ALTAIR SHORT COURSE

I. INTRODUCTION

II. COMPUTER FUNDAMENTALS

1. COMPUTER ORGANIZATION
2. NUMBER SYSTEMS
3. BASIC LOGIC CIRCUITRY
4. PROGRAMMING

III. HARDWARE

1. ALTAIR ORGANIZATION
2. I/O STRUCTURE
3. INTERFACING TECHNIQUES

IV. SOFTWARE

1. PROGRAMMING CONCEPTS
2. MACHINE LANGUAGE
3. ASSEMBLY LANGUAGE
4. BASIC LANGUAGE

V. QUESTIONS & ANSWERS

---------------------------------- NOTES ----------------------------------

1. Protect enable operates on board currently addressed. Other boards are not protected. Use with assembler. Unprotect is the same.

2. Parallel I/O board is in the full duplex mode. Bytes are sent and received simultaneously.

3. DBIN and IN lines must be high to allow input. Control is through an AND gate.

4. Current drain on any card is to be less than 0.6 amps max.

5. Only one TTL load on a Bus line card connection

6. For information on A/D and D/A conversion:

   Ed Currie - (305) 885-9288 - (Florida; evenings)

------------------------------------- NOTES --------------------------------

---------------------------------- NOTES ----------------------------

------------------------------------ NOTES ---------------------------------

------------------------------------- NOTES -------------------------------------

-------------------------------------NOTES -------------------------------

------------------------------------ NOTES ------------------------------

------------------------------------ NOTES --------------------------------

------------------------------------ NOTES --------------------------------

-----------------------------------NOTES ---------------------------------

------------------------------------ NOTES ------------------------------------

------------------------------------- NOTES ---------------------------------

----------------------------------- NOTES -----------------------------------

------------------------------------- NOTES -------------------------------------

----------------------------------- NOTES ---------------------------------

------------------------------------ NOTES ----------------------------------

------------------------------------ NOTES ----------------------------------

------------------------------------ NOTES ---------------------------------

---------------------------------- NOTES ----------------------------------

----------------------------------- NOTES ---------------------------------

The MITS system software consists of three packages, an assembly language development system (called Package I), a machine language debugging package (DBG-8800) and ALTAIR BASIC.

These three packages operate in a stand-alone environment i.e. without disk or other high speed random access storage device. I/O devices supported are asynchronous serial ASCII terminals, parallel ASCII terminals (such as TVT's) and the ACR (cassette) interface board.

Two disk based systems are now under development and should be ready by mid-December. These are Extended BASIC and the DOS. Extended BASIC and the DOS both use the same file structure I/O code; Extended BASIC is an advanced BASIC interpreter while the DOS package is a disk based assembly language development system.

Here is an overview of the features of the packages currently available:

Package I
(System Monitor, Editor & Assembler)

System Monitor - 2K Bytes

Contains I/O drivers for system console, ACR board (supports multiple files on one cassette). Loads programs from paper tape or cassette.

Text Editor - 2K bytes

Facilitates editing of source programs. The editor is line oriented, that is, commands always reference a line or group of lines.

Commands:

P - Print Lines
I - Insert Lines
D - Delete Lines
R - Replace Lines
E - Exit to Monitor
A - Alter a line. Enables user to change, delete or insert minor changes in an already existing line.

F String - Searches from the current line forward for an occurance of the character string given as its argument.

Relative addressing is allowed. P.+6 would print the sixth line after the current one.

Line Feed - Prints and moves the current line pointer to the next line.

Escape - Prints and moves current line pointer to line before current one.

S - Saves File.
L - Loads a File.

Assembler - 3K Bytes

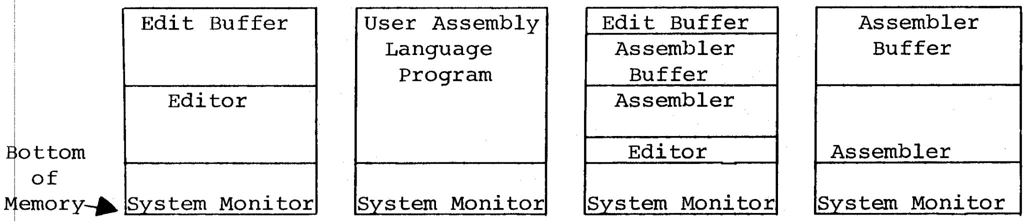
The ALTAIR loading assembler assembles a source program in one pass from paper tape, cassette or from the current Editor buffer. Object code is stored directly into memory as assembly progresses.

Since the assembler is one pass, it is possible to avoid the time consuming process of re-reading source tapes more than once, which is the case with multi-pass assemblers. Also, if the program being assembled resides in the edit buffer, assembly is almost instantaneous and the user may immediately correct and re-assemble his program.

Features not provided by the ALTAIR assembler:

- Conditional Assembly
- Macros
- Cross Reference Listing

ALTERNATIVE MEMORY MAPS FOR PACKAGE I



NOTE: Package I can be used with all memory available. A minimum of 8K is necessary, but any extra memory may be allocated for Package I buffer or program storage.

DEBUG CAPABILITIES OF DBG-8800
2K Bytes

Examine/Modify Commands

A location to be examined can be specified by an octal address, a register name (A,B,C,D,E,H,L, or S for status word), or a period to indicate the address in the current address pointer. In addition a location can be specified with any of the above forms but with a + or - octal offset (e.g. .+7).

The specified location can be examined by typing a / after it. A / causes the following:

- Type the specified (and possibly following locations) in accordance with the current I/O mode.
- Open location(s) for modification.
- A carriage return will close the location.
- A line feed will print the address and contents of the next memory location(s) (depending on I/O mode).
- ↑ acts as a line feed but goes to previous instead of next location(s).
- ; causes contents to typed in octal regardless of I/O mode.
- A tab (control I) will open for examination the address associated with a previously displayed symbolic 3-byte instruction.
- Otherwise input information will be accepted to modify the contents of the current location. Input data must conform to the specified I/O mode.
- A rubout typed at any time will cause input to the current line to be aborted, and a new line will be started.

I/O modes can be respecified at any time by typing an escape followed by one of the following characters:

O	(octal)
A	(ASCII)
S	(symbolic) (instruction format)
W	(two-byte words)
D	(decimal)

Execution Commands

An address (as specified above) if followed by a G will cause execution of the user program to begin at the specified address.

A P will cause execution of the user program to proceed from the most recently encountered break point. An octal number can precede a P to indicate the number of breakpoints that the user wishes to pass over before finally returning control to DEBUG.

Breakpoint Commands

There are 8 possible breakpoints (numbered 0 thru 7). To set a breakpoint an address is followed by an X. The first free breakpoint will be set.

A Y is typed to remove all breakpoints.

A Q will cause a table of all set breakpoints to be displayed.

Memory Block Commands

The contents of a block of memory can be displayed by typing a command of the form:

```
     (ADDRESS A),(ADDRESS B)T
```

This command will cause memory contents beginning with (ADDRESS A) and ending with (ADDRESS B) to be displayed in the current I/O mode.

8K BASIC

ALTAIR BASIC (Version 3.1) requires a minimum of 6K bytes of memory.

Features not normally found in BASIC include Boolean operators (AND, OR, NOT) which can be used in IF statements or for bit manipulation, INP and OUT which can read or write a byte from any I/O port, and PEEK and POKE to read or write a byte from any memory location. Variable length strings (up to 255 characters) are provided, as well as the LEFT$, RIGHT$ and MID$ functions to take substrings of strings, a concatenation operator and VAL and STR$ to convert between strings and numbers. Number representation is 32 bit floating point. Both string and numeric arrays of up to 30 dimensions may be used, and can be allocated dynamically during program execution. Nesting of loops and subroutine calls is limited only by available memory. Intrinsic functions are SIN, COS, TAN, LOG, EXP, SQR, SGN, ABS, INT, FRE, RND and POS, in addition to TAB and SPC in PRINT statements.

Other important features are direct execution of statements, multiple statements per line, and the abiltiy to interrupt program execution and then continue after the examination of variable values.

For the MITS' line of ALTAIR microcomputers, 8K BASIC costs $75 with the pruchase of 8K memory and an I/O interface board.

4K BASIC

The 4K version of BASIC, with less features than 8K BASIC, costs $60 for ALTAIR owners with 4K memory and an I/O board.

The features of 4K BASIC are a subset of those of 8K BASIC. Main restrictions are:

- No strings.
- Matrices of only one dimension.
- Math functions are ABS, INT, SQR, RND, SIN, SGN
- AND, OR, NOT
- No PEEK, POKE, INP, OUT
- No interrupt response subroutines.
- No ON...GOTO, ON...GOSUB
- No CONTinue command.

NOTE: it is often advantageous to run 4K BASIC in an 8K ALTAIR if you have a long program or a program that uses large single dimensioned arrays.

REFERENCE

MATERIAL

MITS
"Creative Electronics"

COMMANDS

A command is usually given after BASIC has typed OK. This is called the "Command Level". Commands may be used as program statements. Certain commands, such as LIST, NEW and CLOAD will terminate program execution when they finish.

NAME	EXAMPLE	PURPOSE/USE
CLEAR	*(SEE PAGE 42 FOR EXAMPLES AND EXPLANATION)	
LIST	LIST LIST 100	Lists current program optionally starting at specified line. List can be control-C'd (BASIC will finish listing the current line)
NULL	NULL 3	(Null command only in 8K version, but paragraph applicable to 4K version also) Sets the number of null (ASCII 0) characters printed after a carriage return/line feed. The number of nulls printed may be set from 0 to 71. This is a must for hardcopy terminals that require a delay after a CRLF* It is necessary to set the number of nulls typed on CRLF to 0 before a paper tape of a program is read in from a Teletype *(TELETYPE is a registered trademark of the TELETYPE CORPORATION)*. In the 8K version, use the null command to set the number of nulls to zero. In the 4K version, this is accomplished by patching location 46 octal to contain the number of nulls to be typed plus 1. (Depositing a 1 in location 46 would set the number of nulls typed to zero.) When you punch a paper tape of a program using the list command, null should be set >=3 for 10 CPS terminals, >=6 for 30 CPS terminals. When not making a tape, we recommend that you use a null setting of 0 or 1 for Teletypes, and 2 or 3 for hard copy 30 CPS terminals. A setting of 0 will work with Teletype compatible CRT's.
RUN	RUN	Starts execution of the program currently in memory at the lowest numbered statement. Run deletes all variables (does a CLEAR) and restores DATA. If you have stopped your program and wish to continue execution at some point in the program, use a direct GOTO statement to start execution of your program at the desired line.

*CRLF=carriage return/line feed

	RUN 200	(8K version only) optionally starting at the specified line number
NEW	NEW	Deletes current program and all variables

THE FOLLOWING COMMANDS ARE IN THE 8K VERSION ONLY

CONT	CONT	Continues program execution after a control/C is typed or a STOP statement is executed. You cannot continue after any error, after modifying your program, or before your program has been run. One of the main purposes of CONT is debugging. Suppose at some point after running your program, nothing is printed. This may be because your program is performing some time consuming calculation, but it may be because you have fallen into an "infinite loop". An infinite loop is a series of BASIC statements from which there is no escape. The ALTAIR will keep executing the series of statements over and over, until you intervene or until power to the ALTAIR is cut off. If you suspect your program is in an infinite loop, type in a control/C. In the 8K version, the line number of the statement BASIC was executing will be typed out. After BASIC has typed out OK, you can use PRINT to type out some of the values of your variables. After examining these values you may become satisfied that your program is functioning correctly. You should then type in CONT to continue executing your program where it left off, or type a direct GOTO statement to resume execution of the program at a different line. You could also use assignment (LET) statements to set some of your variables to different values. Remember, if you control/C a program and expect to continue it later, you must not get any errors or type in any new program lines. If you do, you won't be able to continue and will get a "CN" (continue not) error. It is impossible to continue a direct command. CONT always resumes execution at the next statement to be executed in your program when control/C was typed.

THE FOLLOWING TWO COMMANDS ARE AVAILABLE IN THE 8K CASSETTE VERSION ONLY

CLOAD	CLOAD P	Loads the program named P from the cassette tape. A NEW command is automatically done before the CLOAD command is executed. When done, the CLOAD will type out OK as usual. The one-character program designator may be any printing character. CSAVE and CLOAD use I/O ports 6 & 7. See Appendix I for more information.
CSAVE	CSAVE P	Saves on cassette tape the current program in the ALTAIR's memory. The program in memory is left unchanged. More than one program may be stored on cassette using this command. CSAVE and CLOAD use I/O ports 6 & 7. See Appendix I for more information

OPERATORS

SYMBOL	SAMPLE STATEMENT	PURPOSE/USE
=	A=100 LET Z=2.5	Assigns a value to a variable The LET is optional
-	B=-A	Negation. Note that 0-A is subtraction, while -A is negation.
↑ *(usually a shift/N)*	130 PRINT X↑3	Exponentiation (8K version) (equal to X*X*X in the sample statement) 0↑0=1 0 to any other power = 0 A↑B, with A negative and B not an integer gives an FC error.
*	140 X=R*(B*D)	Multiplication
/	150 PRINT X/1.3	Division
+	160 Z=R+T+Q	Addition
-	170 J=100-I	Subtraction

RULES FOR EVALUATING EXPRESSIONS:

1) Operations of higher precedence are performed before operations of lower precedence. This means the multiplication and divisions are performed before additions and subtractions. As an example, 2+10/5 equals 4, not 2.4. When operations of equal precedence are found in a formula, the left hand one is executed first: 6-3+5=8, not -2.

2) The order in which operations are performed can always be specified explicitly through the use of parentheses. For instance, to add 5 to 3 and then divide that by 4, we would use (5+3)/4, which equals 2. If instead we had used 5+3/4, we would get 5.75 as a result (5 plus 3/4).

The precedence of operators used in evaluating expressions is as follows, in order beginning with the highest precedence:
(Note: Operators listed on the same line have the same precedence.)

1) FORMULAS ENCLOSED IN PARENTHESIS ARE ALWAYS EVALUATED FIRST

2) ↑ EXPONENTIATION *(8K VERSION ONLY)*

3) NEGATION -X WHERE X MAY BE A FORMULA

4) * / MULTIPLICATION AND DIVISION

5) + - ADDITION AND SUBTRACTION

6) RELATIONAL OPERATORS: *(equal precedence for all six)*

=	EQUAL
<>	NOT EQUAL
<	LESS THAN
>	GREATER THAN
<=	LESS THAN OR EQUAL
>=	GREATER THAN OR EQUAL

(8K VERSION ONLY) (These 3 below are Logical Operators)

7) NOT LOGICAL AND BITWISE "NOT"
LIKE NEGATION, NOT TAKES ONLY THE
FORMULA TO ITS RIGHT AS AN ARGUMENT

8) AND LOGICAL AND BITWISE "AND"

9) OR LOGICAL AND BITWISE "OR"

In the 4K version of BASIC, relational operators can only be used once in an IF statement. However, in the 8K version a relational expression can be used as part of any expression.

Relational Operator expressions will always have a value of True (-1) or a value of False (0). Therefore, (5=4)=0, (5=5)=-1, (4>5)=0, (4<5)=-1, etc.

The THEN clause of an IF statement is executed whenever the formula after the IF is not equal to 0. That is to say, IF X THEN... is equivalent to IF X<>0 THEN... .

SYMBOL	SAMPLE STATEMENT	PURPOSE/USE
=	10 IF A=15 THEN 40	Expression Equals Expression
<>	70 IF A<>0 THEN 5	Expression Does Not Equal Expression
>	30 IF B>100 THEN 8	Expression Greater Than Expression
<	160 IF B<2 THEN 10	Expression Less Than Expression
<=,=<	180 IF 100<=B+C THEN 10	Expression Less Than Or Equal To Expression
>=,=>	190 IF Q=>R THEN 50	Expression Greater Than Or Equal To Expression
AND	2 IF A<5 AND B<2 THEN 7	*(8K Version only)* If expression 1 (A<5) AND expression 2 (B<2) are both true, then branch to line 7
OR	IF A<1 OR B<2 THEN 2	*(8K Version only)* If either expression 1 (A<1) OR expression 2 (B<2) is true, then branch to line 2
NOT	IF NOT Q3 THEN 4	*(8K Version only)* If expression "NOT Q3" is true (because Q3 is false), then branch to line 4 *Note: NOT -1=0 (NOT true=false)*

AND, OR and NOT can be used for bit manipulation, and for performing boolean operations.

These three operators convert their arguments to sixteen bit, signed two's, complement integers in the range -32768 to +32767. They then perform the specified logical operation on them and return a result within the same range. If the arguments are not in this range, an "FC" error results.

The operations are performed in bitwise fashion, this means that each bit of the result is obtained by examining the bit in the same position for each argument.

The following truth table shows the logical relationship between bits:

OPERATOR	ARG. 1	ARG. 2	RESULT
AND	1	1	1
	0	1	0
	1	0	0
	0	0	0

(cont.)

OPERATOR	ARG. 1	ARG. 2	RESULT
OR	1	1	1
	1	0	1
	0	1	1
	0	0	0
NOT	1	-	0
	0	-	1

EXAMPLES: *(In all of the examples below, leading zeroes on binary numbers are not shown.)*

63 AND 16=16 — Since 63 equals binary 111111 and 16 equals binary 10000, the result of the AND is binary 10000 or 16.

15 AND 14=14 — 15 equals binary 1111 and 14 equals binary 1110, so 15 AND 14 equals binary 1110 or 14.

-1 AND 8=8 — -1 equals binary 1111111111111111 and 8 equals binary 1000, so the result is binary 1000 or 8 decimal.

4 AND 2=0 — 4 equals binary 100 and 2 equals binary 10, so the result is binary 0 because none of the bits in either argument match to give a 1 bit in the result.

4 OR 2=6 — Binary 100 OR'd with binary 10 equals binary 110, or 6 decimal.

10 OR 10=10 — Binary 1010 OR'd with binary 1010 equals binary 1010, or 10 decimal.

-1 OR -2=-1 — Binary 1111111111111111 (-1) OR'd with binary 1111111111111110 (-2) equals binary 1111111111111111, or -1.

NOT 0=-1 — The bit complement of binary 0 to 16 places is sixteen ones (1111111111111111) or -1. Also NOT -1=0.

NOT X — NOT X is equal to -(X+1). This is because to form the sixteen bit two's complement of the number, you take the bit (one's) complement and add one.

NOT 1=-2 — The sixteen bit complement of 1 is 1111111111111110, which is equal to -(1+1) or -2.

A typical use of the bitwise operators is to test bits set in the ALTAIR's inport ports which reflect the state of some external device.

Bit position 7 is the most significant bit of a byte, while position 0 is the least significant.

For instance, suppose bit 1 of I/O port 5 is 0 when the door to Room X is closed, and 1 if the door is open. The following program will print "Intruder Alert" if the door is opened:

```
10 IF NOT (INP(5) AND 2) THEN 10
```

This line will execute over and over until bit 1 (masked or selected by the 2) becomes a 1. When that happens, we go to line 20 .

```
20 PRINT "INTRUDER ALERT"
```

Line 20 will output "INTRUDER ALERT".

However, we can replace statement 10 with a "WAIT" statement, which has exactly the same effect.

```
10 WAIT 5,2
```

This line delays the execution of the next statement in the program until bit 1 of I/O port 5 becomes 1. The WAIT is much faster than the equivalent IF statement and also takes less bytes of program storage.

The ALTAIR's sense switches may also be used as an input device by the INP function. The program below prints out any changes in the sense switches.

```
10 A=300:REM SET A TO A VALUE THAT WILL FORCE PRINTING
20 J=INP(255):IF J=A THEN 20
30 PRINT J;:A=J:GOTO 20
```

The following is another useful way of using relational operators:

```
125 A=-(B>C)*B-(B<=C)*C
```

This statement will set the variable A to MAX(B,C) = the larger of the two variables B and C.

STATEMENTS

Note: In the following description of statements, an argument of V or W denotes a numeric variable, X denotes a numeric expression, X$ denotes a string expression and an I or J denotes an expression that is truncated to an integer before the statement is executed. Truncation means that any fractional part of the number is lost, e.g. 3.9 becomes 3, 4.01 becomes 4.

An expression is a series of variables, operators, function calls and constants which after the operations and function calls are performed using the precedence rules, evaluates to a numeric or string value.

A constant is either a number (3.14) or a string literal ("FOO").

NAME	EXAMPLE	PURPOSE/USE
DATA	`10 DATA 1,3,-1E3,.04`	Specifies data, read from left to right. Information appears in data statements in the same order as it will be read in the program. IN THE 4K VERSION OF BASIC, DATA STATEMENTS MUST BE THE FIRST STATEMENTS ON A LINE. Expressions may also appear in the 4K version data statements.
	`20 DATA " FOO",ZOO`	*(8K Version)* Strings may be read from DATA statements. If you want the string to contain leading spaces (blanks), colons (:) or commas (,), you must enclose the string in double quotes. It is impossible to have a double quote within string data or a string literal. (""MITS"" is illegal)
DEF	`100 DEF FNA(V)=V/B+C`	*(8K Version)* The user can define functions like the built-in functions (SQR, SGN, ABS, etc.) through the use of the DEF statement. The name of the function is "FN" followed by any legal variable name, for example: FNX, FNJ7, FNKO, FNR2. User defined functions are restricted to one line. A function may be defined to be any expression, but may only have one argument. In the example B & C are variables that are used in the program. Executing the DEF statement defines the function. User defined functions can be redefined by executing another DEF statement for the same function. User defined string functions are not allowed. "V" is called the dummy variable.
	`110 Z=FNA(3)`	Execution of this statement following the above would cause Z to be set to 3/B+C, but the value of V would be unchanged.
DIM	`113 DIM A(3),B(10)`	Allocates space for matrices. All matrix elements are set to zero by the DIM statement.
	`114 DIM R3(5,5),D$(2,2,2)`	*(8K Version)* Matrices can have more than one dimension. Up to 255 dimensions are allowed, but due to the restriction of 72 characters per line the practical maximum is about 34 dimensions.
	`115 DIM Q1(N),Z(2*I)`	Matrices can be dimensioned dynamically during program execution. If a matrix is not explicitly dimensioned with a DIM statement, it is assumed to be a single dimensioned matrix of whose single subscript

	117 A(8)=4	may range from 0 to 10 (eleven elements). If this statement was encountered before a DIM statement for A was found in the program, it would be as if a DIM A(10) had been executed previous to the execution of line 117. All subscripts start at zero (0), which means that DIM X(100) really allocates 101 matrix elements.
END	999 END	Terminates program execution without printing a BREAK message. (see STOP) CONT after an END statement causes execution to resume at the statement after the END statement. END can be used anywhere in the program, and is optional.
FOR	300 FOR V=1 TO 9.3 STEP .6	(see NEXT statement) V is set equal to the value of the expression following the equal sign, in this case 1. This value is called the initial value. Then the statements between FOR and NEXT are executed. The final value is the value of the expression following the TO. The step is the value of the expression following STEP. When the NEXT statement is encountered, the step is added to the variable.
	310 FOR V=1 TO 9.3	If no STEP was specified, it is assumed to be one. If the step is positive and the new value of the variable is <= the final value (9.3 in this example), or the step value is negative and the new value of the variable is => the final value, then the first statement following the FOR statement is executed. Otherwise, the statement following the NEXT statement is executed. All FOR loops execute the statements between the FOR and the NEXT at least once, even in cases like FOR V=1 TO 0.
	315 FOR V=10*N TO 3.4/Q STEP SQR(R)	Note that expressions (formulas) may be used for the initial, final and step values in a FOR loop. The values of the expressions are computed only once, before the body of the FOR....NEXT loop is executed.

```
320 FOR V=9 TO 1 STEP -1
```

When the statement after the NEXT is executed, the loop variable is never equal to the final value, but is equal to whatever value caused the FOR...NEXT loop to terminate. The statements between the FOR and its corresponding NEXT in both examples above (310 & 320) would be executed 9 times.

```
330 FOR W=1 TO 10: FOR W=1 TO :NEXT W:NEXT W
```

Error: do not use nested FOR...NEXT loops with the same index variable.
FOR loop nesting is limited only by the available memory.
(see Appendix D)

GOTO

```
50 GOTO 100
```

Branches to the statement specified.

GOSUB

```
10 GOSUB 910
```

Branches to the specified statement (910) until a RETURN is encountered; when a branch is then made to the statement after the GOSUB. GOSUB nesting is limited only by the available memory.
(see Appendix D)

IF...GOTO

```
32 IF X<=Y+23.4 GOTO 92
```

(8K Version) Equivalent to IF...THEN, except that IF...GOTO must be followed by a line number, while IF...THEN can be followed by either a line number or another statement.

IF...THEN

```
IF X<10 THEN 5
```

Branches to specified statement if the relation is True.

```
20 IF X<0 THEN PRINT "X LESS THAN 0"
```

Executes all of the statements on the remainder of the line after the THEN if the relation is True.

```
25 IF X=5 THEN 50:Z=A
```

WARNING. The "Z=A" will never be executed because if the relation is true, BASIC will branch to line 50. If the relation is false Basic will proceed to the line after line 25.

```
26 IF X<0 THEN PRINT "ERROR, X NEGATIVE": GOTO 350
```

In this example, if X is less than 0, the PRINT statement will be executed and then the GOTO statement will branch to line 350. If the X was 0 or positive, BASIC will proceed to execute the lines after line 26.

INPUT	3 INPUT V,W,W2	Requests data from the terminal (to be typed in). Each value must be separated from the preceeding value by a comma (,). The last value typed should be followed by a carriage return. A "?" is typed as a prompt character. In the 4K version, a value typed in as a response to an INPUT statement may be a formula, such as 2*SIN(.16)-3. However, in the 8K version, only constants may be typed in as a response to an INPUT statement, such as 4.5E-3 or "CAT". If more data was requested in an INPUT statement than was typed in, a "??" is printed and the rest of the data should be typed in. If more data was typed in than was requested, the extra data will be ignored. The 8K version will print the warning "EXTRA IGNORED" when this happens. The 4K version will not print a warning message. *(8K Version)* Strings must be input in the same format as they are specified in DATA statements.
	5 INPUT "VALUE";V	*(8K Version)* Optionally types a prompt string ("VALUE") before requesting data from the terminal. If carriage return is typed to an input statement, BASIC returns to command mode. Typing CONT after an INPUT command has been interrupted will cause execution to resume at the INPUT statement.
LET	300 LET W=X 310 V=5.1	Assigns a value to a variable. "LET" is optional.
NEXT	340 NEXT V	Marks the end of a FOR loop.
	345 NEXT	*(8K Version)* If no variable is given, matches the most recent FOR loop.
	350 NEXT V,W	*(8K Version)* A single NEXT may be used to match multiple FOR statements. Equivalent to NEXT V:NEXT W.

ON...GOTO	100 ON I GOTO 10,20,30,40	*(8K Version)* Branches to the line indicated by the I'th number after the GOTO. That is:
		IF I=1, THEN GOTO LINE 10
		IF I=2, THEN GOTO LINE 20
		IF I=3, THEN GOTO LINE 30
		IF I=4, THEN GOTO LINE 40.

If I=0 or I attempts to select a non-existent line (>=5 in this case), the statement after the ON statement is executed. However, if I is >255 or <0, an FC error message will result. As many line numbers as will fit on a line can follow an ON...GOTO.

```
105 ON SGN(X)+2 GOTO 40,50,60
```

This statement will branch to line 40 if the expression X is less than zero, to line 50 if it equals zero, and to line 60 if it is greater than zero.

ON...GOSUB

```
110 ON I GOSUB 50,60
```

(8K Version) Identical to "ON...GOTO", except that a subroutine call (GOSUB) is executed instead of a GOTO. RETURN from the GOSUB branches to the statement after the ON...GOSUB.

OUT

```
355 OUT I,J
```

(8K Version) Sends the byte J to the output port I. Both I & J must be >=0 and <=255.

POKE

```
357 POKE I,J
```

(8K Version) The POKE statement stores the byte specified by its second argument (J) into the location given by its first argument (I). The byte to be stored must be =>0 and <=255, or an FC error will occur. The address (I) must be =>0 and <=32767, or an FC error will result. Careless use of the POKE statement will probably cause you to "poke" BASIC to death; that is, the machine will hang, and you will have to reload BASIC and will lose any program you had typed in. A POKE to a non-existent memory location is harmless. One of the main uses of POKE is to pass arguments to machine language subroutines. (see Appendix J) You could also use PEEK and POKE to write a memory diagnostic or an assembler in BASIC.

PRINT

```
360 PRINT X,Y;Z
370 PRINT
380 PRINT X,Y;
390 PRINT "VALUE IS";A
400 PRINT A2,B,
```

Prints the value of expressions on the terminal. If the list of values to be printed out does not end with a comma (,) or a semicolon (;), then a carriage return/line feed is executed after all the values have been printed. Strings enclosed in quotes (") may also be printed. If a semicolon separates two expressions in the list, their values are printed next to each other. If a comma appears after an

		expression in the list, and the print head is at print position 56 or more, then a carriage return/line feed is executed. If the print head is before print position 56, then spaces are printed until the carriage is at the beginning of the next 14 column field (until the carriage is at column 14, 28, 42 or 56...). If there is no list of expressions to be printed, as in line 370 of the examples, then a carriage return/line feed is executed.
	410 PRINT MID$(A$,2);	*(8K Version)* String expressions may be printed.
READ	490 READ V,W	Reads data into specified variables from a DATA statement. The first piece of data read will be the first piece of data listed in the first DATA statement of the program. The second piece of data read will be the second piece listed in the first DATA statement, and so on. When all of the data have been read from the first DATA statement, the next piece of data to be read will be the first piece listed in the second DATA statement of the program. Attempting to read more data than there is in all the DATA statements in a program will cause an OD (out of data) error. In the 4K version, an SN error from a READ statement can mean the data it was attempting to read from a DATA statement was improperly formatted. In the 8K version, the line number given in the SN error will refer to the line number where the error actually is located.
REM	500 REM NOW SET V=0	Allows the programmer to put comments in his program. REM statements are not executed, but can be branched to. A REM statement is terminated by end of line, but not by a ":".
	505 REM SET V=0: V=0	In this case the V=0 will never be executed by BASIC.
	506 V=0: REM SET V=0	In this case V=0 will be executed
RESTORE	510 RESTORE	Allows the re-reading of DATA statements. After a RESTORE, the next piece of data read will be the first piece listed in the first DATA statement of the program. The second piece of data read will be the second piece listed in the first DATA statement, and so on as in a normal READ operation.

RETURN	50 RETURN	Causes a subroutine to return to the statement after the most recently executed GOSUB.
STOP	9000 STOP	Causes a program to stop execution and to enter command mode. *(8K Version)* Prints BREAK IN LINE 9000. (as per this example) CONT after a STOP branches to the statement following the STOP.
WAIT	805 WAIT I,J,K 806 WAIT I,J	*(8K Version)* This statement reads the status of input port I, exclusive OR's K with the status, and then AND's the result with J until a non-zero result is obtained. Execution of the program continues at the statement following the WAIT statement. If the WAIT statement only has two arguments, K is assumed to be zero. If you are waiting for a bit to become zero, there should be a one in the corresponding position of K. I, J and K must be =>0 and <=255.

4K INTRINSIC FUNCTIONS

ABS(X)	120 PRINT ABS(X)	Gives the absolute value of the expression X. ABS returns X if X>=0, -X otherwise.
INT(X)	140 PRINT INT(X)	Returns the largest integer less than or equal to its argument X. For example: INT(.23)=0, INT(7)=7, INT(-.1)=-1, INT(-2)= -2, INT(1.1)=1. The following would round X to D decimal places: INT(X*10↑D+.5)/10↑D
RND(X)	170 PRINT RND(X)	Generates a random number between 0 and 1. The argument X controls the generation of random numbers as follows: X<0 starts a new sequence of random numbers using X. Calling RND with the same X starts the same random number sequence. X=0 gives the last random number generated. Repeated calls to RND(0) will always return the same random number. X>0 generates a new random number between 0 and 1. Note that (B-A)*RND(1)+A will generate a random number between A & B.

SGN(X)	230 PRINT SGN(X)	Gives 1 if X>0, 0 if X=0, and -1 if X<0.
SIN(X)	190 PRINT SIN(X)	Gives the sine of the expression X. X is interpreted as being in radians. Note: COS (X)=SIN(X+3.14159/2) and that 1 Radian =180/PI degrees=57.2958 degrees; so that the sine of X degrees= SIN(X/57.2958).
SQR(X)	180 PRINT SQR(X)	Gives the square root of the argument X. An FC error will occur if X is less than zero.
TAB(I)	240 PRINT TAB(I)	Spaces to the specified print position (column) on the terminal. May be used only in PRINT statements. Zero is the leftmost column on the terminal, 71 the rightmost. If the carriage is beyond position I, then no printing is done. I must be =>0 and <=255.
USR(I)	200 PRINT USR(I)	Calls the user's machine language subroutine with the argument I. See POKE, PEEK and Appendix J.

<u>8K FUNCTIONS</u> *(Includes all those listed under 4K INTRINSIC FUNCTIONS plus the following in addition.)*

ATN(X)	210 PRINT ATN(X)	Gives the arctangent of the argument X. The result is returned in radians and ranges from -PI/2 to PI/2. (PI/2=1.5708)
COS(X)	200 PRINT COS(X)	Gives the cosine of the expression X. X is interpreted as being in radians.
EXP(X)	150 PRINT EXP(X)	Gives the constant "E" (2.71828) raised to the power X. (E↑X) The maximum argument that can be passed to EXP without overflow occuring is 87.3365.
FRE(X)	270 PRINT FRE(0)	Gives the number of memory bytes currently unused by BASIC. Memory allocated for STRING space is not included in the count returned by FRE. To find the number of free bytes in STRING space, call FRE with a STRING argument. (see FRE under STRING FUNCTIONS)
INP(I)	265 PRINT INP(I)	Gives the status of (reads a byte from) input port I. Result is =>0 and <=255.

LOG(X)	160 PRINT LOG(X)	Gives the natural (Base E) logarithm of its argument X. To obtain the Base Y logarithm of X use the formula LOG(X)/LOG(Y). Example: The base 10 (common) log of 7 = LOG(7)/ LOG(10).
PEEK	356 PRINT PEEK(I)	The PEEK function returns the contents of memory address I. The value returned will be =>0 and <=255. If I is >32767 or <0, an FC error will occur. An attempt to read a non-existent memory address will return 255. (see POKE statement)
POS(I)	260 PRINT POS(I)	Gives the current position of the terminal print head (or cursor on CRT's). The leftmost character position on the terminal is position zero and the rightmost is 71.
SPC(I)	250 PRINT SPC(I)	Prints I space (or blank) characters on the terminal. May be used only in a PRINT statement. X must be =>0 and <=255 or an FC error will result.
TAN(X)	200 PRINT TAN(X)	Gives the tangent of the expression X. X is interpreted as being in radians.

STRINGS *(8K Version Only)*

1) A string may be from 0 to 255 characters in length. All string variables end in a dollar sign ($); for example, A$, B9$, K$, HELLO$.

2) String matrices may be dimensioned exactly like numeric matrices. For instance, DIM A$(10,10) creates a string matrix of 121 elements, eleven rows by eleven columns (rows 0 to 10 and columns 0 to 10). Each string matrix element is a complete string, which can be up to 255 characters in length.

3) The total number of characters in use in strings at any time during program execution cannot execeed the amount of string space, or an OS error will result. At initialization, you should set up string space so that it can contain the maximum number of characters which can be used by strings at any one time during program execution.

NAME	EXAMPLE	PURPOSE/USE
DIM	25 DIM A$(10,10)	Allocates space for a pointer and length for each element of a string matrix. No string space is allocated. See Appendix D.

LET	27 LET A$="FOO"+V$	Assigns the value of a string expression to a string variable. LET is optional.
= > < <= >= <>		String comparison operators. Comparison is made on the basis of ASCII codes, a character at a time until a difference is found. If during the comparison of two strings, the end of one is reached, the shorter string is considered smaller. Note that "A " is greater than "A" since trailing spaces are significant.
+	30 LET Z$=R$+Q$	String concatentation. The resulting string must be less than 256 characters in length or an LS error will occur.
INPUT	40 INPUT X$	Reads a string from the user's terminal. String does not have to be quoted; but if not, leading blanks will be ignored and the string will be terminated on a "," or ":" character.
READ	50 READ X$	Reads a string from DATA statements within the program. Strings do not have to be quoted; but if they are not, they are terminated on a "," or ":" character or end of line and leading spaces are ignored. See DATA for the format of string data.
PRINT	60 PRINT X$ 70 PRINT "FOO"+A$	Prints the string expression on the user's terminal.

STRING FUNCTIONS *(8K Version Only)*

ASC(X$)	300 PRINT ASC(X$)	Returns the ASCII numeric value of the first character of the string expression X$. See Appendix K for an ASCII/number conversion table. An FC error will occur if X$ is the null string.
CHR$(I)	275 PRINT CHR$(I)	Returns a one character string whose single character is the ASCII equivalent of the value of the argument (I) which must be =>0 and <=255. See Appendix K.
FRE(X$)	272 PRINT FRE("")	When called with a string argument, FRE gives the number of free bytes in string space.
LEFT$(X$,I)	310 PRINT LEFT$(X$,I)	Gives the leftmost I characters of the string expression X$. If I<=0 or >255 an FC error occurs.

LEN(X$)	220 PRINT LEN(X$)	Gives the length of the string expression X$ in characters (bytes). Non-printing characters and blanks are counted as part of the length.
MID$(X$,I)	330 PRINT MID$(X$,I)	MID$ called with two arguments returns characters from the string expression X$ starting at character position I. If I>LEN(I$), then MID$ returns a null (zero length) string. If I<=0 or >255, an FC error occurs.
MID$(X$,I,J)	340 PRINT MID$(X$,I,J)	MID$ called with three arguments returns a string expression composed of the characters of the string expression X$ starting at the Ith character for J characters. If I>LEN(X$), MID$ returns a null string. If I or J <=0 or >255, an FC error occurs. If J specifies more characters than are left in the string, all characters from the Ith on are returned.
RIGHT$(X$,I)	320 PRINT RIGHT$(X$,I)	Gives the rightmost I characters of the string expression X$. When I<=0 or >255 an FC error will occur. If I>=LEN(X$) then RIGHT$ returns all of X$.
STR$(X)	290 PRINT STR$(X)	Gives a string which is the character representation of the numeric expression X. For instance, STR$(3.1)=" 3.1".
VAL(X$)	280 PRINT VAL(X$)	Returns the string expression X$ converted to a number. For instance, VAL("3.1")=3.1. If the first non-space character of the string is not a plus (+) or minus (-) sign, a digit or a decimal point (.) then zero will be returned.

SPECIAL CHARACTERS

CHARACTER	USE
@	Erases current line being typed, and types a carriage return/line feed. An "@" is usually a shift/P.
←	*(backarrow or underline)* Erases last character typed. If no more characters are left on the line, types a carriage return/line feed. "←" is usually a shift/O.

CARRIAGE RETURN — A carriage return must end every line typed in. Returns print head or CRT cursor to the first position (leftmost) on line. A line feed is always executed after a carriage return.

CONTROL/C — Interrupts execution of a program or a list command. Control/C has effect when a statement finishes execution, or in the case of interrupting a LIST command, when a complete line has finished printing. In both cases a return is made to BASIC's command level and OK is typed.
(8K Version) Prints "BREAK IN LINE XXXX" , where XXXX is the line number of the next statement to be executed.

: (colon) — A colon is used to separate statements on a line. Colons may be used in direct and indirect statements. The only limit on the number of statements per line is the line length. It is not possible to GOTO or GOSUB to the middle of a line.

(8K Version Only)

CONTROL/O — Typing a Control/O once causes BASIC to suppress all output until a return is made to command level, an input statement is encountered, another control/O is typed, or an error occurs.

? — Question marks are equivalent to PRINT. For instance, ? 2+2 is equivalent to PRINT 2+2. Question marks can also be used in indirect statements. 10 ? X, when listed will be typed as 10 PRINT X.

MISCELLANEOUS

1) To read in a paper tape with a program on it (8K Version), type a control/O and feed in tape. There will be no printing as the tape is read in. Type control/O again when the tape is through.
Alternatively, set nulls=0 and feed in the paper tape, and when done reset nulls to the appropriate setting for your terminal.
Each line must be followed by two rubouts, or any other non-printing character. If there are lines without line numbers (direct commands) the ALTAIR will fall behind the input coming from paper tape, so this in not recommending.

Using null in this fashion will produce a listing of your tape in the 8K version (use control/O method if you don't want a listing). The null method is the only way to read in a tape in the 4K version.

To read in a paper tape of a program in the 4K version, set the number of nulls typed on carriage return/line feed to zero by patching location 46 (octal) to be a 1. Feed in the paper tape. When the tape has finished reading, stop the CPU and repatch location 46 to be the appropriate number of null characters (usually 0, so deposit a 1). When the tape is finished, BASIC will print SN ERROR because of the "OK" at the end of the tape.

2) To punch a paper tape of a program, set the number of nulls to 3 for 110 BAUD terminals (Teletypes) and 6 for 300 BAUD terminals. Then, type LIST; but, do not type a carriage return.
Now, turn on the terminal's paper tape punch. Put the terminal on local and hold down the Repeat, Control, Shift and P keys at the same time. Stop after you have punched about a 6 to 8 inch leader of nulls. These nulls will be ignored by BASIC when the paper tape is read in. Put the terminal back on line.
Now hit carriage return. After the program has finished punching, put some trailer on the paper tape by holding down the same four keys as before, with the terminal on local. After you have punched about a six inch trailer, tear off the paper tape and save for later use as desired.

3) Restarting BASIC at location zero (by toggling STOP, Examine location 0, and RUN) will cause BASIC to return to command level and type "OK". However, typing Control/C is preferred because Control/C is guaranteed not to leave garbage on the stack and in variables, and a Control C'd program may be continued. (see CONT command)

4) The maximum line length is 72 characters** If you attempt to type too many characters into a line, a bell (ASCII 7) is executed, and the character you typed in will not be echoed. At this point you can either type backarrow to delete part of the line, or at-sign to delete thewhole line. The character you typed which caused BASIC to type the bell is not inserted in the line as it occupies the character position one beyond the end of the line.

*CLEAR	CLEAR	Deletes all variables.
	CLEAR X	*(8K Version)* Deletes all variables. When used with an argument "X", sets the amount of space to be allocated for use by string variables to the number indicated by its argument "X".
	10 CLEAR 50	*(8K Version)* Same as above; but, may be used at the beginning of a program to set the exact amount of string space needed, leaving a maximum amount of memory for the program itself. NOTE: If no argument is given, the string space is set at 200 by default. An OM error will occur if an attempt is made to allocate more string space than there is available memory.

**For inputting only.

EXTENDED BASIC

When EXTENDED BASIC is sent out, the BASIC manual will be updated to contain an extensive section about EXTENDED BASIC. Also, at this time the part of the manual relating to the 4K and 8K versions will be revised to correct any errors and explain more carefully the areas users are having trouble with. This section is here mainly to explain what EXTENDED BASIC will contain.

INTEGER VARIABLES These are stored as double byte signed quantities ranging from -32768 to +32767. They take up half as much space as normal variables and are about ten times as fast for arithmetic. They are denoted by using a percent sign (%) after the variable name. The user doesn't have to worry about conversion and can mix integers with other variable types in expressions. The speed improvement caused by using integers for loop variables, matrix indices, and as arguments to functions such as AND, OR or NOT will be substantial. An integer matrix of the same dimensions as a floating point matrix will require half as much memory.

DOUBLE-PRECISION Double-Precision variables are almost the opposite of integer variables, requiring twice as much space (8bytes per value) and taking 2 to 3 times as long to do arithmetic as single-precision variables. Double-Precision variables are denoted by using a number sign (#) after the variable name. They provide over 16 digits of accuracy. Functions like SIN, ATN and EXP will convert their arguments to single-precision, so the results of these functions will only be good to 6 digits. Negation, addition, subtraction, multiplication, division, comparision, input, output and conversion are the only routines that deal with Double-Precision values. Once again, formulas may freely mix Double-Precision values with other numeric values and conversion of the other values to Double-Precision will be done automatically.

PRINT USING Much like COBOL picture clauses or FORTRAN format statements, PRINT USING provides a BASIC user with complete control over his output format. The user can control how many digits of a number are printed, whether the number is printed in scientific notation and the placement of text in output. All of this can be done in the 8K version using string functions such as STR$ and MID$, but PRINT USING makes it much easier.

DISK I/O EXTENDED BASIC will come in two versions, disk and non-disk. There will only be a copying charge to switch from one to the other. With disk features, EXTENDED BASIC will allow the user to save and recall programs and data files from the ALTAIR FLOPPY DISK. Random access as well as sequential access will be provided. Simultaneous use of multiple data files will be allowed. Utilities will format new disks, delete files and print directories. These will be BASIC programs using special BASIC functions to get access to disk information such as file length, etc. User programs can also access these disk functions, enabling the user to write his own file access method or other special purpose disk routine. The file format can be changed to allow the use of other (non-floppy) disks. This type of modification will be done by MITS under special arrangement.

OTHER FEATURES Other nice features which will be added are:

Fancy Error Messages
An ELSE clause in IF statements
RESEQUENCE
LIST, DELETE commands with line range as arguments
Deleting Matrices in a program
TRACE ON/OFF commands to monitor program flow
EXCHANGE statement to switch variable values (this will speed up string sorts by at least a factor of two).

Other features contemplated for future release are:

A multiple user BASIC
Explicit matrix manipulation
Virtual matrices
Statement modifiers
Record I/O
Paramaterized GOSUB
Multi-argument, multi-line user defined functions with string arguments and values allowed
Compilation
Multiple USR functions
"Chaining"

EXTENDED BASIC will use about 11K of memory for its own code (10K for the non-disk version) leaving 1K free on a 12K machine. It will take almost 20 minutes to load from paper tape, 7 minutes from cassette, and less than 5 seconds to load from disk.

We welcome any suggestions concerning current features or possible additions of extra features. Just send them to the ALTAIR SOFTWARE DEPARTMENT.

In January of 1975, we stunned the computer world with the announcement of our *Altair 8800* general purpose computer that sells for $439 in kit form and $621 assembled.

Today we are annoucing the *Altair 680*.

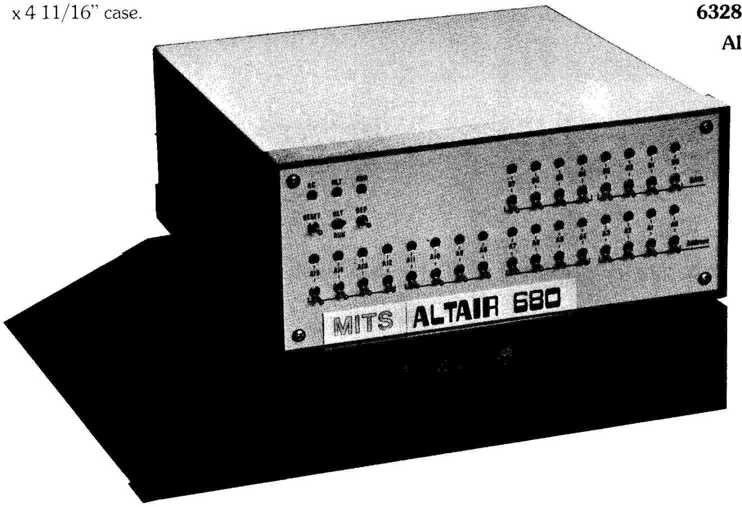
The *Altair 680* is a complete computer built around the 6800 MPU available from Motorola and AMI. It comes with power supply, front panel control board, MPU board with 1K RAM (2102 type 1024 x 1-bit chips), built-in I/O that can be configured for RS232 or 20mA current loop or 60mA current loop, and provisions for 1K of ROM or PROM—all inclosed in an 11" wide x 11" deep x 4 11/16" case.

The *Altair 680* can be utilized for many commercial and industrial products or it can be used as a development system for *Altair 680 MPU Boards*. With a cycle time of 4 microseconds, an 8-bit ALU, 16-bit addressing, the capability of directly addressing 65K of memory and a virtually unlimited number of I/O devices, the Altair 680 can be configured into any number of applications.

Software for the *Altair 680* includes a PROM monitor, assembler, debug, and editor.

MITS
"Creative Electronics"

6328 Linn NE
Albuquerque, NM 87108
505-265-7553 or
262-1951



The *Altair 680* is now selling at a *special introductory price* of $293 in kit form and $420 fully assembled. A turnkey model (complete except for front panel control board) is $240 in kit form and $280 fully assembled.

Introductory *Altair 680* prices are good until December 31, 1975. Contact MITS or your area MITS representative today!

DEVELOPMENT
$293

In January of 1975, we stunned the computer world with the announcement of our *Altair 8800* general purpose computer that sells for $439 in kit form and $621 assembled.

Today we are announcing the *Altair 680 MPU Board.*

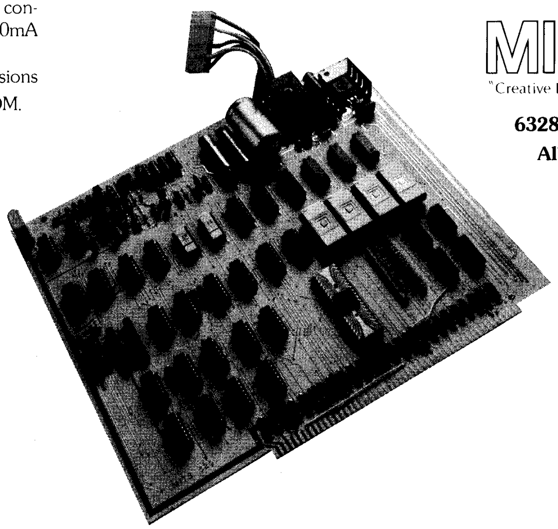
The *Altair 680 MPU Board* is a *complete computer* on a board (less power supply). In addition to the 6800 MPU available from Motorola and AMI, the *Altair 680 MPU Board* comes with 1K of RAM (2102 type 1024 x 1-bit chips), built-in I/O that can be configured for RS232 or 20mA current loop or 60mA current loop, and provisions for 1K of ROM or PROM. It measures 8 3/4" x 10 1/4".

The *Altair 6800 MPU* has 197 instructions—59 of the possible 256 codes being unassigned. Internal registers include the program counter, A and B registers, condition code flags, index register, and stack pointer. Addresses are 16-bits wide, giving the system 65,536 possible locations to use for memory locations and device registers. The idea of using addresses to refer to I/O devices greatly simplifies the internal structure of the *Altair 680.*

Software for the *Altair 680 MPU* includes a PROM monitor, assembler, debug, and editor.

MITS
"Creative Electronics"

6328 Linn NE
Albuquerque, NM 87108
505-265-7553 or
262-1951



The *Altair 680 MPU Board* is now selling at the *special introductory price* of $180 in kit form and $275 assembled.

256 x 8-bit PROM kits (ultraviolet, erasable 1702 devices) are available for $42 in kit form and $57 assembled.

Introductory *Altair 680* prices are good until December 31, 1975. Contact MITS or your area MITS representative today!

PRODUCTION
$180

ALTAIR 8800

THEORY of OPERATION MANUAL & SCHEMATICS

TABLE OF CONTENTS

INTRODUCTION 2

CPU BOARD OPERATION 3

DISPLAY/CONTROL BOARD OPERATION 5

1K STATIC MEMORY BOARD OPERATION 7

POWER SUPPLY OPERATION 9

8800 SYSTEM BUS STRUCTURE 10

SCHEMATICS

PRINTED IN U.S.A.

MITS INC.®
"Creative Electronics"
P.O. BOX 8636
ALBUQUERQUE, NEW MEXICO 87108

INTRODUCTION

The ALTAIR 8800 computer system is designed around Intel's 8080 microprocessor. The Intel 8080 is a complete central processing unit (CPU) on a single LSI chip using n-channel silicon gate MOS technology. This chip uses a separate 16-line address and 8-line bidirectional data bus configuration to greatly simplify design.

The ALTAIR 8800 uses a 100 line bus structure for data transfer between the CPU and memory or I/O devices. This bus structure contains all of the data and address lines, along with the unregulated supply voltages and all control and status signal lines. Cards other than the CPU will have control of the bus only when addressed by the CPU.

The schematic diagrams for the ALTAIR system are located at the end of this section. Specific schematics will be referred to in each particular section of the theory of operation.

On the schematics for each particular board components are identified by letters for the integrated circuits (A, B, C, etc.), and letters and numbers for the resistors and capacitors (R1, R2, C1, C2, etc.). Specific pins on the IC's are identified by numbers external to the symbol for that particular IC. The boxed numbers next to signal lines with arrows that exit or enter a given schematic refer to the bus number for those signals. Other notations on the schematics are self-explanatory.

CPU BOARD OPERATION

The 8800 CPU Board is the "heart" of the ALTAIR system. This board contains the 8080 microprocessor chip, bus drivers, the system clock, miscellaneous gating logic and the system status latch.

Refer to the following schematics for the CPU Board operation: 880-101, 880-102 and 880-103.

BUS DRIVERS

All signals entering or leaving the CPU Board are buffered using 8T97 tri-state drivers. These signals include: 16 address lines through IC's B, C and 4 gates of D; 8 data output lines through IC E and 2 gates of D; 8 data input lines through IC F and 2 gates of H.

The terms "in" and "out" are always defined with respect to the processor. Note that the 8080 bidirectional bus is split at the processor into an input and an output data bus.

The address and data out drivers (IC's B, C, D and E) can be disabled using the $\overline{\text{ADDR DSBL}}$ and $\overline{\text{DO DSBL}}$ bus signals through 2 of the gates of IC M. The data in drivers (IC F and 2 gates of H) are enabled under control of the processor through one gate of IC R, etc. (see schematic).

The 8 status output signals are buffered using 8T97's (4 gates of G and 4 gates of H). These signals are SINTA, SWO, SSTACK, SHLTA, SOUT, SMI, SINP, and SMEMR. The $\overline{\text{STATUS DSBL}}$ signal can be used to disable these outputs.

The 6 command/control output signals are also buffered using an 8T97 (IC J). These signals are SYNC, DBIN, WAIT, WR, HLDA, and INTE. The $\overline{\text{C/C DSBL}}$ signal can be used to disable these outputs.

The 4 command/control input signals (READY, HOLD, INT and RESET) are buffered using 4 gates of the 8T97 IC I. Note that the PRDY and $\overline{\text{PHOLD}}$ signals are synchronized to the leading edge of the Ø2 clock. This is required since the transition of either of these signals during the second half of Ø2 will cause the processor to enter an undefined state.

SYSTEM CLOCK

The ALTAIR 8800 system clock employs a standard TTL oscillator (IC P) with a 2.000 MHz crystal as the feedback element. The correct pulse widths and separations for the two phases are obtained using the dual single-shots (IC Q) and the delay circuit (R43 and C6). The 8080 processor requires a 12 volt swing on the clock. This is accomplished using a 7406 driver (IC N). TTL clock levels are sent to the system bus using 8T97 drivers (2 gates of IC I). The $\overline{\text{CLOC}}$ signal is sent to the system bus through one gate of the 8T97 IC G.

GATING LOGIC

The only external gating logic on the CPU Board consists of IC O (3 gates) and IC R (1 gate). If we define the output on IC O pin 13 to be G1 ENB (Data Input Enable), then:

$$\text{G1 ENB} = (\text{DBIN} + \text{HLDA}) \bullet (\text{RUN} + \text{SS}) \ *$$

Further, if we define G1 DSB = $\overline{\text{G1 ENB}}$; then the output of IC R pin 8, which is the disable input for the input data drivers, is:

$$\text{DI DSB} = \text{G1 DSB} + \text{SSW DSB}$$

G1 DSB, as can be seen from the schematic, is a processor generated signal. When the 8080 is ready for input data, it will allow G1 DSB to go low thus enabling the input data drivers.

$\overline{\text{SSW DSB}}$ is a signal generated on the Display/Control Board. This signal is used to disable the input data drivers when an input from the sense switches (device address 377_8) takes place.** This is necessary since the sense switch inputs are tied directly to the bidirectional data bus at the processor.

SYSTEM STATUS LATCH

The system status latch consists of IC K (8212). At the start of each machine cycle the processor places the system status on the bidirectional data bus. When SYNC and ϕ1 are coincident, this data is latched by IC K and remains latched for the remainder of the machine cycle.

*In these notations, + means <u>or</u>, and • means <u>and</u>.

**This address is listed in octal format. It is the same as the decimal address "255" listed in the assembly manual.

DISPLAY/CONTROL BOARD OPERATION

The 8800 Display/Control Board provides the operator with RUN/STOP and Single Step control of the processor. It also allows him to examine and modify the contents of any location in memory using the front panel switches.

Refer to the following schematics for the Display/Control Board operation: 880-104, 880-105 and 880-106.

The primary function of the D/C Board is controlling the ready line (PRDY), or a combination of the PRDY and the bidirectional data bus, to allow the above functions to be performed. Control of the PRDY line is exercised at IC O (7430). The output of IC O pin 8 (PRDY) logically appears:

$$\text{PRDY} = \text{RUN} + \text{SS} + \text{EXM} + \text{EXM NXT} \quad *$$

For the ready line to be released one of these inputs to IC O must go high. The circuitry preceding IC O will insure that only one of these signals is high at any given time.

RUN/STOP

The RUN/STOP circuit consists of an R-S flip-flop and gating to establish the stop condition. The RUN/STOP flip-flop exercises control over PRDY as described above through its Q output. The gating insures that a STOP will occur when D05, φ2 and PSYNC are true and the STOP switch is depressed.

SS

The Single Step circuit consists of a dual single shot (IC M) for debounce and the SGL STP flip-flop (R-S type). When the machine is in a stopped mode, depressing the SS switch will set the SGL STP flip-flop. (The machine must be stopped for any of the front panel switches except RESET to be active.) This allows PRDY to go high. The machine will execute one machine cycle and PSYNC, on the next cycle, will reset the SGL STP flip-flop. This will pull PRDY low, stopping the machine.

EXM

The Examine circuit consists of a dual single shot (IC L) for debounce, a 2-bit counter (IC J), the top 3 sets of 7405's on schematic 880-106 (IC's A, B, C and 2 gates of D), and some gating.

* In this notation, + means or.

When the Examine switch is depressed the counter (IC J) is started. On the first count, a jump instruction (JMP 303) is strobed directly onto the bi-directional data bus at the processor. This is accomplished by enabling 2 gates of IC C and 2 gates of IC D through the output pin 6 of one gate of IC T. These open collecter gates then pull down data lines D2, D3, D4 and D5. This puts a 303 on the data bus, which is the code for a JMP.

On the second count, the settings of switches SA 0 through SA 7 are strobed onto the data bus in a similar manner to the JMP instruction through IC A and 2 gates of B. This provides the first byte of the JMP address.

The third count strobes the settings for switches SA 8 through SA 15 onto the bus. This provides the second byte of the JMP address. The processor will then execute the JMP to the location set on the switches SA 0 through SA 15, allowing the examination of the contents of that particular memory location.

The fourth count resets the counter and pulls the EXM line low, which in turn pulls PRDY low and stops the processor.

EXM NXT

Examine Next operates in the same manner as Examine, except a NOP is strobed onto the data lines through 4 gates of IC D and 4 gates of IC E. This causes the processor to step the program counter.

DEP

The Deposit circuit places a write pulse on the MWRITE line and enables the switches SA 0 through SA 7. This causes the contents of these eight switches to be stored in the memory location currently addressed.

DEP NXT

The Deposit Next circuit simply causes a sequential operation of the EXM NXT and the DEP circuits.

1K STATIC MEMORY BOARD OPERATION

The 8800 1K Static Memory Board is designed around the Intel 8101 256 X 4 bit static RAM. Two of these RAM's provide 256 8-bit bytes of memory. The board may be configured with a minimum of two 8101's (256 bytes) and may be expanded in increments of 256 bytes by adding pairs of 8101's up to 1024 bytes.

In addition to the RAM's, the board includes 4 circuit units: Address Decoding, Processor Slow Down Circuit, Memory Protect Circuit and Buffers and Buffer Disabling Gating.

Refer to the following schematics for the 1K Static Memory Board operation: 880-107 & 880-108.

ADDRESS DECODING

The address decoding circuitry is in the lower left corner of schematic 880-107. Address bits A10 through A15 are used to select a particular 1K of memory, using IC A and IC B. By patching the inputs of IC B to either the "1" or "0" address inputs for A10 through A15 a board can be assigned any address for a 1K block from 0 to 63.

Address bits A9 and A8 are used to select a particular 256 bytes within the 1K on the board. The gating (IC D, IC F, 2 gates of IC C and 4 gates of IC E) forms a standard 2 to 4 line decoder.

PROCESSOR SLOW DOWN CIRCUIT

Since the 8101 RAM's require 850 nanoseconds for stable data on a read output, it is necessary to insert 2 wait cycles (1us) when the processor reads data from memory.

This is accomplished by IC K, 2 gates of IC N and 1 gate of IC C. This circuit causes the output from pin 8 of IC K to go low for approximately 2 clock cycles starting with PSYNC. If the 1K card has been addressed, and the processor is in a memory read cycle, two of the drivers of IC H will be enabled. This will transmit the low from IC K pin 8 to PRDY on the bus; which will in turn cause the processor to wait for 1us for the data from memory to stabilize.

MEMORY PROTECT CIRCUIT

The Memory Protect circuit consists of an R-S flip-flop (IC L) which can be set or reset by the PROT and UNPROT lines from the system bus when the card is addressed (CE is true).

When the flip-flop is set the pin 11 output of IC N is disabled and MWRITE pulses from the bus cannot get to the 8101's. A status signal, $\overline{\text{PS}}$, is returned to the front panel display via the system bus to indicate when the protect flip-flop for a particular memory card is set.

BUFFERS

The output drivers on the 1K board are 8T97 tri-state drivers (IC's J & H). Gating for enabling and disabling these drivers is accomplished with IC G and 1 gate of IC C.

The logic for this is as follows: *

$$\overline{\text{G2}} = \text{SINP} + \text{SOUT} + \overline{\text{CE}}$$

OR

$$\text{G2} = \overline{\text{SINP}} \bullet \overline{\text{SOUT}} \bullet \text{CE}$$

AND

$$\overline{\text{G1}} = \overline{\text{SMEMR}} + \overline{\text{CE}}$$

OR

$$\text{G1} = \text{SMEMR} \bullet \text{CE}$$

* In this notation + means "or" and ● means "and".

POWER SUPPLY OPERATION

The 8800 Power Supply provides two +8 and + & - 16 volts to the system bus and the display/control board. These voltages are unregulated until they reach the individual cards. Each separate card has all the necessary regulation for its own operation.

Refer to schematic 880-109 for the Power Supply operation.

Transformer T1 provides +8 volts unregulated to the system bus. This voltage is rated at 8 Amps. All boards except the display/control board use this supply for their regulated +5 volts.

Transformer T2 provides two unregulated voltages; +8 volts rated at 1 Amp for the display/control board, and +16 rated at .8 Amps to the system bus.

Transformer T3 provides the -16 volt supply rated at .3 Amps to the system bus.

All of the AC and DC voltages are wired to a terminal block for distribution to the other boards.

8800 SYSTEM BUS STRUCTURE

The 8800 system bus structure consists of 100 lines. These are arranged 50 on each side of the plug-in boards. Refer to drawing # 880-110 for the following explanation.

The following general rules apply to the 8800 system bus:

SYMBOLS: "P" prefix indicates a processor command/control signal

"S" prefix indicates a processor status signal

LOADING: All inputs to a card will be loaded with a maximum of 1 TTL low power load.

LEVELS: All bus signals except the power supply are TTL

BUS DEFINITION

No.	SYMBOL	NAME	FUNCTION
1	+8V	+8 volts	Unregulated input to 5v regulators
2	+16V	+16 volts	Positive unregulated voltage
3	XRDY	External Ready	For special applications: Pulling this line low will cause the processor to enter a WAIT state and allows the status of the normal Ready line (PRDY) to be examined
4	VI0	Vectored Interrupt Line #0	
5	VI1	Vectored Interrupt Line #1	
6	VI2	Vectored Interrupt Line #2	
7	VI3	Vectored Interrupt Line #3	
8	VI4	Vectored Interrupt Line #4	

BUS DEFINITION

No.	SYMBOL	NAME	FUNCTION
9	VI5	Vectored Interrupt Line #5	
10	VI6	Vectored Interrupt Line #6	
11	VI7	Vectored Interrupt Line #7	
12 to 17	TO BE DIFINED		
18	$\overline{\text{STA DSB}}$	$\overline{\text{STATUS DISABLE}}$	Allows the buffers for the 8 status lines to be tri-stated
19	$\overline{\text{C/C DSB}}$	$\overline{\text{COMMAND/CONTROL}}$ $\overline{\text{DISABLE}}$	Allows the buffers for the 6 output command/control lines to be tri-stated
20	UNPROT	UNPROTECT	Input to the memory protect flip-flop on a given memory board
21	SS	SINGLE STEP	Indicates that the machine is in the process of performing a single step
22	$\overline{\text{ADD DSB}}$	$\overline{\text{ADDRESS DISABLE}}$	Allows the buffers for the 16 address lines to be tri-stated
23	$\overline{\text{DO DSB}}$	$\overline{\text{DATA OUT DISABLE}}$	Allows the buffers for the 8 data output lines to be tri-stated
24	Φ2	Phase 2 Clock	
25	Φ1	Phase 1 Clock	
26	PHLDA	Hold Acknowledge	Processor command/control output signal which appears in response to the HOLD signal; indicates that the data and address bus will go to the high impedance state

BUS DEFINITION

No.	SYMBOL	NAME	FUNCTION
27	PWAIT	WAIT	Processor command/control output signal which acknowledges that the processor is in a WAIT state
28	PINTE	INTERRUPT ENABLE	Processor command/control output signal indicating interrupts are enabled: indicates the content of the CPU internal interrupt flip-flop; F-F may be set or reset by EI and DI instruction and inhibits interrupts from being accepted by the CPU if it is reset
29	A5	Address Line #5	
30	A4	Address Line #4	
31	A3	Address Line #3	
32	A15	Address Line #15	
33	A12	Address Line #12	
34	A9	Address Line #9	
35	DO1	Data Out Line #1	
36	DO0	Data Out Line #0	
37	A10	Address Line #10	
38	DO4	Data Out Line #4	
39	DO5	Data Out Line #5	
40	DO6	Data Out Line #6	
41	DI2	Data In Line #2	
42	DI3	Data In Line #3	
43	DI7	Data In Line #7	

BUS DEFINITION

No.	SYMBOL	NAME	FUNCTION
44	SM1	M1	Status output signal that indicates that the processor is in the fetch cycle for the first byte of an instruction
45	SOUT	OUT	Status output signal which indicates that the address bus contains the address of an output device and the data bus will contain the output data when $\overline{\text{PWR}}$ is active
46	SINP	INP	Status output signal which indicates that the address bus contains the address of an input device and the input data should be placed on the data bus when PDBIN is active
47	SMEMR	MEMR	Status output signal which indicates that the data bus will be used for memory read data
48	SHLTA	HLTA	Status output signal which acknowledges a HALT instruction
49	$\overline{\text{CLOCK}}$	$\overline{\text{CLOCK}}$	Inverted output of the 2MHz oscillator that generates the 2 phase clock
50	GND	GROUND	
51	+8V	+8 volts	Unregulated input to 5v regulators
52	-16V	-16 volts	Negative unregulated voltage
53	$\overline{\text{SSW DSB}}$	$\overline{\text{SENSE SWITCH DISABLE}}$	Disables the data input buffers so the input from the sense switches may be strobed onto the bidirectional data bus right at the processor
54	$\overline{\text{EXT CLR}}$	$\overline{\text{EXTERNAL CLEAR}}$	Clear signal for I/O devices (front panel switch closure to ground)

BUS DEFINITION

No.	SYMBOL	NAME	FUNCTION
55 to 67	TO BE DEFINED		
68	MWRT	MEMORY WRITE	Indicates that the current data on the Data Out Bus is to be written into the memory location currently on the address bus
69	$\overline{\text{PS}}$	$\overline{\text{PROTECT STATUS}}$	Indicates the status of the memory protect flip-flop on the memory board currently addressed
70	PROT	PROTECT	Input to the memory protect flip-flop on the memory board currently addressed
71	RUN	RUN	Indicates that the RUN/STOP flip-flop is Reset
72	PRDY	READY	Processor command/control input that controls the run state of the processor; if the line is pulled low the processor will enter a wait state until the line is released
73	$\overline{\text{PINT}}$	$\overline{\text{INTERRUPT REQUEST}}$	The processor recognizes an interrupt request on this line at the end of the current instruction or while halted. If the processor is in the HOLD state or the Interrupt Enable flip-flop is reset, it will not honor the request.
74	$\overline{\text{PHOLD}}$	$\overline{\text{HOLD}}$	Processor command/control input signal which requests the processor to enter the HOLD state; allows an external device to gain control of address and data buses as soon as the processor has completed its use of these buses for the current machine cycle

BUS DEFINITION

No.	SYMBOL	NAME	FUNCTION
75	$\overline{\text{PRESET}}$	$\overline{\text{RESET}}$	Processor command/control input; while activated the content of the program counter is cleared and the instruction register is set to 0
76	PSYNC	SYNC	Processor command/control output provides a signal to indicate the beginning of each machine cycle
77	$\overline{\text{PWR}}$	$\overline{\text{WRITE}}$	Processor command/control output used for memory write or I/O output control: data on the data bus is stable while the $\overline{\text{PWR}}$ is active
78	PDBIN	DATA BUS IN	Processor command/control output signal indicates to external circuits that the data bus is in the input mode
79	A0	Address Line #0	
80	A1	Address Line #1	
81	A2	Address Line #2	
82	A6	Address Line #6	
83	A7	Address Line #7	
84	A8	Address Line #8	
85	A13	Address Line #13	
86	A14	Address Line #14	
87	A11	Address Line #11	
88	D02	Data Out Line #2	
89	D03	Data Out Line #3	
90	D07	Data Out Line #7	
91	DI4	Data In Line #4	

BUS DEFINITION

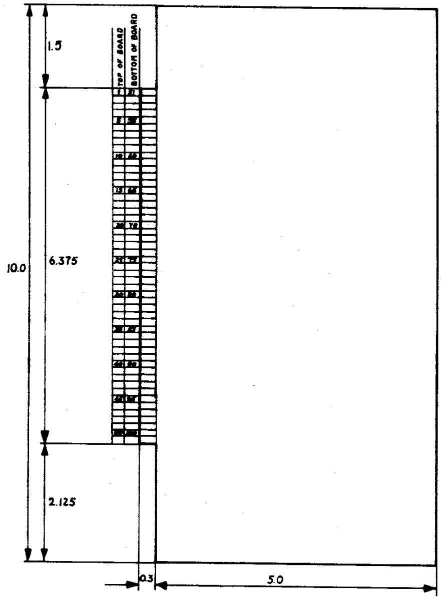
No.	SYMBOL	NAME	FUNCTION
92	DI5	Data In Line #5	
93	DI6	Data In Line #6	
94	DI1	Data In Line #1	
95	DI0	Data In Line #0	
96	SINTA	INTA	Status output signal to acknowledge signal for INTERRUPT request
97	SWO	WO	Status output signal indicates that the operation in the current machine cycle will be a WRITE memory or output function
98	SSTACK	STACK	Status output signal indicates that the address bus holds the pushdown stack address from the Stack Pointer
99	$\overline{\text{POC}}$	Power-On Clear	
100	GND	GROUND	

TOP OF BOARD

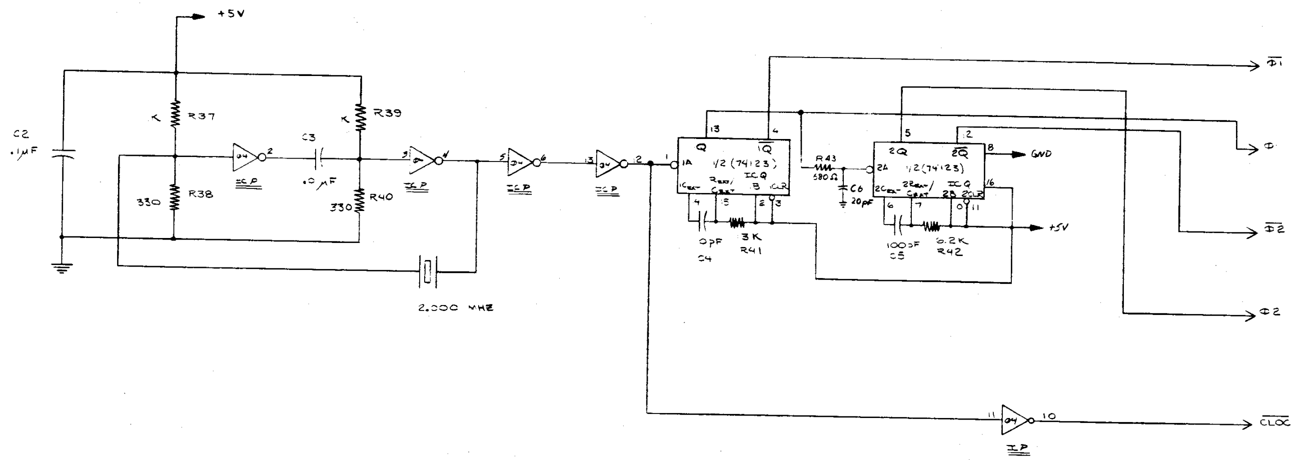
1	+5 V
2	+15 V
3	XRDY
4	VI Ø
5	VI 1
6	VI 2
7	VI 3
8	VI 4
9	VI 5
10	VI 6
11	VI 7
12	
13	
14	
15	
16	
17	
18	$\overline{\text{STAT DISABLE}}$
19	$\overline{\text{C/C DISABLE}}$
20	UNPROTECT
21	SS
22	$\overline{\text{ADDR DISBL}}$
23	$\overline{\text{DO DISBL}}$
24	Φ2
25	Φ1
26	PHLDA
27	PWAIT
28	PINTE
29	A5
30	A4
31	A3
32	A15
33	A12
34	A9
35	DO1
36	DOØ
37	A10
38	DO4
39	DO5
40	DO6
41	DI2
42	DI3
43	DI7
44	SM1
45	SOUT
46	SINP
47	SMEMR
48	SHLTA
49	$\overline{\text{CLOCK}}$ (2 MHz)
50	GND

BOTTOM OF BOARD

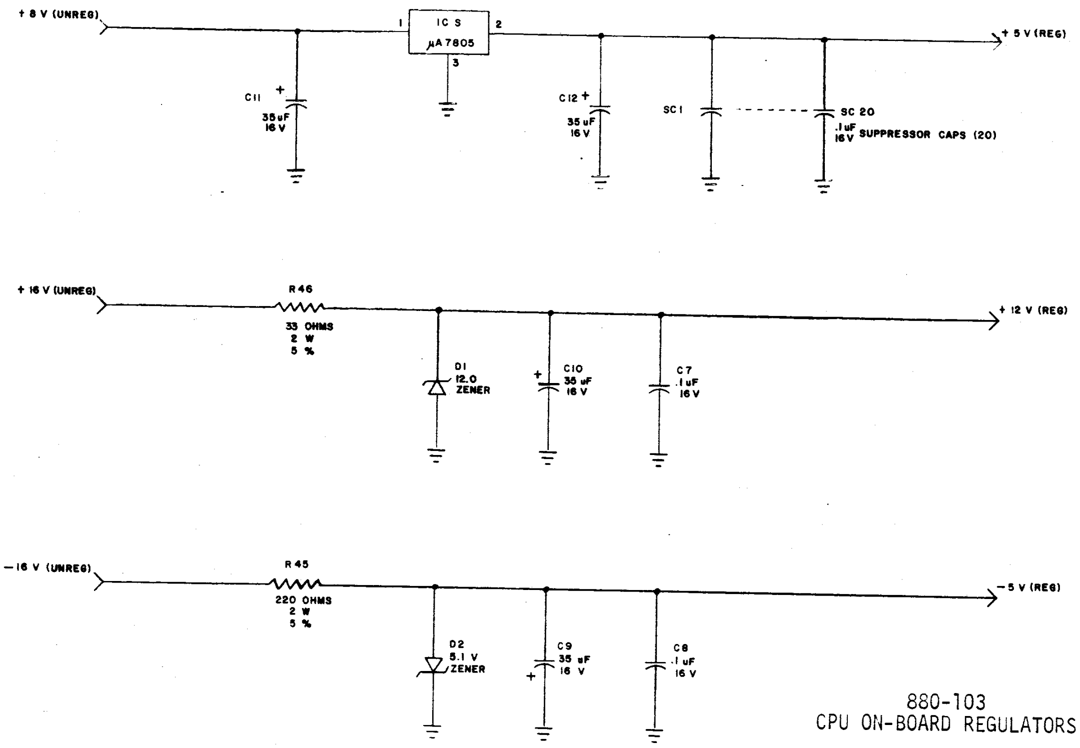
51	+ 5 V	A
52	- 15 V	B
53	$\overline{\text{SSW DSB}}$	C
54	$\overline{\text{EXT CLR}}$	D
55		E
56		F
57		H
58		J
59		K
60		L
61		M
62		N
63		P
64		R
65		S
66		T
67		U
68	MWRITE	V
69	$\overline{\text{PS}}$	W
70	PROTECT	X
71	RUN	Y
72	PRDY	Z
73	$\overline{\text{PINT}}$	a
74	$\overline{\text{PHOLD}}$	b
75	$\overline{\text{PRESET}}$	c
76	PSYNC	d
77	$\overline{\text{PWR}}$	e
78	PDBIN	f
79	A Ø	h
80	A 1	j
81	A 2	k
82	A 6	l
83	A 7	m
84	A 8	n
85	A 13	p
86	A 14	r
87	A 11	s
88	DO 2	t
89	DO 3	u
90	DO 7	v
91	DI 4	w
92	DI 5	x
93	DI 6	y
94	DI 1	z
95	DI Ø	AA
96	SINTA	AB
97	$\overline{\text{SWO}}$	AC
98	SSTACK	AD
99	$\overline{\text{POC}}$	AE
100	GND	AF



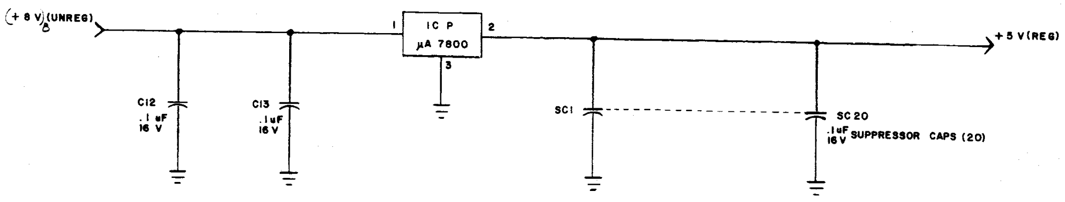
880-110
SYSTEM BUS STRUCTURE



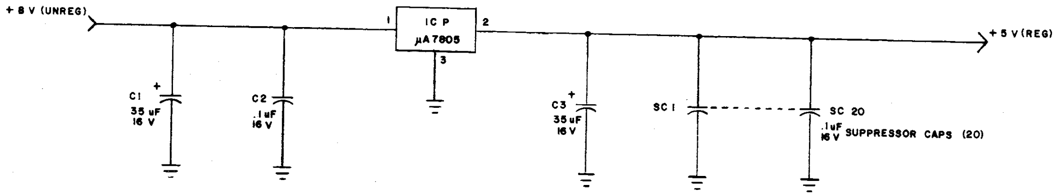
880-102
SYSTEM CLOCK



880-103
CPU ON-BOARD REGULATORS

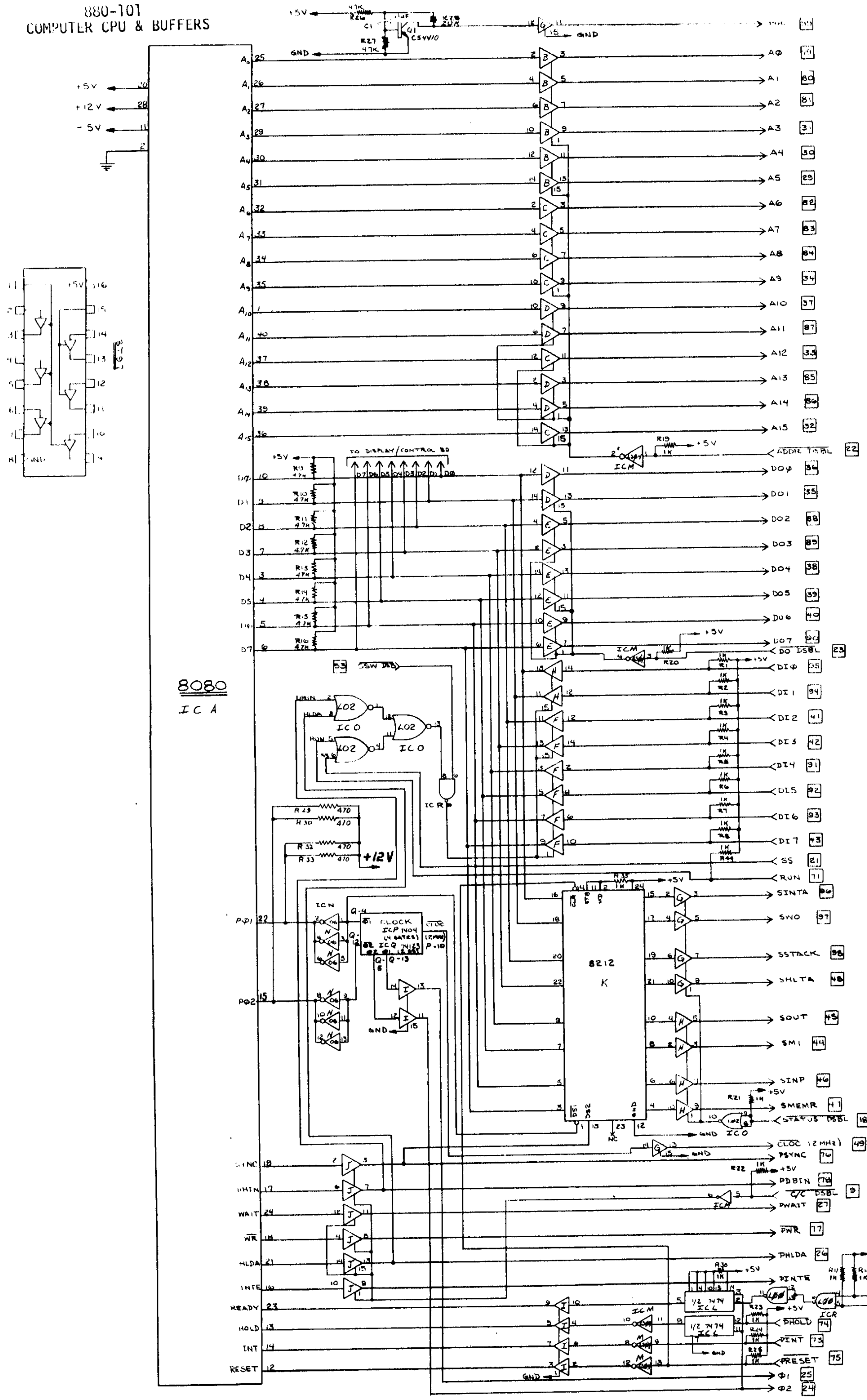


880-104
DISPLAY/CONTROL ON-BOARD REGULATOR

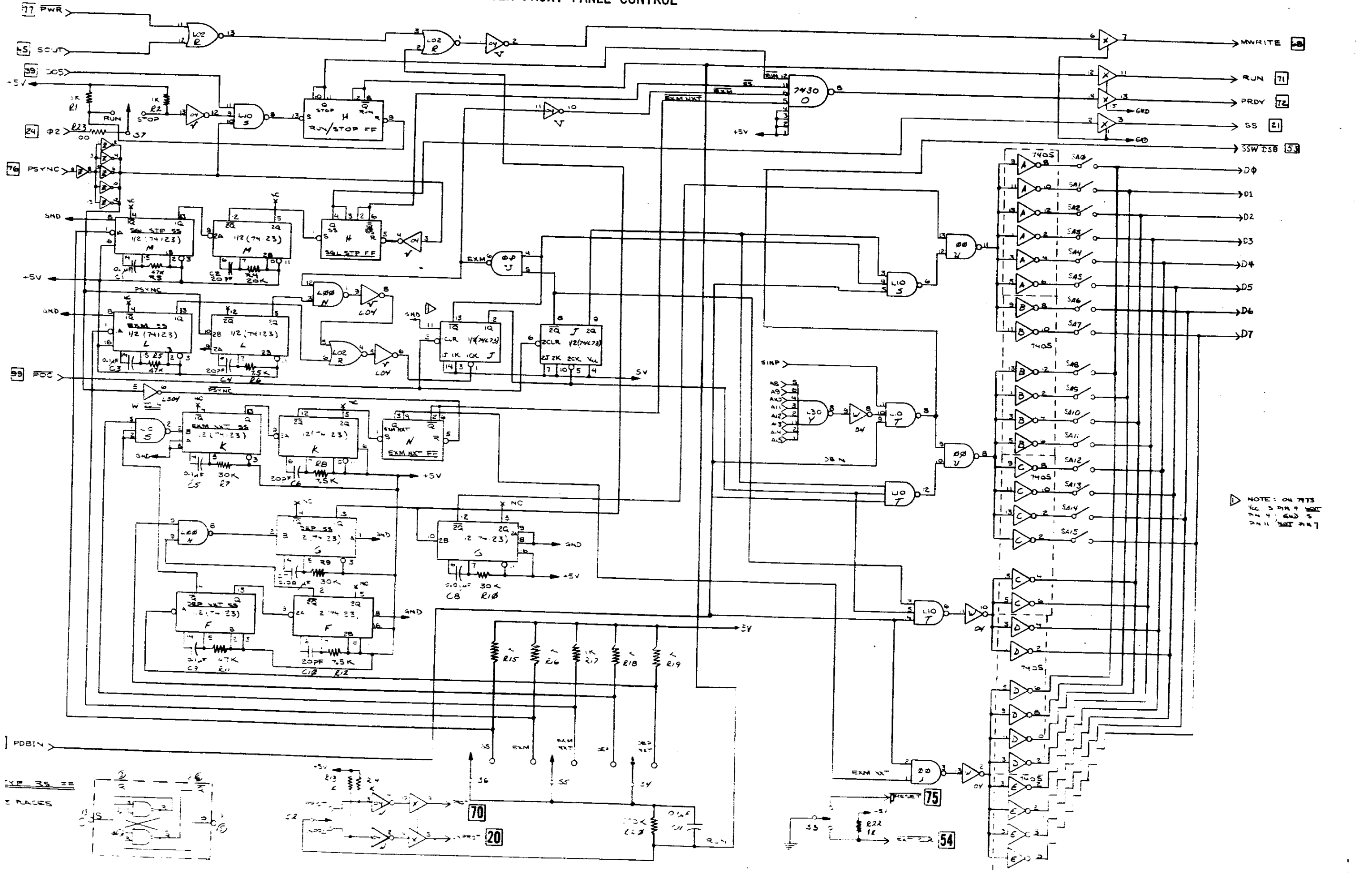


880-108
1K STATIC MEMORY ON-BOARD REGULATOR

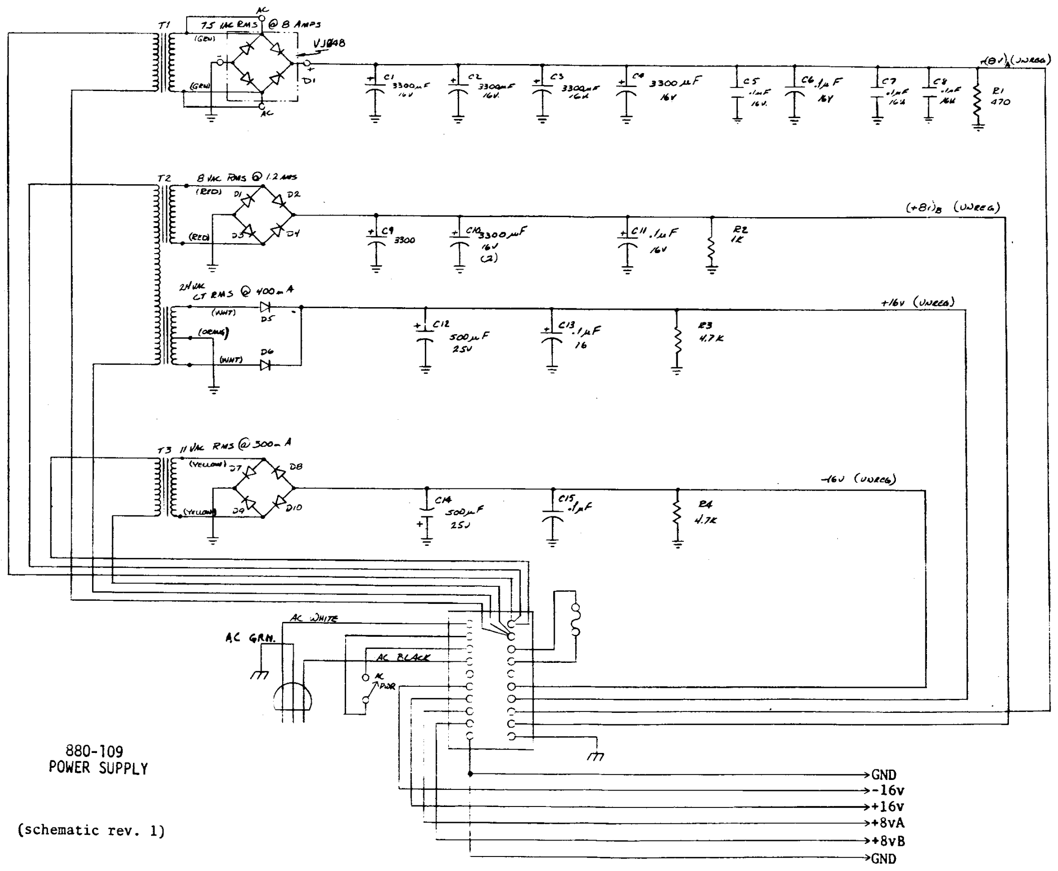
880-101
COMPUTER CPU & BUFFERS



880-106
COMPUTER FRONT PANEL CONTROL

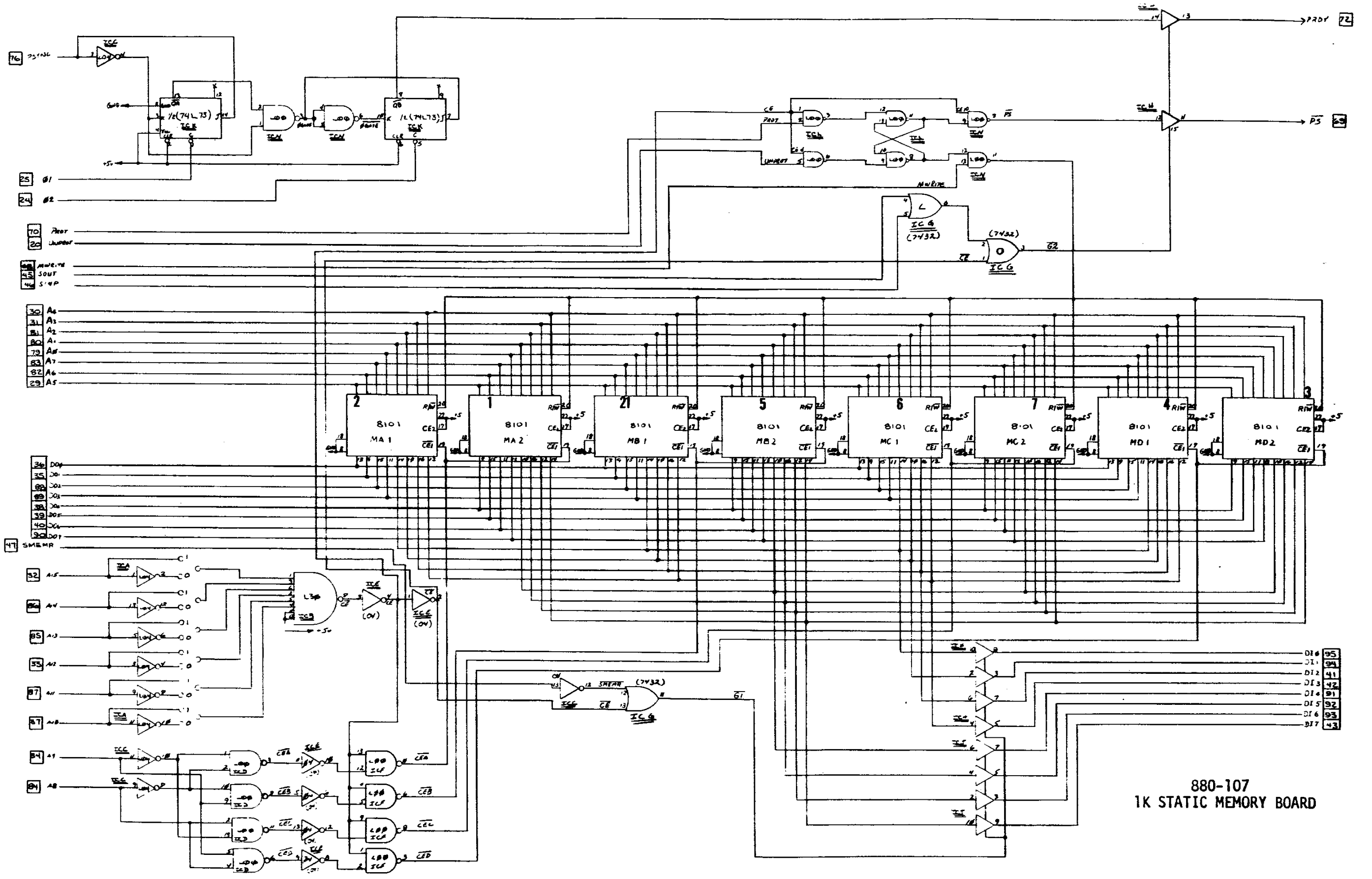


NOTE: ON 7473 Vcc 5 PIN 4 NOT PIN 4, GND 5 PIN 11 NOT PIN 7



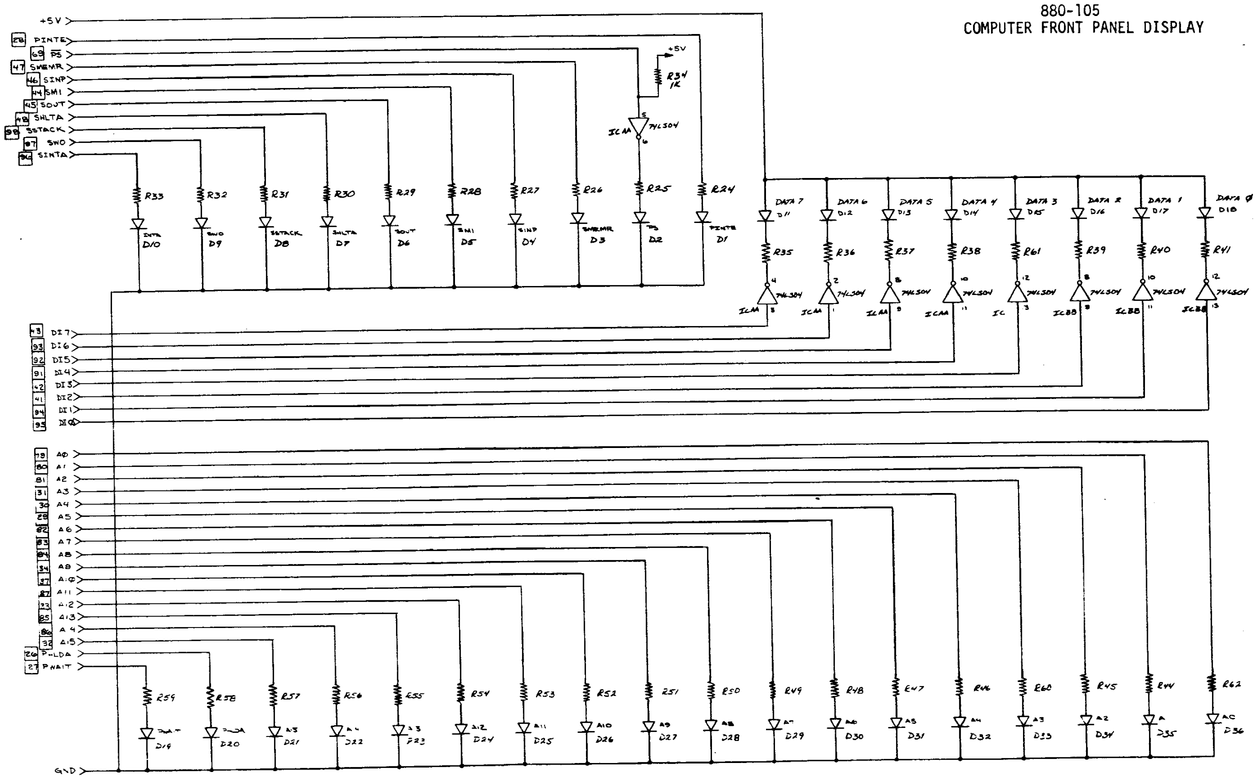
880-109
POWER SUPPLY

(schematic rev. 1)



880-107
1K STATIC MEMORY BOARD

880-105
COMPUTER FRONT PANEL DISPLAY



October 2, 1975

ARROWHEAD TIPS

8800 COMPUTER

Page 11 (D/C Board): Capacitor C7 has been changed several times; you may find change notices referring to various stages of this change process, as well as extra parts. C7 was changed from .001 MFD to .0047 MFD, then to .0068 MFD. Now, the absolute last final ultimate change (as of August 10th!) makes C7 = .01 MFD and changes C8 from .01 to .1 MFD.

Page 14 (D/C Board): In connecting the AC switch wires to the board, use heat-shrink tubing to protect the stripped wire as follows: cut 1/2" of clear heat-shrink tubing and slide it onto the wire, well past the stripped end; then solder the wire onto the board; finally slide the tubing into place and shrink it with the heat of a match or soldering iron. You may prefer to cut the lands around the AC switch and solder the wires directly to the switch pins.

Page 18 (D/C Board): Don't bolt the printed-circuit board to the sub-panel; the switches will hold it fine. The switches come with extra mounting nuts and extra guide washers; you can safely throw away all this extra hardware - you don't need it. Mount the switches as shown with only one nut each for best results.

8800 Tips cont'd

Page 21 (D/C Board): Installing the LED's is slick if you leave one of the leads unsoldered until all the LED's are in place and aligned. You might like to line them up flush with the front dress-panel for higher contrast (so that no light falls on them). After you've adjusted the position, recheck the polarity of all the LED's before soldering the second lead into place.

Pages 40-46 (1K Memory): Until May, only the 1K memory board was available, and most systems were ordered with 256 words of memory on one of these boards. Now that the 2K and 4K memories are available, it isn't sensible to require some 1K boards in every system, but the instructions are still embedded in the CPU manual. CPU kits don't have any memory in them.

Page 50 (P/S Board): The bridge rectifier seems to cause more than its share of problems. Be sure the leads are clean - several of us have found that solder won't wet the leads, and it's a mess to try to clean the partially-soldered assembly. Run the leads through some alcohol and/or steel wool before installing the bridge. The spacer-and-washer arrangement on Page 51 is a jig to get the bridge flat at the right position; it will later be bolted directly to the chassis.

Page 52 (P/S Board): You'll laugh, but I got capacitor C14 in backwards; nobody else has made this mistake. Tell us about your experiences with the kit, and we'll publish them, with or without your name, as you please. Your reward: A replacement fuse (1 amp slo-blow).

Page 58 (Chassis): MITS doesn't ask you to cut wires to close tolerances. If you follow the instructions here without trimming transformer leads, and use all the #20 wire, you'll have long loops of slack. This is good for allowing boards, etc. to be moved without breaking wires, but you may want to install the terminal lugs after consulting the wiring diagrams on Pages 59 and 62.

Page 66 (Motherboard): The way the instructions spell it out, you'll have a slack loop of cable. Hold everything in place on the chassis to see the actual length required. We are enclosing a sorted list of wires to help you check your progress. Using masking tape, group the wires in decades after protecting them with a cable clamp. Then, install them on the motherboard, by decades; 50's, 60's, 70's, 20's, 80's, 30's, 90's, and finally, 40's.

Install the cable clamps by bolting them to the printed circuit board only. If you put screws through the sub-panel, then the dress-panel won't fit flush against it and you won't be able to screw the chassis into the case!

8800 Tips cont'd

Page 68 (Expander Boards): The card guides are maybe sorta optional. They look nice, but they really aren't required to hold the boards in place - the edge connectors are plenty strong for that.

Page 74 (CPU Chip): Many people advocate postponing installation of the CPU chip until after the regulator and zener diodes on the CPU have been tested.

Page 77 (Nameplate): This beauty gets a lot of criticism: "Mine was off-color, kinda pinkish." That's a sticky plastic cover to protect it until you've installed it. Peel the covering off afterward. The white lettering on the dress-panel can be chipped off by hard use. If you decide to protect it with clear acrylic spray, use a matte-finish product. Ours looks funny with a glossy krylon finish.

(Checkout): You can see your machine run, even without any memory. When the machine tries to execute an instruction at a non-existent memory location, the value returned is 11 111 111 (377 in octal). 377 is a restart instruction, used to jump to an interrupt (at location 70) handling routine. It pushes the 2-word program counter onto the stack before taking the jump. At location 70, the lack of memory will yield another 377, so the restart will repeat every 11 clock cycles, or 181,818 times per second. This will cycle the stack pointer through the 16-bit address space about 5 times per second. So, if you run with no memory, the address light A15 will blink at that speed.

Page 39 (CPU Board): The wafer connector is about 5% too large to fit the board (or you might say the holes in the board are too close). To prevent the connector plug from arching, clamp it down flat while soldering it on. If you can't clamp it, then try cutting it with a hacksaw into two 4-pin connectors.

Checkout): After turning the computer on you should reset it. To reset the computer you have to hold the stop switch raised while raising the reset switch. Release the reset switch first. (No, we don't know why, but it's traditional!)

DATA/CONTROL BOARD CONNECTIONS TO SYSTEM BUS

ORGANIZED BY DECADE

0's	10's	20's	30's	40's
		20	30	41
		21	31	42
		24	32	43
		26	33	44
		27	34	45
		28	37	46 (1!)
		29	39	47
				48

50's	60's	70's	80's	90's
53	68	70	80	91
54	69	71	81	92
		72	82	93
		75	83	94
		76	84	95
		77	85	96
		78	86	97
		79	87	98
				99

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

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

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

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

Revision O only

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

[illegible] VALUE

[illegible] Van Haslen

[illegible] percentage of the people [illegible] had trouble with their [illegible] have shown this symptom. "All [illegible] lites on at all times". [illegible] is a list of the most common [illegible]:

A. Check the mother board for shorts using VOM.

B. Insure that all regulated voltages are OK.

C. Make sure the memory is install- [illegible] the address strapping is [illegible]. Be sure that the connector from the front panel for the data lines is on the CPU.

D. Are all the data lines high on the D/C board? If they aren't, check the continuity of the CPU connector.

E. Check the timing of 01 &02.

F. Check the drivers J&H on the static memory board. These tri-state drivers need a low on pins 1&15 to pass a signal. If 1&15 are high the outputs of these drivers will be high all the time.

G. See if line 68 on the bus is high all the time. If it is, back-track thru the logic. Most probable cause is IC6 on the front panel being bad or a short or solder bridge in the vicinity of IC6.

H. Check IC's U&T on D/C board to insure the signal levels are correct,i.e.; no 1.5 volt levels as opposed to ≤.5V or ≥4+VDC.

(c) Slide about 1" of heat shrink tubing (3/32 to 7/64 diam.) over the AC switch wires. Solder these wires directly to the AC switch terminals. Slide the heat shrink tubing down so it covers the switch terminals and uninsulated ends of the switch wires. Heat the tubing once it has been properly positioned to keep it in place.

(4) CLOCK SPECS (CPU Board)

(a) Phase 1 pulse width (measured at 90% points), 60 nano-seconds minimum.

(b) Phase 2 pulse width (measured at 90% points), 220 nano-seconds minimum.

(c) Delay from leading edge (90%) of Phase one to leading edge (10%) of Phase 2. 130 nana-seconds minimum.

(d) Delay from trailing edge (10%) of Phase 2 to leading edge (10%) of phase 1. 70 nano-seconds minimum.

We have discovered that many Signetic 2604 4K RAM's found on some of our 4K Dynamic Memory Boards do not meet the required specifications for access time and refresh period. They are identified on the package as S 2604. If you have an Altair 4K Board that does not work properly

8800 MOD

(1) DEPOSIT PROBLEM (D/C board)
SYMPTOMS: Machine won't deposit at all or deposits all ones when using the front panel deposit switch.
FIX: Change the timing capacitors for the deposit single shot (IC G on the Display/Control board). Change C7 to 0.01MF, C8 to 0.1 MF.

(2) 12 VOLT ZENER (CPU board)
SYMPTOMS: Zener running very hot.
FIX: Change R46 on CPU board to 43 ohms (Use either a 1W or 2W resistor). This should be done only if there are four or less cards on the bus. If there are more than four cards R46 should be 33 ohms.

(3) AC SWITCH (Display/Control board).
PROBLEM: The tracks on the D/C board which connect the AC switch terminals to the pads where the switch wires are connected to the board have 115VAC on them. If these are inadvertently shorted to other tracks on the board several IC's are wiped out.
FIX:(a) remove the AC switch lines from the D/C board.
(b) Cut the tracks leading from the pads where the switch is mounted to the pads where the AC switch wires attach to the D/C board. Cut the tracks as close as possible to the switch pads so there will be no length of tra with 115VAC on it.

The Status Signal $\overline{WO}$ is incorrectly labelled on the front panel as WO.

There is a problem with the protect operation on the 4K RAM board. Pin 10 of ICT is tied to +5V and should be tied to ground. The easiest way to fix this is to lift Pin 10 of ICT and run a short jumper from the lifted Pin to Pin 11 of ICT which is tied to ground.