phrase "to think".

What I'm getting at here is that your favorite "wonder-box" of a computer needs to be told what to do. The beauty of a computer system lies in its ability to perform our commands or "programs" very rapidly and very precisely. What we call a "program" is a set of instructions that tells the computer (or more specifically, the microprocessor) what functions we want it to perform. The program that is used by the processor resides in memory and is referred to as software. It is called "software" because it is easily modified as opposed to the actual electronic circuitry (hardware) which is not so easily changed.

THE SYSTEM

Down to business. A microcomputer is called such because it employs the use of a microprocessor, or so-called "computer-on-a-chip". This computer-on-a-chip is an integrated circuit that consists of literally thousands of transistors and other circuit devices. These make up logic circuits to perform the required functions of the processor. In this article the Altair 680b system will be used for examples, but the concepts presented are generally applicable to most microprocessor systems.

The microcomputer. then employs the use of a microprocessor which is the "heart and brains" of the system. This is because it performs all of the arithmetic computations and also controls the rest of the computer system. The rest of the computer system consists of memory

-continued on page 22

Altair 8800b
-continued from page 1

NEW FRONT PANEL CAPABILITIES

Added Front Panel Switch Functions

Five new front panel switch functions have been added to the Altair 8800b computer to expand the front panel capability:

1. SLOW: Permits execution of a program at a rate of approximately 2 machine cycles per second or slower. The normal machine speed is approximately 500,000 machine cycles per second. Useful in debugging programs where it would be too time consuming to single step through the code.

2. DISPLAY ACCUMULATOR: Displays the contents of the CPU accumulator register on the front panel data LEDs.

3. LOAD ACCUMULATOR: Loads the CPU accumulator register with the data present on the lower eight front panel address switches.

4. INPUT ACCUMULATOR: Inputs the data present at an input/output device into the CPU accumulator register. The input/output device is selected on the upper eight front panel address switches.

5. OUTPUT ACCUMULATOR: Outputs the contents of the CPU accumulator register to a selected input/output device. The input/output device is selected on the upper eight front panel address switches.

Dress Panel

A new multi-color dress panel with functionally designed graphics is used in the Altair 8800b. The front surface of the dress panel has a protective sheet of mylar to insure that the graphics are not rubbed or scratched off. The LED indicators are now back-lit through the panel and the toggle switches have 50% longer handles that are flatted (instead of round) for easier use.

Front Panel I/O Capability

The 8800b has I/O channel 255, and effectively channel 254, dedicated to the front panel. As with the Altair 8800, an input from channel 255 (octal 377) will input the contents of the Sense Switches (A15--A8) to the accumulator. The 8800b has the added feature that an output to 255 will display the contents of the accumulator on the data LEDs. In addition, one can configure this I/O channel (by means of patching jumpers) so that all outputs (to any I/O channel number) are shown on the data LEDs and/or all inputs (from any I/O channel number) are shown on the data LEDs.

NEW DISPLAY/CONTROL LOGIC

Electronically the Display/Control Board has been completely redesigned. The logic design is now totally synchronous. The design approach used in the Altair 8800b is to allow the Display/Control logic to assume control of the CPU and jam the instructions necessary to implement the Front Panel functions. For example: To implement an EXAMINE, the Display/Control Board causes the CPU to execute a jump (JMP) to the address selected on the front panel address switches (AØ--A15).

In order for the Display/Control Board logic to jam instructions to the processor (that is, cause the processor to execute a specific series of instructions), two things are necessary:

1. The Display/Control logic must have control of the processor READY line (RDY). (See section entitled "New Bus Lines.")

2. The Display/Control logic must have access to the processor data bus.

If these two conditions are satisfied, the Display/Control logic can cause the processor to execute a series of instructions by successively placing the instructions on the data bus and activating the READY line to cause the processor to execute the required instructions.

The block diagram, Figure 1, summarizes the interface between the Display/Control logic, the CPU, and the Memory and I/O. On the block diagram, note that:

1. The Display/Control logic has control of the READY (RDY) line.

2. The Display/Control logic has access to the data bus through its own data input drivers (FDIØ--FDI7). By activating the Bus Control signal, it can enable its own drivers and disable the standard data input drivers (DIØ--DI7) from the memory and I/O.

The block diagram, Figure 2, shows the Display/Control logic itself (from a functional block viewpoint).

1. The front panel switches are debounced and the examine, deposit, accumulator and I/O function switches (8 switches) are encoded to the upper four address lines (RA7--RA4) of the control PROM.

2. The outputs from the RUN, STOP, SINGLE STEP and SLOW switches go (in pairs) to similar circuits whose outputs (RUN and SS) control the RDY line ([RDY] = [RUN] OR [SS]). (See Figure 3.) Both of these circuits consist basically of flip-flops which, when set, force the outputs (RUN, SS) high and, when reset, force the outputs low.

The RUN/STOP flip-flop is set asynchronously as soon as the input from the run switch (S[RUN]) goes high. This in turn causes RDY to go high and the processor will start to execute. The flip-flop will reset when the input from the stop switch (S[STOP]) goes high and the following stop condition is true:

STOP COND =
(PSYNC) AND (DO5 = SM1) AND (STSTB).

-continued

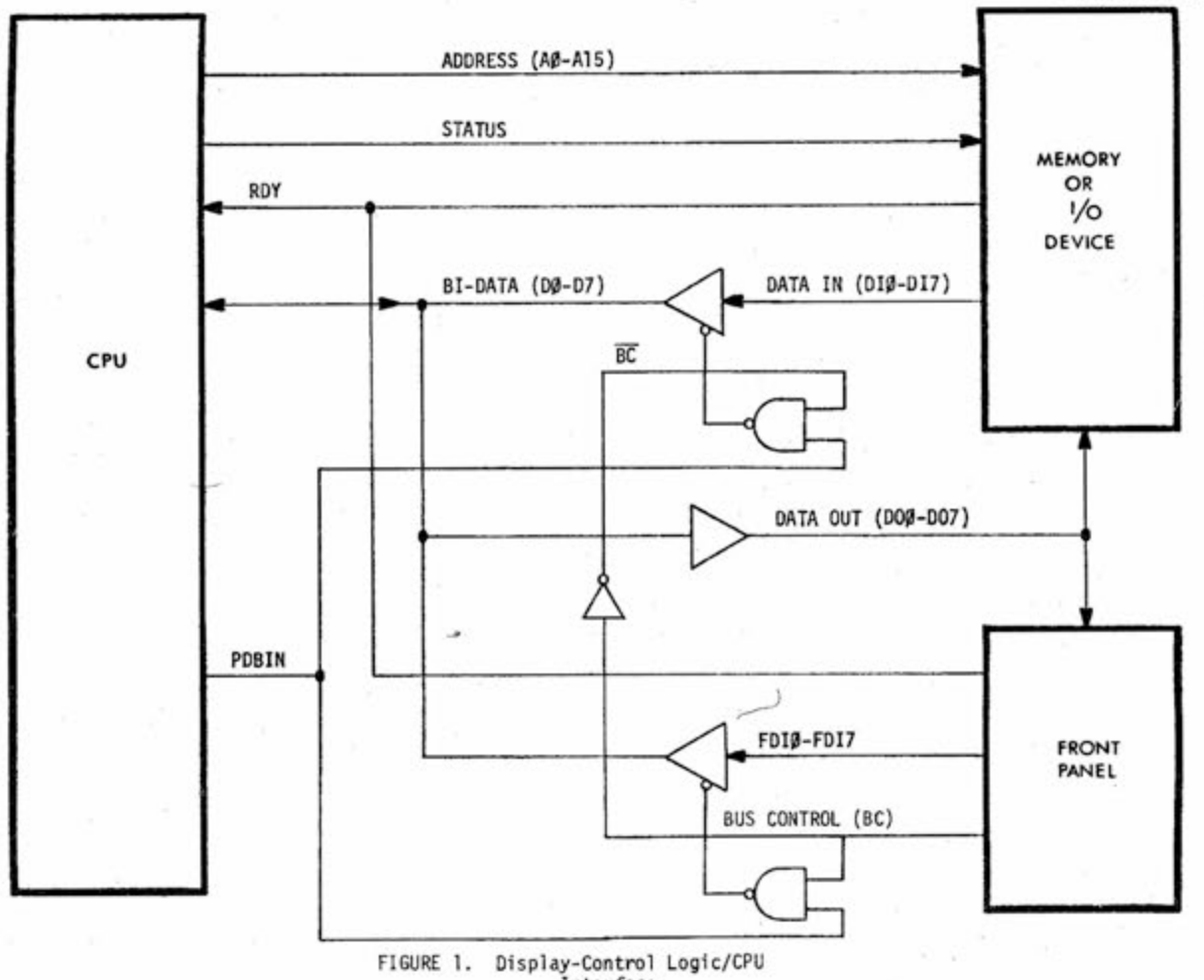


FIGURE 1. Display-Control Logic/CPU Interface

Altair 8800b -continued

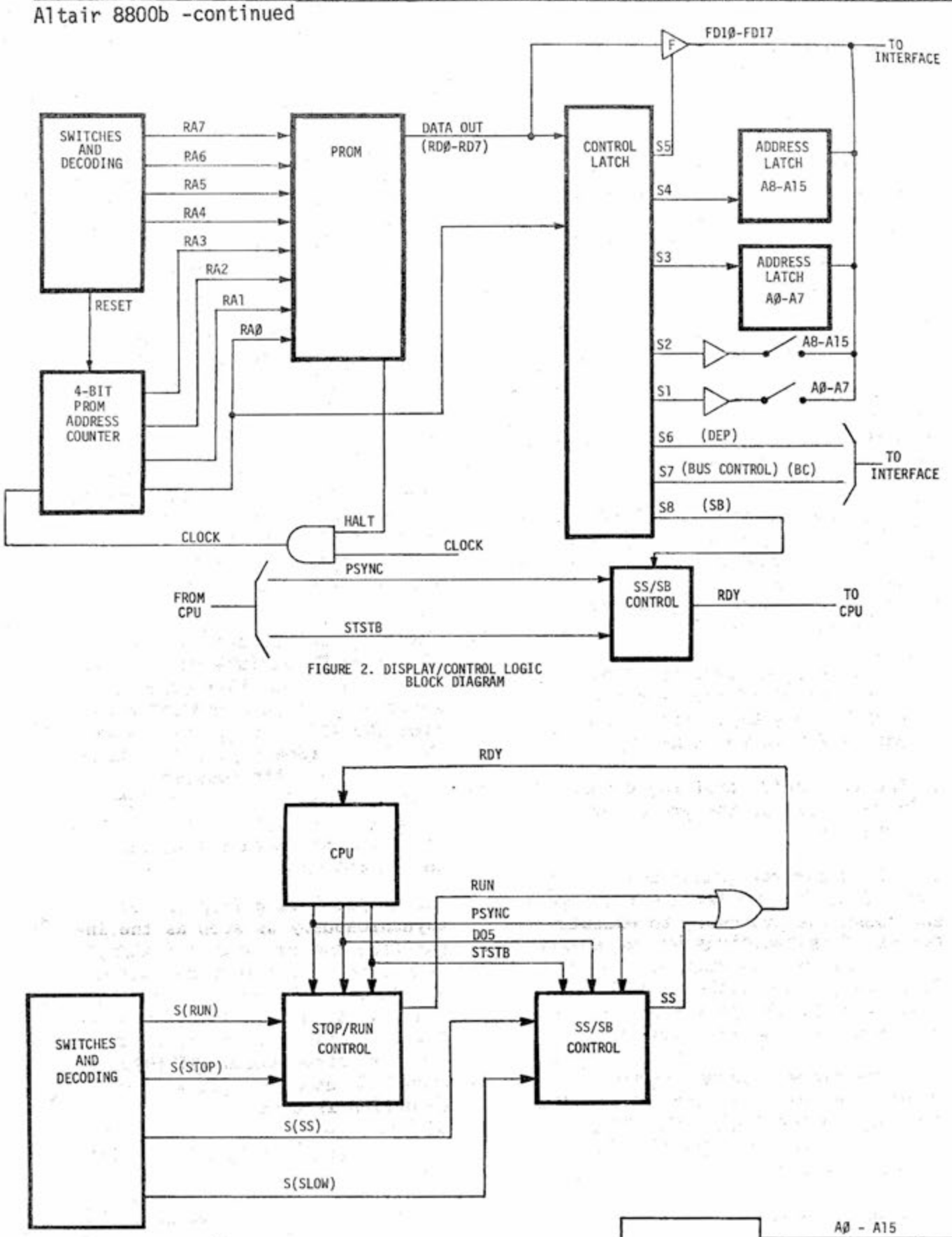


FIGURE 2. DISPLAY/CONTROL LOGIC BLOCK DIAGRAM

FIGURE 3. DISPLAY/CONTROL LOGIC READY LINE CONTROL CIRCUITRY

This insures that the processor will stop during the M1 machine cycle of an instruction cycle.

3. The SS/SB circuit's flip-flop is set by the switch inputs from SINGLE STEP or SLOW, or by the SB input from the control PROM. This circuit, however, has two sets of stop conditions.

   One is associated with the SINGLE STEP/SLOW switch and one with the control PROM that generates the Front Panel functions. The latter will always stop execution after a single machine cycle has been executed. The former can be configured via patching jumpers to stop execution after either a single machine cycle or a complete instruction cycle.

4. The 8 switches which are encoded as addresses to the control PROM represent those functions that are implemented by jamming instructions to the CPU:

   EXAMINE
   EXAMINE NEXT
   DEPOSIT
   DEPOSIT NEXT
   LOAD ACCUMULATOR
   DISPLAY ACCUMULATOR
   INPUT
   OUTPUT

   Pressing one of these switches causes a unique address to be set up on the upper four address lines of the PROM (thus selecting a 16-byte sector within the PROM). At the same time, the 4-bit PROM address counter is cleared and clock pulses are applied to its input. This causes the lower four PROM address lines to begin counting from zero and continue sequentially through the 16 bytes of the selected sector. This will continue until a stop code is encountered in the PROM which will stop the address counter. The instructions stored as data in the PROM may be roughly divided into two categories:

   D/C Logic Control instructions at the even address locations;

   "Processor" instructions at the odd address locations.

   The D/C Logic Control instructions are output from the PROM and stored in the CONTROL LATCH. These instructions configure the D/C logic so the subsequent "processor" instruction can be jammed

-continued on page 19

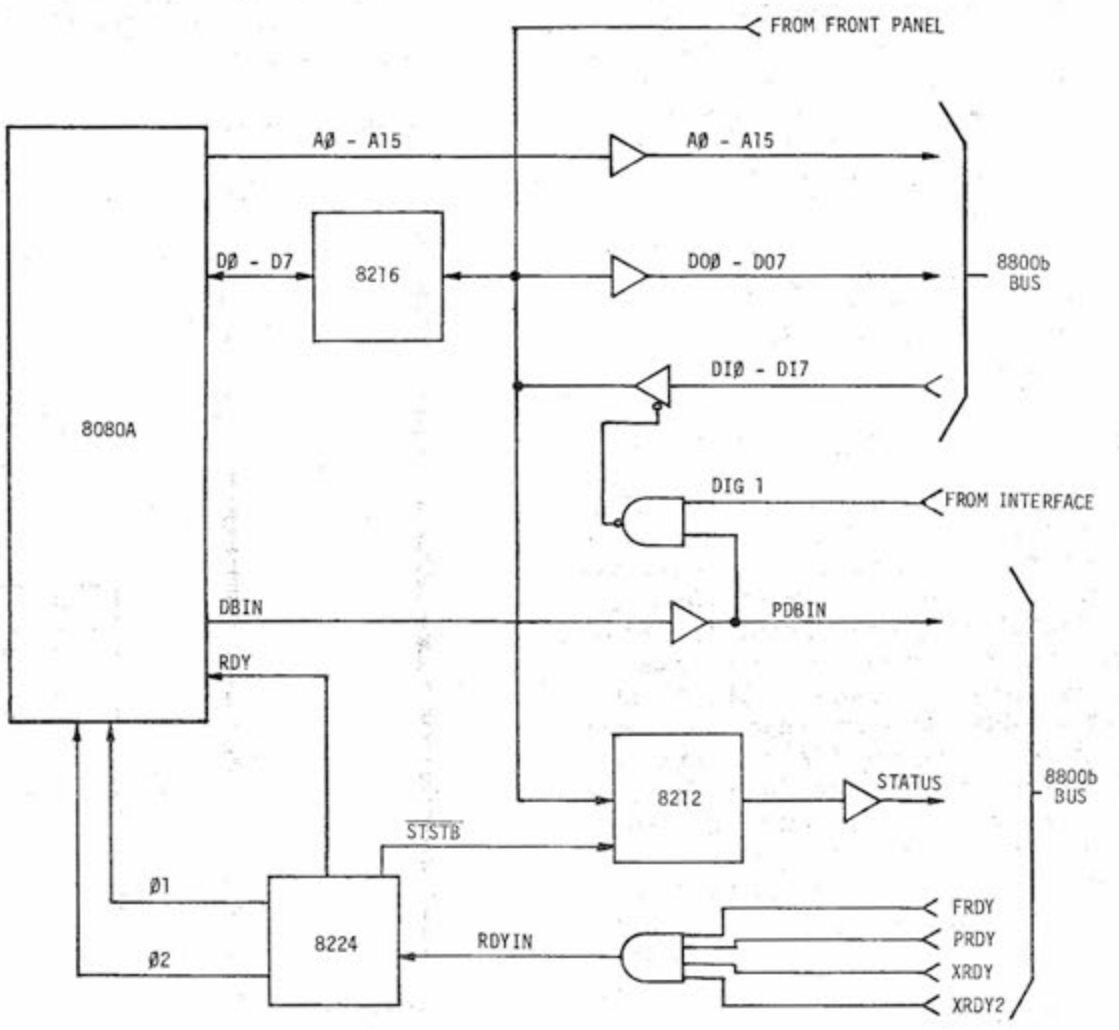


FIGURE 4. CPU BLOCK DIAGRAM

Version 3.4 Altair BASIC

by Paul Allen

Version 3.4 will be released only in the disk version because version 4.0 will be released within a month for all four versions of BASIC. Version 4.0 will allow cassettes of programs to be interchangeable between the different versions of BASIC (8K, Extended, Disk), so it was decided to release 3.4 only in the disk version until 4.0 was ready. Users who have ordered 3.4 will receive 4.0 (except for those who have ordered Disk 3.4). 4.0 in the Extended and Disk versions will have constant compression and line pointers which should speed up program execution in these versions signifigantly.

3.4 and 4.0 will have all the features of 3.3 which was described in detail in the Disk documentation. This means that the Extended and Disk versions will have long lines (255 characters), the INSTR function, CONSOLE, the WIDTH command for setting terminal width, single quote (') remarks, and multiple assembly language subroutines (DEFUSR). The 8K version , Extended version, and Disk version all have octal constants and CLOADing and CSAVEing of matrices on cassette.

NOTE

The Extended version of 4.0 BASIC will require 16K bytes minimum for execution (Extended BASIC 4.0 itself requires 12K).

BASIC version 3.4 has a number of added features as well as a number of bug fixes.

The bug fixes are:

1.) BASIC (all versions) now works properly with the 4PIO board as described in previous Extended BASIC documentation. The correct status bits are now used, and BASIC does an IN from octal channel 23 to clear the output status bit after each character is output. This IN is done no matter what I/O board is used, so it is not recommended that a board other than a 4PIO be used at I/O port 23.

2.) (Extended, Disk versions) The FRE function now returns a positive number if the amount of free memory exceeds 32K bytes.

3.) (Disk version) When a random file is deleted, all the space used by the random file is freed up. Previously, if a random file was extended incrementally, only the first group (8 records) would be freed when the file was deleted.

4.) (Disk version) When simultaneously accessing two files OPENed on different disks, BASIC sometimes forgot which disk it was currently accessing. This has been fixed.

5.) (3.2 8K and larger versions) Typing in a line with a large number of ? marks could case BASIC to be wiped out. Fixed.

6.) (Disk version) The INSTR function did not free up its string temporaries properly, causing spurious "STRING FORMULA TOO COMPLEX" error messages. Fixed.

7.) (Extended 3.2 only) When subtracting double precision numbers of the same exponent of opposite sign, the sign was incorrect, i.e. PRINT 2-3 gave 1 as an answer. Fixed.

8.) (Disk Version 3.3) Use of the line printer caused unpredictable problems. Fixed.

9.) (Disk version 3.3) Use of the RND function with a negative argument caused the random number generator to return the same value over and over again. Fixed.

10.) (Disk version 3.3) Input or Output to sequential data files caused the current terminal position (POS) to be set to zero. Fixed.

11.) (All versions prior to 3.4 not fixed in 4K 3.4) If a direct GOSUB was given to a subroutine which did INPUT from the terminal, the INPUT would wipe out the direct statement, causing unpredictable results when a later RETURN was executed. Under these circumstances, 3.4 will immediately print OK and return to system level if a RETURN is executed back to a direct statement which has been destroyed by an INPUT.

The features and changes listed below are in order of the version for which they are applicable, i.e. features for 4K version first, 8K next, etc.

Additions to 4K and larger versions

Changes for 8K and Larger Versions

Control-C Interrupts INPUT statements

Control-C is now the only way to interrupt an INPUT statement. If a carriage return is typed in response to an INPUT statement, execution of the program will continue at the next statment after the INPUT without changing the values of the variables specified in the INPUT statement.

Rubout and Control-U

The rubout (octal 177) can now be used instead of backarrow (_) or underline to delete characters on an input line. The difference is that rubout prints each character that is deleted and precedes the first character deleted with a backslash (\). If deletion was in progress using rubouts and a new character is typed, a backslash will be echoed and then the new character will be typed.

Example:

```
100 X=\=X\Y=10
```

(In this case two rubouts were typed after 'X=' had been typed.)

Control-U may now be used to delete a line in the same fashion as the at-sign (@). A carriage return is printed and the current line of input is deleted.

Spaces No Longer Allowed in Reserved Words

Spaces may no longer appear inside reserved words such as THEN or AND. The only exception is GOTO which may have embedded spaces. The reason for this is to avoid statements like:

```
100 R=F OR Q
```

Being LISTed as:

```
100 R=FOR Q
```

With the corresponding SYNTAX (SN) error when the line is executed.

Pause (Control-S) and Proceed (Control-Q)

- CONTINUED -

VERSION 3.4 ALTAIR BASIC - CONTINUED -

When executing a program, Control-S may be used to cause program execution to pause so that output may be examined and then resumed with Control-O. This is especially useful when using high speed CRT terminals. After executing a BASIC statement, Control-S will cause BASIC to pause until Control-Q or Control-C is typed. Control-C will cause a BREAK and return to command level. Control-S and Control-Q are not echoed and have no effect when a program is not being executed.

Hexadecimal Constants

Hexadecimal (base sixteen) constants are now available by preceding the number with &H. If the hexadecimal value contains a character which is not A-F or 0-9 a SYNTAX (SN) error will occur. If the hexadecimal value is greater than 16 bits of signifigance (more than four hex digits), an OVERFLOW (OV) error will occur.

Examples:

```
PRINT &HFF
 255

100 LADDR=ADDR AND _&HFF 'mask off low byte
```

Octal contants may optionally be expressed either with a preceding & or with a preceding &O.

Features Available Only in Extended and Larger Versions

Control-C interrupts LINE INPUT

Control-C is now the only way to interrupt a LINE INPUT and return to command level. In version 3.3, a BEL (Control-G) was used to perform this function.

Control-C and Control-O Printing Changed

Control-C and Control-O now print as ^C and ^O when they are typed. Control-U in the Extended version also prints as ^U.

The Tab (Control-I) Character

Tab (Control-I) is used on either input or output to move the terminal carriage or cursor to the next eight column field on the terminal. The tab stops are columns 1,9,17,25,33, etc.

This is especially useful for formatting lines continued with <line feed>:

```
100<Tab>        FOR I=1 TO 10:<line feed>
<tab><tab>      FOR J=1 TO 10:
<tab><tab><tab>         A(I,J)=0:
<tab>   NEXT J,I<carriage return>
```

LISTs as:

```
100     FOR I=1 TO 10:
                FOR J=1 TO 10:
                        A(I,J)=0:
        NEXT J,I
```

> NOTE
>
> <tab> characters always print as the appropriate number of spaces.

Lower Case Input

Lower case alphabetic characters are now accepted by BASIC. Lower case characters are always echoed as lower case, but when lower case is used as part of a direct command or program statement, translation of lower case to upper case is performed if the lower case character is not part of a quoted string literal, REMark statement, or single quote (') remark.

Thus, a line input as:

```
100 print a,b:rem print out the values of a and b
```

Will be LISTed as:

```
100 PRINT A,B:REM print out the values of a and b
```

or:

```
150 if a$="basic" then 200 'test for BASIC command
```

is LISTed as:

```
150 IF A$="basic" THEN 200 'test for BASIC command
```

Brackets now Allowed as Matrix Subscript delimiters

Brackets [,] are now interchangeable with parentheses as delimiters for matrix subscripts. Thus:

```
100 A[I]=0
```

is equivalent to:

```
100 A(I)=0
```

This has been done for compatibility with other BASICs, notably HP BASIC.

CONTinue Possible after Errors

It is now possible to CONTinue after an error in a direct statement. Also, errors no longer cause loss of the current FOR...NEXT context and subroutine (GOSUB...RETURN) context.

EDIT Command Types BEL on Errors

The EDIT command will now type a BEL character (control-G) if it receives a command which it does not recognize (i.e. Y).

Error Trapping

Often it is desirable to trap execution of errors within a BASIC program in order to take action to recover from the error, or to give a better explanation of why the error occurred than a simple error message.

This facility has been added to BASIC through the use of the ON ERROR GOTO, RESUME and ERROR statments, and with the ERR and ERL variables.

Enabling Error Trapping

The ON ERROR GOTO statement is used to specify which line of the BASIC program the error handling subroutine starts. The ON ERROR GOTO statement should be executed before the user expects any errors to occur. Once an ON ERROR GOTO statement has been executed, all errors detected during the execution of the BASIC program will cause BASIC to start execution of the specified error handling routine. If the<line number> specified in the ON ERROR GOTO statement does not exist, an UNDEFINED STATEMENT error will occur.

Syntax of the ON ERROR GOTO statement:

ON ERROR GOTO <line number>

Example:

```
10 ON ERROR GOTO 1000
```

Disabling the Error Routine

IF the user desires to disable the trapping of errors he should place an ON ERROR GOTO 0 statement in his program. This disables trapping of errors, and any error will cause BASIC to print an ERROR message and stop program execution.

- CONTINUED -

VERSION 3.4 ALTAIR BASIC - CONTINUED -

If an ON ERROR GOTO 0 statement appears in error trapping subroutine, it will cause BASIC to stop and print the error message wnich caused the trap. It is recommended that all error trapping subroutines execute an ON ERROR GOTO 0 subroutine if an error is encountered for which they have no recovery action.

> NOTE
>
> If an error occurs during the execution of an error trap routine, the error will immediately be "forced". An error message will be printed for the error detected inside the error trap routine.

The ERR and ERL Variables

When the error handling subroutine is entered, the variable ERR contains the error code for the error. The error codes and their meanings are listed below.

THIS SECTION TO BE ADDED LATER

The ERL variable contains the line number of the line where the error was detected. For instance, if the error occurred on line 1000, ERL will be equal to 1000.

If the statement which caused the error was a direct (immediate mode) statement, the line number will be equal to 65535 decimal.

> NOTE
>
> Neither ERL nor ERR may appear to the left of the = sign in a LET or assignment statement.

The RESUME statement

The RESUME statement is used to continue execution of the BASIC program after the error recovery procedure has been performed. The user has three options. The user may RESUME execution at the statement that caused the error, at the statement after the one that caused the error, or the user may RESUME execution on a different line than caused the error.

To RESUME execution at the statement which caused the error, the user should use:

```
RESUME
```

or

```
RESUME 0
```

To RESUME execution at the statement immediately after the one which caused the error, the user should use:

```
RESUME NEXT
```

To RESUME execution at a line dfferent than the one where the error occurred, use:

```
RESUME <line number>
```

Where <line number> is not equal to zero.

Error Routine Example

The following example shows how a simple error trapping subroutine operates.

```
100 ON ERROR GOTO 500
200 INPUT "WHAT ARE THE NUMBERS TO DIVIDE";X,Y
210 Z=X/Y
220 PRINT "QUOTIENT IS";Z
230 GOTO 200
500 IF ERR=4 AND ERL=210 THEN 520
510 ON ERROR GOTO 0
520 PRINT "YOU CANT HAVE A DIVISOR OF ZERO!"
530 RESUME 200
```

The ERROR statement

In order to force an error to occur in a program, an ERROR statement has been provided. The primary use of the error statement is to allow the user to define his own error codes which can then conveniently be handled by a centralized error trap routine as described above. The format of the ERROR statement is:

```
ERROR <numeric formula>
```

Example:

```
ERROR 5
SYNTAX ERROR
```

When defining his own error codes, the user should pick values which are greater than the ones used by BASIC. Since further error messages may be added to BASIC in the future, it is recommended that error codes which are allocated from the last possible value (255) down to lower codes be used. If the <numeric formula> used in an ERROR statement is less than zero or greater than 255 decimal, a FUNCTION CALL error will occur.

If an attempt is made to print out an error message for an error which is greater than the highest defined system error, an FC error will be printed instead.

Of course, the ERROR statement may also be used to force SYNTAX or other standard BASIC errors.

Assigning String Substrings - The MID$ statement

A new statement has been added that makes it much easier to change a single character or sequence of characters inside a string without altering the other characters in the string. As an added benefit, using such a statement does not incur the numerous string allocations if concatenation is used to perform this function.

The format of the MID$ statement is:

```
MID$(<string variable>,<numeric formula 1>
 [,<numeric formula 2>])=<string formula>
```

Examples:

```
100 MID$(A$,3,2)=" "
500 MID$(N$(I),2)="TEST"
```

<numeric formula 1> specifies the first character of the <string variable> that will be replaced by the <string formula> to the right of the '=' sign. If <numeric formula 1> is greater than the length of the <string variable>, then a FUNCTION CALL error will occur.

The optional <numeric formula 2> specifies how many characters to copy into the <string variable> from the <string formula>.

Characters are copied from the <string formula> into the <string variable>, starting at the character position specified by <numeric formula 1>. They will be copied until either the end of the <string variable> is reached, the end of the <string formula> is reached, or <numeric formula 2> characters have been copied, whichever occurs first.

- CONTINUED -

VERSION 3.4 ALTAIR BASIC - CONTINUED -

More Examples:

Suppose T$="TEST"
Then:
MID$(T$,2)="ORT"
T$ now equals "TORT"

or

MID$(T$,3,1)=" "
T$ now equals "TE T"

or

MID$(T$,3,2)="XTEND"
T$ now equals "TEXT"

Zero Bytes Allowed in Sequential Disk Files

Zero bytes are now allowed as valid data bytes in sequential data files on the disk. In version 3.3, zero bytes could not be written to sequential files.

Features Added to the DISK Version Only

FILES Command Prints Files Across Line

The FILES command now prints the files on the floppy disk in columns across the page instead of down the page. This is much more convenient for CRT terminals.

*
*

Disk Hardware Notes
-continued from page 13

6. Our dealers now have Pertec FD-400 service manuals. If you suspect difficulty with the FD-400, contact your nearest dealer for his advice and service.

7. If you can't remedy the difficulty, don't try to save postage by just returning the FD-400 alone. Please return your complete 88-DCDD including Cables, Controller Boards, and Drive Chassis. This will allow us to check your system out completely and save you time, money, and hassle.

B. Disk Drive Chassis:

1. On the Buffer Card the most common difficulty is incorrect wiring or incorrectly installed ICs.

2. On the Power Supply Board be sure X1 and X3 are properly installed as indicated on the errata sheet.

3. If you suspect difficulty with the Disk Drive, DO NOT attempt to service it. Any work done on the Pertec FD-400 will void the warranty. Typical service charges for customer damaged FD-400's are $100.00.

4. Do not plug the FD-400 connector in backwards. Be sure to install the polarizing key as the instructions indicate. Plugging in the connector backwards will destroy 5-10 ICs and will cost at least $100.00 for repair.

5. If you must ship the Pertec FD-400 or complete Disk Drive unit, reinstall the Disk door block or strap. Any damage to the mechanism as a result of incorrect shipping typically costs the customer $100.00 in repair charges.

to the CPU. In general this will involve setting the SB and BUS CONTROL bits and selecting the source of the "processor" instruction. It must be noted here that by "processor" instruction we mean any of the bytes that may be required to make a complete 8080 instruction (not just the OP code). There are five sources for "processor" instructions:

CONTROL PROM (via tri-state drivers), enable: S5

Upper address switches (A15--A8), enable: S2

Lower address switches (A7--AØ), enable: S1

Upper address latch, enable: S3

Lower address latch, enable: S4

A typical PROM sequence to complete an examine would be as follows:

1) Control Instruction: Set up SB, BUS CONTROL (BC), S5

2) "Processor Instruction": JMP = 303 (octal)

3) Control Instruction: Set up SB, BC, S1

4) "Processor Instruction": 000_8. The contents of PROM are immaterial here since data is coming from address switches AØ--A7.

5) Control Instruction: Set up SB, BC, S2

-Altair 8800b- continued from page 15

6) "Processor Instruction": A8--A15

7) Control Instruction: Clear Control Latch

8) "Processor Instruction": Stop Code for PROM address counter

FRONT PANEL INTERFACE BOARD

All the lines between the 8800b bus and the Display/Control Board are now buffered through a Front Panel Interface Board. (The bus lines no longer directly drive anything on the Display/Control Board.) The Front Panel Interface Board connects to the Display/Control Board by means of two 34-conductor ribbon cable assemblies, eliminating the wiring harness between the Display/Control Board and the bus.

NEW CPU BOARD

The CPU Board consists of four major functional blocks:

8080A CPU Chip
8224 Clock Generator Chip
8212 Status Latch
Drivers and Receivers

The diagram, Figure 4, shows the relationship between these four blocks. Several points of interest are:

1. The DIG1 signal (see section entitled "New Bus Lines") controls enabling of the input data drivers (DIØ--DI7) from the bus.

2. The ready input to the 8224 (RDYIN) is the logical product of (PRDY) AND (FRDY) AND (XRDY) AND (XRDY2).

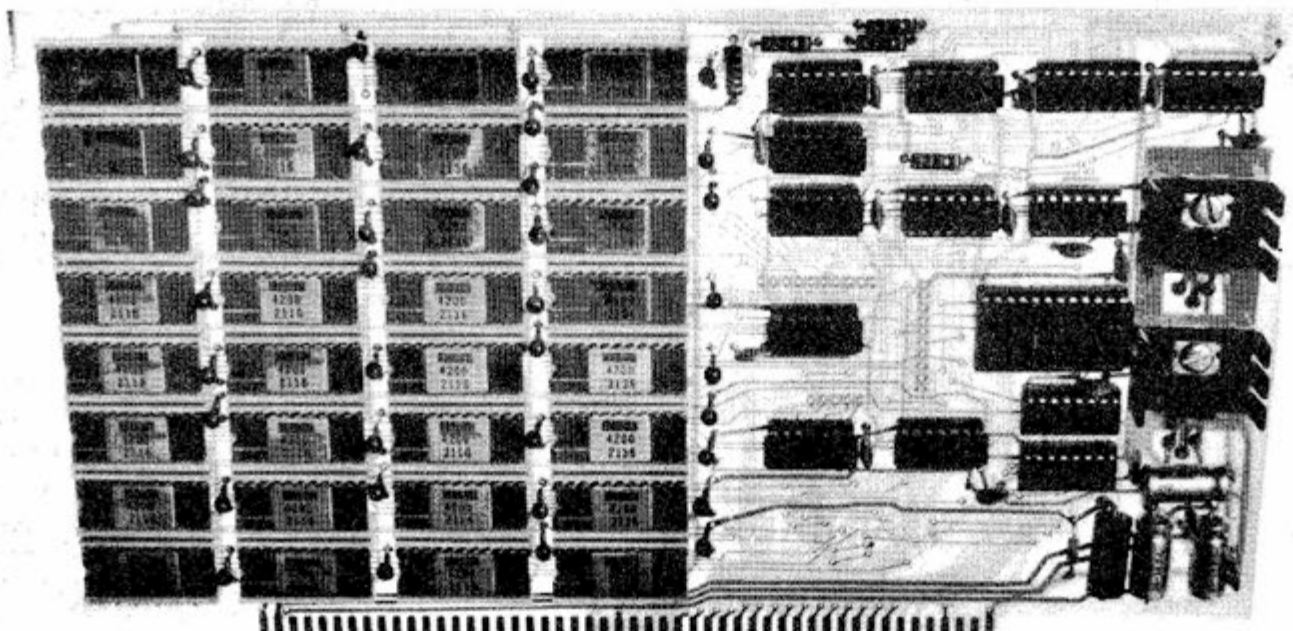
3. The bidirectional data bus to (and from) the 8080A is completely buffered (8216s).

The 8080A, the microprocessor chip itself, exercises control over the CPU board and the rest of the system. It executes the instructions stored in memory and controls all the data transfers.

The 8224 clock generator chip provides the two-phase clock (at the specified voltage levels) required by the 8080A. In addition, it synchronizes the READY and RESET inputs to the 8080A and provides a status signal ($\overline{STSTB}$) that can be used to load the 8212 status latch. This guarantees that status data will be available as soon as possible in a machine cycle. The master timing reference for the 8224 is an external crystal (18MHz). By changing this crystal it is possible to generate the clocks used by the faster versions of the 8080A: the 8080A-1 (1.3us cycle time) and the 8080A-2 (1.5us cycle time).

-continued on page 21

One Slot!



Altair™ 16K Static

Almost too good to be true, the Altair 16K Static RAM board is easily the most advanced memory module yet developed for the Altair 8800, 8800a and 8800b computers.

Four Altair 16K Static boards add up to the entire 64K of memory directly accessible by the Altair.

The Altair 16K Static board offers two surprise features—minimal power requirements and fast access time. One Altair 16K Static board draws less current than any 8800 compatible 4K boards, thus four Altair 16K Static boards can be plugged into the Altair 8800 without beefing up the power supply.

The maximum access time of the Altair 16K Static board is 215 nanoseconds, which makes this board the **fastest Altair compatible static board in existence.**

The Altair 16K Static is now in full production. Special introductory price is $765 in kit form and $945 assembled.

MAIL THIS COUPON TODAY

Enclosed is check for $_____

BankAmericard #_____

or Master Charge #_____

☐ Altair 16K Static ☐ Kit ☐ Assembled
(include $3 for postage and handling)

☐ Please send free information package and price sheet.

NAME_____

ADDRESS_____

CITY_____ STATE AND ZIP_____

MITS/2450 Alamo SE/Albuquerque, NM 87106/505-243-7821

Prices, delivery and specifications subject to change. Allow up to 60 days for delivery.

2450 Alamo SE/Albuquerque, NM 87106/505-243-7821

Troubleshooting

-continued from page 12

If all bits are HIGH: ACIA is not selected due to one or more bad control signals or the output buffer to the CPU is not enabled. After single stepping 12 times (from the beginning of the echo routine) in the echo routine, the ACIA should contain:

(HIGH and LOW are in TTL levels, .8v or less for LOW, 2v or more for HIGH)

CSO	Pin 8	HIGH
RS	Pin 11	LOW
CS1	Pin 10	HIGH
R/W	Pin 13	HIGH
E	Pin 14	HIGH
$\overline{CS2}$	Pin 9	LOW
IC P	Pin 8	LOW
IC O	Pin 8	LOW
IC S	Pin 11	HIGH

NOTE: R/W (pin 13 of the 6850) is LOW for outputting. Continue single stepping 10 more times and Pin 13 should be LOW. IC P, Pin 8, will be HIGH, while Pin 6 will be LOW. SINP will also be LOW. All other pins will be the same as before

Final Notes

If the ACIA is not reset, the Teletype may run open when it is turned on. To correct this, simply flip the Altair ON/OFF switch a couple of times. In some cases an etching error will short out SOUT. This etching error is located on the back of the 2SIO board between IC S, Pin 11, and the gold fingers. Usually SOUT is shorted to address line A6. This etching short should be cut. For those of you who bought the BASIC manual before the extended BASIC section was written, the 2SIO echo routine and bootstrap are listed below: Note that the first 4 bytes reset the ACIA and clear its internal registers. The next 4 bytes tell the ACIA what type of parity, the number of bits/character, and the interrupt information that will be used.

NOTE: There is a misprint in the Appendix, page 101, of the Extended BASIC Manual. The corrections are listed below:

025 in Location 005 is for 1 stop bit.

021, used in Location 005, is for 2 stop bits.

2SIO Echo Routine

OCTAL ADDRESS	OCTAL CODE		OCTAL ADDRESS	OCTAL CODE
000	076		013	322
001	003		014	010
002	323		015	000
003	*020		016	333
004	076		017	**021
005	021	(=2 stop bits, 025=1 stop bit)	020	325
006	323		021	**021
007	*020		022	303
010	333		023	010
011	*020		024	000
012	017			

* - Control channel

** - Data channel

Altair 8800b

-continued from page 19

The 8212 status latch outputs the status signals that define the current machine cycle for all devices attached to the bus. The status latch was used in the 8800b instead of the 8228 bus controller because it was necessary to maintain bus compatibility with the original Altair 8800.

The majority of the system bus lines either originate or terminate at the CPU board. All output lines from the board are driven by tri-state bus drivers (74367 or 74368).

ADDED BUS LINES

All of the original Altair 8800 bus lines have been maintained, and five new lines have been added:

Bus Number	Signal
12	XRDY2
58	FRDY
55	RTC
56	$\overline{STSTB}$
57	DIG1

XRDY2 and FRDY

XRDY2 and FRDY are additional ready inputs to the CPU Board. Formerly, the READY signal consisted of two inputs, PRDY and RDY. The READY signal input to the processor that determines the RUN/WAIT state of the 8080A is now defined as the logical product of these four signals:

READY = (PRDY) AND (FRDY) AND (XRDY) AND (XRDY2)

Therefore, if any of the four "ready" signals on the bus are pulled low, the READY input to the 8080A will go low, causing the CPU to enter a series of .5 microsecond wait states. The four "ready" signals on the bus are used as follows:

PRDY: Used by memory and I/O to synchronize the CPU to slower memory or I/O

FRDY: Used by the Display/Control logic

XRDY and XRDY2:
External ready signals. XRDY and XRDY2 are available to devices that have to stop the processor (by pulling READY low), but must also be able to sense the state of PRDY and FRDY. (For example: DMA)

RTC

RTC is a 60Hz signal used as a timing reference by the Real Time Clock/Vectored Interrupt Board.

$\overline{STSTB}$

$\overline{STSTB}$ is a strobe signal provided by the 8224 clock generator chip. Its basic function is to strobe the 8212 status latch to allow status signals to be set up as soon as possible. This signal is also used by the 8800b Display/Control logic.

DIG1

DIG1 is a signal that controls enabling of the CPU Data Input (DI) drivers. The 8800b employs two sets of DI drivers: one is the standard set used by all memory and I/O devices; the other is used exclusively by the Display/Control logic. If G1 is defined to be the enable signal for the first set of drivers and G2 to be the enable for the second set, then:

G1 = (DIG1) AND (PDBIN)

G2 = ($\overline{DIG1}$) AND (PDBIN)

POWER SUPPLY

Specifications: The power supply furnishes the following voltages to the 8800b bus at the indicated full load currents.

8 volts at 18 amps

+18 volts at 2 amps

-18 volts at 2 amps

The +18 and -18 volt supplies are pre-regulated (series pass transistor) to provide a constant voltage to the bus over the load range of the supplies (0 - 2 amps).

The +8 volt supply is not pre-regulated. Instead, the 8 volt secondary of the transformer is tapped at 3 points. By changing the tap that drives the 8 volt supply, the bus voltage can be maintained between 7.5 volts and 9 volts over a load range of 1 amp to 18 amps.

The primary of the power transformer is tapped to allow for either 115 volt AC or 230 volt AC operation. In addition there are "HIGH LINE" and "LOW LINE" taps for 130 VAC, 100 VAC, 260 VAC and 200 VAC operation. The supply will operate at the above specifications on either 50Hz or 60Hz line frequencies.

-continued on page 23

My Computer Just Stares
-continued from page 13

and I/O (input/output). The memory does just what its name infers: it remembers. It has unique locations (addresses) and can store information (data) within these locations. The I/O section of the computer is what allows the computer to communicate with the rest of the world, e.g. Teletypes, CRTs, other computers.

There has to be a way for these different sections of the computer to communicate with each other. This is done via a data bus and an address bus (see Figure 1). In many microcomputers such as the Altair 8800 and the Altair 680b, the data bus consists of 8 data lines, because information is transferred and stored in 8-bit words called bytes. Each byte of data stored in memory is assigned a unique 16-bit (2-byte) address. The 16-bit address is used by the MPU to gain access to the contents of a particular memory location. Addresses are sent out to memory from the microprocessor on the address bus.

Each I/O device in the system also has one or more 16-bit addresses assigned to it. These I/O devices are addressed the same as any memory byte location.

A LOOK INSIDE THE MICROPROCESSOR

Within the microprocessor is an Arithmetic Logic Unit (ALU). This portion of the chip logic performs all of the basic arithmetic types of operations (add, subtract, compare, etc.) on two operands. Because it must perform these operations in a particular sequence, the ALU is controlled by one of the MPU registers called the Program Counter (PC) (see Figure 2). Once a program (sequence of instructions) is loaded into memory, the Program Counter is loaded with the address of the first instruction of the program. When the computer is put into the RUN mode, the MPU puts the address contained in the PC onto the address bus and reads the contents of that location via the data bus. The instruction that has been read is executed after the PC is incremented to point to the next instruction. This sequence is repeated until the processor is halted.

There are two accumulators, within the MPU, labelled ACCA and ACCB. The purpose of the accumulators is to temporarily store data, either before or after it has been operated upon by the arithmetic logic unit (ALU). For example, to add two numbers, first you must load one of the numbers from memory into ACCA. Second, the other number must be loaded into ACCB. Then an add instruction (which, like the instructions to load the accumulators, will be in the program stored in memory) will be executed to add the contents of the accumulators. After they are added together, the result is then stored in ACCA. This storage in ACCA is temporary. In order to see what the result is, it is necessary to have an instruction in the program which tells the MPU to store the contents of ACCA at a particular memory address location. Located at this memory address location could be either memory or an I/O device. (Some of you super-software-types might be grumbling now since there is an instruction which allows data at a memory address to be directly added to ACCA. You say to first load ACCB is inefficient because it takes extra steps. I agree, but for the sake of example, chose to implement the use of ACCB.)

Perhaps you're wondering at this point just how the MPU knows what instruction to execute. Well, inside of the MPU is an Instruction Decoder. The Instruction Decoder decodes the instructions that are read from the program and has the MPU perform them. Thus, when it reads an ABA (add ACCA to ACCB), in a binary form, it sets up the logic circuitry to take the contents of ACCA and ACCB, adds them together in the ALU, and then stores the result in ACCA.

The Condition Code Register (CCR) is used by the MPU to control program flow during system operation. It consists of 6 bits that can be set to either a one or a zero. A one in a particular CCR bit is considered a set condition, while a zero in a particular bit is considered to be a cleared condition. For example, one of the CCR bits is the Carry bit and it gets set whenever there is a carry from the most significant bit (bit 7) of a result. This could happen, for example, when adding two numbers in the accumulators.

Carry Bit		
0	1 0 1 1 1 0 1 0	ACCA before addition
0	1 1 0 0 0 1 0 1	ACCB before addition
1	0 1 1 1 1 1 1 1	ACCA after addition

Once the carry bit is set, it can be tested or checked to designate program flow, i.e., determine what part of the program should be executed next.

The Index Register (IX) is a two-byte register that is used to temporarily store data or a sixteen bit memory address. In a real world application it can be used for indexing into a table. For example, if you wanted to clear a section of memory (set all of the bytes equal to zero), a starting address would be loaded into the IX. The address location in memory designated by the IX would then be cleared. The IX then could be incremented by one and the next byte would be cleared. This would continue until the last address of the block had been cleared. You may now ask, "How does the MPU know when the last address has been cleared?" This would be taken care of by comparing the IX with a memory byte containing a specified number. The operation would end, for example, when the address in the IX equalled the specified number in memory.

Finally, within the MPU is the Stack Pointer. Very simply, the Stack Pointer is used to point to the Stack. The Stack is a section of memory that is used for temporary storage of MPU register contents. Say, for instance, that you just finished a calculation and the answer is in ACCA. Instead of storing ACCA into a memory address location while performing another operation, it is at times more efficient to just push it onto the stack. Using the stack is more efficient because it takes fewer bytes to implement than a memory store and read. However, it is a sequential read-write operation rather than random access as with normal memory store. This means that each byte has to be read in a last-in-first-out basis. Random access memory usage, however, means that any byte can be accessed at any time regardless of its position. The Push Instruction (PSHA) causes the contents of the indicated accumulator (A in this example) to be stored on the stack, in memory, at the location indicated by the Stack Pointer. The Stack Pointer is automatically decremented by one following the storage operation and is "pointing" to the next empty stack location. The Pull instruction (PULA or PULB) causes the last byte stacked to be loaded into the appropriate accumulator. The Stack Pointer is automatically incremented by one just prior to the data transfer so that it will point to the last byte stacked rather than the next empty location.

Another useful implementation of the Stack is in subroutine linkage. A subroutine is a program within a program. Say, for example, that you had a long program which performed many operations. One of the things that it has to do frequently is multiply two numbers together. What we can do is write a small program (subroutine) that multiplies two numbers, and go to the subroutine at every place in the program where necessary.

The stack is used to store the current address of the Program Counter every time the program goes to the subroutine. This is how the MPU keeps track of where it left off in the main program. When the subroutine is finished, the Program Counter address is pulled off the Stack and reloaded into the Program Counter. The main program then picks up where it left off before having called the subroutine.

-continued

My Computer Just Stares -continued

In this way, the multiplication routine does not have to be written into the program many times, but instead just once. This can save much memory space, which can be critical when writing long programs.

In the next issue of Computer Notes we'll see how all of these parts of the MPU, along with the memory and I/O work together to run the programs that are entered into the computer.

References:

Primer on Microprocessors, Electronics Products Magazine, Jan. 20, 1975, p. 25-32.

M6800 Microprocessor Programming Manual, Motorola, Inc., 1975.

Altair 680 Programming Manual, MITS, Inc., 1976

Figure 2 courtesy of Motorola, Inc.

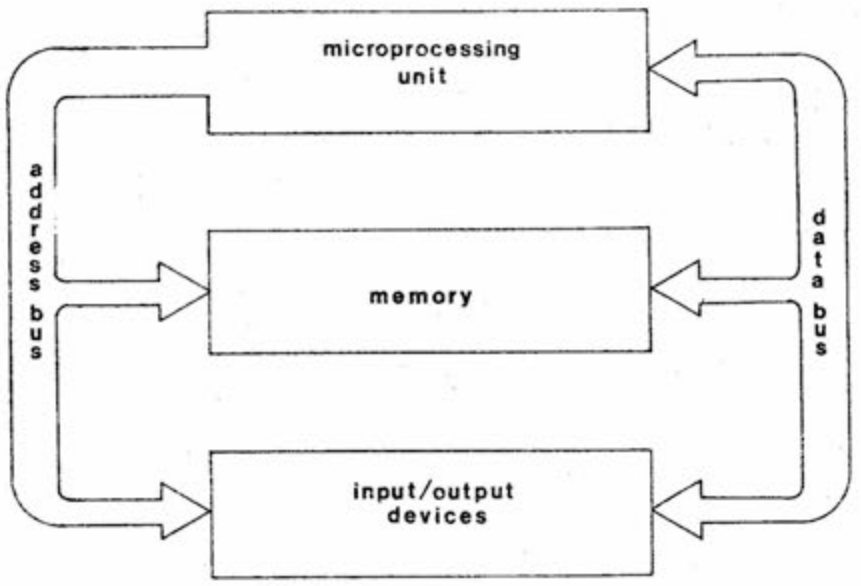


Figure 1 Microcomputer System Block Diagram

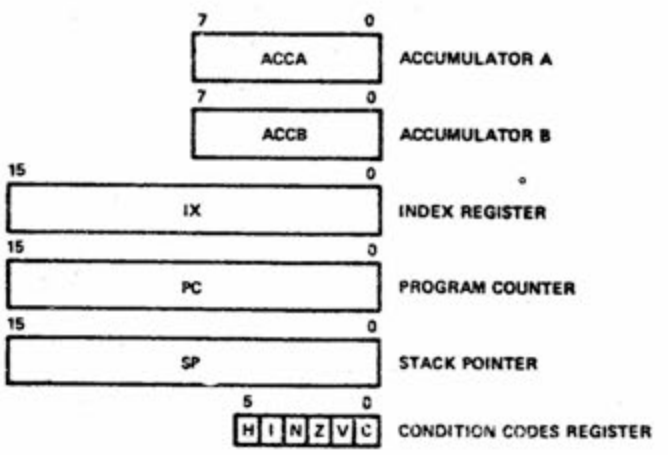


Figure 2 Programming Model of M6800

Altair 8800b
-continued from page 21

MISCELLANEOUS

18-Slot Motherboard

The four-slot expander cards in the Altair 8800 have been replaced by a single piece 18-slot motherboard. The 18-slot motherboard contains 100 solder lands which comprise the 100 pin bus. The need for expander board wiring has been completely eliminated. Assembled units may be ordered with 6, 12 or 18 edge connectors.

Single Step/Slow

Single Step: The 8800b has provisions for selecting one of two modes for the single step operation by means of a patching jumper. In the first mode a single machine cycle will be executed each time the switch is activated. The second mode allows a complete instruction cycle to be executed.

Slow: The SLOW mode on the 8800b will operate in the same manner as single step as far as the mode is concerned. The speed of the slow mode is selectable by patching jumpers for three different speeds.

Data LEDs

The front panel data LEDs are driven (in the STOP mode) by the Data Out lines (DOØ--DO7). (In the Altair 8800 they are driven by the Data Input lines, DIØ--DI7.) If single step is operated in the single machine cycle mode, the correct data will be displayed on the LEDs during memory write and output machine cycles.

RESET Switch

The RESET switch on the front panel has provisions for wiring to the front panel switch enable line (instead of to ground). If this is done, the machine can be RESET only in the STOP mode.

Control PROM

The front panel control PROM (1702A) has been divided into 16 sectors, each 16 bytes long. The even addresses within any sector are used to control the front panel circuit. Since the last address must contain a stop code for the PROM Address Counter, there are 7 bytes available in each sector for machine code. This means that there is some flexibility in redefining the front panel switch functions (for special applications) by re-programming the control RPOM. The functions are constrained by the fact that there are only 7 bytes of machine code available to execute them.

PRICES

Altair 8800b Computer Kit (Kit includes 2 edge connectors and card guide sets)	$ 840.00
Assembled Altair 8800b Computer	
(With 6 edge connectors)	1,100.00
(With 12 edge connectors)	1,175.00
(With 18 edge connectors)	1,250.00
8800b-U (kit only)* The 8800b-U will update the 8800 to the 8800b. The only items not included in this package that are included in the 8800b are the switches, LEDs, case and motherboard.	489.00
8800b-SL (kit only)* The 8800b-SL is a further update to the 8800b-U and contains the switches and LEDs for the 8800, to duplicate the 8800b switches and LEDs.	78.00
8800b-PS (kit only)* The 8800b-PS is for those desiring the power supply modification but not the 8800b-U in its entirety.	147.00

*(add $3 for postage and handling)

Easy as Spaghetti

"Learning to use a computer should be roughly equivalent to learning how to make spaghetti."

Personal Computing is the new people/computer magazine that understands this concept. We believe that if you are bright enough to:

1. Brown ½ pound of hamburger in a large saucepan. Add celery, 1 clove crushed garlic . . .

and etcetera, then you are probably bright enough to learn how to make practical use of your own, personal computer. You are probably bright enough to play games with it or make use of it for your own business or educational purposes.

Personal Computing looks at the computer as a handy, powerful mind tool. One that expands your ability to keep track of the many complicated aspects of a modern society.

The first issue of **Personal Computing** includes the following articles:

1. Part one of **Spaghetti BASIC.** Easy to learn course on programming a microcomputer in the simplest of computer languages, BASIC.

2. **Ten Steps to Becoming a Computer Hobbyist.** Tells you about the phenomena of this newest breed of electronic tinkerer. And if you'd like to join the fun, we'll try to guide you in the right direction.

3. **The Equalizer.** Zany feature by Nels Winkless III that views the personal computer as "the most powerful equalizer since the Colt 45 in the old frontier."

Personal Computing will provide educational articles on basic computer jargon, computer architecture, and computer programming. These articles will be written in easy to understand language for the beginner and they will serve as a reference for people already knowledgeable in the field.

Another regular feature on **Personal Computing** will be a section on "Future Computing." Also, each issue will include a poster sized, four color computer graphic.

Personal Computing is a new kind of magazine, completely different from existing hobbyist publications.

Benwill Publishing, the publishers of **Digital Design** and **Minicomputer News,** invites you to take advantage of a special, charter subscription offer. For a limited time only, you can subscribe to the first year of **Personal Computing** for only $6. This includes a free copy of the initial, October-November kickoff issue, plus the six bi-monthly issues scheduled for 1977.

To subscribe to **Personal Computing,** fill out this coupon and return it with your check to:

Personal Computing

Benwill Publishing Corp., 167 Corey Road, Brookline, MA 02146, USA.

NAME_____

ADDRESS_____

CITY_____ STATE & ZIP_____

Note: $6 special charter subscription rates apply to U.S. only. Mexico and Canada, $12 surface mail and $16 airmail. Foreign airmail: $24. Charter rates increased $2 after September, 1976.