ALTAIR

BASIC

REFERENCE MANUAL

MITS ALTAIR BASIC

REFERENCE MANUAL

Table of Contents:

PRINTED IN U.S.A.

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

Introduction

Before a computer can perform any useful function, it must be "told" what to do. Unfortunately, at this time, computers are not capable of understanding English or any other "human" language. This is primarily because our languages are rich with ambiguities and implied meanings. The computer must be told precise instructions and the exact sequence of operations to be performed in order to accomplish any specific task. Therefore, in order to facilitate human communication with a computer, programming languages have been developed.

ALTAIR BASIC* is a programming language both easily understood and simple to use. It serves as an excellent "tool" for applications in areas such as business, science and education. With only a few hours of using BASIC, you will find that you can already write programs with an ease that few other computer languages can duplicate.

Originally developed at Dartmouth University, BASIC language has found wide acceptance in the computer field. Although it is one of the simplest computer languages to use, it is very powerful. BASIC uses a small set of common English words as its "commands". Designed specifically as an "interactive" language, you can give a command such as "PRINT 2 + 2", and ALTAIR BASIC will immediately reply with "4". It isn't necessary to submit a card deck with your program on it and then wait hours for the results. Instead the full power of the ALTAIR is "at your fingertips".

Generally, if the computer does not solve a particular problem the way you expected it to, there is a "Bug" or error in your program, or else there is an error in the data which the program used to calculate its answer. If you encounter any errors in BASIC itself, please let us know and we'll see that it's corrected. Write a letter to us containing the following information:

1) System Configuration

2) Version of BASIC

3) A detailed description of the error
   Include all pertinent information such as a listing of the program in which the error occurred, the data placed into the program and BASIC's printout.

All of the information listed above will be necessary in order to properly evaluate the problem and correct it as quickly as possible. We wish to maintain as high a level of quality as possible with all of our ALTAIR software.

* BASIC *is a registered trademark of Dartmouth University.*

We hope that you enjoy ALTAIR BASIC, and are successful in using it to solve all of your programming needs.

In order to maintain a maximum quality level in our documentation, we will be continuously revising this manual. If you have any suggestions on how we can improve it, please let us know.

If you are already familiar with BASIC programming, the following section may be skipped. Turn directly to the Reference Material on page 22.

NOTE: MITS ALTAIR BASIC is available under license or purchase agreements. Copying or otherwise distributing MITS software outside the terms of such an agreement may be a violation of copyright laws or the agreement itself.

If any immediate problems with MITS software are encountered, feel free to give us a call at (505) 265-7553. The Software Department is at Ext. 3; and the joint authors of the ALTAIR BASIC Interpreter, Bill Gates, Paul Allen and Monte Davidoff, will be glad to assist you.

GETTING STARTED WITH BASIC

MITS
"Creative Electronics"

This section is not intended to be a detailed course in BASIC programming. It will, however, serve as an excellent introduction for those of you unfamiliar with the language.

The text here will introduce the primary concepts and uses of BASIC enough to get you started writing programs. For further reading suggestions, see Appendix M.

If your ALTAIR does not have BASIC loaded and running, follow the procedures in Appendices A & B to bring it up.

We recommend that you try each example in this section as it is presented. This will enhance your "feel" for BASIC and how it is used.

Once your I/O device has typed " OK ", you are ready to use ALTAIR BASIC.

NOTE: All commands to ALTAIR BASIC should end with a carriage return. The carriage return tells BASIC that you have finished typing the command. If you make a typing error, type a back-arrow (←), usually shift/O, or an underline to eliminate the last character. Repeated use of " ← " will eliminate previous characters. An at-sign (@) will eliminate the entire line that you are typing.

Now, try typing in the following:

```
PRINT 10-4  (end with carriage return)
```

ALTAIR BASIC will immediately print:

```
 6

OK
```

The print statement you typed in was executed as soon as you hit the carriage return key. BASIC evaluated the formula after the "PRINT" and then typed out its value, in this case 6.

Now try typing in this:

```
PRINT 1/2,3*10          ("*" means multiply, "/" means divide)
```

ALTAIR BASIC will print:

```
 .5            30
```

As you can see, ALTAIR BASIC can do division and multiplication as well as subtraction. Note how a " , " (comma) was used in the print command to print two values instead of just one. The comma divides the 72 character line into 5 columns, each 14 characters wide. The last two of the positions on the line are not used. The result is a " , " causes BASIC to skip to the next 14 column field on the terminal, where the value 30 was printed.

Commands such as the "PRINT" statements you have just typed in are called Direct Commands. There is another type of command called an Indirect Command. Every Indirect command begins with a Line Number. A Line Number is any integer from 0 to 65529.

Try typing in the following lines:

```
10 PRINT 2+3
20 PRINT 2-3
```

A sequence of Indirect Commands is called a "Program". Instead of executing indirect statements immediately, ALTAIR BASIC saves Indirect Commands in the ALTAIR's memory. When you type in RUN , BASIC will execute the lowest numbered indirect statement that has been typed in first, then the next highest, etc. for as many as were typed in.

Suppose we type in RUN now:

```
RUN
```

ALTAIR BASIC will type out:

```
 5
-1

OK
```

In the example above, we typed in line 10 first and line 20 second. However, it makes no difference in what order you type in indirect statements. BASIC always puts them into correct numerical order according to the Line Number.

If we want a listing of the complete program currently in memory, we type in LIST . Type this in:

```
LIST
```

ALTAIR BASIC will reply with:

```
10 PRINT 2+3
20 PRINT 2-3
OK
```

Sometimes it is desirable to delete a line of a program altogether. This is accomplished by typing the Line Number of the line we wish to delete, followed only by a carriage return.

Type in the following:

```
10
LIST
```

ALTAIR BASIC will reply with:

```
20 PRINT 2-3
OK
```

We have now deleted line 10 from the program. There is no way to get it back. To insert a new line 10, just type in 10 followed by the statement we want BASIC to execute.

Type in the following:

```
10 PRINT 2*3
LIST
```

ALTAIR BASIC will reply with:

```
10 PRINT 2*3
20 PRINT 2-3
OK
```

There is an easier way to replace line 10 than deleting it and then inserting a new line. You can do this by just typing the new line 10 and hitting the carriage return. BASIC throws away the old line 10 and replaces it with the new one.

Type in the following:

```
10 PRINT 3-3
LIST
```

ALTAIR BASIC will reply with:

```
10 PRINT 3-3
20 PRINT 2-3
OK
```

It is not recommended that lines be numbered consecutively. It may become necessary to insert a new line between two existing lines. An increment of 10 between line numbers is generally sufficient.

If you want to erase the complete program currently stored in memory, type in " NEW ". If you are finished running one program and are about to read in a new one, be sure to type in " NEW " first. This should be done in order to prevent a mixture of the old and new programs.

Type in the following:

```
NEW
```

ALTAIR BASIC will reply with:

```
OK
```

Now type in:

```
LIST
```

ALTAIR BASIC will reply with:

```
OK
```

Often it is desirable to include text along with answers that are printed out, in order to explain the meaning of the numbers.

Type in the following:

```
PRINT "ONE THIRD IS EQUAL TO",1/3
```

ALTAIR BASIC will reply with:

```
ONE THIRD IS EQUAL TO        .333333

OK
```

As explained earlier, including a " , " in a print statement causes it to space over to the next fourteen column field before the value following the " , " is printed.

If we use a " ; " instead of a comma, the value next will be printed immediately following the previous value.

NOTE: Numbers are always printed with at least one trailing space. Any text to be printed is always to be enclosed in double quotes.

Try the following examples:

A)
```
PRINT "ONE THIRD IS EQUAL TO";1/3
ONE THIRD IS EQUAL TO .333333

OK
```

B)
```
PRINT 1,2,3
 1             2             3

OK
```

C)
```
PRINT 1;2;3
 1  2  3

OK
```

D)
```
PRINT -1;2;-3
-1  2 -3
```

OK

We will digress for a moment to explain the format of numbers in ALTAIR BASIC. Numbers are stored internally to over six digits of accuracy. When a number is printed, only six digits are shown. Every number may also have an exponent (a power of ten scaling factor).

The largest number that may be represented in ALTAIR BASIC is $1.70141*10^{38}$, while the smallest positive number is $2.93874*10^{-39}$.

When a number is printed, the following rules are used to determine the exact format:

1) If the number is negative, a minus sign (-) is printed. If the number is positive, a space is printed.

2) If the absolute value of the number is an integer in the range 0 to 999999, it is printed as an integer.

3) If the absolute value of the number is greater than or equal to .1 and less than or equal to 999999, it is printed in fixed point notation, with no exponent.

4) If the number does not fall under categories 2 or 3, scientific notation is used.

Scientific notation is formatted as follows: SX.XXXXXESTT .
(each X being some integer 0 to 9)

The leading "S" is the sign of the number, a space for a positive number and a " - " for a negative one. One non-zero digit is printed before the decimal point. This is followed by the decimal point and then the other five digits of the mantissa. An "E" is then printed (for exponent), followed by the sign (S) of the exponent; then the two digits (TT) of the exponent itself. Leading zeroes are never printed; i.e. the digit before the decimal is never zero. Also, trailing zeroes are never printed. If there is only one digit to print after all trailing zeroes are suppressed, no decimal point is printed. The exponent sign will be " + " for positive and " - " for negative. Two digits of the exponent are always printed; that is zeroes are not suppressed in the exponent field. The value of any number expressed thus is the number to the left of the "E" times 10 raised to the power of the number to the right of the "E".

No matter what format is used, a space is always printed following a number. The 8K version of BASIC checks to see if the entire number will fit on the current line. If not, a carriage return/line feed is executed before printing the number.

The following are examples of various numbers and the output format ALTAIR BASIC will place them into:

NUMBER	OUTPUT FORMAT
+1	1
-1	-1
6523	6523
-23.460	-23.46
1E20	1E+20
-12.3456E-7	-1.23456E-06
1.234567E-10	1.23457E-10
1000000	1E+06
999999	999999
.1	.1
.01	1E-02
.000123	1.23E-04

A number input from the terminal or a numeric constant used in a BASIC program may have as many digits as desired, up to the maximum length of a line (72 characters). However, only the first 7 digits are significant, and the seventh digit is rounded up.

```
PRINT 1.2345678901234567890
 1.23457

OK
```

The following is an example of a program that reads a value from the terminal and uses that value to calculate and print a result:

```
10 INPUT R
20 PRINT 3.14159*R*R
RUN
? 10
 314.159

OK
```

Here's what's happening. When BASIC encounters the input statement, it types a question mark (?) on the terminal and then waits for you to type in a number. When you do (in the above example 10 was typed), execution continues with the next statement in the program after the variable (R) has been set (in this case to 10). In the above example, line 20 would now be executed. When the formula after the PRINT statement is evaluated, the value 10 is substituted for the variable R each time R appears in the formula. Therefore, the formula becomes 3.14159*10*10, or 314.159.

If you haven't already guessed, what the program above actually does is to calculate the area of a circle with the radius "R".

If we wanted to calculate the area of various circles, we could keep re-running the program over each time for each successive circle. But, there's an easier way to do it simply by adding another line to the program as follows:

```
30 GOTO 10
RUN
? 10
 314.159
? 3
 28.2743
? 4.7
 69.3977
?

OK
```

By putting a " GOTO " statement on the end of our program, we have caused it to go back to line 10 after it prints each answer for the successive circles. This could have gone on indefinitely, but we decided to stop after calculating the area for three circles. This was accomplished by typing a carriage return to the input statement (thus a blank line).

NOTE: Typing a carriage return to an input statement in the 4K version of BASIC will cause a SN error (see Reference Material).

The letter "R" in the program we just used was termed a "variable". A variable name can be any alphabetic character and may be followed by any alphanumeric character.

In the 4K version of BASIC, the second character must be numeric or omitted. In the 8K version of BASIC, any alphanumeric characters after the first two are ignored. An alphanumeric character is any letter (A-Z) or any number (0-9).

Below are some examples of legal and illegal variable names:

LEGAL	ILLEGAL
IN 4K VERSION	
A	% (1st character must be alphabetic)
Z1	Z1A (variable name too long)
	QR (2nd character must be numeric)
IN 8K VERSION	
TP	TO (variable names cannot be reserved words)
PSTG$	
COUNT	RGOTO (variable names cannot contain reserved words)

The words used as BASIC statements are "reserved" for this specific purpose. You cannot use these words as variable names or inside of any variable name. For instance, "FEND" would be illegal because "END" is a reserved word.

The following is a list of the reserved words in ALTAIR BASIC:

4K RESERVED WORDS

ABS CLEAR DATA DIM END FOR GOSUB GOTO IF INPUT

INT LET LIST NEW NEXT PRINT READ REM RESTORE

RETURN RND RUN SGN SIN SQR STEP STOP TAB(THEN

TO USR

8K RESERVED WORDS INCLUDE ALL THOSE ABOVE, AND IN ADDITION

ASC AND ATN CHR$ CLOAD CONT COS CSAVE DEF EXP

FN FRE INP LEFT$ LEN LOG MID$ NULL ON OR NOT

OUT PEEK POKE POS RIGHT$ SPC(STR$ TAN VAL WAIT

Remember, in the 4K version of BASIC variable names are only a letter or a letter followed by a number. Therefore, there is no possibility of a conflict with a reserved word.

Besides having values assigned to variables with an input statement, you can also set the value of a variable with a LET or assignment statement.

Try the following examples:

```
A=5

OK

PRINT A,A*2
 5            10

OK
LET Z=7

OK

PRINT Z, Z-A
 7            2

OK
```

As can be seen from the examples, the "LET" is optional in an assignment statement.

BASIC "remembers" the values that have been assigned to variables using this type of statement. This "remembering" process uses space in the ALTAIR's memory to store the data.

The values of variables are thrown away and the space in memory used to store them is released when one of four things occur:

1) A new line is typed into the program or an old line is deleted

2) A CLEAR command is typed in

3) A RUN command is typed in

4) NEW is typed in

Another important fact is that if a variable is encountered in a formula before it is assigned a value, it is automatically assigned the value zero. Zero is then substituted as the value of the variable in the particular formula. Try the example below:

```
PRINT Q,Q+2,Q*2
 0            2            0

OK
```

Another statement is the REM statement. REM is short for remark. This statement is used to insert comments or notes into a program. When BASIC encounters a REM statement the rest of the line is ignored.

This serves mainly as an aid for the programmer himself, and serves no useful function as far as the operation of the program in solving a particular problem.

Suppose we wanted to write a program to check if a number is zero or not. With the statements we've gone over so far this could not be done. What is needed is a statement which can be used to conditionally branch to another statement. The "IF-THEN" statement does just that.

Try typing in the following program: (remember, type NEW first)

```
10 INPUT B
20 IF B=0 THEN 50
30 PRINT "NON-ZERO"
40 GOTO 10
50 PRINT "ZERO"
60 GOTO 10
```

When this program is typed into the ALTAIR and run, it will ask for a value for B. Type any value you wish in. The ALTAIR will then come to the "IF" statement. Between the "IF" and the "THEN" portion of the statement there are two expressions separated by a relation.

A relation is one of the following six symbols:

RELATION	MEANING
=	EQUAL TO
>	GREATER THAN
<	LESS THAN
<>	NOT EQUAL TO
<=	LESS THAN OR EQUAL TO
=>	GREATER THAN OR EQUAL TO

The IF statement is either true or false, depending upon whether the two expressions satisfy the relation or not. For example, in the program we just did, if 0 was typed in for B the IF statement would be true because 0=0. In this case, since the number after the THEN is 50, execution of the program would continue at line 50. Therefore, "ZERO" would be printed and then the program would jump back to line 10 (because of the GOTO statement in line 60).

Suppose a 1 was typed in for B. Since 1=0 is false, the IF statement would be false and the program would continue execution with the next line. Therefore, "NON-ZERO" would be printed and the GOTO in line 40 would send the program back to line 10.

Now try the following program for comparing two numbers:

```
10 INPUT A,B
20 IF A<=B THEN 50
30 PRINT "A IS BIGGER"
40 GOTO 10
50 IF A<B THEN 80
60 PRINT "THEY ARE THE SAME"
70 GOTO 10
80 PRINT "B IS BIGGER"
90 GOTO 10
```

When this program is run, line 10 will input two numbers from the terminal. At line 20, if A is greater than B, A<=B will be false. This will cause the next statement to be executed, printing "A IS BIGGER" and then line 40 sends the computer back to line 10 to begin again.

At line 20, if A has the same value as B, A<=B is true so we go to line 50. At line 50, since A has the same value as B, A<B is false; therefore, we go to the following statement and print "THEY ARE THE SAME". Then line 70 sends us back to the beginning again.

At line 20, if A is smaller than B, A<=B is true so we go to line 50. At line 50, A<B will be true so we then go to line 80. "B IS BIGGER" is then printed and again we go back to the beginning.

Try running the last two programs several times. It may make it easier to understand if you try writing your own program at this time using the IF-THEN statement. Actually trying programs of your own is the quickest and easiest way to understand how BASIC works. Remember, to stop these programs just give a carriage return to the input statement.

One advantage of computers is their ability to perform repetitive tasks. Let's take a closer look and see how this works.

Suppose we want a table of square roots from 1 to 10. The BASIC function for square root is "SQR"; the form being SQR(X), X being the number you wish the square root calculated from. We could write the program as follows:

```
10 PRINT 1,SQR(1)
20 PRINT 2,SQR(2)
30 PRINT 3,SQR(3)
40 PRINT 4,SQR(4)
50 PRINT 5,SQR(5)
60 PRINT 6,SQR(6)
70 PRINT 7,SQR(7)
80 PRINT 8,SQR(8)
90 PRINT 9,SQR(9)
100 PRINT 10,SQR(10)
```

This program will do the job; however, it is terribly inefficient. We can improve the program tremendously by using the IF statement just introduced as follows:

```
10 N=1
20 PRINT N,SQR(N)
30 N=N+1
40 IF N<=10 THEN 20
```

When this program is run, its output will look exactly like that of the 10 statement program above it. Let's look at how it works.

At line 10 we have a LET statement which sets the value of the variable N at 1. At line 20 we print N and the square root of N using its current value. It thus becomes 20 PRINT 1,SQR(1), and this calculation is printed out.

At line 30 we use what will appear at first to be a rather unusual LET statement. Mathematically, the statement N=N+1 is nonsense. However, the important thing to remember is that in a LET statement, the symbol " = " does not signify equality. In this case " = " means "to be replaced with". All the statement does is to take the current value of N and add 1 to it. Thus, after the first time through line 30, N becomes 2.

At line 40, since N now equals 2, N<=10 is true so the THEN portion branches us back to line 20, with N now at a value of 2.

The overall result is that lines 20 through 40 are repeated, each time adding 1 to the value of N. When N finally equals 10 at line 20, the next line will increment it to 11. This results in a false statement at line 40, and since there are no further statements to the program it stops.

This technique is referred to as "looping" or "iteration". Since it is used quite extensively in programming, there are special BASIC statements for using it. We can show these with the following program.

```
10 FOR N=1 TO 10
20 PRINT N,SQR(N)
30 NEXT N
```

The output of the program listed above will be exactly the same as the previous two programs.

At line 10, N is set to equal 1. Line 20 causes the value of N and the square root of N to be printed. At line 30 we see a new type of statement. The "NEXT N" statement causes one to be added to N, and then if N<=10 we go back to the statement following the "FOR" statement. The overall operation then is the same as with the previous program.

Notice that the variable following the "FOR" is exactly the same as the variable after the "NEXT". There is nothing special about the N in this case. Any variable could be used, as long as they are the same in both the "FOR" and the "NEXT" statements. For instance, "Z1" could be substituted everywhere there is an "N" in the above program and it would function exactly the same.

Suppose we wanted to print a table of square roots from 10 to 20, only counting by two's. The following program would perform this task:

```
10 N=10
20 PRINT N,SQR(N)
30 N=N+2
40 IF N<=20 THEN 20
```

Note the similar structure between this program and the one listed on page 12 for printing square roots for the numbers 1 to 10. This program can also be written using the "FOR" loop just introduced.

```
10 FOR N=10 TO 20 STEP 2
20 PRINT N,SQR(N)
30 NEXT N
```

Notice that the only major difference between this program and the previous one using "FOR" loops is the addition of the "STEP 2" clause. This tells BASIC to add 2 to N each time, instead of 1 as in the previous program. If no "STEP" is given in a "FOR" statement, BASIC assumes that one is to be added each time. The "STEP" can be followed by any expression.

Suppose we wanted to count backwards from 10 to 1. A program for doing this would be as follows:

```
10 I=10
20 PRINT I
30 I=I-1
40 IF I>=1 THEN 20
```

Notice that we are now checking to see that I is greater than or equal to the final value. The reason is that we are now counting by a negative number. In the previous examples it was the opposite, so we were checking for a variable less than or equal to the final value.

The "STEP" statement previously shown can also be used with negative numbers to accomplish this same purpose. This can be done using the same format as in the other program, as follows:

```
10 FOR I=10 TO 1 STEP -1
20 PRINT I
30 NEXT I
```

"FOR" loops can also be "nested". An example of this procedure follows:

```
10 FOR I=1 TO 5
20 FOR J=1 TO 3
30 PRINT I,J
40 NEXT J
50 NEXT I
```

Notice that the "NEXT J" comes before the "NEXT I". This is because the J-loop is inside of the I-loop. The following program is incorrect; run it and see what happens.

```
10 FOR I=1 TO 5
20 FOR J=1 TO 3
30 PRINT I,J
40 NEXT I
50 NEXT J
```

It does not work because when the "NEXT I" is encountered, all knowledge of the J-loop is lost. This happens because the J-loop is "inside" of the I-loop.

It is often convenient to be able to select any element in a table of numbers. BASIC allows this to be done through the use of matrices.

A matrix is a table of numbers. The name of this table, called the matrix name, is any legal variable name, "A" for example. The matrix name "A" is distinct and separate from the simple variable "A", and you could use both in the same program.

To select an element of the table, we subscript "A" : that is to select the I'th element, we enclose I in parenthesis "(I)" and then follow "A" by this subscript. Therefore, "A(I)" is the I'th element in the matrix "A".

NOTE: In this section of the manual we will be concerned with one-dimensional matrices only. (See Reference Material)

"A(I)" is only one element of matrix A, and BASIC must be told how much space to allocate for the entire matrix.

This is done with a "DIM" statement, using the format "DIM A(15)". In this case, we have reserved space for the matrix index "I" to go from 0 to 15. Matrix subscripts always start at 0; therefore, in the above example, we have allowed for 16 numbers in matrix A.

If "A(I)" is used in a program before it has been dimensioned, BASIC reserves space for 11 elements (0 through 10).

As an example of how matrices are used, try the following program to sort a list of 8 numbers with you picking the numbers to be sorted.

```
10 DIM A(8)
20 FOR I=1 TO 8
30 INPUT A(I)
50 NEXT I
70 F=0
80 FOR I=1 TO 7
90 IF A(I)<=A(I+1) THEN 140
100 T=A(I)
110 A(I)= A(I+1)
120 A(I+1)=T
130 F=1
140 NEXT I
150 IF F=1 THEN 70
160 FOR I=1 TO 8
170 PRINT A(I),
180 NEXT I
```

When line 10 is executed, BASIC sets aside space for 9 numeric values, A(0) through A(8). Lines 20 through 50 get the unsorted list from the user. The sorting itself is done by going through the list of numbers and upon finding any two that are not in order, we switch them. "F" is used to indicate if any switches were done. If any were done, line 150 tells BASIC to go back and check some more.

If we did not switch any numbers, or after they are all in order, lines 160 through 180 will print out the sorted list. Note that a subscript can be any expression.

Another useful pair of statements are "GOSUB" and "RETURN". If you have a program that performs the same action in several different places, you could duplicate the same statements for the action in each place within the program.

The "GOSUB"-"RETURN" statements can be used to avoid this duplication. When a "GOSUB" is encountered, BASIC branches to the line whose number follows the "GOSUB". However, BASIC remembers where it was in the program before it branched. When the "RETURN" statement is encountered, BASIC goes back to the first statement following the last "GOSUB" that was executed. Observe the following program.

```
10 PRINT "WHAT IS THE NUMBER";
30 GOSUB 100
40 T=N
50 PRINT "WHAT IS THE SECOND NUMBER";
70 GOSUB 100
80 PRINT "THE SUM OF THE TWO NUMBERS IS",T+N
90 STOP
100 INPUT N
```

```
110 IF N = INT(N) THEN 140
120 PRINT "SORRY, NUMBER MUST BE AN INTEGER.  TRY AGAIN."
130 GOTO 100
140 RETURN
```

What this program does is to ask for two numbers which must be integers, and then prints the sum of the two. The subroutine in this program is lines 100 to 130. The subroutine asks for a number, and if it is not an integer, asks for a number again. It will continue to ask until an integer value is typed in.

The main program prints " WHAT IS THE NUMBER ", and then calls the subroutine to get the value of the number into N. When the subroutine returns (to line 40), the value input is saved in the variable T. This is done so that when the subroutine is called a second time, the value of the first number will not be lost.

" WHAT IS THE SECOND NUMBER " is then printed, and the second value is entered when the subroutine is again called.

When the subroutine returns the second time, " THE SUM OF THE TWO NUMBERS IS " is printed, followed by the value of their sum. T contains the value of the first number that was entered and N contains the value of the second number.

The next statement in the program is a "STOP" statement. This causes the program to stop execution at line 90. If the "STOP" statement was not included in the program, we would "fall into" the subroutine at line 100. This is undesirable because we would be asked to input another number. If we did, the subroutine would try to return; and since there was no "GOSUB" which called the subroutine, an RG error would occur. Each "GOSUB" executed in a program should have a matching "RETURN" executed later, and the opposite applies, i.e. a "RETURN" should be encountered only if it is part of a subroutine which has been called by a "GOSUB".

Either "STOP" or "END" can be used to separate a program from its subroutines. In the 4K version of BASIC, there is no difference between the "STOP" and the "END". In the 8K version, "STOP" will print a message saying at what line the "STOP" was encountered.

Suppose you had to enter numbers to your program that didn't change each time the program was run, but you would like it to be easy to change them if necessary. BASIC contains special statements for this purpose, called the "READ" and "DATA" statements.

Consider the following program:

```
10 PRINT "GUESS A NUMBER";
20 INPUT G
30 READ D
40 IF D=-999999 THEN 90
50 IF D<>G THEN 30
60 PRINT "YOU ARE CORRECT"
70 END
90 PRINT "BAD GUESS, TRY AGAIN."
95 RESTORE
```

```
100 GOTO 10
110 DATA 1,393,-39,28,391,-8,0,3.14,90
120 DATA 89,5,10,15,-34,-999999
```

This is what happens when this program is run. When the "READ" statement is encountered, the effect is the same as an INPUT statement. But, instead of getting a number from the terminal, a number is read from the "DATA" statements.

The first time a number is needed for a READ, the first number in the first DATA statement is returned. The second time one is needed, the second number in the first DATA statement is returned. When the entire contents of the first DATA statement have been read in this manner, the second DATA statement will then be used. DATA is always read sequentially in this manner, and there may be any number of DATA statements in your program.

The purpose of this program is to play a little game in which you try to guess one of the numbers contained in the DATA statements. For each guess that is typed in, we read through all of the numbers in the DATA statements until we find one that matches the guess.

If more values are read than there are numbers in the DATA statements, an out of data (OD) error occurs. That is why in line 40 we check to see if -999999 was read. This is not one of the numbers to be matched, but is used as a flag to indicate that all of the data (possible correct guesses) has been read. Therefore, if -999999 was read, we know that the guess given was incorrect.

Before going back to line 10 for another guess, we need to make the READ's begin with the first piece of data again. This is the function of the "RESTORE". After the RESTORE is encountered, the next piece of data read will be the first piece in the first DATA statement again.

DATA statements may be placed anywhere within the program. Only READ statements make use of the DATA statements in a program, and any other time they are encountered during program execution they will be ignored.

THE FOLLOWING INFORMATION APPLIES TO THE 8K VERSION OF BASIC ONLY

A list of characters is referred to as a "String". MITS, ALTAIR, and THIS IS A TEST are all strings. Like numeric variables, string variables can be assigned specific values. String variables are distinguished from numeric variables by a "$" after the variable name.

For example, try the following:

```
A$="ALTAIR 8800"

OK
PRINT A$
ALTAIR 8800

OK
```

In this example, we set the string variable A$ to the string value "ALTAIR 8800". Note that we also enclosed the character string to be assigned to A$ in quotes.

Now that we have set A$ to a string value, we can find out what the length of this value is (the number of characters it contains). We do this as follows:

```
PRINT LEN(A$),LEN("MITS")
 11            4

OK
```

The "LEN" function returns an integer equal to the number of characters in a string.

The number of characters in a string expression may range from 0 to 255. A string which contains 0 characters is called the "NULL" string. Before a string variable is set to a value in the program, it is initialized to the null string. Printing a null string on the terminal will cause no characters to be printed, and the print head or cursor will not be advanced to the next column. Try the following:

```
PRINT LEN(Q$);Q$;3
 0  3

OK
```

Another way to create the null string is: Q$=""

Setting a string variable to the null string can be used to free up the string space used by a non-null string variable.

Often it is desirable to access parts of a string and manipulate them. Now that we have set A$ to "ALTAIR 8800", we might want to print out only the first six characters of A$. We would do so like this:

```
PRINT LEFT$(A$,6)
ALTAIR

OK
```

"LEFT$" is a string function which returns a string composed of the leftmost N characters of its string argument. Here's another example:

```
FOR N=1 TO LEN(A$):PRINT LEFT$(A$,N):NEXT N
A
AL
ALT
ALTA
ALTAI
ALTAIR
ALTAIR
ALTAIR 8
ALTAIR 88
```

```
ALTAIR 8800
ALTAIR 8800

OK
```

Since A$ has 11 characters, this loop will be executed with N=1,2, 3,...,10,11. The first time through only the first chatacter will be printed, the second time the first two characters will be printed, etc.

There is another string function called "RIGHT$" which returns the right N characters from a string expression. Try substituting "RIGHT$" for "LEFT$" in the previous example and see what happens.

There is also a string function which allows us to take characters from the middle of a string. Try the following:

```
FOR N=1 TO LEN(A$):PRINT MID$(A$,N):NEXT N
ALTAIR 8800
LTAIR 8800
TAIR 8800
AIR 8800
IR 8800
R 8800
 8800
8800
800
00
0

OK
```

"MID$" returns a string starting at the Nth position of A$ to the end (last character) of A$. The first position of the string is position 1 and the last possible position of a string is position 255.

Very often it is desirable to extract only the Nth character from a string. This can be done by calling MID$ with three arguments. The third argument specifies the number of characters to return.

For example:

```
FOR N=1 TO LEN(A$):PRINT MID$(A$,N,1),MID$(A$,N,2):NEXT N
A             AL
L             LT
T             TA
A             AI
I             IR
R             R
               8
8             88
8             80
0             00
0             0

OK
```

See the Reference Material for more details on the workings of "LEFT$", "RIGHT$" and "MID$".

Strings may also be concatenated (put or joined together) through the use of the "+" operator. Try the following:

```
B$="MITS"+" "+A$

OK
PRINT B$
MITS ALTAIR 8800

OK
```

Concatenation is especially useful if you wish to take a string apart and then put it back together with slight modifications. For instance:

```
C$=LEFT$(B$,4)+"-"+MID$(B$,6,6)+"-"+RIGHT$(B$,4)

OK
PRINT C$
MITS-ALTAIR-8800

OK
```

Sometimes it is desirable to convert a number to its string representation and vice-versa. "VAL" and "STR$" perform these functions. Try the following:

```
STRING$="567.8"

OK
PRINT VAL(STRING$)
 567.8

OK
STRING$=STR$(3.1415)

OK
PRINT STRING$,LEFT$(STRING$,5)
 3.1415        3.14

OK
```

"STR$" can be used to perform formatted I/O on numbers. You can convert a number to a string and then use LEFT$, RIGHT$, MID$ and concatenation to reformat the number as desired.

"STR$" can also be used to conveniently find out how many print columns a number will take. For example:

```
PRINT LEN(STR$(3.157))
 6
```

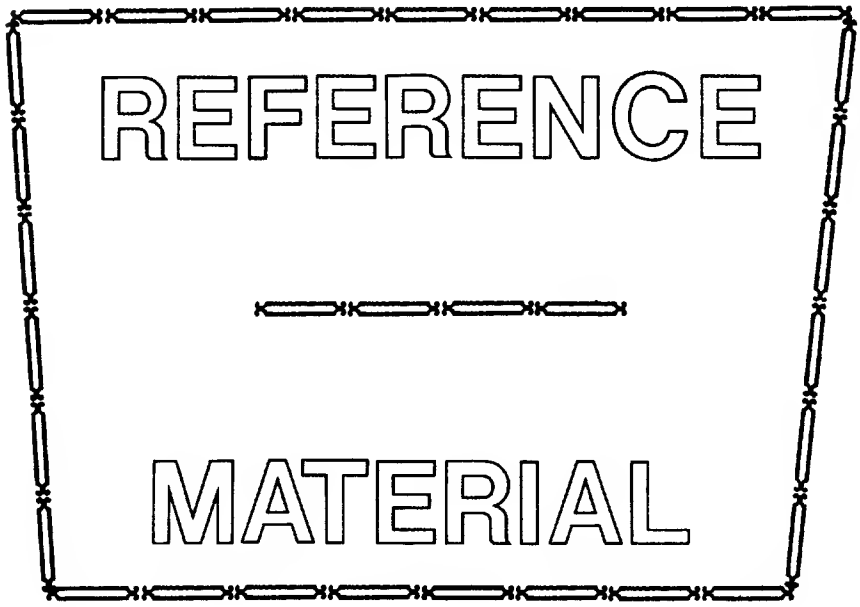
```
OK
```

If you have an application where a user is typing in a question such as "WHAT IS THE VOLUME OF A CYLINDER OF RADIUS 5.36 FEET, OF HEIGHT 5.1 FEET?" you can use "VAL" to extract the numeric values 5.36 and 5.1 from the question. For further functions "CHR$" and "ASC" see Appendix K.

The following program sorts a list of string data and prints out the sorted list. This program is very similar to the one given earlier for sorting a numeric list.

```
100 DIM A$(15):REM ALLOCATE SPACE FOR STRING MATRIX
110 FOR I=1 TO 15:READ A$(I):NEXT I:REM READ IN STRINGS
120 F=0:I=1:REM SET EXCHANGE FLAG TO ZERO AND SUBSCRIPT TO 1
130 IF A$(I)<=A$(I+1) THEN 180:REM DON'T EXCHANGE IF ELEMENTS
    IN ORDER
140 T$=A$(I+1):REM USE T$ TO SAVE A$(I+1)
150 A$(I+1)=A$(I):REM EXCHANGE TWO CONSECUTIVE ELEMENTS
160 A$(I)=T$
170 F=1:REM FLAG THAT WE EXCHANGED TWO ELEMENTS
180 I=I+1: IF I<15 GOTO 130
185 REM ONCE WE HAVE MADE A PASS THRU ALL ELEMENTS, CHECK
187 REM TO SEE IF WE EXCHANGED ANY.  IF NOT, DONE SORTING.
190 IF F THEN 120:REM EQUIVALENT TO IF F<>0 THEN 120
200 FOR I=1 TO 15:PRINT A$(I):NEXT I: REM PRINT SORTED LIST
210 REM STRING DATA FOLLOWS
220 DATA APPLE,DOG,CAT,MITS,ALTAIR,RANDOM
230 DATA MONDAY,"***ANSWER***"," FOO"
240 DATA COMPUTER,   FOO,ELP,MILWAUKEE,SEATTLE,ALBUQUERQUE
```

BASIC LANGUAGE



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 <u>not</u> 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
20 PRINT "INTRUDER ALERT"
```

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 .

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

may range from 0 to 10 (eleven elements).

```
117 A(8)=4
```

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
345 NEXT
```

Marks the end of a FOR loop. *(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.

8K FUNCTIONS *(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 exeeed 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

APPENDICES

APPENDIX A

HOW TO LOAD BASIC

When the ALTAIR is first turned on, there is random garbage in its memory. BASIC is supplied on a paper tape or audio cassette. Somehow the information on the paper tape or cassette must be transfered into the computer. Programs that perform this type of information transfer are called loaders.

Since initially there is nothing of use in memory; you must toggle in, using the switches on the front panel, a 20 instruction bootstrap loader. This loader will then load BASIC.

To load BASIC follow these steps:

1) Turn the ALTAIR on.

2) Raise the STOP switch and RESET switch simultaneously.

3) Turn your terminal (such as a Teletype) to LINE.

Because the instructions must be toggled in via the switches on the front panel, it is rather inconvenient to specify the positions of each switch as "up" or "down". Therefore, the switches are arranged in groups of 3 as indicated by the broken lines below switches 0 through 15. To specify the positions of each switch, we use the numbers 0 through 7 as shown below:

3 SWITCH GROUP

LEFTMOST	MIDDLE	RIGHTMOST	OCTAL NUMBER
Down	Down	Down	0
Down	Down	Up	1
Down	Up	Down	2
Down	Up	Up	3
Up	Down	Down	4
Up	Down	Up	5
Up	Up	Down	6
Up	Up	Up	7

So, to put the octal number 315 in switches 0 through 7, the switches would have the following positions:

7	6	5	4	3	2	1	0	← SWITCH
UP	UP	DOWN	DOWN	UP	UP	DOWN	UP	← POSITION
3			1			5		← OCTAL NO.

Note that switches 8 through 15 were not used. Switches 0 through 7 correspond to the switches labeled DATA on the front panel. A memory address would use all 16 switches.

The following program is the bootstrap loader for users loading from paper tape, and not using a REV 0 Serial I/O Board.

OCTAL ADDRESS	OCTAL DATA	
000	041	
001	175	
002	037	(for 8K; for 4K use 017)
003	061	
004	022	
005	000	
006	333	
007	000	
010	017	
011	330	
012	333	
013	001	
014	275	
015	310	
016	055	
017	167	
020	300	
021	351	
022	003	
023	000	

The following 21 byte bootstrap loader is for users loading from a paper tape and using a REV 0 Serial I/O Board on which the update changing the flag bits has not been made. If the update has been made, use the above bootstrap loader.

OCTAL ADDRESS	OCTAL DATA	
000	041	
001	175	
002	037	(for 8K; for 4K use 017)
003	061	
004	023	
005	000	
006	333	
007	000	
010	346	
011	040	
012	310	
013	333	
014	001	
015	275	
016	310	
017	055	
020	167	

OCTAL ADDRESS (cont.)	OCTAL DATA
021	300
022	351
023	003
024	000

The following bootstrap loader is for users with BASIC supplied on an audio cassette.

OCTAL ADDRESS	OCTAL DATA	
000	041	
001	256 ~~125~~	
002	037	(for 8K; for 4K use 017)
003	061	
004	022	
005	000	
006	333	
007	006	
010	017	
011	330	
012	333	
013	007	
014	275	
015	310	
016	055	
017	167	
020	300	
021	351	
022	003	
023	000	

To load a bootstrap loader:

1) Put switches 0 through 15 in the down position.

2) Raise EXAMINE.

3) Put 041 (data for address 000) in switches 0 through 7.

4) Raise DEPOSIT.

5) Put the data for the next address in switches 0 through 7.

6) Depress DEPOSIT NEXT.

7) Repeat steps 5 & 6 until the entire loader is toggled in.

8) Put switches 0 through 15 in the down position.

9) Raise EXAMINE.

10) Check that lights D0 through D7 correspond with the data that should

be in address 000. A light on means the switch was up, a light off means the switch was down. So for address 000, lights D1 through D4 and lights D6 & D7 should be off, and lights D0 and D5 should be on.

If the correct value is there, go to step 13. If the value is wrong, continue with step 11.

11) Put the correct value in switches 0 through 7.

12) Raise DEPOSIT.

13) Depress EXAMINE NEXT.

14) Repeat steps 10 through 13, checking to see that the correct data is in each corresponding address for the entire loader.

15) If you encountered any mistakes while checking the loader, go back now and re-check the whole program to be sure it is corrected.

16) Put the tape of BASIC into the tape reader. Be sure the tape is positioned at the beginning of the leader. The leader is the section of tape at the beginning with 6 out of the 8 holes punched.

If you are loading from audio cassette, put the cassette in the recorder. Be sure the tape is fully rewound.

17) Put switches 0 through 15 in the down position.

18) Raise EXAMINE.

19) If you have connected to your terminal a REV 0 Serial I/O Board on which the update changing the flag bits has not been made, raise switch 14; if you are loading from an audio cassette, raise switch 15 also.

If you have a REV 0 Serial I/O Board which has been updated, or have a REV 1 I/O Board, switch 14 should remain down and switch 15 should be raised only if you are loading from audio cassette.

20) Turn on the tape reader and then depress RUN. Be sure RUN is depressed while the reader is still on the leader. Do not depress run before turning on the reader, since this may cause the tape to be read incorrectly.

If you are loading from a cassette, turn the cassette recorder to Play. Wait 15 seconds and then depress RUN.

21) Wait for the tape to be read in. This should take about 12 minutes for 8K BASIC and 6 minutes for 4K BASIC. It takes about 4 minutes to load 8K BASIC from cassette, and about 2 minutes for 4K BASIC.

Do not move the switches while the tape is being read in.

22) If a C or an O is printed on the terminal as the tape reads in, the tape has been mis-read and you should start over at step 1 on page 46.

23) When the tape finishes reading, BASIC should start up and print MEMORY SIZE?. See Appendix B for the initialization procedure.

24) If BASIC refuses to load from the Audio Cassette, the ACR Demodulator may need alignment. The flip side of the cassette contains 90 seconds of 125's (octal) which were recorded at the same tape speed as BASIC. Use the Input Test Program described on pages 22 and 28 of the ACR manual to perform the necessary alignment.

APPENDIX B

INITIALIZATION DIALOG

STARTING BASIC

Leave the sense switches as they were set for loading BASIC (Appendix A). After the initialization dialog is complete, and BASIC types OK, you are free to use the sense switches as an input device (I/O port 255).

After you have loaded BASIC, it will respond:

MEMORY SIZE?

If you type a carriage return to MEMORY SIZE?, BASIC will use all the contiguous memory upwards from location zero that it can find. BASIC will stop searching when it finds one byte of ROM or non-existent memory.

If you wish to allocate only part of the ALTAIR's memory to BASIC, type the number of bytes of memory you wish to allocate in decimal. This might be done, for instance, if you were using part of the memory for a machine language subroutine.

There are 4096 bytes of memory in a 4K system, and 8192 bytes in an 8K system.

BASIC will then ask:

TERMINAL WIDTH?	This is to set the output line width for PRINT statements only. Type in the number of characters for the line width for the particular terminal or other output device you are using. This may be any number from 1 to 255, depending on the terminal. If no answer is given (i.e. a carriage return is typed) the line width is set to 72 characters.

Now ALTAIR BASIC will enter a dialog which will allow you to delete some of the arithmetic functions. Deleting these functions will give more memory space to store your programs and variables. However, you will not be able to call the functions you delete. Attempting to do so will result in an FC error. The only way to restore a function that has been deleted is to reload BASIC.

The following is the dialog which will occur:

4K Version

WANT SIN?	Answer " Y " to retain SIN, SQR and RND If you answer " N ", asks next question
WANT SQR?	Answer " Y " to retain SQR and RND. If you answer " N ", asks next question.

```
WANT RND?
```
Answer " Y " to retain RND.
Answer " N " to delete RND.

8K Version

```
WANT SIN-COS-TAN-ATN?
```
Answer " Y " to retain all four of the functions, " N " to delete all four, or " A " to delete ATN only.

Now BASIC will type out:

```
XXXX BYTES FREE

ALTAIR BASIC VERSION 3.0
[FOUR-K VERSION]
     (or)
[EIGHT-K VERSION]

OK
```

"XXXX" is the number of bytes available for program, variables, matrix storage and the stack. It does not include string space.

You will now be ready to begin using ALTAIR BASIC.

APPENDIX C

ERROR MESSAGES

After an error occurs, BASIC returns to command level and types OK. Variable values and the program text remain intact, but the program can not be continued and all GOSUB and FOR context is lost.

When an error occurs in a direct statement, no line number is printed.

Format of error messages:

Direct Statement ?XX ERROR

Indirect Statement ?XX ERROR IN YYYYY

In both of the above examples, "XX" will be the error code. The "YYYYY" will be the line number where the error occured for the indirect statement.

The following are the possible error codes and their meanings:

ERROR CODE	MEANING
4K VERSION	
BS	Bad Subscript. An attempt was made to reference a matrix element which is outside the dimensions of the matrix. In the 8K version, this error can occur if the wrong number of dimensions are used in a matrix reference; for instance, LET A(1,1,1)=Z when A has been dimensioned DIM A(2,2).
DD	Double Dimension. After a matrix was dimensioned, another dimension statement for the same matrix was encountered. This error often occurs if a matrix has been given the default dimension 10 because a statement like A(I)=3 is encountered and then later in the program a DIM A(100) is found.
FC	Function Call error. The parameter passed to a math or string function was out of range. FC errors can occur due to: a) a negative matrix subscript (LET A(-1)= b) an unreasonably large matrix subscript (>32767) c) LOG-negative or zero argument d) SQR-negative argument

e) A↑B with A negative and B not an integer

f) a call to USR before the address of the machine language subroutine has been patched in

g) calls to MID$, LEFT$, RIGHT$, INP, OUT, WAIT, PEEK, POKE, TAB, SPC or ON...GOTO with an improper argument.

ID — Illegal Direct. You cannot use an INPUT or *(in 8K Version)* DEFFN statement as a direct command.

NF — NEXT without FOR. The variable in a NEXT statement corresponds to no previously executed FOR statement.

OD — Out of Data. A READ statement was executed but all of the DATA statements in the program have already been read. The program tried to read too much data or insufficient data was included in the program.

OM — Out of Memory. Program too large, too many variables, too many FOR loops, too many GOSUB's, too complicated an expression or any combination of the above. (see Appendix D)

OV — Overflow. The result of a calculation was too large to be represented in BASIC's number format. If an underflow occurs, zero is given as the result and execution continues without any error message being printed.

SN — Syntax error. Missing parenthesis in an expression, illegal character in a line, incorrect punctuation, etc.

RG — RETURN without GOSUB. A RETURN statement was encountered without a previous GOSUB statement being executed.

US — Undefined Statement. An attempt was made to GOTO, GOSUB or THEN to a statement which does not exist.

/0 — Division by Zero.

8K VERSION (Includes all of the previous codes in addition to the following.)

CN — Continue error. Attempt to continue a program when none exists, an error occured, or after a new line was typed into the program.

LS	Long String. Attempt was made by use of the concatenation operator to create a string more than 255 characters long.
OS	Out of String Space. Save your program on paper tape or cassette, reload BASIC and allocate more string space or use smaller strings or less string variables.
ST	String Temporaries. A string expression was too complex. Break it into two or more shorter ones.
TM	Type Mismatch. The left hand side of an assignment statement was a numeric variable and the right hand side was a string, or vice versa; or, a function which expected a string argument was given a numeric one or vice versa.
UF	Undefined Function. Reference was made to a user defined function which had never been defined.

APPENDIX D

SPACE HINTS

In order to make your program smaller and save space, the following hints may be useful.

1) Use multiple statements per line. There is a small amount of overhead (5bytes) associated with each line in the program. Two of these five bytes contain the line number of the line in binary. This means that no matter how many digits you have in your line number (minimum line number is 0, maximum is 65529), it takes the same number of bytes. Putting as many statements as possible on a line will cut down on the number of bytes used by your program.

2) Delete all unnecessary spaces from your program. For instance:

```
10 PRINT X, Y, Z
```

uses three more bytes than

```
10 PRINTX,Y,Z
```

Note: All spaces between the line number and the first non-blank character are ignored.

3) Delete all REM statements. Each REM statement uses at least one byte plus the number of bytes in the comment text. For instance, the statement 130 REM THIS IS A COMMENT uses up 24 bytes of memory.

In the statement 140 X=X+Y: REM UPDATE SUM, the REM uses 14 bytes of memory including the colon before the REM.

4) Use variables instead of constants. Suppose you use the constant 3.14159 ten times in your program. If you insert a statement

```
10 P=3.14159
```

in the program, and use P instead of 3.14159 each time it is needed, you will save 40 bytes. This will also result in a speed improvement.

5) A program need not end with an END; so, an END statement at the end of a program may be deleted.

6) Reuse the same variables. If you have a variable T which is used to hold a temporary result in one part of the program and you need a temporary variable later in your program, use it again. Or, if you are asking the terminal user to give a YES or NO answer to two different questions at two different times during the execution of the program, use the same temporary variable A$ to store the reply.

7) Use GOSUB's to execute sections of program statements that perform identical actions.

8) If you are using the 8K version and don't need the features of the 8K version to run your program, consider using the 4K version instead. This will give you approximately 4.7K to work with in an 8K machine, as opposed to the 1.6K you have available in an 8K machine running the 8K version of BASIC.

9) Use the zero elements of matrices; for instance, A(0), B(0,X).

STORAGE ALLOCATION INFORMATION

Simple (non-matrix) numeric variables like V use 6 bytes; 2 for the variable name, and 4 for the value. Simple non-matrix string variables also use 6 bytes; 2 for the variable name, 2 for the length, and 2 for a pointer.

Matrix variables use a minimum of 12 bytes. Two bytes are used for the variable name, two for the size of the matrix, two for the number of dimensions and two for each dimension along with four bytes for each of the matrix elements.

String variables also use one byte of string space for each character in the string. This is true whether the string variable is a simple string variable like A$, or an element of a string matrix such as Q1$(5,2).

When a new function is defined by a DEF statement, 6 bytes are used to store the definition.

Reserved words such as FOR, GOTO or NOT, and the names or the intrinsic functions such as COS, INT and STR$ take up only one byte of program storage. All other characters in programs use one byte of program storage each.

When a program is being executed, space is dynamically allocated on the stack as follows:

1) Each active FOR...NEXT loop uses 16 bytes.

2) Each active GOSUB (one that has not returned yet) uses 6 bytes.

3) Each parenthesis encountered in an expression uses 4 bytes and each temporary result calculated in an expression uses 12 bytes.

APPENDIX E

SPEED HINTS

The hints below should improve the execution time of your BASIC program. Note that some of these hints are the same as those used to decrease the space used by your programs. This means that in many cases you can increase the efficiency of both the speed and size of your programs at the same time.

1) Delete all unnecessary spaces and REM's from the program. This may cause a small decrease in execution time because BASIC would otherwise have to ignore or skip over spaces and REM statements.

2) *THIS IS PROBABLY THE MOST IMPORTANT SPEED HINT BY A FACTOR OF 10.* Use variables instead of constants. It takes more time to convert a constant to its floating point representation than it does to fetch the value of a simple or matrix variable. This is especially important within FOR...NEXT loops or other code that is executed repeatedly.

3) Variables which are encountered first during the execution of a BASIC program are allocated at the start of the variable table. This means that a statement such as 5 A=0:B=A:C=A, will place A first, B second, and C third in the symbol table (assuming line 5 is the first statement executed in the program). Later in the program, when BASIC finds a reference to the variable A, it will search only one entry in the symbol table to find A, two entries to find B and three entries to find C, etc.

4) *(8K Version)* NEXT statements without the index variable. NEXT is somewhat faster than NEXT I because no check is made to see if the variable specified in the NEXT is the same as the variable in the most recent FOR statement.

5) Use the 8K version instead of the 4K version. The 8K version is about 40% faster than the 4K due to improvements in the floating point arithmetic routines.

6) The math functions in the 8K version are much faster than their counterparts simulated in the 4K version. (see Appendix G)

APPENDIX F

DERIVED FUNCTIONS

The following functions, while not intrinsic to ALTAIR BASIC, can be calculated using the existing BASIC functions.

FUNCTION	FUNCTION EXPRESSED IN TERMS OF BASIC FUNCTIONS
SECANT	SEC(X) = 1/COS(X)
COSECANT	CSC(X) = 1/SIN(X)
COTANGENT	COT(X) = 1/TAN(X)
INVERSE SINE	ARCSIN(X) = ATN(X/SQR(-X*X+1))
INVERSE COSINE	ARCCOS(X) = -ATN(X/SQR(-X*X+1))+1.5708
INVERSE SECANT	ARCSEC(X) = ATN(SQR(X*X-1))+(SGN(X)-1)*1.5708
INVERSE COSECANT	ARCCSC(X) = ATN(1/SQR(X*X-1))+(SGN(X)-1)*1.5708
INVERSE COTANGENT	ARCCOT(X) = -ATN(X)+1.5708
HYPERBOLIC SINE	SINH(X) = (EXP(X)-EXP(-X))/2
HYPERBOLIC COSINE	COSH(X) = (EXP(X)+EXP(-X))/2
HYPERBOLIC TANGENT	TANH(X) = -EXP(-X)/(EXP(X)+EXP(-X))*2+1
HYPERBOLIC SECANT	SECH(X) = 2/(EXP(X)+EXP(-X))
HYPERBOLIC COSECANT	CSCH(X) = 2/(EXP(X)-EXP(-X))
HYPERBOLIC COTANGENT	COTH(X) = EXP(-X)/(EXP(X)-EXP(-X))*2+1
INVERSE HYPERBOLIC SINE	ARGSINH(X) = LOG(X+SQR(X*X+1))
INVERSE HYPERBOLIC COSINE	ARGCOSH(X) = LOG(X+SQR(X*X-1))
INVERSE HYPERBOLIC TANGENT	ARGTANH(X) = LOG((1+X)/(1-X))/2
INVERSE HYPERBOLIC SECANT	ARGSECH(X) = LOG((SQR(-X*X+1)+1)/X)
INVERSE HYPERBOLIC COSECANT	ARGCSCH(X) = LOG((SGN(X)*SQR(X*X+1)+1)/X)
INVERSE HYPERBOLIC COTANGENT	ARGCOTH(X) = LOG((X+1)/(X-1))/2

APPENDIX G

SIMULATED MATH FUNCTIONS

The following subroutines are intended for 4K BASIC users who want to use the transcendental functions not built into 4K BASIC. The corresponding routines for these functions in the 8K version are much faster and more accurate. The REM statements in these subroutines are given for documentation purposes only, and should not be typed in because they take up a large amount of memory.

The following are the subroutine calls and their 8K equivalents:

8K EQUIVALENT	SUBROUTINE CALL
P9=X9↑Y9	GOSUB 60030
L9=LOG(X9)	GOSUB 60090
E9=EXP(X9)	GOSUB 60160
C9=COS(X9)	GOSUB 60240
T9=TAN(X9)	GOSUB 60280
A9=ATN(X9)	GOSUB 60310

The unneeded subroutines should not be typed in. Please note which variables are used by each subroutine. Also note that TAN and COS require that the SIN function be retained when BASIC is loaded and initialized.

```
60000 REM EXPONENTIATION: P9=X9↑Y9
60010 REM NEED: EXP, LOG
60020 REM VARIABLES USED: A9,B9,C9,E9,L9,P9,X9,Y9
60030 P9=1 : E9=0 : IF Y9=0 THEN RETURN
60040 IF X9<0 THEN IF INT(Y9)=Y9 THEN P9=1-2*Y9+4*INT(Y9/2) : X9=-X9
60050 IF X9<>0 THEN GOSUB 60090 : X9=Y9*L9 : GOSUB 60160
60060 P9=P9*E9 : RETURN
60070 REM NATURAL LOGARITHM: L9=LOG(X9)
60080 REM VARIABLES USED: A9,B9,C9,E9,L9,X9
60090 E9=0 : IF X9<=0 THEN PRINT "LOG FC ERROR"; : STOP
60095 A9=1 : B9=2 : C9=.5 : REM THIS WILL SPEED UP THE FOLLOWING
60100 IF X9>=A9 THEN X9=C9*X9 : E9=E9+A9 : GOTO 60100
60110 IF X9<C9 THEN X9=B9*X9 : E9=E9-A9 : GOTO 60110
60120 X9=(X9-.707107)/(X9+.707107) : L9=X9*X9
60130 L9=(((.598979*L9+.961471)*L9+2.88539)*X9+E9-.5)*.693147
60135 RETURN
60140 REM EXPONENTIAL: E9=EXP(X9)
60150 REM VARIABLES USED: A9,E9,L9,X9
60160 L9=INT(1.4427*X9)+1 : IF L9<127 THEN 60180
60170 IF X9>0 THEN PRINT "EXP OV ERROR"; : STOP
60175 E9=0 : RETURN
60180 E9=.693147*L9-X9 : A9=1.32988E-3-1.41316E-4*E9
60190 A9=((A9*E9-8.30136E-3)*E9+4.16574E-2)*E9
60195 E9=(((A9-.166665)*E9+.5)*E9-1)*E9+1 : A9=2
60197 IF L9<=0 THEN A9=.5 : L9=-L9 : IF L9=0 THEN RETURN
```

```
60200 FOR X9=1 TO L9 : E9=A9*E9 : NEXT X9 : RETURN
60210 REM COSINE: C9=COS(X9)
60220 REM N.B. SIN MUST BE RETAINED AT LOAD-TIME
60230 REM VARIABLES USED: C9,X9
60240 C9=SIN(X9+1.5708) : RETURN
60250 REM TANGENT: T9=TAN(X9)
60260 REM NEEDS COS. (SIN MUST BE RETAINED AT LOAD-TIME)
60270 REM VARIABLES USED: C9,T9,X9
60280 GOSUB 60240 : T9=SIN(X9)/C9 : RETURN
60290 REM ARCTANGENT: A9=ATN(X9)
60300 REM VARIABLES USED: A9,B9,C9,T9,X9
60310 T9=SGN(X9): X9=ABS(X9): C9=0 : IF X9>1 THEN C9=1 : X9=1/X9
60320 A9=X9*X9 : B9=((2.86623E-3*A9-1.61657E-2)*A9+4.29096E-2)*A9
60330 B9=((((B9-7.5289E-2)*A9+.106563)*A9-.142089)*A9+.199936)*A9
60340 A9=((B9-.333332)*A9+1)*X9 : IF C9=1 THEN A9=1.5708-A9
60350 A9=T9*A9 : RETURN
```

APPENDIX H

CONVERTING BASIC PROGRAMS NOT WRITTEN FOR THE ALTAIR

Though implementations of BASIC on different computers are in many ways similar, there are some incompatibilites which you should watch for if you are planning to convert some BASIC programs that were not written for the ALTAIR.

1) Matrix subscripts. Some BASICs use " [" and "] " to denote matrix subscripts. ALTAIR BASIC uses " (" and ") ".

2) Strings. A number of BASICs force you to dimension (declare) the length of strings before you use them. You should remove all dimension statements of this type from the program. In some of these BASICs, a declaration of the form DIM A$(I,J) declares a string matrix of J elements each of which has a length I. Convert DIM statements of this type to equivalent ones in ALTAIR BASIC: DIM A$(J).

ALTAIR BASIC uses " + " for string concatenation, not " , " or " & ".

ALTAIR BASIC uses LEFT$, RIGHT$ and MID$ to take substrings of strings. Other BASICs use A$(I) to access the Ith character of the string A$, and A$(I,J) to take a substring of A$ from character position I to character position J. Convert as follows:

OLD	NEW
A$(I)	MID$(A$,I,1)
A$(I,J)	MID$(A$,I,J-I+1)

This assumes that the reference to a substring of A$ is in an expression or is on the right side of an assignment. If the reference to A$ is on the left hand side of an assignment, and X$ is the string expression used to replace characters in A$, convert as follows:

OLD	NEW
A$(I)=X$	A$=LEFT$(A$,I-1)+X$+MID$(A$,I+1)
A$(I,J)=X$	A$=LEFT$(A$,I-1)+X$+MID$(A$,J+1)

3) Multiple assignments. Some BASICs allow statements of the form: 500 LET B=C=0. This statement would set the variables B & C to zero.

In 8K ALTAIR BASIC this has an entirely different effect. All the " ='s " to the right of the first one would be interpreted as logical comparison operators. This would set the variable B to -1 if C equaled 0. If C did not equal 0, B would be set to 0. The easiest way to convert statements like this one is to rewrite them as follows:

```
500 C=0:B=C.
```

4) Some BASICs use " \ " instead of " : " to delimit multiple statements per line. Change the " \'s " to " :'s " in the program.

5) Paper tapes punched by other BASICs may have no nulls at the end of each line, instead of the three per line recommended for use with ALTAIR BASIC.

To get around this, try to use the tape feed control on the Teletype to stop the tape from reading as soon as ALTAIR BASIC types a carriage return at the end of the line. Wait a second, and then continue feeding in the tape.

When you have finished reading in the paper tape of the program, be sure to punch a new tape in ALTAIR BASIC's format. This will save you from having to repeat this process a second time.

6) Programs which use the MAT functions available in some BASICs will have to be re-written using FOR...NEXT loops to perform the appropriate operations.

APPENDIX I

USING THE ACR INTERFACE

NOTE: The cassette features, CLOAD and CSAVE, are only present in 8K BASICs which are distributed on cassette. 8K BASIC on paper tape will give the user about 130 more bytes of free memory, but it will not recognize the CLOAD or CSAVE commands.

The CSAVE command saves a program on cassette tape. CSAVE takes one argument which can be any printing character. CSAVE can be given directly or in a program. Before giving the CSAVE command start your audio recorder on Record, noting the position of the tape.

CSAVE writes data on channel 7 and expects the device status from channel 6. Patches can easily be made to change these channel numbers.

When CSAVE is finished, execution will continue with the next statement. What is written onto the tape is BASIC's internal representation of the program in memory. The amount of data written onto the tape will be equal to the size of the program in memory plus seven.

Variable values are not saved on the tape, nor are they affected by the CSAVE command. The number of nulls being printed on your terminal at the start of each line has no affect on the CSAVE or CLOAD commands.

CLOAD takes its one character argument just like the CSAVE command. For example, CLOAD E.

The CLOAD command first executes a "NEW" command, erasing the current program and all variable values. The CLOAD command should be given before you put your cassette recorder on Play.

BASIC will read a byte from channel 7 whenever the character ready flag comes up on channel 6. When BASIC finds the program on the tape, it will read all characters received from the tape into memory until it finds three consecutive zeros which mark the end of the program. Then BASIC will return to command level and type "OK".

Statements given on the same line as a CLOAD command are ignored. The program on the cassette is not in a checksummed format, so the program must be checked to make sure it read in properly.

If BASIC does not return to command level and type "OK", it means that BASIC either never found a file with the right filename character, or that BASIC found the file but the file never ended with three consecutive zeros. By carefully watching the front panel lights, you can tell if BASIC ever finds a file with the right name.

Stopping the ALTAIR and restarting it at location 0 will prevent BASIC from searching forever. However, it is likely that there will either be no program in the machine, or a partial program that has errors. Typing NEW will always clear out whatever program is in the machine.

Reading and writing data from the cassette is done with the INP, OUT and WAIT statements. Any block of data written on the tape should have its beginning marked with a character. The main thing to be careful of is allowing your program to fall behind while data passes by unread.

Data read from the cassette should be stored in a matrix, since

there isn't time to process data as it is being read in. You will probably want to detect the end of data on the tape with a special character.

USRLOC = 8275, 8276

APPENDIX J

BASIC/MACHINE LANGUAGE INTERFACE

In all versions of BASIC the user can link to a machine language subroutine. The first step is to set aside enough memory for the subroutine. When BASIC asks "MEMORY SIZE?", you shouldn't type a return, because BASIC would then write into all of memory trying to find out how much memory your machine has and then use whatever memory it finds.

The memory that BASIC actually uses is constantly modified, so you cannot store your machine language routine in those locations.

BASIC always uses memory starting at location 0 and as high upwards as you let it. BASIC cannot use non-contiguous blocks of memory. Therefore, it is best to reserve the top locations of memory for your machine language program.

For example, if you have a 4K machine and want to use a 200 byte subroutine, you should set memory size to 3896. Remember, BASIC always accepts numbers in decimal and that 4K is really 2↑12=4096 rather than 4000. Now BASIC will not use any location >= 3896.

If you try to allocate too much memory for your machine language program, you will get an OM (out of memory) error. This is because there is a certain amount of memory that BASIC must have or it will give an OM error and go back to the "MEMORY SIZE?" question.

The starting location of your routine must be stored in a location known as "USRLOC". The exact octal location of USRLOC will be given with each distributed version of BASIC. It is not the same for the 4K and 8K versions.

USRLOC for Version 3.0: 8K (both paper tape & cassette) = 111(octal)
4K = 103(octal)

Initially USRLOC is set up to contain the address of "ILLFUN", which is the routine that gives an FC (function call) error. USRLOC is the two byte absolute address of the location BASIC calls when USR is invoked.

USR is a function just like ABS or INT and is called as follows:
10 X=USR(3).

When your routine is called the stack pointer is set up and you are allowed to use up to 8 levels of stack (16 bytes). If you want to use more, you have to save BASIC's stack pointer (SP), set up your own, and restore BASIC's before you return back to BASIC.

All of the registers (A, B, C, D, E, H, L and PSW) can be changed. It is dangerous to modify locations in BASIC itself unless you know what you are doing. This is unlikely unless you have purchased a source copy of BASIC. Popping more entries off of the stack than you put on is almost guaranteed to cause trouble.

To retrieve the argument passed to USR, you must call the routine whose address is given in location 4 and 5 (DEINT). The low order 8 bits of an address are always stored in the lower address (4 in this case), and the high order 8 bits are stored in the next (higher) memory address (5 in this case).

The argument to USR is truncated to an integer (calling USR with 3.8 is the same as calling it with 3). If the argument is greater than 32767 or less than -32768, an FC error will result. When DEINT returns, the two byte signed value of the argument will be in registers D & E. The high order byte would be in D, the low order byte in E. For instance: if the argument to USR was -1, D would equal 255 and E would equal 255; if the argument was 400, D would equal 1 and E would equal 144.

To pass back a value from USR, set up a two byte value in registers A & B and call the routine whose address is given in locations 6 and 7. A & B should be set up in the same manner that D & E are when a value is passed to USR (A should contain the high order byte and B the low order byte).

If the routine whose address is given in locations 6 and 7 is not called, the function USR in the user's program will be an identity function. That is, USR(X) will equal X.

At the end of the USR routine a RET must be done to get back to BASIC. The BASIC program is completely stopped while USR is being executed and the program will not be continued until USR returns.

In the 4K version, the USR routine should not enable interrupts from a device. 4K BASIC uses the RST 7 location (56 decimal, 70 octal) to store a subroutine. If an interrupt occurs, this subroutine will be called which will have an undetermined and undesirable effect on the way BASIC behaves.

In the 8K BASIC, locations 56, 57 and 58 decimal have been set aside to store a JMP to a user-provided interrupt service routine. Initially a RET instruction is stored at location 56, so until a user sets up the call to his interrupt service routine, interrupts will have no effect.

Care must be taken in interrupt routines to save and restore the stack pointer, (A, B, C, D, E, H & L) and the PSW. Interrupt routines can pass data using PEEK, and can receive data using POKE.

The interrupt service routine should re-enable interrupts with an EI instruction before it returns, as interrupts are automatically disabled when the interrupt occurs. If this procedure is not followed, the interrupt service routine will never "see" another interrupt.

Though there is only one way of calling a machine language subroutine, this does not restrict the user to a single subroutine. The argument passed to USR can be used to determine which routine gets called. Multiple arguments to a machine language routine can be passed with POKE or through multiple calls to USR by the BASIC program.

The machine language routine can be loaded from paper tape or cassette before or after BASIC is loaded. The checksum loader, an unchecksummed loader, the console switches, or more conveniently the POKE function can be used to load the routine.

A common use of USR for 4K users will be doing IN's and OUT's to special devices. For example, on a 4K machine a user wants USR to pass back the value of the front panel switch register:

```
Answer to MEMORY SIZE? : 4050
USRLOC patched to contain [17,322]=7722 Base 8=4050 decimal
```

At location 4050=7722 Base 8 put:

```
7722/333        IN      255     ;(255 Base 10=377 Base 8) Get
7723/377                        ;the value of the switches in A
7724/107        MOV     B,A     ;B gets low part of answer
7725/257        XRA     A       ;A gets high part of answer
7726/052        LHLD    6       ;get address of routine
7727/006
7730/000                        ;that floats [A,B]
7731/351        PCHL            ;go to that routine which will
                                ;return to BASIC
                                ;with the answer
```

MORE ON PEEK AND POKE *(8K VERSION ONLY)*

As mentioned before, POKE can be used to set up your machine language routine in high memory. BASIC does not restrict which addresses you can POKE. Modifying USRLOC can be accomplished using two successive calls to POKE. Patches which a user wishes to include in his BASIC can also be made using POKE.

Using the PEEK function and OUT statement of 8K BASIC, the user can write a binary dump program in BASIC. Using INP and POKE it is possible to write a binary loader.

PEEK and POKE can be used to store byte oriented information. When you initialize BASIC, answer the MEMORY SIZE? question with the amount of memory in your ALTAIR minus the amount of memory you wish to use as storage for byte formatted data.

You are now free to use the memory in the top of memory in your ALTAIR as byte storage. See PEEK and POKE in the Reference Material for a further description of their parameters.

APPENDIX K

ASCII CHARACTER CODES

DECIMAL	CHAR.	DECIMAL	CHAR.	DECIMAL	CHAR.
000	NUL	043	+	086	V
001	SOH	044	,	087	W
002	STX	045	-	088	X
003	ETX	046	.	089	Y
004	EOT	047	/	090	Z
005	ENQ	048	0	091	[
006	ACK	049	1	092	\
007	BEL	050	2	093	]
008	BS	051	3	094	↑
009	HT	052	4	095	←
010	LF	053	5	096	`
011	VT	054	6	097	a
012	FF	055	7	098	b
013	CR	056	8	099	c
014	SO	057	9	100	d
015	SI	058	:	101	e
016	DLE	059	;	102	f
017	DC1	060	<	103	g
018	DC2	061	=	104	h
019	DC3	062	>	105	i
020	DC4	063	?	106	j
021	NAK	064	@	107	k
022	SYN	065	A	108	l
023	ETB	066	B	109	m
024	CAN	067	C	110	n
025	EM	068	D	111	o
026	SUB	069	E	112	p
027	ESCAPE	070	F	113	q
028	FS	071	G	114	r
029	GS	072	H	115	s
030	RS	073	I	116	t
031	US	074	J	117	u
032	SPACE	075	K	118	v
033	!	076	L	119	w
034	"	077	M	120	x
035	#	078	N	121	y
036	$	079	O	122	z
037	%	080	P	123	{
038	&	081	Q	124	|
039	'	082	R	125	}
040	(	083	S	126	~
041	)	084	T	127	DEL
042	*	085	U		

LF=Line Feed FF=Form Feed CR=Carriage Return DEL=Rubout

CHR$ is a string function which returns a one character string which contains the ASCII equivalent of the argument, according to the conversion table on the preceeding page. ASC takes the first character of a string and converts it to its ASCII decimal value.

One of the most common uses of CHR$ is to send a special character to the user's terminal. The most often used of these characters is the BEL (ASCII 7). Printing this character will cause a bell to ring on some terminals and a "beep" on many CRT's. This may be used as a preface to an error message, as a novelty, or just to wake up the user if he has fallen asleep. (Example: PRINT CHR$(7);)

A major use of special characters is on those CRT's that have cursor positioning and other special functions (such as turning on a hard copy printer).

As an example, try sending a form feed (CHR$(12)) to your CRT. On most CRT's this will usually cause the screen to erase and the cursor to "home" or move to the upper left corner.

Some CRT's give the user the capability of drawing graphs and curves in a special point-plotter mode. This feature may easily be taken advantage of through use of ALTAIR BASIC's CHR$ function.

APPENDIX L

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
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).
Multi-Argument, user defined functions with string arguments and values allowed

Other features contemplated for future release are:

A multiple user BASIC
Explicit matrix manipulation
Virtual matrices
Statement modifiers
Record I/O
Paramaterized GOSUB
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.

APPENDIX M

BASIC TEXTS

Below are a few of the many texts that may be helpful in learning BASIC.

1) BASIC PROGRAMMING, John G. Kemeny, Thomas E Kurtz, 1967, p145

2) BASIC, Albrecht, Finkel and Brown, 1973

3) A GUIDED TOUR OF COMPUTER PROGRAMMING IN BASIC, Thomas A Dwyer and Michael S. Kaufman; Boston: Houghton Mifflin Co., 1973

Books numbered 1 & 2 may be obtained from:

People's Computer Company
P.O. Box 310
Menlo Park, California
94025

They also have other books of interest, such as:

101 BASIC GAMES, Ed. David Ahl, 1974 p250

WHAT TO DO AFTER YOU HIT RETURN or PCC's FIRST BOOK OF COMPUTER GAMES

COMPUTER LIB & DREAM MACHINES, Theodore H. Nelson, 1974, p186

8/12/76
JEZ

+ recursive functions
+ MID$ start?
INSTR
Hex constants $4FF
[illegible]

MODS TO ALTAIR MANUAL

1) NO NULL STATEMENT

2) AUTO 1000,100

GENERATE LINE NUMBERS STARTING AT 1000, INCREMENTED BY 100. LINES INSERTED INTO PROGRAM, TERMINATE ON ↑C.

on entering line, 1500,10 changes auto params (add ? line #)

3) OPERATORS ADDED ARE

MOD	x MOD y =	x - INT(x/y) * y
MIN	x MIN y =	IF(x < y) THEN x ELSE y
MAX	x MAX y =	IF(x > y) THEN x ELSE y

4) ~~DEF FN~~ New Def statement redone. The FUNCTION can have any type, determined from the name. The function can any number of arguments, including zero.

Example:

```
DEF FN upd$($$, b, e, u$) = LEFT$($$,b-1)+u$+MID$($$,E+1)
DEF FNds = SQR((X(I)-X(J))↑2 + (Y(I)-Y(J))↑2)
```

5) INPUT PROMPT STRING; as above? [illegible]

6) SET GOTO (ON/OFF) ; tracing flag
SET PRINT (ON/OFF) ; parity flag
SET LIST 0 ; turns off auto [illegible]

MID$(stringvar, pos, len) = string expr
Index (PL/I) = INSTR (string expr, string expr)

7) New functions
HEX$(byte) gives two character hex representation
UPPER$(string) gives upper case version of string

Special Characters
@ETX ≡ ↑C - BREAK
@BS ≡ ↑H - backspace input

i) EDIT ~~[illegible]~~ 100[, 200] ← copies 200 to 100 before editing new line 100...
tab ≡ - copy until following char reached (if [illegible])
i - goto to insert mode
r - goto replace mode
d - delete [illegible] chars - type chars prefixed with \
blank - advance
@ETX; ↑C abort
@ESC ≡ ↑K exits insert, replace modes
@CR ≡ ↑M reprint line, update if at beginning.

i) Conditional Expression

IF b THEN c ELSE d END

if b is true evaluates c, skips d
if b is false evaluate d, skipping c.

$2^3 = 2 \wedge 3$

edit line#1, line#2
  ↳ if 2nd line appears, it is copied a given number of line#1
  → then may be edited.

C/R finishes edit if no changes made
INT aborts edit (none of the changes will be done)

space copies characters as-is
backspace can't be used
d deletes one char
i start inserting the following chars until C/R or escape
r start replacing with " "
tab ⇥ skip over (like space) until char ⇥ found.

1st 3 characters are enough: edi lis , or / works for range: lis 50/75

list #1,#2	50,	lists 50 to end	50,70	lists 50 thru 70
	,50	list beginning thru 50	50	lists 50 only.

new erases program

deletes line
text of program replaces or inserts line

delete #1,#2 deletes range of lines

auto #1,#2 prompts for inserting lines, starting at #1, incrementing by #2
if after it prompts you type #1 or #1,#2 it will restart from those numbers.

To save a program on paper tape,
ready the punch by pressing in the red button,
then type to BASIC CSAVE "P"
↑ must be upper case.
Basic will punch the program on tape.

TO save on WYLBUR, logon & COLLECT UNN CLEAR
then switch to Basic (>X) and type CSAVE "K"
↑ upper case
The program will be transmitted to the
wylbur Active file. Then switch the keyboard to wylbur by Repeat Int >k
↑ lowercase
and hit INT twice to stop wylbur from COLLECTING.
Then save the Active file (SAVE #stuff) for example.

```
                        ;
                        ;
                        ;  SYSTEM INTERFACE
                        ;
                            file    "8K Basic"

                        BASIC:                              ;FULL RESTART INITIALIZATION
                        SYSINITJ:
0000 C30000                 JMP     INITIALZ
                        REENTERBASIC:                       ;REENTER AFTER PAUSE
0003 C30000                 JMP     cmndrstr

                        ;
                        ;  Monitor Routines
                        ;
0006 0406               co  equ     406h    ;c -> screen
0006 0409               cinb        equ     409h    ;keyboard -> ac, carry set if any
0006 0538               dclr        equ     538h    ;clear screen
0006 04F4               xco equ     4f4h    ;c -> printer (blocking)
                        ;
                        ;  NON-BLOCKING INPUT
                        ;    CHAR IN AC IF NOT ZERO
                        ;    ZERO SET IF NONE
                        ;
                        SYSKEYIN:
0006 C5                     push    b
0007 D5                     push    d
0008 E5                     push    h
0009 CD0904                 call    cinb    ;get char
000C D20000                 jnc     syskeynone
000F FE00                   cpi     0
0011 CA0000                 jz      clearscreen
0014 FE1F                   cpi     1fh     ;us to break to fourteen
0016 CA0000                 jz      gomonitor
                        syskeyinret:
0019 E1                     pop     h
001A D1                     pop     d
001B C1                     pop     b
001C C9                     ret
                        syskeynone:
001D 97                     sub     a       ;set zero
001E C31900                 jmp     syskeyinret
                        clearscreen:
0021 CD3805                 call    dclr    ;clear screen
0024 C31D00                 jmp     syskeynone
                        gomonitor:
0027 CF                     rst     1       ;about using us (↑←)
0028 00                     nop
0029 00                     nop
                        ;
                        ;  SEND AC TO SCREEN
                        ;
                        SYSDISPL:
002A F5                     push    psw
002B C5                     push    b
002C D5                     push    d
```

```
002D E5          push    h
002E 4F          mov     c,a
002F CD0604      call    co       ;c to screen
0032 3A0000      lda     p3010    ;print on the 3010 if zero
0035 A7          ana     a
0036 CCF404      cz      xco      ;yes print
0039 E1          pop     h
003A D1          pop     d
003B C1          pop     b
003C F1          pop     psw
003D C9          RET
                 ;
                 ;  CHECK FOR BREAK REQUEST
                 ;    SET ZERO TO BREAK
                 ;
                 SYSBREAK:
003E CD0600      call    syskeyin
0041 CA0000      jz      nobreak
0044 97          sub     a
0045 C9          ret
                 nobreak:
0046 3E01        mvi     a,1
0048 B7          ora     a
0049 C9          ret
                 ;
                 ;  DELAY
                 ;
                 SYSWAIT:
004A C9          RET
                 ;
                 ;  RETURN TO MONITOR
                 ;
004B B400        MONITOR     EQU     0B400H
                 SYSQUIT:
004B C300B4      JMP     MONITOR
```

```
004E 000D   CR  EQU     0DH
004E 000A   LF  EQU     0AH
004E 0007   BEL EQU     07H
004E 0008   BS  EQU     08H
004E 0009   TAB EQU     09H
004E 0009   HT  EQU     09H
004E 0011   DC1 EQU     11H
004E 007F   DEL EQU     7FH
004E 000F   SI  EQU     0FH
004E 0003   ETX EQU     03H
004E 000C   FF  EQU     0CH
004E 001B   ESC EQU     1BH
```

```
004E 0080   KEYSTM  EQU 80H             ;STATEMENT COOES
004E 0080   KEYDAT  EQU KEYSTM
004E 0081   KEYREM  EQU KEYDAT+1
004E 0082   KEYLSAL EQU KEYREM+1
004E 0082   KEYEND  EQU KEYLSAL
004E 0083   KEYFOR  EQU KEYEND+1
004E 0084   KEYNEX  EQU KEYFOR+1
004E 0085   KEYINPT EQU KEYNEX+1
004E 0086   KEYDIM  EQU KEYINPT+1
004E 0087   KEYREA  EQU KEYOIM+1
004E 0088   KEYLET  EQU KEYREA+1
004E 0089   KEYGTO  EQU KEYLET+1
004E 008A   KEYRUN  EQU KEYGTO+1
004E 008B   KEYIF   EQU KEYRUN+1
004E 008C   KEYELS  EQU KEYIF+1
004E 008D   KEYRES  EQU KEYELS+1
004E 008E   KEYGSB  EQU KEYRES+1
004E 00BF   KEYRET  EQU KEYGSB+1
004E 0090   KEYSTOP EQU KEYRET+1
004E 0091   KEYON   EQU KEYSTOP+1
004E 0092   KEYAUT  EQU KEYON+1
004E 0093   KEYDEL  EQU KEYAUT+1
004E 0094   KEYPLT  EQU KEYDEL+1
004E 0095   KEYWAI  EQU KEYPLT+1
004E 0096   KEYPRT  EQU KEYWAI+1
004E 0097   KEYOEF  EQU KEYPRT+1
004E 009B   KEYCON  EQU KEYDEF+1
004E 0099   KEYLIS  EQU KEYCON+1
004E 009A   KEYEDI  EQU KEYLIS+1
004E 009B   KEYCLR  EQU KEYEDI+1
004E 009C   KEYCLD  EQU KEYCLR+1
004E 009D   KEYCSV  EQU KEYCLD+1
004E 009E   KEYNEW  EQU KEYCSV+1
004E 009F   KEYSET  EQU KEYNEW+1
004E 00A0   KEYSUGR EQU KEYSET+1
004E 00A0   KEYLSBL EQU KEYSUGR
004E 00A0   KEYTHEN EQU KEYSUGR
004E 00A1   KEYTO   EQU KEYTHEN+1
004E 00A2   KEYSTEP EQU KEYTO+1
004E 00A3   KEYLSBH EQU KEYSTEP+1
004E 00A3   KEYPRM  EQU KEYLSBH
004E 00A4   KEYLINE EQU KEYPRM+1
004E 00A5   KEYLSAH EQU KEYLINE+1
004E 00A5   KEYTAB  EQU KEYLSAH
004E 00A6   KEYSPC  EQU KEYTAB+1
004E 00A7   KEYFN   EQU KEYSPC+1
004E 00A8   KEYNOT  EQU KEYFN+1
004E 00A9   KEYOFF  EQU KEYNOT+1
            ;
004E 00AA   KEYOPR  EQU KEYOFF+1        ;OPERATOR CODES
004E 00AA   KEYADD  EQU KEYOPR
004E 00AB   KEYSUB  EQU KEYADD+1
004E 00AC   KEYMUL  EQU KEYSUB+1
004E 00AD   KEYDIV  EQU KEYMUL+1
004E 00AE   KEYMOD  EQU KEYDIV+1
004E 00AF   KEYEXPT EQU KEYMOD+1
```

```
004E 00B0   KEYAND  EQU KEYEXPT+1
004E 00B1   KEYOR   EQU KEYAND+1
004E 00B2   KEYMAX  EQU KEYOR+1
004E 00B3   KEYMIN  EQU KEYMAX+1
            ;
004E 00B4   KEYREL  EQU KEYMIN+1        ;RELATION CODES
004E 00B4   KEYGT   EQU KEYREL
004E 00B5   KEYEQ   EQU KEYGT+1
004E 00B6   KEYLT   EQU KEYEQ+1
            ;
004E 00B7   KEYFCT  EQU KEYLT+1         ;FUNCTION CODES
004E 00B7   KEYSGN  EQU KEYFCT
004E 00B8   KEYINT  EQU KEYSGN+1
004E 00B9   KEYABS  EQU KEYINT+1
004E 00BA   KEYSQR  EQU KEYABS+1
004E 00BB   KEYRND  EQU KEYSQR+1
004E 00BC   KEYLOG  EQU KEYRND+1
004E 00BD   KEYEXP  EQU KEYLOG+1
004E 00BE   KEYCOS  EQU KEYEXP+1
004E 00BF   KEYSIN  EQU KEYCOS+1
004E 00C0   KEYTAN  EQU KEYSIN+1
004E 00C1   KEYATA  EQU KEYTAN+1
004E 00C2   KEYUSR  EQU KEYATA+1
004E 00C3   KEYFRE  EQU KEYUSR+1
004E 00C4   KEYPORT EQU KEYFRE+1
004E 00C5   KEYPOS  EQU KEYPORT+1
004E 00C6   KEYMEM  EQU KEYPOS+1
004E 00C7   KEYLEN  EQU KEYMEM+1
004E 00C8   KEYSTR  EQU KEYLEN+1
004E 00C9   KEYVAL  EQU KEYSTR+1
004E 00CA   KEYASC  EQU KEYVAL+1
004E 00CB   KEYCHR  EQU KEYASC+1
004E 00CC   KEYHEX  EQU KEYCHR+1
004E 00CD   KEYHXV  EQU KEYHEX+1
004E 00CE   KEYUPR  EQU KEYHXV+1
004E 00CF   KEYLFT  EQU KEYUPR+1
004E 00D0   KEYRIG  EQU KEYLFT+1
004E 00D1   KEYMID  EQU KEYRIG+1
004E 00D2   KEYINS  EQU KEYMID+1
            ;
004E 00D3   KEYS    EQU KEYINS+1        ;LAST ENTRY
```

```
                STMTABL:                ;STATEMENT ROUTINES
004E 0000           DW      DATSTM
0050 0000           DW      REMSTM
                                        ;LISTED WITH BLANK AFTER
0052 0000           DW      ENDSTM
0054 0000           DW      FORSTM
0056 0000           DW      NEXSTM
0058 0000           DW      INPSTM
005A 0000           DW      DIMSTM
005C 0000           DW      REASTM
005E 0000           DW      LETSTM
0060 0000           DW      GTOSTM
0062 0000           DW      RUNSTM
0064 0000           DW      IFSTM
0066 0000           DW      ELSSTM
0068 0000           DW      RESSTM
006A 0000           DW      GSBSTM
006C 0000           DW      RETSTM
006E 0000           DW      STPSTM
0070 0000           DW      ONSTM
0072 0000           DW      AUTSTM
0074 0000           DW      DELSTM
0076 0000           DW      PLTSTM
0078 0000           DW      WAISTM
007A 0000           DW      PRTSTM
007C 0000           DW      DEFSTM
007E 0000           DW      CONSTM
0080 0000           DW      LISSTM
0082 0000           DW      EDISTM
0084 0000           DW      CLRSTM
0086 0000           DW      CLDSTM
0088 0000           DW      CSVSTM
008A 0000           DW      NEWSTM
008C 0000           DW      SETSTM
```

```
                OPRTABL:                ;OPERATORS AND PRECEDENCE
008E 79             DB      79H
008F 0000           DW      ADDOPR
0091 79             DB      79H
0092 0000           DW      SUBOPR
0094 7B             DB      7BH
0095 0000           DW      MULOPR
0097 7B             DB      7BH
0098 0000           DW      DIVDPR
009A 7B             DB      7BH
009B 0000           DW      MDDDPR
009D 7F             DB      7FH
009E 0000           DW      EXPOPR
00A0 50             DB      50H
00A1 0000           DW      ANDDPR
00A3 46             DB      46H
00A4 0000           DW      ORNOPR
00A6 76             DB      76H
00A7 0000           DW      MAXDPR
00A9 76             DB      76H
00AA 0000           DW      MINOPR
```

```
                FCTTABL:                ;FUNCTION ROUTINES
00AC 0000           DW      SGNFCT
00AE 0000           DW      INTFCT
00B0 0000           DW      ABSFCT
00B2 0000           DW      SQRFCT
00B4 0000           DW      RNDFCT
00B6 0000           DW      LOGFCT
00B8 0000           DW      EXPFCT
00BA 0000           DW      COSFCT
00BC 0000           DW      SINFCT
00BE 0000           DW      TANFCT
00C0 0000           DW      ATNFCT
00C2 0000           DW      ERRAFC
00C4 0000           DW      FREFCT
00C6 0000           DW      PORFCT
00C8 0000           DW      POSFCT
00CA 0000           DW      MEMFCT
00CC 0000           DW      LENFCT
00CE 0000           DW      STRFCT
00D0 0000           DW      VALFCT
00D2 0000           DW      ASCFCT
00D4 0000           DW      CHRFCT
00D6 0000           DW      HEXFCT
00D8 0000           DW      HXVFCT
00DA 0000           DW      UPRFCT
00DC 0000           DW      LFTFCT
00DE 0000           DW      RIGFCT
00E0 0000           DW      MIDFCT
00E2 0000           DW      INSFCT
```

```
                KEYWADDS:                       ;POINTERS TO KEYWORD GROUPS
00E4 000000         DW      KEYWRD0, KEYWRD1, KEYWRD2, KEYWRD3
00E7 000000
00EA 0000
00EC 000000         DW      KEYWRD4, KEYWRD5, KEYWRD6, KEYWRD7
00EF 000000
00F2 0000
00F4 000000         DW      KEYWRD8, KEYWRD9, KEYWRDA, KEYWRDB
00F7 000000
00FA 0000
00FC 000000         DW      KEYWRDC, KEYWRDD, KEYWRDE, KEYWRDF
00FF 000000
0102 0000
                KEYWORDS:
                KEYWRD0:
0104 94504C         DB      KEYPLT,     "PLO", 'T+128
0107 4FD4
0109 965052         DB      KEYPRT,     "PRIN", 'T+128
010C 494ED4
010F A35052         D8      KEYPRM,     "PROMP", 'T+128
0112 4F4D50
0115 D4
0116 C4504F         DB      KEYPORT,    "POR", 'T+128
0119 52D4
011B 45504F         DB      KEYPOS-B0H, "PO", 'S+128
011E D3
                KEYWRD1:
011F 924155         DB      KEYAUT,     "AUT", 'O+128
0122 54CF
0124 B0414E         DB      KEYAND,     "AN", 'D+128
0127 C4
012B B94142         DB      KEYABS,     "A8", 'S+128
012B D3
012C C14154         DB      KEYATA,     "AT", 'N+128
012F CE
0130 4A4153         DB      KEYASC-80H, "AS", 'C+128
0133 C3
                KEYWRD2:
0134 815245         DB      KEYREM,     "RE", 'M+128
0137 CD
0138 875245         DB      KEYREA,     "REA", 'D+128
013B 41C4
013D 8A5255         DB      KEYRUN,     "RU", 'N+128
0140 CE
0141 8D5245         DB      KEYRES,     "RESTOR", 'E+128
0144 53544F
0147 52C5
0149 8F5245         DB      KEYRET,     "RETUR", 'N+128
014C 545552
014F CE
0150 BB524E         D8      KEYRND,     "RN", 'D+128
0153 C4
0154 505249         DB      KEYRIG-80H, "RIGHT", '$+128
0157 474854
015A A4
                KEYWRD3:
```

```
015B 905354     DB      KEYSTOP,    "STO", 'P+128
015E 4FD0
0160 98434F     DB      KEYCON,     "CON", 'T+128
0163 4ED4
0165 9B434C     DB      KEYCLR,     "CLEA", 'R+128
016B 4541D2
016B 9D5341     DB      KEYCSV,     "SAV", 'E+128
016E 56C5
0170 9F5345     DB      KEYSET,     "SE", 'T+128
0173 D4
0174 A25354     DB      KEYSTEP,    "STE", 'P+128
0177 45D0
0179 A65350     DB      KEYSPC,     "SP", 'C+128
017C C3
017D B75347     DB      KEYSGN,     "SG", 'N+128
01B0 CE
0181 BA5351     DB      KEYSQR,     "SQ", 'R+128
01B4 D2
0185 BE434F     DB      KEYCOS,     "CO", 'S+12B
0188 D3
01B9 BF5349     DB      KEYSIN,     "SI", 'N+12B
01BC CE
01BD CB5354     DB      KEYSTR,     "STR", '$+128
0190 52A4
0192 4B434B     DB      KEYCHR-80H, "CHR", '$+12B
0195 52A4
                KEYWRD4:
0197 B04441     DB      KEYDAT,     "DAT", 'A+128
019A 54C1
019C B64449     DB      KEYDIM,     "DI", 'M+128
019F CD
01A0 934445     DB      KEYDEL,     "DELET", 'E+12B
01A3 4C4554
01A6 C5
01A7 974445     DB      KEYDEF,     "DE", 'F+12B
01AA C6
01AB A0544B     DB      KEYTHEN,    "THE", 'N+128
01AE 45CE
01B0 A154CF     DB      KEYTO,      "T", 'O+12B
01B3 A55441     DB      KEYTAB,     "TA", 'B+12B
01B6 C2
01B7 405441     DB      KEYTAN-80H, "TA", 'N+128
01BA CE
                KEYWRD5:
01BB B2454E     DB      KEYEND,     "EN", 'D+128
01BE C4
01BF BC454C     DB      KEYELS,     "ELS", 'E+128
01C2 53C5
01C4 9A4544     DB      KEYEDI,     "EDI", 'T+128
01C7 49D4
01C9 BD455B     DB      KEYEXP,     "EX", 'P+128
01CC D0
01CD C25553     DB      KEYUSR,     "US", 'R+12B
01D0 D2
01D1 4E5550     DB      KEYUPR-80H, "UPPER", '$+12B
01D4 504552
```

```
01D7 A4
                KEYWRD6:
01D8 83464F         DB      KEYFOR,     "FO", 'R+128
01DB D2
01DC A746CE         DB      KEYFN,      "F", 'N+128
01DF C34652         DB      KEYFRE,     "FR", 'E+128
01E2 C5
01E3 495641         DB      KEYVAL-80H, "VA", 'L+128
01E6 CC
                KEYWRD7:
01E7 89474F         DB      KEYGTO,     "GOT", 'O+128
01EA 54CF
01EC 8E474F         DB      KEYGS8,     "GOSU", '8+128
01EF 5355C2
01F2 155741         DB      KEYWAI-80H, "WAI", 'T+128
01F5 49D4
                KEYWRD8:
01F7 CC4845         D8      KEYHEX,     "HEX", '$+128
01FA 58A4
01FC 4D4845         DB      KEYHXV-80H, "HEX", 'V+128
01FF 58D6
                KEYWRD9:
0201 85494E         DB      KEYINPT,    "INPU", 'T+128
0204 5055D4
0207 BB49C6         DB      KEYIF,      "I", 'F+128
020A B8494E         DB      KEYINT,     "IN", 'T+128
020D D4
020E 52494E         DB      KEYINS-80H, "INST", 'R+128
0211 5354D2
                KEYWRDA:
0214 2CAA           DB      KEYMUL-80H, '*+128
                KEYWRDB:
0216 2AAB           DB      KEYADD-80H, '++128
                KEYWRDC:
0218 B84C45         DB      KEYLET,     "LE", 'T+128
0218 D4
021C 994C49         DB      KEYLIS,     "LIS", 'T+128
021F 53D4
0221 9C4C4F         DB      KEYCLD,     "LOA", 'D+128
0224 41C4
0226 A44C49         DB      KEYLINE,    "LIN", 'E+128
0229 4EC5
022B B6BC           DB      KEYLT,      '<+128
022D BC4C4F         DB      KEYLOG,     "LO", 'G+128
0230 C7
0231 C74C45         DB      KEYLEN,     "LE", 'N+128
0234 CE
0235 4F4C45         D8      KEYLFT-80H, "LEFT", '$+128
0238 4654A4
                KEYWRDD:
023B ABAD           DB      KEYSUB,     '-+128
023D AE4D4F         DB      KEYMOD,     "MO", 'D+128
0240 C4
0241 B24D41         DB      KEYMAX,     "MA", 'X+128
0244 D8
0245 B34D49         DB      KEYMIN,     "MI", 'N+128
```

```
0248 CE
0249 B5BD       DB      KEYEQ,      '=+128
024B C64D45     DB      KEYMEM,     "ME", 'M+128
024E CD
024F 514D49     DB      KEYMID-80H, "MID", '$+128
0252 44A4
             KEYWRDE:
0254 B44E45     DB      KEYNEX,     "NEX", 'T+128
0257 58D4
0259 9E4E45     DB      KEYNEW,     "NE", 'W+128
025C D7
025D A84E4F     DB      KEYNOT,     "NO", 'T+128
0260 D4
0261 AFDE       DB      KEYEXPT,    '↑+128
0263 34BE       DB      KEYGT-80H,  '>+128
             KEYWRDF:
0265 968F       DB      KEYPRT,     '?+128
0267 914FCE     DB      KEYON,      "O", 'N+128
026A A94F46     DB      KEYOFF,     "OF", 'F+128
026D C6
026E ADAF       DB      KEYDIV,     '/+128
0270 314FD2     DB      KEYOR-80H,  "O", 'R+128
```

```
                ERRN:                                   ;ERROR CODES
                ERRNCN:
0273 434F4E         DB      "CONTINUE",0                ;CONTINUE ERROR
0276 54494E
0279 554500
                ERRNSL:
027C 444556         DB      "DEVICE",0                  ;SAVE/LOAD DEVICE ERROR
027F 494345
0282 00
                ERRNDD:
0283 44494D         DB      "DIMENSION",0               ;DOUBLE DIMENSION
0286 454E53
0289 494F4E
028C 00
                ERRNID:
028D 444952         DB      "DIRECT",0                  ;ILLEGAL DIRECT
0290 454354
0293 00
                ERRNDO:
0294 444956         DB      "DIVIDE BY 0",0             ;DIVISION BY ZERO
0297 494445
029A 204259
029D 203000
                ERRNFC:
02A0 46554E         DB      "FUNCTION CALL",0           ;FUNCTION CALL
02A3 435449
02A6 4F4E20
02A9 43414C
02AC 4C00
                ERRNLS:
02AE 4C4F4E         DB      "LONG STRING",0             ;LONG STRING
02B1 472053
02B4 545249
02B7 4E4700
                ERRNOM:
02BA 4D454D         DB      "MEMORY SPACE",0            ;OUT OF MEMORY
02BD 4F5259
02C0 205350
02C3 414345
02C6 00
                ERRNNF:
02C7 4E4558         DB      "NEXT W/O FOR",0            ;NEXT WITHOUT FOR
02CA 542057
02CD 2F4F20
02D0 464F52
02D3 00
                ERRNOD:
02D4 4F5554         DB      "OUT OF DATA",0             ;OUT OF DATA
02D7 204F46
02DA 204441
02DD 544100
                ERRNOV:
02E0 4F5645         DB      "OVERFLOW",0                ;OVERFLOW
02E3 52464C
02E6 4F5700
                ERRNRG:
```

```
02E9 524554     DB      "RETN W/O GOSUB",0      ;RETURN WITHOUT GOSUB
02EC 4E2057
02EF 2F4F20
02F2 474F53
02F5 554200
             ERRNOS:
02F8 535452     DB      "STRING SPACE",0        ;OUT OF STRING SPACE
02FB 494E47
02FE 205350
0301 414345
0304 00
             ERRNST:
0305 535452     DB      "STRING TEMPS",0        ;STRING TEMPORARIES
0308 494E47
030B 205445
030E 4D5053
0311 00
             ERRNBS:
0312 535542     DB      "SUBSCRIPT",0           ;BAD SUBSCRIPT
0315 534352
0318 495054
031B 00
             ERRNSN:
031C 53594E     DB      "SYNTAX",0              ;SYNTAX ERROR
031F 544158
0322 00
             ERRNTM:
0323 545950     DB      "TYPE",0                ;TYPE MISMATCH
0326 4500
             ERRNUF:
0328 554E44     DB      "UNDFND FUNCTION",0     ;UNDEFINED FUNCTION
032B 464E44
032E 204655
0331 4E4354
0334 494F4E
0337 00
             ERRNUS:
0338 554E44     DB      "UNDFND LINE",0         ;UNDEFINED STATEMENT
033B 464E44
033E 204C49
0341 4E4500
             ERRNUV:
0344 554E44     DB      "UNDFND VARIABLE",0     ;UNDEFINED VARIABLE
0347 464E44
034A 205641
034D 524941
0350 424C45
0353 00
             ERRNFI:
0354 46696C     DB      "File not Saved",0      ;unknown file name
0357 65206E
035A 6F7420
035D 536176
0360 656400
```

```
                ;
                ;  INTERPRETER VARIABLES
                ;
                ;            VARIABLES MARKED WITH SAME CHARACTER IN COLUMN 71
                ;               ARE FIXED IN THAT ORDER.
                ;

0363 01     p3010:      db      1               ;0 to print on 3010
0364 00     REAINPFL:   DB      0               ;READ/INPUT FLAG
0365 00     PRINTFLG:   DB      0               ;PRINT/NO PRINT FLAG
0366 01     TRACEFLG:   DB      1               ;TRACE/NO TRACE FLAG
0367 00     SCANPFLG:   DB      0               ;SCAN/NOSCAN PARENTHESIS FLAG
0368 01     SCANPFLE:   DB      1               ;ARRAY NAME FOR ERASE
0369 00AB   SCANPFLD    EQU     KEYS-'( ;NO ARRAY ELEMENTS WANTED
0369 00     MATSCCNT:   DB      0               ;SUBSCRIPT COUNT
036A 00     MATDMFLG:   DB      0               ;SCANNING FOR VAR/DIMENSION  V
036B 00     TYPEFLG:    DB      0               ;TYPE FLAG                   V
036C 0002   TYPEINTG    EQU     2               ;TYPE OF INTEGER
036C 0003   TYPESTRG    EQU     3               ;TYPE OF STRING
036C 0004   TYPESING    EQU     4               ;TYPE OF SINGLE FLOATING POINT
036C 0008   TYPEDUBL    EQU     8               ;TYPE OF DOUBLE FLOATING POINT
036C 0020   TYPEDEF     EQU     080H/4          ;MARKING BIT FOR USER-FUNCTION

036C 00     STRGTMPL:   DB      0               ;TEMP STRING DESCRPTR, LEN   S
036D 0000   STRGTMPA:   DW      0               ;TEMP STRING DESCRPTR, ADDR  S
036F 0000   SCANPTR1:   DW      0               ;SCAN POINTER
0371 0000   SCANPTR2:   DW      0               ;SCAN POINTER
0373 FFFF   CURLINE:    DW      -1              ;CURRENT LINE NUMBER
0375 0000   CURLINES:   DW      0               ;SAVED CURRENT LINE NUMBER
0377 0000   PROGCNTR:   DW      ENDINTRP+12     ;CURRENT PROGRAM LOCATION
0379 0377   VARINDEX    EQU     PROGCNTR        ;INDEX VARIABLE OF FOR
0379 0000   PROGCNTS:   DW      0               ;SAVED CURRENT PROGRAMLOCATION
037B 0000   CURLDATA:   DW      0               ;CURRENT DATA LINE NUMBER
037D 0000   CURDATAP:   DW      ENDINTRP        ;CURRENT DATA POINTER
037F 0000   INPTBUFR:   DW      INITSTSP        ;INPUT BUFFER ADDRESS
03B1 0064   PREDREL     EQU     064H            ;PRECEDENCE OF RELATION
0381 0070   PREDNUM     EQU     070H            ;LOWER BNDRY OF NUM OP PREC.
03B1 005A   PREDNOT     EQU     05AH            ;PRECEDENCE DF NOT OPERATOR
03B1 007D   PREDUMIN    EQU     07DH            ;PRECEDENCE OF UNARY MINUS
03B1 009D   LINESYZE    EQU     79+7B           ;DEFAULT LINESYZE
0381 000E   ITEMSIZE    EQU     14              ;DEFAULT WIDTH OF PRINT ITEM
```

```
                ;
                ;  MEMORY ALLOCATION POINTERS
                ;


                ;
0381 8000   LIMLOWER    EQU     08000H
0381 AF00   LIMUPPER    EQU     0AF00H
                ;
                ;  MEMORY LAYOUT
                ;
                ;  ENCOOE BUFFER
                ;  PROGRAM
                ;  VARIABLES
                ;  ARRAYS
                ;  FREE SPACE  /  STACK (INCLUOING BUFFERS)
                ;  FREE STRING SPACE
                ;  STRINGS
                ;  STRING TEMPORARIES
                ;  FREE STRING TEMPORARIES
                ;
0381 0000   PROGBASE:   OW      ENDINTRP+13     ;BASE OF PROGRAM SPACE
0383 0000   VARTABLE:   OW      ENDINTRP+15     ;BASE OF VARIABLE TABLE
0385 0000   MATTABLE:   DW      ENDINTRP+15     ;BASE OF ARRAY TABLE
03B7 0000   FREELIMT:   OW      ENDINTRP+15     ;LOWER LIMIT OF FREE SPACE
0389 0000   STCKBASE:   DW      INITSTCK        ;BASE OF STACK
03BB 0000   STRGFREE:   DW      INITSTCK+10     ;FIRST FREE STRING SPACE
03BD 0000   STRGBASE:   DW      INITSTCK+10     ;BASE OF STRING SPACE
038F 0000   STRGTMPP:   OW      INITSTCK+11     ;STRING TEMPORARY ALLOC PTR
0391 0000   STRGTLIM:   OW      INITSTCK+10+2*3 ;STRING TEMPORARY LIMIT


0393 0000   ACCUMLTR:   DB      0,0     ;ACCUMULATOR                    A
0395 00     FLACCMSB:   DB      0       ;SIGN-BIT/HIGH-OROER MANTISSA   A
0396 00     FLACCEXP:   DB      0       ;EXPONENT                       A
0397 00     FLACCSSV:   DB      0       ;SAVEO SIGN                     A

0398 01     NULLCNT:    DB      1       ;# OF NULLS TO INSERT AFTER (CR)
0399 01     CURSPOS:    DB      1       ;CHARACTER CURSOR POSITION      C
039A 63     CURSLIM:    DB      -LINESYZE       ;OUTPUT CURSOR LIMIT     C

039B 00     FLSCR0:     DB      0       ;FLOATING POINT SCRATCH AREA
039C 01     FLSCR1:     OB      1
039D 02     FLSCR2:     DB      2
039E 03     FLSCR3:     OB      3

039F 039B   INOTINS     EQU     FLSCR0  ;INPUT/OUTPUT INSTRUCTIONS
039F 00DB   OPCINP      EQU     0DBH    ;INPUT INSTRUCTION
039F 00D3   OPCOUT      EQU     0D3H    ;OUTPUT INSTRUCTION
039F 00C9   OPCRET      EQU     0C9H    ;RETURN INSTRUCTION

039F 52C74F RNOFCTSO:   OB      052h, 0c7h, 04fh, 080h  ;RANDOM SEED
03A2 80
```

```
                ;
                ;
                ;  GENERAL USE SUBROUTINES
                ;
                ;
                ;  SCAN ONE CHARACTER AND CLASSIFY
                ;
                SCANNXTV:
03A3 7E             MOV     A,M     ;SCAN CURRENT BYTE,
03A4 E3             XTHL
03A5 BE             CMP     M       ;VERIFY MATCH,
03A6 23             INX     H
03A7 E3             XTHL
03A8 C20000         JNZ     ERRASN  ;SQUAWK ABOUT SYNTAX ERROR
                SCANNXT:
03AB 23             INX     H       ;SCAN FOR NEXT NON-BLANK CHAR
03AC 7E             MOV     A,M     ;C=NUMERIC CHARACTER
03AD FE3A           CPI     ":"     ;Z=END OF STATEMENT
03AF D0             RNC
03B0 FE20           CPI     " "
03B2 CAAB03         JZ      SCANNXT
03B5 FE30           CPI     "0"
03B7 3F             CMC
03B8 3C             INR     A
03B9 3D             DCR     A
03BA C9             RET
```

```
                ;
                ;  TEST FOR ALPHABETIC CHARACTER
                ;
                ALPHACHK:
03BB 7E             MOV     A,M     ;TEST FOR ALPHABETIC CHARACTER
                ALPHACHA:
03BC FE7B           CPI     'z+1    ;LOWER CASE
03BE D0             RNC
03BF FE61           CPI     "a"     ;LOWER CASE
03C1 D20000         JNC     ALPHACHL
03C4 FE5B           CPI     'Z+1    ;C=ALPHABETIC
03C6 D0             RNC
03C7 FE41           CPI     "A"     ;UPPER CASE
03C9 3F             CMC
03CA C9             RET
                ALPHACHL:
03CB C6E0           ADI     'A-'a   ;CONVERT LOWER TO UPPER
03CD C9             RET

                ;
                ;  MATCH CHARACTER OF BUFFER AGAINST CHARACTER IN A
                ;
                CHARMTCH:
03CE AE             XRA     M       ;MAKE MATCH TEST
03CF C8             RZ              ;Z=SUCCESS]
03D0 FE20           CPI     'a-'A   ;LOWER CASE - UPPER CASE
03D2 C0             RNZ             ;NOT LOWER-UPPER DIFFERENCE
03D3 CDBB03         CALL    ALPHACHK        ;ALPHABETIC?
03D6 9F             SBB     A
03D7 3C             INR     A       ;Z=C,S=0
03D8 C9             RET

                ;
                ;  CHECK TYPE OF EXPRESSION                LEN  CHAR
                ;             RETURNS:  S => INTEGER        2    %
                ;                       Z => STRING         3    $
                ;                      PO => SINGLE         4    @
                ;                      NC => DOUBLE         8    #
                ;
                TYPECHK:
03D9 3A6B03         LDA     TYPEFLG
                TYPECHKA:
03DC FE05           CPI     TYPESING+1
03DE 3D             DCR     A
03DF 3D             DCR     A
03E0 3D             DCR     A
03E1 B7             ORA     A
03E2 37             STC
03E3 C9             RET
```

```
                ;
                ;  SCAN A PAIR OF LINE NUMBER PARAMETERS
                ;
                SCANLPRZ:
03E4 010000         LXI     B,0     ;DEFAULT SECOND IS FIRST
                SCANLPRM:
03E7 C40000         CNZ     SCANLINN        ;DEFAULT FIRST IS IN DE
03EA F5             PUSH    PSW
03EB 78             MOV     A,B
03EC B1             ORA     C       ;ZERO DEFAULT IS FIRST PARAMETER
03ED C20000         JNZ     SCANLPR1
03F0 42             MOV     B,D
03F1 4B             MOV     C,E
                SCANLPR1:
03F2 F1             POP     PSW
03F3 EB             XCHG
03F4 E3             XTHL            ;PUT FIRST ONTO STACK
03F5 E5             PUSH    H
03F6 EB             XCHG
03F7 50             MOV     D,B
03F8 59             MOV     E,C
03F9 C8             RZ
03FA FEAD           CPI     KEYDIV  ;SEPARATOR MUST BE "/",
03FC CA0000         JZ      SCANLPR2
03FF CDA303         CALL    SCANNXTV        ;bscan (val)
0402 2C             DB      ","     ;   OR ","
0403 2B             DCX     H
                SCANLPR2:
0404 11FFFF         LXI     D,0FFFFH        ;EMPTY SECOND OPERAND = END
0407 CDAB03         CALL    SCANNXT ;bscan ,
040A C8             RZ

                ;
                ;  SCAN A LINE NUMBER
                ;
                SCANLINN:
040B 2B             DCX     H       ;SCAN LINE # IN COMMAND/STATEMENT
                SCANLINR:
040C 110000         LXI     D,0     ;DEFAULT LINE IS 0, INITIALIZE
                SCANLINL:
040F CDAB03         CALL    SCANNXT ;bscan ,
0412 D0             RNC
0413 E5             PUSH    H
0414 F5             PUSH    PSW
0415 219819         LXI     H,0FFFFH/10-1
0418 CD0000         CALL    CMHLLTDE
041B DA0000         JC      ERRASN
041E 62             MOV     H,D
041F 6B             MOV     L,E     ;HL=10*DE
0420 19             DAD     D
0421 29             DAD     H
0422 19             DAD     D
0423 29             DAD     H
0424 F1             POP     PSW
0425 D630           SUI     "0"     ;GET VALUE OF NEXT DIGIT
```

```
0427 5F         MOV     E,A
0428 1600       MVI     D,000H
042A 19         DAD     D       ;AND ADD IT ON
042B EB         XCHG
042C E1         POP     H
042D C30F04     JMP     SCANLINL
```

```
                ;
                ;  SEARCH FOR A GIVEN LINE NUMBER
                ;
                LINESRCH:
0430 2A8103         LHLD    PROGBASE        ;LOOK FOR LINE NUMBER IN DE
                LINESRCL:
0433 E5             PUSH    H       ;C=LINE FOUND
0434 CD0000         CALL    LINELINK        ;BC=LINE LOCATION, IF FOUND
0437 CA0000         JZ      POPHLRET        ;=NEXT LINE, IF NOT FOUND
043A C5             PUSH    B       ;ADDRESS OF NEXT LINE
043B 7E             MOV     A,M     ;GET NUMBER OF CURRENT LINE
043C 23             INX     H
043D 66             MOV     H,M     ;(from HL,MA)
043E 6F             MOV     L,A
043F CD0000         CALL    CMHLLTDE
0442 E1             POP     H       ;HL=NEXT LINE
0443 C1             POP     B
0444 3F             CMC
0445 C8             RZ
0446 D23304         JNC     LINESRCL
0449 60             MOV     H,B
044A 69             MOV     L,C
044B 3F             CMC
044C C9             RET

                ;
                ;  LINK TO NEXT LINE
                ;
                LINELINK:
044D E5             PUSH    H       ;FIND ADDRESS OF NEXT LINE
044E 4E             MOV     C,M     ;Z=END OF PROGRAM
044F 23             INX     H
0450 46             MOV     B,M
0451 23             INX     H
0452 E3             XTHL
0453 09             DAD     B       ;ADD LENGTH TO ADDRESS
0454 E3             XTHL
0455 78             MOV     A,B
0456 B1             ORA     C
0457 C1             POP     B
0458 C9             RET
```

```
                ;
                ;  INSERT/REPLACE LINE OF PROGRAM
                ;
                LINEINS:
0459 D5             PUSH    D       ;DE=LINE NUMBER
045A D40000         CNC     KEYSCAN ;C=ALREADY KEY-SCANNED
045D CDAB03         CALL    SCANNXT ;bscan ,        ;NC=MUST BE KEY-SCANNED
0460 D1             POP     D
0461 E5             PUSH    H       ;HL=TEXT TO INSERT
0462 D5             PUSH    D
0463 C5             PUSH    B       ;BC=LENGTH OF TEXT
0464 F5             PUSH    PSW     ;Z=DELETE, NO REPLACE
0465 CD3004         CALL    LINESRCH        ;LOOK FOR LINE
0468 C5             PUSH    B       ;SAVE LOCATION
0469 DC0000         CC      LINEDEL ;DELETE IF PRESENT
046C D1             POP     D
046D F1             POP     PSW
046E CA0000         JZ      POPHL3RT        ;EXIT IF NOTHING MORE
0471 2A8703         LHLD    FREELIMT        ;PULL APART FOR NEW LINE
0474 E3             XTHL
0475 C1             POP     B
0476 E5             PUSH    H
0477 09             DAD     B
0478 CD0000         CALL    COPYCHK
047B EB             XCHG
047C C1             POP     B
047D 71             MOV     M,C     ;BEGINNING OF NEW LINE
047E 23             INX     H
047F 70             MOV     M,B
0480 23             INX     H
0481 D1             POP     D
0482 73             MOV     M,E     ;INSERT LINE NUMBER
0483 23             INX     H
0484 72             MOV     M,D
0485 23             INX     H
0486 EB             XCHG
0487 E1             POP     H       ;RECOVER TEXT POINTER
                LINEINSL:
0488 7E             MOV     A,M     ;INSERT TEXT OF NEW LINE
0489 12             STAX    D
048A 23             INX     H
048B 13             INX     D
048C B7             ORA     A
048D C28804         JNZ     LINEINSL
0490 C30000         JMP     LINEDELU
```

```
                ;
                ;  DELETE TEXT FROM PROGRAM
                ;
                LINEDEL:
0493 EB             XCHG            ;BC=BEGINNING OF TEXT TO REMOVE
0494 79             MOV     A,C
0495 93             SUB     E       ;COMPUTE NEGATIVE OF
0496 6F             MOV     L,A     ;NUMBER OF BYTES DELETED
0497 7B             MOV     A,B
049B 9A             SBB     D
0499 67             MOV     H,A
049A E5             PUSH    H
049B 2AB703         LHLD    FREELIMT        ;HL=BEGINNING OF TEXT SURVIVING
                LINEDELL:
049E 1A             LDAX    D
049F 02             STAX    B
04A0 03             INX     B
04A1 13             INX     D
04A2 CD0000         CALL    CMHLLTDE
04A5 D29E04         JNC     LINEDELL
04AB C1             POP     B
                LINEDELU:
04A9 2AB703         LHLD    FREELIMT        ;UPDATE DATA POINTERS
04AC 09             DAD     B       ;BC=INCREMENT
04AD 22B703         SHLD    FREELIMT
04B0 2AB503         LHLD    MATTABLE
04B3 09             DAD     B
04B4 228503         SHLD    MATTABLE
04B7 2AB303         LHLD    VARTABLE
04BA 09             DAD     B
04BB 22B303         SHLD    VARTABLE
04BE C30000         JMP     CLEARPCN

                ;
                ;  MAKE SIXTEEN BIT COMPARISON
                ;
                CMHLLTDE:
04C1 7C             MOV     A,H     ;COMPARE DE VS HL
04C2 92             SUB     D       ;C=HL<DE
04C3 C0             RNZ
04C4 7D             MOV     A,L
04C5 93             SUB     E
04C6 C9             RET
```

```
                ;
                ;  MOVE LONG TO HIGHER ADDRESS
                ;
                COPYCHK:
04C7 CD0000         CALL    SPACECHK
                COPYTEXT:
04CA C5             PUSH    B       ;COPY SECTION DE-BC TO AREA
04CB E3             XTHL            ;ENOING AT HL
04CC C1             POP     B
                COPYTXTL:
04CD CDC104         CALL    CMHLLTDE
04D0 7E             MOV     A,M
04D1 02             STAX    B
04D2 C8             RZ
04D3 0B             DCX     B
04D4 2B             DCX     H
04D5 C3CD04         JMP     COPYTXTL

                ;
                ;  CHECK SPACE FOR STACK ALLOCATION
                ;
                SPACESTK:
04D8 E5             PUSH    H       ;VERIFY STACK HAS ROOM ENOUGH
04D9 2A8703         LHLD    FREELIMT        ;C=NUMBER OF WORDS NEEDED
04DC 0600           MVI     B,000H
04DE 09             DAD     B
04DF 09             DAD     B
04E0 CD0000         CALL    SPACECHK
04E3 E1             POP     H
04E4 C9             RET

                ;
                ;  CHECK SPACE FOR PROGRAM OR VARIABLE ALLOCATION
                ;
                SPACECHK:
04E5 D5             PUSH    D       ;CHECK THAT ENOUGH SPACE IS LEFT
04E6 EB             XCHG            ;ON STACK ABOVE HL
04E7 21DAFF         LXI     H,-38
04EA 39             DAD     SP
04EB CDC104         CALL    CMHLLTDE
04EE EB             XCHG
04EF D1             POP     D
04F0 D0             RNC
                ERRAOM:
04F1 1E47           MVI     E,ERRNOM-ERRN
04F3 C30000         JMP     ERRMSG
```

```
                ;
                ;  RE-INITIALIZATION ROUTINES
                ;
                NEWSTM:
04F6 C0             RNZ             ;NEW COMMAND
                CLEARPGM:
04F7 2A8103         LHLD    PROG8ASE        ;CLEAR PROGRAM
04FA AF             XRA     A
04F8 77             MOV     M,A
04FC 23             INX     H
04FD 77             MOV     M,A
04FE 23             INX     H
                NEWLOAD:
04FF 228303         SHLD    VARTA8LE
                CLEARSET:
0502 CD0000         CALL    CLEARPCN        ;CLEAR PROGRAM POINTERS
                CLEARVST:
0505 227703         SHLD    PROGCNTR        ;UPDATE PROGRAM COUNTER
0508 CD0000         CALL    CLEARVAR        ;CLEAR VARIABLES
                CLEARSTK:
050B C1             POP     8       ;RESET STACK,
050C 2A8903         LHLD    STCK8ASE
050F F9             SPHL
0510 2160FF         LXI     H,0-LINESYZE-3
0513 39             DAD     SP
0514 F9             SPHL            ;CREATE INPUT BUFFER
0515 227F03         SHLD    INPTBUFR
0518 2A8D03         LHLD    STRG8ASE        ;CLEAR STRING TEMPORARIES,
051B 23             INX     H
051C 228F03         SHLD    STRGTMPP
051F 210000         LXI     H,0
0522 E5             PUSH    H
0523 227903         SHLD    PROGCNTS        ;SET NO CONTINUE
0526 2A7703         LHLD    PROGCNTR
0529 C5             PUSH    B
052A C9             RET

                CLEARVAR:
052B 2A8303         LHLD    VARTABLE        ;CLEAR ALL VARIABLES
052E 228503         SHLD    MATTA8LE
0531 228703         SHLD    FREELIMT
0534 2A8D03         LHLD    STRG8ASE
0537 228B03         SHLD    STRGFREE
053A C9             RET

                CLEARPCN:
053B 210000         LXI     H,0     ;CLEAR PROGRAM POINTERS
053E 227903         SHLD    PROGCNTS
0541 2A8103         LHLD    PROGBASE
0544 2B             DCX     H
0545 3600           MVI     M,0     ;END OF LINE -1
0547 227703         SHLD    PROGCNTR
054A AF             XRA     A
```

```
                ;
                ;  RESTORE:  REWIND DATA STATEMENTS
                ;
                RESSTM:
0548 CA0000         JZ      RESSTMDF        ;RESTDRE STATEMENT
054E CD0B04         CALL    SCANLINN
0551 E5             PUSH    H
0552 CD3004         CALL    LINESRCH
0555 D20000         JNC     ERRAUS
0558 E1             POP     H
0559 E8             XCHG
055A C30000         JMP     RESSTM8U
                RESSTMDF:
055D E8             XCHG            ;DEFAULT IS RESTORE TO BEGINNING
055E 2A8103         LHLD    PRDG8ASE
                RESSTMBU:
0561 2B             DCX     H       ;BACK UP BEFORE LINE
                RESDTPTR:
0562 227D03         SHLD    CURDATAP        ;SET DATA POINTER
0565 E8             XCHG
0566 C9             RET


                ;
                ;  CLEAR:  CLEAR VARIABLES, REALLOCATE STRING SPACE
                ;
                CLRSTM:
0567 CA0505         JZ      CLEARVST        ;CLEAR STATEMENT
056A CD0000         CALL    VALINTDE
056D 28             DCX     H       ;bscan -
056E CDAB03         CALL    SCANNXT ;bscan ,
0571 C0             RNZ
0572 E5             PUSH    H
0573 2A8D03         LHLD    STRGBASE
0576 7D             MOV     A,L
0577 93             SUB     E
0578 5F             MOV     E,A
0579 7C             MOV     A,H
057A 9A             SBB     D
0578 57             MOV     D,A
057C DA0000         JC      ERRASN
057F 2A8303         LHLD    VARTA8LE
0582 012800         LXI     B,40
0585 09             DAD     B
0586 CDC104         CALL    CMHLLTDE
0589 D2F104         JNC     ERRAOM
058C EB             XCHG
058D 228903         SHLD    STCK8ASE
0590 E1             POP     H
0591 C30505         JMP     CLEARVST
```

```
                ;
                ;  LOW-LEVEL CHARACTER I/O ROUTINES
                ;
                PRNTCHRI:
0594 E3             XTHL
0595 7E             MOV     A,M
0596 23             INX     H
0597 E3             XTHL
                PRNTCHRA:
0598 F5             PUSH    PSW     ;TRANSMIT CHARACTER
0599 3A6503         LDA     PRINTFLG
059C B7             ORA     A
059D C20000         JNZ     POPAFRET
05A0 F1             POP     PSW
05A1 F5             PUSH    PSW
05A2 FE20           CPI     " "
05A4 DA0000         JC      PRNTCHRW
05A7 E5             PUSH    H
05A8 2A9903         LHLD    CURSPOS ;LINE TOO LONG?
05AB 7C             MOV     A,H
05AC 85             ADD     L
05AD 7D             MOV     A,L
05AE E1             POP     H
05AF DC0000         CC      PRNTCRLF
05B2 3C             INR     A
05B3 329903         STA     CURSPOS
                PRNTCHRW:
05B6 F1             POP     PSW     ;SEND CHARACTER
05B7 CD2A00         CALL    SYSDISPL
05BA C9             RET


                INPTCHAR:
05BB CD0600         CALL    SYSKEYIN        ;RECEIVE A CHARACTER
05BE CABB05         JZ      INPTCHAR        ;WAIT FOR ONE
05C1 FE0F           CPI     SI
05C3 C0             RNZ
05C4 3A6503         LDA     PRINTFLG
05C7 2F             CMA
05C8 326503         STA     PRINTFLG
05CB C3BB05         JMP     INPTCHAR
```

```
                ;
                ;  ERROR PROCESSING
                ;
                MSGERROR:
05CE 204552         DB      " ERROR",0
05D1 524F52
05D4 00
                MSGIN:
05D5 20494E         DB      " IN ",0
05D8 2000
                MSGOK:
05DA 0D0A4F         DB      CR,LF,"OK",CR,LF,0
05DD 4B0D0A
05E0 00
                MSGBREAK:
05E1 0D0A42         DB      CR,LF,"BREAK",0
05E4 524541
05E7 4B00

                ERRDATA:
05E9 2A7B03         LHLD    CURLDATA
05EC 227303         SHLD    CURLINE
                ERRASN:
05EF 1EA9           MVI     E,ERRNSN-ERRN
                ERRMSG:
05F1 CD0B05         CALL    CLEARSTK
05F4 AF             XRA     A
05F5 326503         STA     PRINTFLG        ;TURN ON PRINTING
05F8 326703         STA     SCANPFLG        ;ALLOW SUBSCRIPTING
05FB CD0000         CALL    PRNTCRLF
05FE 217302         LXI     H,ERRN
0601 57             MOV     D,A
0602 CD9405         CALL    PRNTCHRI        ;print (val)
0605 3F             DB      "?"
0606 19             DAD     D       ;PRINT ERROR CODE
0607 CD0000         CALL    PRNTMSG
060A 21CE05         LXI     H,MSGERROR
                ERRMSGPR:
060D CD0000         CALL    PRNTMSG
0610 2A7303         LHLD    CURLINE
0613 7C             MOV     A,H
0614 A5             ANA     L
0615 3C             INR     A
0616 C40000         CNZ     ERRMSGIN
```

```
                ;
                ;  COMMAND/LINE INPUT
                ;
                CMNDSTRT:
0619 AF             XRA     A       ;TOP LEVEL EXECUTIVE
061A 326503         STA     PRINTFLG        ;TURN ON PRINTING
061D 326703         STA     SCANPFLG        ;ALLOW SUBSCRIPTING
0620 21FFFF         LXI     H,-1
0623 227303         SHLD    CURLINE
0626 21DA05         LXI     H,MSGOK
0629 CD0000         CALL    PRNTMSG ;REQUEST COMMAND
                CMNDINPT:
062C 110000         LXI     D,MSGSTARS+2    ;INPUT COMMAND
062F CD0000         CALL    INPTRQST
0632 DA2C06         JC      CMNDINPT
0635 CDAB03         CALL    SCANNXT ;bscan ,
0638 F5             PUSH    PSW
0639 CD0B04         CALL    SCANLINN        ;SCAN OFF LINE NUMBER
063C D5             PUSH    D
063D CD0000         CALL    KEYSCAN ;SCAN STATEMENT
0640 D1             POP     D
0641 F1             POP     PSW
0642 D20000         JNC     EXECUTE ;DIRECT IF NO LINE NUMBER
0645 CD5904         CALL    LINEINS ;INSERT LINE AS REQUESTED
0648 C32C06         JMP     CMNDINPT

                CMNDRSTR:
064B CD0B05         CALL    CLEARSTK        ;ENTRY FOR RESTARTING
064E CD0000         CALL    PRNTCRLF
0651 210000         LXI     H,MSGREDO+11    ;TELL HIM WE"RE STARTING
0654 C30D06         JMP     ERRMSGPR
```

```
                ;
                ;  AUTOMATIC LINE-NUMBERED INPUT
                ;
                AUTSTMIN:
0657 D5             PUSH    D       ;SAVE LINE NUMBER
065B CD5904         CALL    LINEINS ;INSERT LINE
065B E1             POP     H       ;RECOVER LINE NUMBER,
065C D1             POP     D       ;INCREMENT
065D 19             DAD     D
065E DA0000         JC      ERRAOV
0661 C30000         JMP     AUTSTMN

                AUTSTM:
0664 C1             POP     B       ;REMOVE CALLER
                AUTSTMS:
0665 11EB03         LXI     D,1000  ;DEFAULT STARTING LINE NUMBER
0668 016400         LXI     B,100   ;DEFAULT INCREMENT VALUE
066B CDE703         CALL    SCANLPRM        ;SCAN PARAMETERS
066E C2EF05         JNZ     ERRASN
0671 E1             POP     H
0672 CD0000         CALL    PRNTCRLF
                AUTSTMN:
0675 D5             PUSH    D       ;SAVE INCREMENT
0676 E5             PUSH    H       ;AND NEXT LINE NUMBER
0677 CD0000         CALL    ENCODEHL        ;PROMPT WITH LINE NUMBER
067A EB             XCHG
067B 13             INX     D
067C CD0000         CALL    INPTRQST
067F D1             POP     D
06B0 DA0000         JC      AUTSTMBR
06B3 CDAB03         CALL    SCANNXT ;bscan ,
06B6 D25706         JNC     AUTSTMIN
06B9 3F             CMC
                AUTSTMBR:
06BA D1             PDP     D       ;TAKE A BREAK
06BB DA1906         JC      CMNDSTRT        ;END OF AUTO
06BE C36506         JMP     AUTSTMS ;GET NEW LINE NUMBER, INCREMENT
```

```
                ;
                ;  LEXICAL SCANNER / KEYWORD RECOGNITION
                ;
                KEYSCAN:
0691 0E05           MVI     C,5     ;SCAN INPUT LINE FOR KEYWORDS,
0693 54             MOV     D,H     ;CDNDENSE LINE DN TDP DF SELF
0694 5D             MOV     E,L
0695 2B             DCX     H       ;bscan -
0696 E5             PUSH    H
0697 CDAB03         CALL    SCANNXT ;bscan +
                KEYSCANL:
069A 7E             MOV     A,M
069B FE20           CPI     " "
069D CA0000         JZ      KEYSCANH        ;DELETE BLANKS
06A0 47             MOV     B,A
06A1 FE22           CPI     '"
06A3 CA0000         JZ      KEYSCANI        ;SWALLOW WHOLE STRING
06A6 B7             ORA     A
06A7 CA0000         JZ      KEYSCANX
06AA FE30           CPI     "0"     ;NON-KEYWORD
06AC DA0000         JC      KEYSCANK
06AF FE3C           CPI     "<"     ;   SD WE DDN"T  SCAN
06B1 DA0000         JC      KEYSCANP
                KEYSCANK:
06B4 C5             PUSH    B       ;SCAN FDR MATCHING KEYWORD
06B5 D5             PUSH    D
06B6 E5             PUSH    H
06B7 E60F           ANI     00FH    ;HASH CHARACTER
06B9 5F             MOV     E,A
06BA 1600           MVI     D,0
06BC 21E400         LXI     H,KEYWADDS      ;ADDRESS C"SPONDING KEYWORDS
06BF 19             DAD     D
06C0 19             DAD     D
06C1 5E             MOV     E,M
06C2 23             INX     H
06C3 56             MOV     D,M
06C4 EB             XCHG
06C5 C30000         JMP     KEYSCANB

                KEYSCANZ:
06C8 1A             LDAX    D
06C9 B7             DRA     A
06CA F20000         JP      KEYSCANN
                KEYSCANM:
06CD 78             MOV     A,B     ;MATCH, GET SYMBDL NUMBER
06CE F680           ORI     080H
06D0 C30000         JMP     KEYSCANF
                KEYSCANN:
06D3 23             INX     H       ;ADDRESS NEXT CHAR IN LINE
06D4 13             INX     D
06D5 0C             INR     C
                KEYSCANC:
06D6 1A             LDAX    D       ;CDMPARE LINE WITH KEYWDRD
06D7 E67F           ANI     07FH
06D9 CDCE03         CALL    CHARMTCH        ;COMPARE CHARACTERS
```

```
06DC CAC806     JZ      KEYSCANZ
06DF 79         MOV     A,C     ;MATCH ENOUGH YET?
06E0 FE03       CPI     3
06E2 DA0000     JC      KEYSCANA

06E5 CDBB03     CALL    ALPHACHK        ;STOP ON BREAK CHAR OK
06E8 2B         DCX     H
06E9 D2CD06     JNC     KEYSCANM
            KEYSCANA:
06EC EB         XCHG
            KEYSCANW:
06ED B6         ORA     M       ;SKIP OVER REST OF KEYWORD
06EE 23         INX     H
06EF F2ED06     JP      KEYSCANW
06F2 A8         XRA     B
            KEYSCANB:
06F3 46         MOV     B,M     ;GET CODE FOR KEYWORD
06F4 23         INX     H
06F5 EB         XCHG
06F6 E1         POP     H       ;RESTORE STARTING POSITION
06F7 E5         PUSH    H
06F8 0E00       MVI     C,0
06FA F2D606     JP      KEYSCANC
06FD 7E         MOV     A,M     ;NO MATCH, GET CHARACTER
            KEYSCANF:
06FE D1         POP     D       ;RECOVER OUTPUT POINTER
06FF D1         POP     D
0700 C1         POP     B
0701 063A       MVI     B,":"   ;CHECK FOR SPECIAL PROCESSING
0703 FE8C       CPI     KEYELS
0705 C20000     JNZ     KEYSCAND
0708 EB         XCHG
0709 70         MOV     M,B     ;INSERT COLON BEFORE ELSE
070A EB         XCHG
070B 13         INX     D
070C 0C         INR     C
            KEYSCAND:
070D FE80       CPI     KEYDAT
070F CA0000     JZ      KEYSCANI
0712 0600       MVI     B,0
0714 FE81       CPI     KEYREM
0716 CA0000     jz      keyscani
0719 FE9C       cpi     keycld  ;pass file name in load and save
071B CA0000     jz      keyscani
071E FE9D       cpi     keycsv
            KEYSCANI:
0720 CC0000     CZ      KEYSCANV
0723 B7         ORA     A
0724 CA0000     JZ      KEYSCANX
            KEYSCANP:
0727 12         STAX    D       ;INSERT SYMBOL IN MEMORY
0728 13         INX     D
0729 0C         INR     C
            KEYSCANH:
072A 23         INX     H
072B C39A06     JMP     KEYSCANL
```

```
                KEYSCANX:
072E E1             POP     H       ;EXIT KEYWORD TRANSLATION
072F 12             STAX    D       ;END OF STATEMENT
0730 13             INX     D
0731 12             STAX    D       ;END OF "PROGRAM"
0732 13             INX     D
0733 12             STAX    D
0734 47             MOV     B,A     ;LENGTH IN BC
0735 C9             RET

                ;
                ;  COPY BUFFER TEXT WITHOUT PROCESSING
                ;
                KEYSCANV:
0736 12             STAX    D       ;COPY TEXT VERBATIM TO STOPPER
0737 0C             INR     C
0738 13             INX     D
0739 23             INX     H
073A 7E             MOV     A,M
073B B7             ORA     A
073C C8             RZ
073D B8             CMP     B
073E C8             RZ
073F FE22           CPI     '"      ;STRING WITHIN TEXT?
0741 C23607         JNZ     KEYSCANV
0744 C5             PUSH    B
0745 47             MOV     B,A
0746 CD3607         CALL    KEYSCANV
0749 F1             POP     PSW
074A 47             MOV     B,A
074B 7E             MOV     A,M
074C B7             ORA     A       ;STRING TERMINATE ON END OF LINE?
074D C8             RZ
074E C33607         JMP     KEYSCANV
```

```
                ;
                ;  LINE INPUT ROUTINE
                ;
                INPTLNBS:
0751 2B             DCX     H       ;DELETE A CHARACTER FROM INPUT
0752 05             DCR     B
0753 CA0000         JZ      INPTLNRD
0756 CD9405         CALL    PRNTCHRI        ;print (val)
0759 5C             DB      '\
075A 0C             inr     c               ;char count
075B C30000         JMP     INPTLINL
                INPTLNRD:
075E 210000         LXI     H,MSGSTARS      ;BREAK ENTERED
0761 CD0000         CALL    PRNTMSG ;TELL HIM WE GOT IT
0764 05             DCR     B       ;BREAK AT BEGINNING MEANS BREAK
0765 CA0000         JZ      INPTEXIT

                INPTCRLF:
0768 CD0000         CALL    PRNTCRLF        ;ON THE NEXT LINE
                INPTRQST:
076B 62             MOV     H,D
076C 6B             MOV     L,E     ;PRINT USER"S PROMPT MESSAGE
076D CD0000         CALL    PRNTMSG
0770 2A7F03         LHLD    INPTBUFR        ;INPUT A LINE FROM RECEIVER
0773 010001         LXI     B,1*256
0776 CD9405         CALL    PRNTCHRI        ;print (val)
0779 20             DB      " "             ;OK, WE"RE READY FOR INPUT
                INPTLINL:
077A 3600           MVI     M,0     ;MAINTAIN ENDING ZERO
077C CDBB05         CALL    INPTCHAR
                INPTLINC:
077F FE07           CPI     BEL
0781 CA0000         JZ      INPTLNST        ;BELL"S OK
0784 FE0D           CPI     CR
0786 CA0000         JZ      INPTCRTN        ;CARRIAGE RTN IS END OF LINE
0789 FE08           CPI     BS
078B CA5107         JZ      INPTLNBS        ;BACKSPACE IS DELETE
078E FE03           CPI     ETX     ;CONTROL C IS ABORT
0790 CA5E07         JZ      INPTLNRD        ;FORGET THIS LINE, START OVER
0793 FE0C           CPI     FF      ;FORM FEEDS ARE ECHOED
0795 CA0000         JZ      INPTLNEC
0798 FE20           CPI     " "
079A DA7A07         JC      INPTLINL        ;IGNORE OTHER CONTROL CHARS
```

```
                INPTLNST:
079D 77             MOV     M,A     ;STORE THE CHARACTER
079E 78             MOV     A,B
079F FE9D           CPI     LINESYZE
07A1 3E07           MVI     A,BEL
07A3 D20000         JNC     INPTLNEC
07A6 04             INR     B
07A7 B1             ORA     C
07A8 4E             MOV     C,M
07A9 23             INX     H
07AA 3E0A           MVI     A,LF
07AC FC9805         CM      PRNTCHRA
07AF 79             MOV     A,C
                INPTLNEC:
07B0 CD9805         CALL    PRNTCHRA        ;ac -> screen   ;ECHO CHARACTER
07B3 C37A07         JMP     INPTLINL

                INPTCRTN:
07B6 05             DCR     B       ;CARRIAGE RETURN AT BEGINNING
07B7 CA6807         JZ      INPTCRLF        ;GETS ANOTHER TURN
                INPTEXIT:
07BA 2A7F03         LHLD    INPTBUFR
07BD 2B             DCX     H
07BE CD0000         CALL    PRNTCRLF
07C1 90             SUB     B       ;SET CONDITION CODES
07C2 3F             CMC                     ;S=C=NZ = BREAK
07C3 9F             SBB     A       ;NS=NC=Z = NON-EMPTY LINE
07C4 C9             RET

                MSGSTARS:
07C5 2A2A2A         DB      "***",0
07CB 00
```

```
                ;
                ;  SET OPTIONS COMMAND
                ;
                SETSTM:
07C9 CAEF05         JZ      ERRASN  ;TURN OPTION ON OR OFF
07CC FE99           CPI     KEYLIS
07CE CA0000         JZ      SETSTMLS
07D1 F5             PUSH    PSW     ;SAVE OPTION
07D2 CDAB03         CALL    SCANNXT ;bscan ,
07D5 CAEF05         JZ      ERRASN
07D8 D691           SUI     KEYON
07DA 47             MOV     8,A     ;SAVE FLAG
07DB CDAB03         CALL    SCANNXT ;bscan +
07DE F1             POP     PSW     ;WHICH OPTION
07DF FE89           CPI     KEYGTO
07E1 CA0000         JZ      SETSTMGT        ;GOTO
07E4 FE96           CPI     KEYPRT
07E6 C2EF05         JNZ     ERRASN
                SETSTMPR:
07E9 78             MOV     A,B
07EA 326303         sta     p3010           ;used to be printflg **
07ED C9             RET
                SETSTMGT:
07EE 78             MOV     A,B
07EF 326603         STA     TRACEFLG
07F2 C9             RET
                SETSTMLS:
07F3 23             INX     H
07F4 CD0000         CALL    VAL8YTE
07F7 2F             CMA                     ;FIND NEGATIVE OF 8YTE
07F8 3C             INR     A
07F9 329A03         STA     CURSLIM
07FC C9             RET

                ;
                ;  DELETE COMMAND PROCESSOR
                ;
                DELSTM:
07ED 11FFFF         LXI     D,0FFFFH        ;DELETE COMMAND
0800 CDE403         CALL    SCANLPRZ
0803 E3             XTHL            ;SAVE SCAN POINTER
0804 E8             XCHG
0805 CDC104         CALL    CMHLLTDE        ;VERIFY  FIRST<=LAST
0808 DAEF05         JC      ERRASN
080B E5             PUSH    H
080C CD3004         CALL    LINESRCH        ;LOOK FOR FIRST LINE
080F D1             POP     D
0810 C5             PUSH    B
0811 CD3004         CALL    LINESRCH        ;LOOK FOR LAST LINE
0814 C1             POP     B
0815 CD9304         CALL    LINEDEL
0818 E1             POP     H
0819 C9             RET
```

```
                ;
                ;  LIST COMMAND PROCESSOR
                ;
                LISSTM:
081A 110000         LXI     D,0     ;LIST COMMAND
081D 01FFFF         LXI     B,0FFFFH        ;TOTAL DEFAULT IS ENTIRE FILE
0820 CA0000         JZ      LISSTMSC
0823 010000         LXI     8,0     ;ELSE DEFAULT IS ONLY ONE LINE
                LISSTMSC:
0826 CDE703         CALL    SCANLPRM        ;SCAN LINE PARAMETERS
0829 C2EF05         JNZ     ERRASN
082C E3             XTHL
082D EB             XCHG
082E E5             PUSH    H
082F CD3004         CALL    LINESRCH
0832 C5             PUSH    8
                LISSTMLP:
0833 C1             POP     8       ;MOVE ON TO NEXT LINE
0834 D1             POP     D
0835 E1             POP     H
0836 CD3E00         CALL    SYSBREAK        ;ALLOW BREAK
0839 CA0000         JZ      EXECUTEB
083C C5             PUSH    B
083D E3             XTHL
083E CD4D04         CALL    LINELINK
0841 CA0000         JZ      POPHLRET        ;END OF PROGRAM, QUIT
0844 D5             PUSH    D
0845 C5             PUSH    B
0846 E5             PUSH    H       ;SAVE TEXT FOR LATER
0847 4E             MOV     C,M     ;FETCH LINE NUMBER
0848 23             INX     H
0849 46             MOV     B,M
084A 60             MOV     H,8
084B 69             MOV     L,C
084C E8             XCHG
084D CDC104         CALL    CMHLLTDE
0850 DA0000         JC      LISSTMXT        ;LAST LINE REACHED?
0853 CD0000         CALL    PRNTCRLF        ;LIST CURRENT LINE
0856 EB             XCHG
0857 CD0000         CALL    PRINTINT        ;PRINT LINE NUMBER
085A CD9405         CALL    PRNTCHRI        ;print (val)
085D 20             DB      " "     ;FOLLOWED BY BLANK
085E E1             POP     H
085F CD0000         CALL    LISEDIXP        ;EXPAND TEXT
0862 CD0000         CALL    PRNTMSG ;AND PRINT IT
0865 21A000         LXI     H,0+LINESYZE+3
0868 39             DAD     SP
0869 F9             SPHL            ;DEALLOCATE EXPANDED TEXT
086A C33308         JMP     LISSTMLP
```

```
                LISSTMXT:
086D E1             POP     H
                POPHL3RT:
086E E1             POP     H
086F E1             POP     H
                POPHLRET:
0870 E1             POP     H
0871 C9             RET


                ;
                ;  EXPAND KEYWORDS IN LINE / INVERSE OF KEYSCAN
                ;
                LISEDIXP:
0872 0E4E           MVI     C,LINESYZE/2    ;SPACE ENOUGH TO EXPAND LINE?
0874 CDD804         CALL    SPACESTK
0877 EB             XCHG                    ;SAVE POINTER TO LINE TO EXPAND
0878 C1             POP     B               ;AND CALLER
0879 2160FF         LXI     H,0-LINESYZE-3
087C 39             DAD     SP
087D F9             SPHL                    ;CREATE TEXT BUFFER ON STACK
087E C5             PUSH    B               ;PUT BACK CALLER
087F EB             XCHG
0880 23             INX     H
0881 23             INX     H               ;plus 2
0882 E5             PUSH    H               ;SAVE TEXT POINTER
0883 210400         LXI     H,4             ;CREATE POINTER TO EXPAND TEXT
0886 39             DAD     SP
0887 EB             XCHG
0888 069D           MVI     B,LINESYZE      ;INITIALIZE LENGTH COUNTER
088A C30000         JMP     LISEDIKD
                LISEDISC:
088D CD0000         CALL    LISEDIST        ;STUFF ONE CHARACTER OF LINE
                LISEDIKD:
0890 E1             POP     H               ;DO REST OF LINE
0891 7E             MOV     A,M
                LISEDINC:
0892 23             INX     H
0893 FE3A           CPI     ":"
0895 C20000         JNZ     LISEDIKT
0898 7E             MOV     A,M
0899 FE8C           CPI     KEYELS  ;:ELSE BECOMES ELSE
089B CA9208         JZ      LISEDINC
089E 3E3A           MVI     A,":"
                LISEDIKT:
08A0 A7             ANA     A               ;MOVE HIGH ORDER INTO S-FLAG
08A1 CA0000         JZ      LISEDIXT
08A4 E5             PUSH    H
08A5 F28D08         JP      LISEDISC
08A8 4F             MOV     C,A
```

```
08A9 21A0A3     LXI     H,KEYLSBH*256+KEYLSBL
0BAC CD0000     CALL    LISEDISB        ;OPTIONAL BLANK BEFORE KEYWORD
08AF 210401     LXI     H,KEYWORDS      ;SEARCH FOR KEYWORD
0BB2 C30000     JMP     LISEDIKS
            LISEDIKL:
08B5 B6         ORA     M
08B6 23         INX     H
08B7 F2B508     JP      LISEDIKL
            LISEDIKS:
08BA 7E         MOV     A,M     ;FETCH KEYWORD NUMBER
0BBB F680       ORI     080H
08BD 23         INX     H
0BBE A9         XRA     C
0BBF C2B508     JNZ     LISEDIKL
            LISEDIKY:
08C2 7E         MOV     A,M     ;EXPAND KEYWORD
0BC3 07         RLC
08C4 A7         ANA     A       ;HIGH-ORDER TO CARRY
0BC5 1F         RAR
0BC6 CD0000     CALL    LISEDIST        ;STUFF THIS CHARACTER
08C9 23         INX     H
0BCA D2C20B     JNC     LISEDIKY        ;DO THEM ALL
0BCD 79         MOV     A,C
0BCE 21B2A5     LXI     H,KEYLSAH*256+KEYLSAL
0BD1 CD0000     CALL    LISEDISB        ;OPTIONAL BLANK AFTER KEYWORD
0BD4 C3900B     JMP     LISEDIKD

            LISEDISB:
0BD7 BD         CMP     L       ;INSERT BLANK IN LINE IF
0BDB DB         RC              ;L <= A < H
08D9 BC         CMP     H
0BDA D0         RNC
0BDB 3E20       MVI     A," "   ;GENERATE BLANK
            LISEDIST:
0BDD 12         STAX    D
0BDE 13         INX     D
0BDF 05         DCR     B
0BE0 C0         RNZ             ;TRUNCATE TOO LONG A LINE
0BE1 04         INR     B
0BE2 2B         DCX     H
0BE3 C9         RET

            LISEDIXT:
0BE4 12         STAX    D
0BE5 3E9E       MVI     A,LINESYZE+1    ;COMPUTE LENGTH OF OUTPUT
0BE7 90         SUB     B
0BEB 47         MOV     B,A
0BE9 210200     LXI     H,2     ;CREATE POINTER TO EXPAND TEXT
08EC 39         DAD     SP
0BED C9         RET             ;AND RETURN
```

```
                ;
                ;  EDIT COMMAND PROCESSOR
                ;
                EDISTM:
08EE 110000         LXI     D,0     ;SCAN PARAMETERS
08F1 CDE403         CALL    SCANLPRZ
08F4 E3             XTHL            ;SAVE SCAN,
08F5 226F03         SHLD    SCANPTR1        ;AND OUTPUT LINE NUMBER
08F8 CD3004         CALL    LINESRCH        ;LOOK UP LINE
08FB D20000         JNC     ERRAUS  ;NOT FOUND...
08FE 60             MOV     H,8
08FF 69             MOV     L,C
0900 23             INX     H
0901 23             INX     H       ;plus 2
0902 CD7208         CALL    LISEDIXP        ;EXPAND LINE
0905 2A6F03         LHLD    SCANPTR1        ;RECOVER LINE NUM8ER
0908 E5             PUSH    H
                EDISTMLS:
0909 CD0000         CALL    EDISTMCR        ;GIVE HIM A LOOK AT IT
090C CD0000         CALL    PRNTMSG ;PRINT COPY OF TEXT
090F CD0000         CALL    EDISTMCR        ;A NEW EDIT LINE
0912 0E01           MVI     C,1     ;POSITION COUNTER
                EDISTMNX:
0914 CD0000         CALL    EDISTMCH        ;OK MASTER, TELL ME WHAT TO DO
0917 FE20           CPI     " "     ;MOVE ALONG
0919 CA0000         JZ      EDISTMAD
091C CDBC03         CALL    ALPHACHA        ;CONVERT LOWER TO UPPER
091F FE44           CPI     "D"     ;DELETE
0921 CA0000         JZ      EDISTMDL
0924 FE49           CPI     "I"     ;INSERT
0926 CA0000         JZ      EDISTMIN
0929 FE52           CPI     "R"     ;REPLACE
092B CA0000         JZ      EDISTMRP
                EDISTMER:
092E 3E07           MVI     A,8EL   ;SQUAWK ABOUT ERROR
                EDISTMEC:
0930 CD9805         CALL    PRNTCHRA        ;ac -> screen
0933 C31409         JMP     EDISTMNX

                ;  ADVANCE
                ;
                EDISTMAD:
0936 79             MOV     A,C
0937 88             CMP     B       ;CAN WE STILL ADVANCE?
0938 D22E09         JNC     EDISTMER
0938 0C             INR     C       ;ADVANCE POSITION COUNTER
093C 7E             MOV     A,M
093D 23             INX     H       ;PRINT CHARACTER PASSED OVER
093E C33009         JMP     EDISTMEC
```

```
                ;  DELETE
                ;
                EDISTMDL:
0941 79             MOV     A,C
0942 B8             CMP     B       ;ANYTHING TO DELETE?
0943 D22E09         JNC     EDISTMER
0946 05             DCR     B       ;DECREASE CHARACTER COUNT
0947 E5             PUSH    H       ;SAVE CURRENT POSITION
0948 7E             MOV     A,M
0949 CD9805         CALL    PRNTCHRA        ;LIST CHARACTER DELETED
094C 54             MOV     D,H
094D 5D             MOV     E,L
                EDISTMDM:
094E 23             INX     H
094F 7E             MOV     A,M     ;MOVE THIS CHARACTER DOWNWARD
0950 12             STAX    D
0951 13             INX     D
0952 B7             ORA     A
0953 C24E09         JNZ     EDISTMDM
0956 E1             POP     H
0957 C31409         JMP     EDISTMNX

                ;  INSERT
                ;
                EDISTMIN:
095A CD0000         CALL    EDISTMCH        ;GET SOMETHING TO PUT IN
095D 57             MOV     D,A     ;SAVE COPY OF CHARACTER
                EDISTMRI:
095E 78             MOV     A,B
095F FE9D           CPI     LINESYZE        ;ROOM AT THE INNPUT BUFFER?
0961 D22E09         JNC     EDISTMER
0964 04             INR     B       ;COUNT NEWCOMER
0965 0C             INR     C       ;NEXT ONE GOES AFTER HIM
0966 7A             MOV     A,D
0967 CD9805         CALL    PRNTCHRA        ;ac -> screen   ;PRINT NEWCOMER
096A E5             PUSH    H       ;SAVE CURRENT POSITION
                EDISTMIM:
096B 5E             MOV     E,M
096C 77             MOV     M,A     ;MOVE CHARACTERS UP ONE BYTE
096D B7             ORA     A
096E 7B             MOV     A,E
096F 23             INX     H
0970 C26B09         JNZ     EDISTMIM
0973 E1             POP     H
0974 23             INX     H
0975 C35A09         JMP     EDISTMIN
```

```
                ;  REPLACE
                ;
                EDISTMRP:
0978 CD0000         CALL    EDISTMCH        ;GET UPDATE CHARACTER
097B 57             MOV     D,A
097C 79             MOV     A,C
097D B8             CMP     B       ;REPLACING END OF LINE?
097E D25E09         JNC     EDISTMRI        ;IF SO, GO TO INSERT
0981 72             MOV     M,D     ;UPDATE THE CHARACTER
0982 0C             INR     C
0983 23             INX     H
0984 7A             MOV     A,D
0985 CD9805         CALL    PRNTCHRA        ;ac -> screen   ;PRINT NEWCOMER
0988 C37809         JMP     EDISTMRP

                ;  SEARCH
                ;
                EDISTMSR:
098B CD0000         CALL    EDISTMCH        ;FIND CHARACTER TO SEARCH FOR
098E CDBC03         CALL    ALPHACHA        ;CONVERT TO STANDARD CASE
0991 57             MOV     D,A
0992 1E00           MVI     E,0
                EDISTMSL:
0994 79             MOV     A,C
0995 B8             CMP     B
0996 D22E09         JNC     EDISTMER        ;NO MORE, TERMINATE SEARCH
0999 CDBB03         CALL    ALPHACHK        ;FETCH CHARACTER IN STANDARD CASE
099C BB             CMP     E
099D CA1409         JZ      EDISTMNX        ;GOTTA MATCH?
09A0 CD9805         CALL    PRNTCHRA        ;ac -> screen   ;LIST FAILURES
09A3 0C             INR     C
09A4 23             INX     H
09A5 5A             MOV     E,D
09A6 C39409         JMP     EDISTMSL        ;AND KEEP LOOKING
```

```
            EDISTMXT:
09A9 0D         DCR     C       ;BEGINNING CR MEANS DONE, UPDATE
09AA C20909     JNZ     EDISTMLS        ;OTHERWISE, LIST, MORE EDITS
09AD D1         POP     D       ;RETRIEVE LINE NUMBER
09AE 210000     LXI     H,0
09B1 39         DAD     SP      ;POINT TO TEXT
09B2 CD5904     CALL    LINEINS ;AND REINSERT
            EDISTMQT:
09B5 21A000     LXI     H,0+LINESYZE+3
09B8 39         DAD     SP
09B9 F9         SPHL            ;DEALLOCATE TEXT BUFFER
09BA E1         POP     H       ;RECOVER SCAN POINTER
09BB C9         RET             ;AND RETURN

            ;  LIST LINE, PREPARE FOR UPDATES
            ;
            EDISTMCR:
09BC D1         POP     D
09BD E1         POP     H       ;RETRIEVE COPY OF LINE NUMBER
09BE E5         PUSH    H       ;SAVE IT,
09BF D5         PUSH    D
09C0 C5         PUSH    B       ;AND LINE LENGTH
09C1 CD0000     CALL    PRNTCRLF
09C4 CD0000     CALL    PRINTINT        ;PRINT LINE NUMBER
09C7 CD9405     CALL    PRNTCHRI        ;print (val)
09CA 20         DB      " "
09CB 210600     LXI     H,6
09CE 39         DAD     SP      ;CREATE POINTER TO TEXT BUFFER
09CF C1         POP     B
09D0 C9         RET

            ;  GET OPTION CHARACTER
            ;
            EDISTMCH:
09D1 CDBB05     CALL    INPTCHAR        ;GET CHARACTER ROUTINE
09D4 FE20       CPI     " "
09D6 D0         RNC             ;NOT CONTROL, RETURN
09D7 FE07       CPI     BEL
09D9 C8         RZ
09DA D1         POP     D       ;REMOVE CALLER
09DB FE09       CPI     HT      ;SEARCH (TAB)
09DD CA8B09     JZ      EDISTMSR
09E0 FE0D       CPI     CR      ;LIST, OR UPDATE
09E2 CAA909     JZ      EDISTMXT
09E5 FE1B       CPI     ESC     ;TERMINATE OPTION
09E7 CA1409     JZ      EDISTMNX
09EA FE03       CPI     ETX     ;ABORT, NO UPDATE
09EC C22E09     JNZ     EDISTMER
09EF 21C507     LXI     H,MSGSTARS      ;TYPE BREAK MESSAGE
09F2 CD0000     CALL    PRNTMSG
09F5 D1         POP     D
09F6 C3B509     JMP     EDISTMQT
```

```
                ;
                ;  SCAN STACK FOR "FOR" LOOP
                ;
09F9 0010   FORBLCK     EQU     16      ;SIZE OF "FOR" STACK ENTRY

                FDRCHK:
09F9 210400         LXI     H,4     ;LOOK FOR MARK ON STACK
09FC 39             DAD     SP
                FORCHKL:
09FD 7E             MOV     A,M
09FE 23             INX     H
09FF FE83           CPI     KEYFOR
0A01 C0             RNZ
0A02 3E04           MVI     A,TYPESING
0A04 326B03         STA     TYPEFLG ;SET CORRECT TYPE FLAG
0A07 4E             MOV     C,M     ;MARK IS PRESENT
0A08 23             INX     H
0A09 46             MOV     B,M
0A0A 23             INX     H
0A0B E5             PUSH    H
0A0C 60             MOV     H,B
0A0D 69             MOV     L,C
0A0E 7A             MOV     A,D     ;LODKING FOR PARTICULAR VARIABLE?
0A0F B3             ORA     E
0A10 EB             XCHG
0A11 CA0000         JZ      FDRCHKXT
0A14 EB             XCHG            ;IS THIS IT?
0A15 CDC104         CALL    CMHLLTDE
                FORCHKXT:
0A18 010D00         LXI     B,FORBLCK-3
0A1B E1             POP     H
0A1C C8             RZ
0A1D 09             DAD     B
0A1E C3FD09         JMP     FDRCHKL

                ;
                ;  FOR STATEMENT PROCESSOR
                ;
                FDRSTM:
0A21 3EAB           MVI     A,SCANPFLD      ;FDR STATEMENT
0A23 326703         STA     SCANPFLG
0A26 CD0000         CALL    LETSTM
0A29 CDD903         CALL    TYPECHK
0A2C EA0000         JPE     ERRATM  ;MUST BE SINGLE INDEX
0A2F E3             XTHL            ;SAVE SCANPTR, REMOVE CALLER
0A30 EB             XCHG
0A31 227703         SHLD    VARINDEX
0A34 EB             XCHG
0A35 CDF909         CALL    FORCHK
0A38 D1             PDP     D
0A39 C20000         JNZ     FDRSTMNF
0A3C 09             DAD     B
0A3D F9             SPHL
                FDRSTMNF:
0A3E EB             XCHG
0A3F 0E08           MVI     C,FORBLCK+1/2
```

```
0A41 CDD804     CALL    SPACESTK

0A44 E5         PUSH    H
0A45 CD0000     CALL    DATSTM  ;FIND FIRST STATEMENT IN FOR LOOP
0A48 E3         XTHL            ;AND SAVE
0A49 E5         PUSH    H
0A4A 2A7303     LHLD    CURLINE ;SAVE CURRENT LINE NUMBER
0A4D E3         XTHL
0A4E CDA303     CALL    SCANNXTV        ;bscan (val)
0A51 A1         DB      KEYTO   ;SCAN LIMIT VALUE,
0A52 CD0000     CALL    VALNUMBR        ;bscan numbr
0A55 E5         PUSH    H
0A56 CD0000     CALL    LDRGAC
0A59 E1         POP     H
0A5A C5         PUSH    B       ;SAVE ON STACK
0A5B D5         PUSH    D
0A5C 010081     LXI     B,08100H        ;LOAD DEFAULT STEP=1.0
0A5F 51         MOV     D,C
0A60 5A         MOV     E,D
0A61 7E         MOV     A,M
0A62 FEA2       CPI     KEYSTEP ;CHECK FOR EXPLICIT STEP SIZE
0A64 3E01       MVI     A,001H
0A66 C20000     JNZ     FORSTMST
0A69 CDAB03     CALL    SCANNXT ;bscan +
0A6C CD0000     CALL    VALNUMBR        ;bscan numbr
0A6F E5         PUSH    H
0A70 CD0000     CALL    LDRGAC
0A73 E1         POP     H
0A74 CD0000     CALL    SIGNACC
            FORSTMST:
0A77 C5         PUSH    B       ;SAVE STEP SIZE ON STACK
0A78 D5         PUSH    D
0A79 F5         PUSH    PSW     ;SAVE DIRECTION
0A7A 33         INX     SP
0A7B E5         PUSH    H
0A7C 2A7703     LHLD    VARINDEX        ;SAVE INDEX VARIABLE
0A7F E3         XTHL
            FORMARK:
0A80 0683       MVI     B,KEYFOR        ;MARK STACK WITH "FOR"
0A82 C5         PUSH    B
0A83 33         INX     SP
```

```
                ;
                ;  INTERPRETER EXECUTIVE
                ;
                EXECUTEL:
0AB4 CD0000         CALL    BREAKCHK        ;USER HAVE ANY COMMENTS?
0A87 227703         SHLD    PROGCNTR
0ABA 7E             MOV     A,M
0A8B FE3A           CPI     ":"
0A8D CA0000         JZ      EXECUTE ;MULTIPLE STATEMENTS ON LINE?
0A90 B7             ORA     A
0A91 C2EF05         JNZ     ERRASN
0A94 23             INX     H       ;END OF LINE,
0A95 7E             MOV     A,M
0A96 23             INX     H
0A97 B6             ORA     M
0A98 23             INX     H
0A99 CA0000         JZ      ENDPROGM        ;END OF PROGRAM?
0A9C 5E             MOV     E,M
0A9D 23             INX     H
0A9E 56             MOV     D,M
0A9F EB             XCHG
0AA0 227303         SHLD    CURLINE ;MOVE TO NEXT LINE
0AA3 EB             XCHG
                EXECUTE:
0AA4 CDAB03         CALL    SCANNXT ;bscan ,        ;EXECUTE STATEMENT
0AA7 11B40A         LXI     D,EXECUTEL
0AAA D5             PUSH    D
                EXECUTEC:
0AAB C8             RZ
                EXECUTES:
0AAC FEB0           CPI     KEYSTM  ;WHAT KIND OF STATEMENT?
0AAE DA0000         JC      LETSTM
0AB1 FEA0           CPI     KEYSUGR
0AB3 D20000         JNC     EXECUTE2
0AB6 B7             ADD     A
0AB7 4F             MOV     C,A
0ABB 0600           MVI     B,000H
0ABA EB             XCHG
0ABB 214E00         LXI     H,STMTABL
0ABE 09             DAD     B
0ABF 4E             MOV     C,M
0AC0 23             INX     H
0AC1 46             MOV     B,M
0AC2 C5             PUSH    B
0AC3 EB             XCHG
0AC4 C3AB03         JMP     SCANNXT
```

```
                BREAKCHK:
0AC7 CD3E00         CALL    SYSBREAK        ;TIME TO TAKE A BREAK?
                STPSTM:
0ACA C0             RNZ             ;STOP STATEMENT
0ACB 3C             INR     A
                EXECUTEB:
0ACC 227703         SHLD    PROGCNTR
                INPSTMBR:
0ACF C1             POP     B       ;THROW AWAY CALLER
                ENDPROGM:
0AD0 F5             PUSH    PSW
0AD1 2A7303         LHLD    CURLINE
0AD4 7D             MOV     A,L
0AD5 A4             ANA     H
0AD6 3C             INR     A
0AD7 CA0000         JZ      ENDSTMC
0ADA 227503         SHLD    CURLINES        ;SAVE INFORMATION FOR CONTINUE
0ADD 2A7703         LHLD    PROGCNTR
0AE0 227903         SHLD    PROGCNTS
                ENDSTMC:
0AE3 AF             XRA     A
0AE4 326503         STA     PRINTFLG
0AE7 F1             POP     PSW
0AE8 21E105         LXI     H,MSGBREAK
0AEB C20D06         JNZ     ERRMSGPR
0AEE C31906         JMP     CMNDSTRT

                CONSTM:
0AF1 C0             RNZ             ;CONT COMMAND
0AF2 1E00           MVI     E,ERRNCN-ERRN
0AF4 2A7903         LHLD    PROGCNTS
0AF7 7C             MOV     A,H
0AF8 B5             ORA     L
0AF9 CAF105         JZ      ERRMSG
0AFC EB             XCHG
0AFD 2A7503         LHLD    CURLINES
0B00 227303         SHLD    CURLINE
0B03 EB             XCHG
0B04 C9             RET

                RUNSTM:
0B05 CA0205         JZ      CLEARSET        ;RUN COMMAND
0B08 CD0505         CALL    CLEARVST
0B0B 01840A         LXI     B,EXECUTEL
0B0E C30000         JMP     RUNSTMC

                ENDSTM:
0B11 CACC0A         JZ      EXECUTEB        ;END STATEMENT
0B14 CDA303         CALL    SCANNXTV        ;bscan (val)
0B17 8A             DB      KEYRUN
0B18 C34B00         JMP     SYSQUIT
```

```
                ;
                ;  GOSUB/GOTO STATEMENTS
                ;
                GSBSTM:
0B1B 0E03           MVI     C,3     ;GOSUB STATEMENT
0B1D CDD804         CALL    SPACESTK
0B20 C1             POP     B
0B21 E5             PUSH    H
0B22 E5             PUSH    H
0B23 2A7303         LHLD    CURLINE
0B26 E3             XTHL
0B27 168E           MVI     D,KEYGSB        ;MARK STACK WITH GOSUB
0B29 D5             PUSH    D
0B2A 33             INX     SP
                RUNSTMC:
0B2B C5             PUSH    B
                GTOSTM:
0B2C CD0B04         CALL    SCANLINN        ;GOTO STATEMENT
0B2F D5             PUSH    D
0B30 CD0000         CALL    REMSTM
0B33 D1             POP     D
0B34 E5             PUSH    H
0B35 CD0000         CALL    TRACE
0B38 2A7303         LHLD    CURLINE
0B3B CDC104         CALL    CMHLLTDE
0B3E E1             POP     H
0B3F 23             INX     H
0B40 DC3304         CC      LINESRCL
0B43 D43004         CNC     LINESRCH
0B46 60             MOV     H,B
0B47 69             MOV     L,C
0B48 2B             DCX     H
0B49 D8             RC
                ERRAUS:
0B4A 1EC5           MVI     E,ERRNUS-ERRN
0B4C C3F105         JMP     ERRMSG

                ;
                ;  RETURN STATEMENT
                ;
                RETSTM:
0B4F C0             RNZ             ;RETURN STATEMENT
0B50 16FF           MVI     D,0FFH
0B52 CDF909         CALL    FORCHK  ;KILL ACTIVE FOR LOOPS
0B55 F9             SPHL            ;INSIDE SUBROUTINE
0B56 FE8E           CPI     KEYGSB
0B58 1E76           MVI     E,ERRNRG-ERRN
0B5A C2F105         JNZ     ERRMSG
0B5D D1             POP     D
0B5E CD0000         CALL    TRACE
0B61 EB             XCHG
0B62 227303         SHLD    CURLINE
0B65 21840A         LXI     H,EXECUTEL
0B68 E3             XTHL
                ;   JMP     DATSTM
```

```
                ;
                ;  DATA/ELSE/REM STATEMENTS
                ;
                DATSTM:
0B69 0E3A           MVI     C,":"   ;DATA STATEMENT
0B6B C30000         JMP     SCAN2KEY
                ELSSTM:

                REMSTM:
0B6E 0E00           MVI     C,000H  ;REM STATEMENT
                SCAN2KEY:
0B70 0600           MVI     B,000H  ;SKIP TO KEYWORD IN C
                DATRSKST:
0B72 79             MOV     A,C     ;SET UP TERMINATING BYTE
0B73 48             MOV     C,B
0B74 47             MOV     B,A
                DATRSKIP:
0B75 7E             MOV     A,M     ;SKIP TO TERMINATING BYTE
0B76 B7             ORA     A
0B77 C8             RZ
0B78 BB             CMP     B
0B79 CB             RZ
0B7A 23             INX     H
0B7B FE22           CPI     '"      ;STRING TO SKIP?
0B7D CA720B         JZ      DATRSKST
0B80 FEBB           CPI     KEYIF
0BB2 C2750B         JNZ     DATRSKIP
0BB5 14             INR     D       ;COUNT NUMBER OF IFS WE SKIP
0B86 C3750B         JMP     DATRSKIP

                ;
                ;  PROGRAM BRANCH TRACING
                ;
                TRACE:
0B89 3A6603         LDA     TRACEFLG        ;TRACING?
0BBC B7             ORA     A
0B8D C0             RNZ
0BBE C5             PUSH    B
0BBF D5             PUSH    D       ;SAVE DESTINATION LINE NUMBER
0B90 CD9405         CALL    PRNTCHRI        ;print (val)
0B93 5B             DB      "["     ;LEFT BRACKET
0B94 2A7303         LHLD    CURLINE
0B97 CD0000         CALL    PRINTINT        ;PRINT CURRENT LINE NUMBER
0B9A CD9405         CALL    PRNTCHRI        ;print (val)
0B9D 2C             DB      ","
0B9E E1             POP     H
0B9F E5             PUSH    H
0BA0 CD0000         CALL    PRINTINT        ;PRINT DESTINATION LINE NUMBER
0BA3 CD9405         CALL    PRNTCHRI        ;print (val)
0BA6 5D             DB      "]"     ;RIGHT BRACKET
                POPDEBCR:
0BA7 D1             POP     D
0BAB C1             POP     B
0BA9 C9             RET
```

```
                ;
                ;  ASSIGNMENT STATEMENT PROCESSOR
                ;
                LETSTM:
0BAA CD0000         CALL    VARSCAN ;LET STATEMENT
0BAD CDA303         CALL    SCANNXTV        ;bscan (val)
0BB0 B5             DB      KEYEQ
                ASSIGNVL:
0BB1 3A6B03         LDA     TYPEFLG
0BB4 F5             PUSH    PSW
0BB5 D5             PUSH    D
0BB6 CD0000         CALL    VALEXPR ;bscan expr
0BB9 D1             POP     D
0BBA F1             POP     PSW
                ASSIGN:
0BBB EB             XCHG            ;MAKE THE ASSIGNMENT
0BBC D5             PUSH    D       ;SAVE SCAN
0BBD E5             PUSH    H       ;SAVE VARIABLE
0BBE CD0000         CALL    COERCE
0BC1 C20000         JNZ     LETSTMNM
0BC4 CD0000         CALL    STRGUNIQ        ;REMOVE CONFLICT PROBLEMS
0BC7 CD0000         CALL    STRGRELT        ;RELEASE STRING TEMPORARY
0BCA E1             POP     H       ;COPY DESCRIPTOR TO DESTINATION
0BCB CD0000         CALL    CDPYVAL
0BCE E1             POP     H
0BCF C9             RET

                LETSTMNM:
0BD0 CD0000         CALL    LDMMAC  ;MAKE NUMERIC ASSIGNMENT
0BD3 D1             POP     D
0BD4 E1             POP     H
0BD5 C9             RET

                STRGUNIQ:
0BD6 2A9303         LHLD    ACCUMLTR        ;GET STRING DESCRIPTDR
0BD9 EB             XCHG            ;IS STRING IN STRING SPACE?
0BDA CD0000         CALL    STRGTEST
0BDD D0             RNC
0BDE CDC104         CALL    CMHLLTDE        ;VARIABLE REFERENCE?
0BE1 D40000         CNC     STRGSTDR        ;IF SD, MAKE NEW COPY
0BE4 C9             RET
```

```
                ;
                ;  COERCE ACCUMULATOR TO TYPE IN A
                ;
                COERCE:
0BE5 CDDC03         CALL    TYPECHKA
                COERCEF:
0BEB E20000         JPO     CSINGLE
0BEB CA0000         JZ      CSTRING
0BEE C30000         JMP     ERRATM

                VALNUMBR:
0BF1 CD0000         CALL    VALEXPR ;bscan expr
                CSINGLE:
0BF4 CDD903         CALL    TYPECHK
0BF7 E0             RPO
0BFB C30000         JMP     ERRATM

                CSTRING:
0BFB CDD903         CALL    TYPECHK
0BFE CB             RZ
0BFF C30000         JMP     ERRATM

                ERRATM:
0C02 1EB0           MVI     E,ERRNTM-ERRN
0C04 C3F105         JMP     ERRMSG

                VALINTDE:
0C07 CDF10B         CALL    VALNUMBR        ;bscan numbr EVAL POSITIVE INTEGER EXPR
                CINTPOS:
0C0A CD0000         CALL    SIGNACC ;CONVERT TO INTEGER
0C0D FA0000         JM      ERRAFC
                CINTEGER:
0C10 3A9603         LDA     FLACCEXP
0C13 FE90           CPI     090H
0C15 DA0000         JC      FIXAC
0C1B 01B090         LXI     B,090B0H
0C1B 110000         LXI     D,00000H
0C1E CD0000         CALL    FLCMP
0C21 51             MOV     D,C
0C22 C8             RZ

                ERRAFC:
0C23 1E2D           MVI     E,ERRNFC-ERRN
0C25 C3F105         JMP     ERRMSG

                VALBYTE2:
0C2B CDA303         CALL    SCANNXTV        ;bscan (val)
0C2B 2C             DB      ","     ;EVAL LATER BYTE ARGUMENTS
                VALBYTE:
0C2C CDF10B         CALL    VALNUMBR        ;bscan numbr EVAL BYTE EXPRESSION
                CBYTE:
0C2F CD0A0C         CALL    CINTPOS ;CONVERT ACC TO BYTE
0C32 7A             MOV     A,D
0C33 B7             ORA     A
0C34 C2230C         JNZ     ERRAFC
```

```
0C37 2B         DCX     H
0C38 CDAB03     CALL    SCANNXT ;bscan ,
0C3B 7B         MOV     A,E
0C3C C9         RET

            EXECUTE2:
0C3D FEC4       CPI     KEYPORT ;PORT OUTPUT?
0C3F CA0000     JZ      PORSTM
0C42 FEC6       CPI     KEYMEM  ;MEMORY ALTERATION?
0C44 CA0000     JZ      MEMSTM

            ;
            ;  MID-STRING ASSIGNMENT STATEMENT
            ;
            MIDSTM:
0C47 CDA303     CALL    SCANNXTV        ;bscan (val)
0C4A D1         DB      KEYMID  ;ENTER POINTING TO "MID$"
0C4B CDA303     CALL    SCANNXTV        ;bscan (val)
0C4E 28         DB      "("
0C4F CD0000     CALL    VARSCAN ;SCAN VARIABLE TO UPDATE
0C52 CDFB0B     CALL    CSTRING ;MAKE SURE IT"S A STRING
0C55 D5         PUSH    D       ;SAVE REFERENCE
0C56 E5         PUSH    H
0C57 CD0000     CALL    STRGTEST        ;WHERE IS STRING NOW?
0C5A D5         PUSH    D       ;SHOULDN"T BE IN PROGRAM
0C5B D40000     CNC     STRGSTOR        ;OR ELSE WE MODIFY OURSELF
0C5E E1         POP     H
0C5F CD0000     CALL    COPYVAL
0C62 E1         POP     H       ;CONTINUE SCAN
0C63 CD280C     CALL    VALBYTE2        ;SCAN STARTING POSITION
0C66 B7         ORA     A
0C67 CA230C     JZ      ERRAFC  ;MUST BE NON-ZERO
0C6A D5         PUSH    D
0C6B 1EFF       MVI     E,0FFH
0C6D 7E         MOV     A,M
0C6E FE29       CPI     ")"     ;DEFAULT LENGTH?
0C70 C4280C     CNZ     VALBYTE2        ;SCAN LENGTH, IF GIVEN
0C73 CDA303     CALL    SCANNXTV        ;bscan (val)
0C76 29         DB      ")"
0C77 C1         POP     B       ;CONDENSE STACK
0C78 51         MOV     D,C
0C79 D5         PUSH    D
0C7A CDA303     CALL    SCANNXTV        ;bscan (val)
0C7D B5         DB      KEYEQ
0C7E CD0000     CALL    VALEXPR ;bscan expr     ;EVALUATE RIGHT HAND SIDE
0C81 226F03     SHLD    SCANPTR1
0C84 CD0000     CALL    LENFCTC ;RELEASE STRING RESOURCE
0C87 4E         MOV     C,M     ;AND LDAD DESCRIPTOR
0CB8 23         INX     H
0CB9 46         MOV     B,M
0CBA D1         POP     D       ;GET BACK LENGTH, START
0C8B BB         CMP     E
0C8C D20000     JNC     MIDSTMLN        ;LENMOV = MIN(LENI, LENS)
0C8F 5F         MOV     E,A
            MIDSTMLN:
0C90 E1         POP     H       ;RECOVER DESTINATION DESCRIPTOR
```

```
0C91 7E         MOV     A,M     ;GET ITS LENGTH
0C92 15         DCR     D
0C93 92         SUB     D       ;SUBTRACT STARTING POSITION
0C94 DA0000     JC      MIDSTMXT        ;NOTHING TO DO IF BEYOND
```

```
0C97 BB          CMP     E
0C98 D20000      JNC     MIDSTMLM
0C9B 5F          MOV     E,A
             MIDSTMLM:
0C9C C5          PUSH    B       ;SAVE SOURCE ADDRESS
0C9D CD0000      CALL    LDICBMM ;COMPUTE DESTINATION ADDRESS
0CA0 6A          MOV     L,D
0CA1 2600        MVI     H,0
0CA3 09          DAD     B
0CA4 EB          XCHG
0CA5 C1          POP     B
0CA6 CD0000      CALL    COPYSTRG        ;COPY STRING
             MIDSTMXT:
0CA9 2A6F03      LHLD    SCANPTR1
0CAC C9          RET

             ;
             ;  LOCATE STRING REFERENCED BY DE
             ;
             STRGTEST:
0CAD D5          PUSH    D       ;DE=STRING REFERENCE
0CAE EB          XCHG
0CAF 23          INX     H       ;GET ADDRESS OF STRING
0CB0 5E          MOV     E,M
0CB1 23          INX     H
0CB2 56          MOV     D,M
0CB3 2A8703      LHLD    FREELIMT        ;BOUNDARY
0CB6 CDC104      CALL    CMHLLTDE        ;NC = STRING IN PROGRAM
0CB9 D1          POP     D       ;C = STRING IN BUFFER
0CBA C9          RET             ;OR STRING SPACE
```

```
                ;
                ;  CASE/CONDITIONAL STATEMENT PROCESSORS
                ;
                ONSTM:
0CBB CD2C0C         CALL    VALBYTE ;ON STATEMENT
0CBE 7E             MOV     A,M
0CBF 47             MOV     B,A
0CC0 FE8E           CPI     KEYGSB  ;GOSUB RATHER THAN GOTO?
0CC2 CA0000         JZ      ONNSTMC
0CC5 CDA303         CALL    SCANNXTV        ;bscan (val)
0CC8 89             DB      KEYGTO  ;MUST BE GOTO...
0CC9 2B             DCX     H
                ONNSTMC:
0CCA 4B             MOV     C,E
                ONNSTMSL:
0CCB 0D             DCR     C       ;LOOK FOR RIGHT LINE NUMBER
0CCC 78             MOV     A,B
0CCD CAAC0A         JZ      EXECUTES        ;THEN EXECUTE STATEMENT
0CD0 CD0C04         CALL    SCANLINR
0CD3 FE2C           CPI     ","
0CD5 C0             RNZ
0CD6 C3CB0C         JMP     ONNSTMSL


                IFSTM:
0CD9 CDF10B         CALL    VALNUMBR        ;bscan numbr    ;IF STATEMENT
0CDC 7E             MOV     A,M
0CDD FE89           CPI     KEYGTO
0CDF CA0000         JZ      IFNSTMC
0CE2 CDA303         CALL    SCANNXTV        ;bscan (val)
0CE5 A0             DB      KEYTHEN
                IFNSTMC:
0CE6 CD0000         CALL    SIGNACC ;TEST CONDITION
0CE9 C20000         JNZ     IFNSTMCH
0CEC 1601           MVI     0,1
                IFNSTMSK:
0CEE 0E8C           MVI     C,KEYELS
0CF0 CD700B         CALL    SCAN2KEY        ;SKIP TO CORRESPONDING ELSE
0CF3 B7             ORA     A
0CF4 C8             RZ              ;OR END OF LINE
0CF5 CDAB03         CALL    SCANNXT ;bscan +
0CF8 15             DCR     D
0CF9 C2EE0C         JNZ     IFNSTMSK
                IFNSTMCH:
0CFC 2B             DCX     H       ;bscan -
0CFD CDAB03         CALL    SCANNXT ;bscan ,        ;CHOICE MADE
0D00 DA2C0B         JC      GTOSTM  ;GOTO A LABEL,
0D03 C3AB0A         JMP     EXECUTEC        ;OR EXECUTE A STATEMENT
```

```
                ;
                ;  PRINT STATEMENT PROCESSOR
                ;
                PRTSTMN:
0D06 FEA5           CPI     KEYTAB  ;TAB OPTION?
0D08 CA0000         JZ      PRNTOPTN
0D0B FEA6           CPI     KEYSPC  ;SPACE OPTION?
0D0D CA0000         JZ      PRNTOPTN
0D10 E5             PUSH    H
0D11 FE2C           CPI     ","
0D13 CA0000         JZ      PRNTCOMA
0D16 FE3B           CPI     ";"
0D18 CA0000         JZ      PRNTSEMI
0D1B C1             POP     B
0D1C CD0000         CALL    VALEXPR ;bscan expr
0D1F 2B             DCX     H       ;bscan -
0D20 E5             PUSH    H
0D21 CDD903         CALL    TYPECHK
0D24 CA0000         JZ      PRTSTRNG
0D27 CD0000         CALL    VALSTRGN        ;CREATE STRING FROM NUMBER
0D2A 2A9303         LHLD    ACCUMLTR        ;VERIFY ROOM ENOUGH ON LINE
0D2D 7E             MOV     A,M
0D2E 219903         LXI     H,CURSPOS
0D31 86             ADD     M
0D32 23             INX     H
0D33 86             ADD     M
0D34 DC0000         CC      PRNTCRLF        ;NO ROOM, FIND ANOTHER LINE
0D37 CD0000         CALL    PRNTSTRT
0D3A CD9405         CALL    PRNTCHRI        ;print (val)
0D3D 20             DB      " "
0D3E 3C             INR     A
                PRTSTRNG:
0D3F CC0000         CZ      PRNTSTRT        ;SEND OUTPUT STRING
0D42 E1             POP     H
0D43 CDAB03         CALL    SCANNXT ;bscan ,
                PRTSTM:
0D46 C2060D         JNZ     PRTSTMN ;PRINT STATEMENT
                PRNTCRLF:
0D49 CD9405         CALL    PRNTCHRI        ;print (val)
0D4C 0D             DB      CR      ;PRINT A CR, LF
0D4D CD9405         CALL    PRNTCHRI        ;print (val)
0D50 0A             DB      LF
                PRNTNULS:
0D51 3A9803         LDA     NULLCNT ;PRINT NULLS AFTER CR
                PRNTNULL:
0D54 3D             DCR     A
0D55 329903         STA     CURSPOS
0D58 C8             RZ
0D59 F5             PUSH    PSW
0D5A AF             XRA     A
0D5B CD9805         CALL    PRNTCHRA        ;ac -> screen
0D5E F1             POP     PSW
0D5F C3540D         JMP     PRNTNULL
```

```
                PRNTCOMA:
0D62 3A9903         LDA     CURSPOS ;COMMA SEPARATOR
0D65 FE8C           CPI     LINESYZE/ITEMSIZE-1*ITEMSIZE
0D67 D4490D         CNC     PRNTCRLF
0D6A D20000         JNC     PRNTSEMI
                PRNTCOML:
0D6D D60E           SUI     ITEMSIZE
0D6F D26D0D         JNC     PRNTCOML
0D72 2F             CMA
0D73 C30000         JMP     PRNTCOMC

                PRNTOPTN:
0D76 F5             PUSH    PSW
0D77 CDAB03         CALL    SCANNXT ;bscan +
0D7A CD0000         CALL    VALPARNS        ;GET OPTION PARAMETER
0D7D CDF40B         CALL    CSINGLE
0D80 CD2F0C         CALL    CBYTE
0D83 28             DCX     H
0D84 F1             POP     PSW
0D85 FEA6           CPI     KEYSPC
0D87 E5             PUSH    H
0D88 7B             MOV     A,E
0D89 CA0000         JZ      PRNTBLNK
0D8C 3A9903         LDA     CURSPOS
0D8F 2F             CMA
0D90 83             ADD     E
0D91 D20000         JNC     PRNTSEMI
                PRNTCOMC:
0D94 3C             INR     A
                PRNTBLNK:
0D95 47             MOV     B,A     ;PAD OUTPUT WITH A BLANKS
0D96 B7             ORA     A
0D97 CA0000         JZ      PRNTSEMI
0D9A 3E20           MVI     A," "
                PRNTBLNL:
0D9C CD9805         CALL    PRNTCHRA        ;ac -> screen
0D9F 05             DCR     B
0DA0 C29C0D         JNZ     PRNTBLNL
                PRNTSEMI:
0DA3 E1             POP     H
0DA4 CDAB03         CALL    SCANNXT ;bscan ,
0DA7 C8             RZ
0DA8 C3060D         JMP     PRTSTMN
```

```
            PRNTNUMS:
0DAB 23         INX     H       ;SENO STRING TO TRANSMITTER
            PRNTMSG:
0DAC C5         PUSH    B
0DAD D5         PUSH    D
0DAE 01A70B     LXI     B,POPDEBCR
0DB1 C5         PUSH    B
0DB2 CD0000     CALL    VALSTRGZ        ;STRING ENDS ON ZERO
            PRNTSTRT:
0DB5 CD0000     CALL    STRGRELA
0DB8 CD0000     CALL    LOOCBMM
0DBB 14         INR     O
            PRNTSTRL:
0OBC 15         DCR     D
0DBD C8         RZ
0DBE 0A         LOAX    B
0DBF CD9805     CALL    PRNTCHRA        ;ac -> screen
0DC2 FE0O       CPI     CR
0OC4 CC510D     CZ      PRNTNULS
0DC7 03         INX     B
0DC8 C3BC0D     JMP     PRNTSTRL

            ;
            ;  RETURN CURRENT POSITION ON OUTPUT LINE
            ;
            POSFCT:
0OCB 3A9903     LDA     CURSPOS ;POS FUNCTION
            FLOATA:
0DCE 47         MOV     B,A     ;RETURN BYTE ANSWER
0DCF AF         XRA     A
0DD0 C30000     JMP     FLOATAB

            ;
            ;  PLOT STATEMENT
            ;
            PLTSTM:
0DD3 CDF10B     CALL    VALNUMBR        ;bscan numbr    ;GET X-COORDINATE
0DD6 CD100C     CALL    CINTEGER
0DD9 D5         PUSH    D
0DDA CDA303     CALL    SCANNXTV        ;bscan (val)
0DDD 2C         DB      ","
0DDE CDF10B     CALL    VALNUMBR        ;bscan numbr    ;GET Y-COORDINATE
0DE1 CO100C     CALL    CINTEGER
0DE4 D5         PUSH    O
0DE5 CDA303     CALL    SCANNXTV        ;bscan (val)
0DE8 2C         DB      ","
0DE9 CDF10B     CALL    VALNUMBR        ;bscan numbr    ;GET OPERATION
0DEC CD100C     CALL    CINTEGER
0DEF 7B         MOV     A,E
0DF0 D1         POP     D
0DF1 C1         POP     B
0DF2 E5         PUSH    H
            ;   CALL    SYSPLOT
0DF3 E1         POP     H
0DF4 C9         RET
```

```
                ;
                ;  INPUT/READ STATEMENT PROCESSORS
                ;
                MSGQUES:
0DF5 3F3F00         DB      "??",0
                MSGREDO:
0DF8 3F5245         DB      "?REDO FROM START",CR,LF,0
0DFB 444F20
0DFE 46524F
0E01 4D2053
0E04 544152
0E07 540D0A
0E0A 00
                MSGEXTRA:
0E0B 3F4558         DB      "?EXTRA IGNORED",CR,LF,0
0E0E 545241
0E11 204947
0E14 4E4F52
0E17 45440D
0E1A 0A00
                ;  INPUT
                ;
                INPSTM:
0E1C AF             XRA     A       ;INPUT STATEMENT
0E1D 326503         STA     PRINTFLG        ;TURN ON PRINTING
                INPSTMRD:
0E20 E5             PUSH    H       ;SAVE SCAN IN CASE OF ERROR
0E21 0E4E           MVI     C,LINESYZE/2
0E23 CDD804         CALL    SPACESTK
0E26 EB             XCHG
0E27 2A7F03         LHLD    INPTBUFR        ;SAVE ADDRESS OF CURRENT BUFFER
0E2A E5             PUSH    H
0E2B 2160FF         LXI     H,0-LINESYZE-3
0E2E 39             DAD     SP
0E2F F9             SPHL            ;AND CREATE A NEW BUFFER
0E30 227F03         SHLD    INPTBUFR
0E33 EB             XCHG
0E34 7E             MOV     A,M
0E35 FE22           CPI     '"
0E37 CA0000         JZ      INPSTMPR
0E3A FEA3           CPI     KEYPRM
0E3C 11F60D         LXI     D,MSGQUES+1
0E3F C20000         JNZ     INPSTMIN
0E42 CDAB03         CALL    SCANNXT ;bscan +
                INPSTMPR:
0E45 CD0000         CALL    VALEXPR ;bscan expr     ;OPTIONAL PROMPT STRING
0E48 CDFB0B         CALL    CSTRING
0E4B CDA303         CALL    SCANNXTV        ;bscan (val)
0E4E 3B             DB      ";"
0E4F E5             PUSH    H
0E50 CDB50D         CALL    PRNTSTRT
0E53 E1             POP     H
0E54 11F70D         LXI     D,MSGQUES+2
                INPSTMIN:
0E57 E5             PUSH    H
0E58 CD0000         CALL    DATAINPT
```

```
0E58 C30000     JMP     REAINPFS

            ;  REAO
            ;
            REASTM:
0E5E E5         PUSH    H       ;READ STATEMENT
0E5F 2A7D03     LHLO    CUROATAP
0E62 7E         MOV     A,M
0E63 B7         ORA     A
0E64 CC0000     CZ      OATASRCH        ;GET DATA IF NECESSARY

            REAINPFS:
0E67 326403     STA     REAINPFL
0E6A C30000     JMP     REAINPLQ
            REAINPLP:
0E6D COA303     CALL    SCANNXTV        ;bscan (val)
0E70 2C         DB      ","
0E71 E3         XTHL
0E72 7E         MOV     A,M
0E73 FE2C       CPI     ","
0E75 C40000     CNZ     DATAGET
            REAINPLQ:
0E78 E3         XTHL
0E79 7E         MOV     A,M
0E7A FEA4       CPI     KEYLINE ;LINE OPTION?
0E7C CA0000     JZ      INPSTMLN
0E7F CD0000     CALL    VARSCAN ;FIND NEXT VARIABLE TO BE INPUT
0E82 E3         XTHL            ;SAVE INPUT LIST POINTER
0E83 D5         PUSH    D       ;SAVE VARIABLE POINTER,
0E84 3A6B03     LOA     TYPEFLG ;AND TYPE
0E87 F5         PUSH    PSW
0E88 CD0000     CALL    REAINPOC        ;DECODE INPUT
            REAINPLA:
0E88 F1         POP     PSW     ;ASSIGN VALUE
0E8C D1         POP     D
0E8D CDBB0B     CALL    ASSIGN
0E90 2B         DCX     H       ;bscan -
0E91 CDA803     CALL    SCANNXT ;bscan ,
0E94 CA0000     JZ      REAINPCM
0E97 FE2C       CPI     ","     ;DATA ITEMS SEPARTED BY COMMAS
0E99 C20000     JNZ     REAINPER
            REAINPCM:
0E9C E3         XTHL
0E9D 2B         OCX     H       ;bscan -        ;MORE VARIABLES?
0E9E CDA803     CALL    SCANNXT ;bscan ,
0EA1 C26D0E     JNZ     REAINPLP
0EA4 D1         POP     D       ;END OF VARLIST
0EA5 3A6403     LDA     REAINPFL
0EA8 B7         ORA     A
0EA9 E8         XCHG
0EAA C26205     JNZ     RESDTPTR
0EAD D5         PUSH    D
0EAE F5         PUSH    PSW
0EAF 86         ORA     M
0EB0 210B0E     LXI     H,MSGEXTRA
            INPSTMER:
```

```
0EB3 C4AC0D     CNZ     PRNTMSG
0EB6 F1         POP     PSW
```

```
            INPSTMXT:
0EB7 D1         POP     D       ;RECOVER SCAN POINTER
0EB8 21A000     LXI     H,0+LINESYZE+3
0EB8 39         DAD     SP
0EBC F9         SPHL            ;DEALLOCATE BUFFER
0E8D E1         POP     H
0E8E 227F03     SHLD    INPT8UFR        ;AND RESTORE ADDRESS OF OLD
0EC1 EB         XCHG
0EC2 D1         POP     D
0EC3 C8         RZ
0EC4 FACF0A     JM      INPSTMBR        ;BREAK TIME...
0EC7 EB         XCHG
0EC8 C3200E     JMP     INPSTMRD        ;OR REDO THE INPUT

            REAINPER:
0ECB 3A6403     LDA     REAINPFL
0ECE B7         ORA     A
0ECF C2E905     JNZ     ERRDATA
0ED2 21F80D     LXI     H,MSGREDO
0ED5 3C         INR     A
0ED6 F5         PUSH    PSW
0ED7 C3830E     JMP     INPSTMER

            ;
            ;  SEARCH FOR DATA STATEMENT
            ;
            DATAGET:
0EDA 3A6403     LDA     REAINPFL
0EDD B7         ORA     A       ;READ OR INPUT?
0EDE 11F50D     LXI     D,MSGQUES
0EE1 CA0000     JZ      DATAINPT        ;INPUT
            DATASRCH:
0EE4 CD690B     CALL    DATSTM  ;LOOK FOR NEXT DATA STATEMENT
0EE7 B7         DRA     A
0EE8 C20000     JNZ     DATASRCK
0EEB 23         INX     H
0EEC 7E         MOV     A,M
0EED 23         INX     H
0EEE B6         ORA     M
0EEF 23         INX     H
0EF0 1E61       MVI     E,ERRNOD-ERRN
0EF2 CAF105     JZ      ERRMSG
0EF5 5E         MOV     E,M
0EF6 23         INX     H
0EF7 56         MOV     D,M
0EF8 EB         XCHG
0EF9 227B03     SHLD    CURLDATA
0EFC EB         XCHG
            DATASRCK:
0EFD CDAB03     CALL    SCANNXT ;bscan ,
0F00 FE80       CPI     KEYDAT
0F02 C2E40E     JNZ     DATASRCH
0F05 C9         RET

            DATAINPT:
```

```
0F06 CD6B07     CALL    INPTRQST
0F09 C8         RZ              ;INPUT OK, RETURN
0F0A C1         POP     B       ;BREAK ***
0F0B C3B70E     JMP     INPSTMXT

            REAINPDC:
0F0E CDAB03     CALL    SCANNXT ;bscan ,
0F11 CDD903     CALL    TYPECHK
0F14 7E         MOV     A,M
0F15 C20000     JNZ     DECODE  ;READ/INPUT A NUMBER
0F18 FE22       CPI     '"
0F1A CA0000     JZ      VALSTRGC
0F1D 163A       MVI     D,":"
0F1F 062C       MVI     B,","
0F21 2B         DCX     H
0F22 C30000     JMP     VALSTRGS        ;READ/INPUT A STRING

            INPSTMLN:
0F25 3A6403     LDA     REAINPFL        ;LINE OPTION VALID ONLY
0F28 B7         ORA     A       ;FOR INPUT STATEMENT
0F29 C2EF05     JNZ     ERRASN
0F2C CDAB03     CALL    SCANNXT ;bscan +
0F2F CD0000     CALL    VARSCAN
0F32 E3         XTHL
0F33 D5         PUSH    D
0F34 3A6B03     LDA     TYPEFLG
0F37 F5         PUSH    PSW
0F38 0600       MVI     B,0
0F3A CD0000     CALL    VALSTRGY        ;SWALLOW REST OF INPUT LINE
0F3D C38B0E     JMP     REAINPLA        ;AND ASSIGN TO STRING VARIABLE
```

```
                ;
                ;  NEXT STATEMENT PROCESSOR
                ;
                NEXSTM:
0F40 110000         LXI     D,0     ;NEXT STATEMENT
                NEXSTML:
0F43 C40000         CNZ     VARSCAN
0F46 227703         SHLD    PROGCNTR
0F49 CDF909         CALL    FORCHK  ;VERIFY WE"RE IN FOR LOOP
0F4C C20000         JNZ     ERRANF
0F4F F9             SPHL            ;BACK UP STACK
0F50 D5             PUSH    D
0F51 7E             MOV     A,M     ;RECOVER SIGN OF STEPSIZE
0F52 23             INX     H
0F53 F5             PUSH    PSW
0F54 D5             PUSH    D
0F55 CD0000         CALL    LDRGACMM        ;RECOVER STEP SIZE
0F58 E3             XTHL
0F59 E5             PUSH    H
0F5A CD0000         CALL    FLADDM  ;INCREMENT CONTROL VARIA8LE
0F5D E1             POP     H
0F5E CD0000         CALL    LDMMAC
0F61 E1             POP     H
0F62 CD0000         CALL    LDRGMM
0F65 E5             PUSH    H
0F66 CD0000         CALL    FLCMP
0F69 E1             POP     H
0F6A C1             POP     8
0F68 90             SU8     8
0F6C CD0000         CALL    LDRGMM  ;RECOVER LINE NUM8R, PROGRAM CNTR
0F6F CA0000         JZ      NEXSTMC ;CHECK LIMIT
0F72 CD890B         CALL    TRACE
0F75 E8             XCHG
0F76 227303         SHLD    CURLINE
0F79 60             MOV     H,B
0F7A 69             MOV     L,C
0F78 C3800A         JMP     FORMARK

                ERRANF:
0F7E 1E54           MVI     E,ERRNNF-ERRN
0F80 C3F105         JMP     ERRMSG

                NEXSTMC:
0F83 F9             SPHL            ;END OF LOOP...
0F84 2A7703         LHLD    PROGCNTR
0F87 7E             MOV     A,M
0F88 FE2C           CPI     ","
0F8A C2840A         JNZ     EXECUTEL        ;MORE INDICES?
0F8D CDA803         CALL    SCANNXT ;bscan ,
0F90 CD430F         CALL    NEXSTML
```

```
                ;
                ;  EVALUATE AN EXPRESSION
                ;
                VALEXPR:
0F93 2B             DCX     H       ;SCAN & EVALUATE AN EXPRESSION
0F94 1600           MVI     D,0     ;INITIAL PRECEDENCE=0
                VALEXPRL:
0F96 D5             PUSH    D
0F97 0E01           MVI     C,1
0F99 CDD804         CALL    SPACESTK
0F9C CD0000         CALL    VALPRMRY        ;bscan prmry
0F9F 227103         SHLD    SCANPTR2
                VALEXPRC:
0FA2 2A7103         LHLD    SCANPTR2
                VALEXPRD:
0FA5 C1             POP     B       ;PREVIOUS PRECEDENCE
0FA6 78             MOV     A,B
0FA7 FE70           CPI     PREDNUM
0FA9 D4F40B         CNC     CSINGLE
0FAC 7E             MOV     A,M
0FAD 1600           MVI     D,000H
                VALEXPRR:
0FAF D6B4           SUI     KEYREL  ;RELATION?
0FB1 DA0000         JC      VALEXPRO
0FB4 FE03           CPI     KEYFCT-KEYREL
0FB6 D20000         JNC     VALEXPRO
0FB9 FE01           CPI     1       ;YES
0FBB 17             RAL
0FBC AA             XRA     D       ;CONVERT 0,1,2 TO 1,2,4
0FBD BA             CMP     D
0FBE 57             MOV     D,A
0FBF DAEF05         JC      ERRASN
0FC2 226F03         SHLD    SCANPTR1
0FC5 CDAB03         CALL    SCANNXT ;bscan ,
0FC8 C3AF0F         JMP     VALEXPRR

                VALEXPRO:
0FCB 7A             MOV     A,D
0FCC B7             ORA     A
0FCD C20000         JNZ     VALREL
0FD0 7E             MOV     A,M
0FD1 226F03         SHLD    SCANPTR1
0FD4 D6AA           SUI     KEYDPR  ;OPERATOR?
0FD6 D8             RC
0FD7 FE0A           CPI     KEYREL-KEYOPR
0FD9 D0             RNC
0FDA 5F             MOV     E,A     ;YES
0FDB CDD903         CALL    TYPECHK ;STRING OPERANDS?
0FDE B3             DRA     E       ;AND CATENATION OPERATOR?
0FDF 7B             MOV     A,E
0FE0 CA0000         JZ      VALCONCT        ;YES
0FE3 B3             ADD     E
0FE4 83             ADD     E
0FE5 5F             MOV     E,A
0FE6 218E00         LXI     H,OPRTABL
```

```
0FE9 19          DAD     D
0FEA 78          MOV     A,B
0FEB 56          MOV     D,M
0FEC BA          CMP     D
0FED D0          RNC
0FEE 23          INX     H
0FEF CDF40B      CALL    CSINGLE
                 VALEXPR2:
0FF2 C5          PUSH    B       ;STACK OPERATION,
0FF3 01A20F      LXI     B,VALEXPRC      ;EVALUATE SECOND OPERAND
0FF6 C5          PUSH    B
0FF7 42          MOV     B,D
0FF8 4B          MOV     C,E
0FF9 CD0000      CALL    PUSHAC
0FFC 50          MOV     D,B
0FFD 59          MOV     E,C
0FFE 4E          MOV     C,M
0FFF 23          INX     H
1000 46          MOV     B,M
1001 C5          PUSH    B
1002 2A6F03      LHLD    SCANPTR1
1005 C3960F      JMP     VALEXPRL

                 ;
                 ;  EVALUATE A RELATION
                 ;
                 VALREL:
1008 210000      LXI     H,RELOPR        ;SCAN & EVALUATE RELATION
100B 3A6B03      LDA     TYPEFLG
100E 07          RLC
100F 07          RLC
1010 07          RLC
1011 B2          ORA     D
1012 5F          MOV     E,A
1013 1664        MVI     D,PREDREL
1015 78          MOV     A,B
1016 BA          CMP     D
1017 D0          RNC
1018 C3F20F      JMP     VALEXPR2

                 RELOPRXT:
101B 3C          INR     A       ;MATCH RESULT OF COMPARISON
101C 8F          ADC     A       ;-1,0,1 TO 1,2,4
101D C1          POP     B       ;VERSUS RELATION TO BE TESTED
101E A0          ANA     B
101F C6FF        ADI     -1
1021 9F          SBB     A
1022 C30000      JMP     FLOATBYT
```

```
                RELOPR:
1025 0000           DW      RELOPRC ;COMPUTE RELATION
                RELOPRC:
1027 79             MOV     A,C
1028 C1             POP     B
1029 D1             POP     D
102A F5             PUSH    PSW
102B 0F             RRC
102C 0F             RRC
102D 0F             RRC
102E E60F           ANI     00FH
1030 CDE50B         CALL    COERCE
1033 211B10         LXI     H,RELOPRXT
1036 E5             PUSH    H
1037 C20000         JNZ     FLCMP   ;NUMERIC COMPARISON?
103A 3E04           MVI     A,TYPESING      ;NO, STRING
103C 326B03         STA     TYPEFLG
103F D5             PUSH    D
1040 CD0000         CALL    STRGRELA        ;RELEASE TEMP OF SECOND OPERAND
1043 D1             POP     D
1044 4E             MOV     C,M
1045 23             INX     H
1046 C5             PUSH    B       ;SAVE LENGTH
1047 4E             MOV     C,M
1048 23             INX     H
1049 46             MOV     B,M
104A C5             PUSH    B       ;AND ADDRESS
104B CD0000         CALL    STRGRELD        ;RELEASE TEMP OF FIRST OPERAND
104E CD0000         CALL    LDDCBMM
1051 E1             POP     H
1052 E3             XTHL
1053 5D             MOV     E,L
1054 E1             POP     H
                RELOPRSL:
1055 7B             MOV     A,E     ;COMPARE CHARACTER BY CHARACTER
1056 B2             ORA     D
1057 C8             RZ
1058 7B             MOV     A,E
1059 D601           SUI     1
105B D8             RC
105C AF             XRA     A
105D BA             CMP     D
105E 3C             INR     A
105F D0             RNC
1060 15             DCR     D
1061 1D             DCR     E
1062 0A             LDAX    B
1063 BE             CMP     M
1064 23             INX     H
1065 03             INX     B
1066 CA5510         JZ      RELOPRSL
1069 3F             CMC
106A C30000         JMP     CMPXT
```

```
                ;
                ;  EVALUATE A PRIMARY
                ;
                VALPRMRY:
106D 3E04           MVI     A,TYPESING      ;SCAN & EVALUATE A PRIMARY
106F 326B03         STA     TYPEFLG
1072 CDAB03         CALL    SCANNXT ;bscan ,
1075 DA0000         JC      DECODE  ;NUMERIC CONSTANT?
1078 CDBB03         CALL    ALPHACHK
107B DA0000         JC      VALVAR  ;VARIABLE?
107E FEAA           CPI     KEYADD
1080 CA6D10         JZ      VALPRMRY
1083 FE2E           CPI     "."
1085 CA0000         JZ      DECODE
1088 FEAB           CPI     KEYSUB
108A CA0000         JZ      VALUMINS
108D FE22           CPI     '"      ;STRING CONSTANT?
108F CA0000         JZ      VALSTRGC
1092 FEA8           CPI     KEYNDT
1094 CA0000         JZ      VALUNOT
1097 FEA7           CPI     KEYFN   ;DEFINED FUNCTION?
1099 CA0000         JZ      VALFCTD
109C FE8B           CPI     KEYIF   ;CONDITIONAL EXPRESSION?
109E CA0000         JZ      VALCOND
10A1 D6B7           SUI     KEYFCT  ;INTRINSIC FUNCTION?
10A3 D20000         JNC     VALFCTN
                VALPARNS:
10A6 CDA303         CALL    SCANNXTV        ;bscan (val)
10A9 28             DB      "("
                VALPARN2:
10AA CD930F         CALL    VALEXPR ;bscan expr
10AD CDA303         CALL    SCANNXTV        ;bscan (val)
10B0 29             DB      ")"
10B1 C9             RET

                VALUMINS:
10B2 167D           MVI     D,PREDUMIN      ;EVALUATE UNARY MINUS
10B4 CD960F         CALL    VALEXPRL
10B7 2A7103         LHLD    SCANPTR2
10BA E5             PUSH    H
10BB CD0000         CALL    CMACCS
                VALRETNM:
10BE CDF40B         CALL    CSINGLE
10C1 E1             POP     H
10C2 C9             RET
```

```
                ;
                ;  EVALUATE A VARIABLE
                ;
                VALVAR:
10C3 CD0000         CALL    VARSCAN ;SCAN & EVALUATE VARIABLE
10C6 E5             PUSH    H
10C7 D5             PUSH    D
10C8 EB             XCHG
10C9 1ED1           MVI     E,ERRNUV-ERRN
10CB C2F105         JNZ     ERRMSG
10CE 229303         SHLD    ACCUMLTR
10D1 CDD903         CALL    TYPECHK
10D4 EB             XCHG
10D5 219303         LXI     H,ACCUMLTR
10D8 C40000         CNZ     CDPYVAL
10DB D1             PDP     D
10DC E1             POP     H
10DD C9             RET

                ;
                ;  EVALUATE CDNDITIDNAL EXPRESSIDN
                ;
                VALCDND:
10DE CDAB03         CALL    SCANNXT ;bscan , EVAL CDNDITIONAL EXPRESSION
10E1 CDF10B         CALL    VALNUMBR        ;bscan numbr
10E4 CDA303         CALL    SCANNXTV        ;bscan (val)
10E7 A0             DB      KEYTHEN
10E8 CD0000         CALL    SIGNACC
10EB CA0000         JZ      VALCDNDF
10EE CD930F         CALL    VALEXPR ;bscan expr     ;TRUE, EVALUATE THEN PORTION
10F1 1601           MVI     D,1
                VALCNDTL:
10F3 0E82           MVI     C,KEYEND
10F5 CD700B         CALL    SCAN2KEY        ;SKIP ELSE PDRTION
10F8 CDA303         CALL    SCANNXTV        ;bscan (val)
10FB 82             DB      KEYEND
10FC 15             DCR     D
10FD C2F310         JNZ     VALCNDTL
1100 C9             RET

                VALCONDF:
1101 1601           MVI     D,1
                VALCNDFL:
1103 0E8C           MVI     C,KEYELS        ;FALSE, SKIP THEN PDRTIDN
1105 CD700B         CALL    SCAN2KEY
1108 CDA303         CALL    SCANNXTV        ;bscan (val)
110B 8C             DB      KEYELS
110C 15             DCR     D
110D C20311         JNZ     VALCNDFL
1110 CD930F         CALL    VALEXPR ;bscan expr     ;EVALUATE ELSE PORTIDN
1113 CDA303         CALL    SCANNXTV        ;bscan (val)
1116 82             DB      KEYEND
1117 C9             RET
```

```
                ;
                ;  EVALUATE INTRINSIC FUNCTION
                ;
                VALFCTN:
1118 0600           MVI     B,000H      ;Scan & EVALUATE INTRINSIC FUNCTION CALL
111A 07             RLC
111B 4F             MOV     C,A
111C C5             PUSH    B
111D CDAB03         CALL    SCANNXT ;bscan ,
1120 79             MOV     A,C
1121 FE2F           CPI     KEYLFT-KEYFCT*2-1       ;LEFT$, MID$, or RIGHT$
1123 DA0000         JC      VALFCTAR
1126 CDA303         CALL    SCANNXTV        ;bscan (val)
1129 28             DB      "("
112A CD930F         CALL    VALEXPR ;bscan expr
112D CDFB0B         CALL    CSTRING
1130 EB             XCHG
1131 2A9303         LHLD    ACCUMLTR
1134 E3             XTHL            ;PUSH STRING ONTO STACK
1135 C30000         JMP     VALFCTLK

                VALFCTAR:
1138 CDA610         CALL    VALPARNS        ;EVALUATE ARGUMENT TO FUNCTION
113B E3             XTHL
113C 11BE10         LXI     D,VALRETNM
113F D5             PUSH    D
                VALFCTLK:
1140 01AC00         LXI     B,FCTTABL       ;BRANCH TO APPROPRIATE ROUTINE
1143 09             DAD     B
1144 4E             MOV     C,M
1145 23             INX     H
1146 66             MOV     H,M
1147 69             MOV     L,C
1148 E9             PCHL            ;CALL FUNCTION
```

```
                ;
                ;  PROCESS STRING CONSTANT
                ;
                VALSTRGN:
1149 CD0000         CALL    ENCODE  ;CREATE STRING FROM NUMBER
                VALSTRGZ:
114C 0680           MVI     B,080H
114E 2B             DCX     H
114F C30000         JMP     VALSTRGY

                VALSTRGC:
1152 0622           MVI     B,'"    ;SCAN & DECODE A STRING CONSTANT
                VALSTRGY:
1154 50             MOV     D,B
                VALSTRGS:
1155 E5             PUSH    H
1156 0EFF           MVI     C,-1
                VALSTRGL:
1158 23             INX     H       ;FIND STRING LENGTH
1159 7E             MOV     A,M
115A 0C             INR     C
115B B7             ORA     A
115C CA0000         JZ      VALSTRGE
115F BA             CMP     D
1160 CA0000         JZ      VALSTRGE
1163 B8             CMP     B
1164 C25811         JNZ     VALSTRGL
                VALSTRGE:
1167 FE22           CPI     '"
1169 CCAB03         CZ      SCANNXT
116C E3             XTHL
116D 23             INX     H
116E EB             XCHG
116F 79             MOV     A,C
1170 CD0000         CALL    STRSTCDS
1173 EB             XCHG
1174 CDAD0C         CALL    STRGTEST        ;LOCATE STRING
1177 3F             CMC
1178 1F             RAR
1179 B0             ORA     B
117A F40000         CP      STRGSTOR        ;MAKE A COPY OF CERTAIN BUFFERS
```

```
                ;
                ;  ALLOCATE STRING TEMPORARY
                ;
                STRGALOT:
117D 116C03         LXI     D,STRGTMPL      ;USE CURRENT DESCRIPTOR
                STRGALOU:
1180 D5             PUSH    D
1181 3E03           MVI     A,TYPESTRG      ;RETURN STRING RESULT
1183 326B03         STA     TYPEFLG
1186 2A8F03         LHLD    STRGTMPP        ;IN A NEW STRING TEMPORARY
1189 229303         SHLD    ACCUMLTR
118C EB             XCHG
118D 2A9103         LHLD    STRGTLIM        ;ANY MORE TEMPORARIES?
1190 CDC104         CALL    CMHLLTDE
1193 DA0000         JC      ERRAST
1196 EB             XCHG
1197 D1             POP     D       ;GET DESCRIPTOR
1198 CD0000         CALL    COPYVAL ;COPY IT
119B 228F03         SHLD    STRGTMPP
119E E1             POP     H
119F C9             RET

                STRGALOV:
11A0 E5             PUSH    H
11A1 C38011         JMP     STRGALOU

                ERRAST:
11A4 1E92           MVI     E,ERRNST-ERRN
11A6 C3F105         JMP     ERRMSG
```

```
                ;
                ;  RELEASE STRING RESOURCES
                ;
                STRGRELA:
11A9 2A9303         LHLD    ACCUMLTR
                STRGRELH:
11AC EB             XCHG
                STRGRELD:
11AD CD0000         CALL    STRGRELT        ;RELEASE TEMPORARY
11B0 EB             XCHG
11B1 C0             RNZ             ;NOT OUR BOY
11B2 D5             PUSH    D
11B3 50             MOV     D,B
11B4 59             MOV     E,C
11B5 1B             DCX     D
11B6 4E             MOV     C,M
11B7 2A8B03         LHLD    STRGFREE
11BA CDC104         CALL    CMHLLTDE
11BD C27008         JNZ     POPHLRET
11C0 47             MOV     B,A     ;RELEASE STRING SPACE
11C1 09             DAD     B
11C2 228B03         SHLD    STRGFREE
11C5 E1             POP     H
11C6 C9             RET

                ;
                ;  RELEASE STRING TEMPORARY
                ;
                STRGRELT:
11C7 2A8F03         LHLD    STRGTMPP        ;RELEASE STRING TEMPORARY
11CA 2B             DCX     H
11CB 46             MOV     B,M
11CC 2B             DCX     H
11CD 4E             MOV     C,M
11CE 2B             DCX     H
11CF CDC104         CALL    CMHLLTDE
11D2 C0             RNZ
11D3 228F03         SHLD    STRGTMPP        ;RELEASE STRING TEMPORARY
11D6 C9             RET
```

```
                ;
                ;  EVALUATE A CATENATION
                ;
                VALCONCT:
11D7 C5             PUSH    B       ;EVALUATE A CONCATENATION
11D8 E5             PUSH    H
11D9 2A9303         LHLD    ACCUMLTR        ;SAVE FIRST OPERAND,
11DC E3             XTHL
11DD CD6D10         CALL    VALPRMRY        ;bscan prmry    ;EVALUATE SECOND
11E0 E3             XTHL
11E1 CDFB0B         CALL    CSTRING
11E4 7E             MOV     A,M     ;ADD LENGTHS,
11E5 E5             PUSH    H
11E6 2A9303         LHLD    ACCUMLTR
11E9 E5             PUSH    H
11EA 86             ADD     M
11EB 1E3B           MVI     E,ERRNLS-ERRN
11ED DAF105         JC      ERRMSG
11F0 CD0000         CALL    STRNGEN ;AND ALLOCATE OUTPUT STRING
11F3 D1             POP     D
11F4 CDAD11         CALL    STRGRELD        ;RELEASE STRING TEMPORARIES
11F7 E3             XTHL
11F8 CDAC11         CALL    STRGRELH
11FB E5             PUSH    H
11FC 2A6D03         LHLD    STRGTMPA        ;COPY STRINGS TO OUTPUT STRING
11FF EB             XCHG
1200 CD0000         CALL    VALCONCP
1203 CD0000         CALL    VALCONCP
1206 21A50F         LXI     H,VALEXPRD
1209 E3             XTHL
120A E5             PUSH    H
120B C37D11         JMP     STRGALOT

                VALCONCP:
120E E1             POP     H       ;COPY STRING FOR CATENATION
120F E3             XTHL
1210 7E             MOV     A,M     ;GET LENGTH,
1211 23             INX     H
1212 4E             MOV     C,M     ;ADDRESS OF STRING
1213 23             INX     H
1214 46             MOV     B,M
1215 6F             MOV     L,A

                COPYSTRG:
1216 2C             INR     L       ;COPY A STRING OF LENGTH L
                COPYSTRL:
1217 2D             DCR     L       ;FROM BC TO DE
1218 C8             RZ
1219 0A             LDAX    B
121A 12             STAX    D
121B 03             INX     B
121C 13             INX     D
121D C31712         JMP     COPYSTRL
```

```
                ;
                ;  DIMENSION STATEMENT PROCESSING
                ;
                DIMSTML:
1220 2B             DCX     H
1221 CDAB03         CALL    SCANNXT ;bscan ,
1224 C8             RZ
1225 CDA303         CALL    SCANNXTV        ;bscan (val)
1228 2C             DB      ","
                DIMSTM:
1229 012012         LXI     B,DIMSTML       ;DIM STATEMENT
122C C5             PUSH    B
122D 3E80           MVI     A,080H
122F C30000         JMP     VARSCANI

                ;
                ;  SCAN A VARIABLE NAME
                ;
                VARSCAN:
1232 AF             XRA     A       ;SCAN FOR VARIABLE
                VARSCANI:
1233 326A03         STA     MATDMFLG
1236 0600           MVI     B,0*TYPEDEF
                VARSCNDF:
1238 CDBB03         CALL    ALPHACHK        ;ENTRY TO SCAN FOR DEFINED FCT
123B D2EF05         JNC     ERRASN
123E B0             ORA     B
123F 47             MOV     B,A
1240 0E3F           MVI     C,"?"
1242 1604           MVI     D,TYPESING      ;ASSUME NUMERIC VARIABLE
1244 CDAB03         CALL    SCANNXT ;bscan ,
1247 DA0000         JC      VARSCAND
124A CDBB03         CALL    ALPHACHK
124D D20000         JNC     VARSCANS
                VARSCAND:
1250 4F             MOV     C,A
                VARSKIPL:
1251 CDAB03         CALL    SCANNXT ;bscan ,        ;SKIP EXTRA ALPHANUMERIC
1254 DA5112         JC      VARSKIPL        ;CHARACTERS IN NAME
1257 CDBB03         CALL    ALPHACHK
125A DA5112         JC      VARSKIPL
                VARSCANS:
125D D624           SUI     "$"     ;STRING VARIABLE?
125F C20000         JNZ     VARNAME
1262 1603           MVI     D,TYPESTRG      ;YES
1264 CDAB03         CALL    SCANNXT ;bscan ,

                VARNAME:
1267 78             MOV     A,B     ;TRANSLATE IDENT TO INTERNAL FORM
1268 D640           SUI     "@"     ;DEF/VARIABLE IS FIRST BIT
126A 07             RLC             ;FIRST CHAR IS NEXT FIVE BITS
126B 07             RLC
126C 47             MOV     B,A
126D 79             MOV     A,C     ;SECOND CHAR IS NEXT SIX BITS
126E D630           SUI     "0"
```

```
1270 0F         RRC
1271 0F         RRC
1272 0F         RRC
1273 0F         RRC
1274 4F         MOV     C,A
1275 A8         XRA     B       ;PACK THREE BYTES INTO TWO
1276 E603       ANI     003H
1278 A8         XRA     B
1279 47         MOV     B,A
127A 7A         MOV     A,D
127B 326B03     STA     TYPEFLG
127E A9         XRA     C       ;TYPE IS LAST FOUR BITS
127F E60F       ANI     00FH
1281 A9         XRA     C
1282 4F         MOV     C,A

1283 3A6703     LDA     SCANPFLG
1286 86         ADD     M
1287 FE28       CPI     "("     ;SUBSCRIPTED?
1289 CA0000     JZ      MATSCANP
128C FE58       CPI     "["     ;BY LEFT BRACKET?
128E CA0000     JZ      MATSCANB
1291 AF         XRA     A
1292 326703     STA     SCANPFLG
1295 E5         PUSH    H

                ;
                ;  LOOK UP VARIABLE IN TABLE
                ;
1296 2A8303         LHLD    VARTABLE
                VARSCANT:
1299 EB             XCHG
129A 2A8503         LHLD    MATTABLE
129D CDC104         CALL    CMHLLTDE        ;LOOK THROUGH VARIABLE TABLE
12A0 CA0000         JZ      VARSCANF
12A3 1A             LDAX    D
12A4 6F             MOV     L,A
12A5 B9             CMP     C
12A6 13             INX     D
12A7 C20000         JNZ     VARSCANM
12AA 1A             LDAX    D
12AB BB             CMP     B
                VARSCANM:
12AC 13             INX     D
12AD CA0000         JZ      VARSCANX
12B0 7D             MOV     A,L
12B1 E60F           ANI     00FH    ;ADDRESS NEXT ENTRY
12B3 6F             MOV     L,A
12B4 2600           MVI     H,0
12B6 19             DAD     D
12B7 C39912         JMP     VARSCANT

                VARSCANF:
12BA C5             PUSH    B       ;NOT FOUND, CREATE ENTRY
12BB 79             MOV     A,C
12BC E60F           ANI     00FH
```

```
12BE C602       AOI     2
12C0 4F         MOV     C,A
12C1 0600       MVI     B,0
12C3 EB         XCHG
12C4 2AB703     LHLO    FREELIMT
12C7 E5         PUSH    H
12C8 09         DAO     B
12C9 C1         POP     B
12CA E5         PUSH    H
12CB CDC704     CALL    COPYCHK ;MOVE ARRAYS FOR SPACE
12CE E1         POP     H
12CF 22B703     SHLO    FREELIMT
12D2 60         MOV     H,B
12D3 69         MOV     L,C
12D4 22B503     SHLO    MATTABLE        ;ALLOCATE, ZERO ENTRY
            VARALLOC:
12D7 2B         DCX     H
12D8 3600       MVI     M,000H
12DA CDC104     CALL    CMHLLTOE
12DD C2D712     JNZ     VARALLOC
12E0 01         POP     O
12E1 73         MOV     M,E
12E2 23         INX     H
12E3 72         MOV     M,D
12E4 23         INX     H
12E5 EB         XCHG            ;EXIT VARIABLE SCAN
12E6 B3         ORA     E       ;NZ=VAR NOT FOUND, CREATED
            VARSCANX:
12E7 E1         POP     H       ;HL=SCAN POINTER
12EB C9         RET             ;BE=VARIABLE REFERENCE

            ;
            ;  LOOK UP ARRAY IN TABLE
            ;
            MATSCANB:
12E9 C601       ADI     ']-'[+'(-')     ;(got me?)
            MATSCANP:
12EB C601       ADI     ')-'(
12E0 E5         PUSH    H       ;SCAN SUBSCRIPT OF VARIABLE
12EE 2A6A03     LHLD    MATDMFLG
12F1 B5         ORA     L
12F2 6F         MOV     L,A
12F3 E3         XTHL            ;SAVE DIMFLAG, CLOSE CHAR, TYPE
12F4 1600       MVI     O,000H
            MATSCANL:
12F6 D5         PUSH    D       ;SCAN SUBSCRIPT LIST
12F7 C5         PUSH    B
12FB CDAB03     CALL    SCANNXT ;bscan ,
12FB CD070C     CALL    VALINTOE        ;EVALUATE SUBSCRIPT
12FE C1         POP     B
12FF F1         POP     PSW
1300 EB         XCHG
1301 E3         XTHL
1302 E5         PUSH    H
1303 EB         XCHG
1304 3C         INR     A       ;COUNT NUMBER OF SUBSCRIPTS
```

```
1305 57          MOV     D,A
1306 7E          MOV     A,M
1307 FE2C        CPI     ","
1309 CAF612      JZ      MATSCANL
130C E3          XTHL
130D 226A03      SHLD    MATDMFLG        ;RESTORE DIMFLAG, TYPE
1310 7D          MOV     A,L
1311 E1          POP     H
1312 AE          XRA     M
1313 87          ADD     A       ;CHECK FOR CORRECT CLOSER
1314 C2EF05      JNZ     ERRASN
1317 227103      SHLD    SCANPTR2
131A D5          PUSH    D
131B 2A8503      LHLD    MATTABLE        ;LOOK FOR NAME IN
131E C30000      JMP     MATSCANO        ;MAT VARIABLE TABLE
            MATSCANN:
1321 19          DAD     D
            MATSCANO:
1322 EB          XCHG
1323 2A8703      LHLD    FREELIMT
1326 E8          XCHG
1327 CDC104      CALL    CMHLLTDE
132A CA0000      JZ      MATSCANC
132D 7E          MOV     A,M
132E B9          CMP     C
132F 23          INX     H
1330 C20000      JNZ     MATSCANM
1333 7E          MOV     A,M
1334 88          CMP     B
            MATSCANM:
1335 23          INX     H
1336 5E          MOV     E,M
1337 23          INX     H
1338 56          MOV     D,M
1339 23          INX     H
133A C22113      JNZ     MATSCANN
133D 3A6A03      LDA     MATDMFLG        ;NAME FOUND
1340 B7          ORA     A
1341 1E10        MVI     E,ERRNDD-ERRN
1343 FAF105      JM      ERRMSG
1346 F1          POP     PSW     ;RIGHT NUMBER OF SUBSCRIPTS?
1347 BE          CMP     M
1348 CA0000      JZ      MATSCANI
            ERRABS:
134B 1E9F        MVI     E,ERRNBS-ERRN
134D C3F105      JMP     ERRMSG

            MATSCANC:
1350 79          MOV     A,C     ;NAME NOT FOUND, CREATE NEW ENTRY
1351 E60F        ANI     00FH
1353 5F          MOV     E,A
1354 1600        MVI     D,0
1356 71          MOV     M,C
1357 23          INX     H
1358 70          MOV     M,B
1359 23          INX     H
```

```
135A F1          POP     PSW
135B 326903      STA     MATSCCNT
135E 4F          MOV     C,A
135F CDD804      CALL    SPACESTK
1362 226F03      SHLD    SCANPTR1
1365 23          INX     H
1366 23          INX     H       ;plus 2
1367 41          MOV     B,C
1368 70          MOV     M,B
1369 23          INX     H
             MATSCNS8:
136A 3A6A03      LDA     MATDMFLG        ;SET SU8SCRIPT RANGES
136D 87          ORA     A
136E 78          MOV     A,8
136F 010B00      LXI     B,11    ;DEFAULT RANGE=0-10
1372 F20000      JP      MATSCNSD
1375 C1          POP     8
1376 03          INX     B
             MATSCNSD:
1377 71          MOV     M,C
1378 23          INX     H
1379 70          MOV     M,8
137A 23          INX     H
1378 F5          PUSH    PSW
137C E5          PUSH    H
137D CD0000      CALL    MUL16   ;UPDATE ARRAY SIZE
1380 EB          XCHG
1381 E1          POP     H
1382 C1          POP     8
13B3 05          DCR     8
1384 C26A13      JNZ     MATSCNS8
13B7 42          MOV     8,D
13BB 4B          MOV     C,E
1389 EB          XCHG            ;ALLOCATE ARRAY,
13BA 19          DAD     D
138B DA4B13      JC      ERRA8S
138E CDE504      CALL    SPACECHK
1391 228703      SHLD    FREELIMT
             MATSCANZ:
1394 2B          DCX     H       ;AND ZERO
1395 3600        MVI     M,000H
1397 CDC104      CALL    CMHLLTDE
139A C29413      JNZ     MATSCANZ
139D 03          INX     8       ;SAVE ENTRY SIZE
139E 67          MOV     H,A
139F 3A6A03      LDA     MATDMFLG
13A2 B7          DRA     A
13A3 3A6903      LDA     MATSCCNT
13A6 6F          MOV     L,A
13A7 29          DAD     H
13A8 09          DAD     B
13A9 EB          XCHG
13AA 2A6F03      LHLD    SCANPTR1        ;AT BEGINNING OF ENTRY
13AD 73          MOV     M,E
13AE 23          INX     H
13AF 72          MOV     M,D
```

```
13B0 23          INX     H
13B1 FA0000      JM      MATSCANX        ;DIM ONLY?
                 MATSCANI:
13B4 23          INX     H       ;INITIALIZE SUBSCRIPT COMPUTATION
13B5 010000      LXI     B,0
13B8 C30000      JMP     MATSCANS
                 MATSCANR:
13BB E1          POP     H       ;COMPUTE SPECIFIC REFERENCE
                 MATSCANS:
13BC 5E          MOV     E,M
13BD 23          INX     H
13BE 56          MOV     D,M
13BF 23          INX     H
13C0 E3          XTHL
13C1 F5          PUSH    PSW
13C2 CDC104      CALL    CMHLLTDE
13C5 D24B13      JNC     ERRABS
13C8 E5          PUSH    H
13C9 CD0000      CALL    MUL16
13CC D1          POP     D
13CD 19          DAD     D
13CE F1          POP     PSW
13CF 3D          DCR     A
13D0 44          MOV     B,H
13D1 4D          MOV     C,L
13D2 C2BB13      JNZ     MATSCANR
13D5 3A6B03      LDA     TYPEFLG
13D8 5F          MOV     E,A
13D9 1600        MVI     D,0
13DB CD0000      CALL    MUL16   ;MULTIPLY BY ENTRY SIZE
13DE C1          POP     B
13DF 09          DAD     B
13E0 EB          XCHG
                 MATSCANX:
13E1 2A7103      LHLD    SCANPTR2
13E4 CDAB03      CALL    SCANNXT ;bscan ,
13E7 BF          CMP     A
13EB C9          RET

                 MUL16:
13E9 210000      LXI     H,0     ;MULTIPLY BC*DE GIVING HL
13EC 7B          MOV     A,B
13ED B1          ORA     C
13EE CB          RZ
13EF 3E10        MVI     A,16
                 MUL16LP:
13F1 29          DAD     H
13F2 DA4B13      JC      ERRABS
13F5 EB          XCHG
13F6 29          DAD     H
13F7 EB          XCHG
13FB D20000      JNC     MUL16XT
13FB 09          DAD     B
13FC DA4B13      JC      ERRABS
                 MUL16XT:
13FF 3D          DCR     A
```

```
1400 C2F113     JNZ     MUL16LP
1403 C9         RET
```

```
                ;
                ;  USER-DEFINED FUNCTION DEFINITION
                ;
                DEFSTM:
1404 CD0000         CALL    SCANFNN ;DEF STATEMENT
1407 E5             PUSH    H       ;CHECK IF IN DIRECT MODE
1408 2A7303         LHLD    CURLINE ;Z=DIRECT MDDE
140B 23             INX     H
140C 7C             MOV     A,H
140D B5             ORA     L
140E E1             POP     H
140F CA0000         JZ      ERRAID
1412 EB             XCHG            ;SAVE REFERENCE TO DEFINITION
1413 73             MOV     M,E
1414 23             INX     H
1415 72             MDV     M,D
1416 EB             XCHG
1417 7E             MOV     A,M
1418 FE28           CPI     "("     ;CHECK FDR VARLIST
                DEFSTML:
141A C2690B         JNZ     DATSTM
141D CDAB03         CALL    SCANNXT ;bscan ,
1420 CD3212         CALL    VARSCAN ;DEFINE VARIABLES IN LIST
1423 7E             MDV     A,M
1424 FE2C           CPI     ","
1426 C31A14         JMP     DEFSTML

                ;  USER-DEFINED FUNCTION EVALUATION
                ;
                VALFCTD:
1429 CD0000         CALL    SCANFNN         SCAN    ;& EVALUATE USER DEFINED FUNCTION
142C 3A6B03         LDA     TYPEFLG ;SAVE TYPE DF FUNCTIDN
142F B7             DRA     A
1430 F5             PUSH    PSW
1431 E5             PUSH    H       ;SAVE CALL ARGUMENTS
1432 EB             XCHG
1433 7E             MDV     A,M
1434 23             INX     H
1435 66             MDV     H,M     ;FETCH FUNCTION DEFINITIDN
1436 6F             MDV     L,A
1437 B4             DRA     H
1438 1EB5           MVI     E,ERRNUF-ERRN
143A CAF105         JZ      ERRMSG  ;MUST BE DEFINED ...
```

```
143D 7E         MOV     A,M
143E FE28       CPI     "("     ;PARAMETERS NEEDED?
1440 C20000     JNZ     VALFCTNA        ;APPARENTLY NOT
1443 CDAB03     CALL    SCANNXT ;bscan ,
1446 E3         XTHL
1447 CDA303     CALL    SCANNXTV        ;bscan (val)
144A 28         DB      "("     ;MUST BE PARAMETERS IN CALL
144B E3         XTHL
144C C30000     JMP     VALFCTDM


            ;  ARGUMENT SCANNING
            ;
            VALFCTDL:
144F CDA303     CALL    SCANNXTV        ;bscan (val)
1452 2C         DB      ","     ;COMMAS BETWEEN ARGUMENTS
1453 E3         XTHL
1454 CDA303     CALL    SCANNXTV        ;bscan (val)
1457 2C         DB      ","     ;AND BETWEEN PARAMETERS
            VALFCTDM:
145B 0E04       MVI     C,4     ;VERIFY SPACE ON STACK
145A CDDB04     CALL    SPACESTK
145D 3EAB       MVI     A,SCANPFLD      ;SCAN NEXT PARAMETER
145F 326703     STA     SCANPFLG
1462 CDC310     CALL    VALVAR  ;GET CURRENT VALUE DF PARAMETER
1465 226F03     SHLD    SCANPTR1        ;SAVE PARAMETER SCAN
146B E1         POP     H
1469 227103     SHLD    SCANPTR2        ;SAVE ARGUMENT SCAN
146C CDD903     CALL    TYPECHK
146F CA0000     JZ      VALFCTPS        ;PUSH STRINGS DIFFERENTLY
1472 CD0000     CALL    PUSHAC1 ;PUSH NUMERIC ACCUMULATDR
1475 E5         PUSH    H       ;SAVE VARIABLE"S ADDRESS
1476 C30000     JMP     VALFCTPT
            VALFCTPS:
1479 CDA011     CALL    STRGALDV        ;CDPY DESCRIPTDR TD TEMPDRARY
147C AF         XRA     A       ;ELIMINATE DRIGINAL DESCRIPTOR
147D 1B         DCX     D
147E 1B         DCX     D
147F 1B         DCX     D       ;plus 3
14B0 12         STAX    D
1481 2A9303     LHLD    ACCUMLTR        ;GET ADDRESS OF DESCRIPTDR
14B4 E5         PUSH    H
1485 D5         PUSH    D       ;PUT IT BACK HERE LATER
```

```
                VALFCTPT:
1486 3A6B03         LDA     TYPEFLG ;SAVE TYPE OF PARAMETER
1489 37             STC
148A D1             POP     D
148B D5             PUSH    D       ;GET COPY OF ADDRESS
148C F5             PUSH    PSW
148D 2A6F03         LHLD    SCANPTR1        ;SAVE PARAMETER SCAN
1490 E5             PUSH    H
1491 2A7103         LHLD    SCANPTR2
1494 CDB10B         CALL    ASSIGNVL        ;UPDATE VALUE OF PARAMETER
1497 7E             MOV     A,M
1498 FE29           CPI     ")"
149A C24F14         JNZ     VALFCTDL        ;MORE ARGUMENTS
149D CDAB03         CALL    SCANNXT ;bscan ,
14A0 E3             XTHL
14A1 CDA303         CALL    SCANNXTV        ;bscan (val)
14A4 29             DB      ")"     ;MUST BE END OF PARAMETERS TOO

                ;  EVALUATE EXPRESSION
                ;
                VALFCTNA:
14A5 CDA303         CALL    SCANNXTV        ;bscan (val)
14A8 B5             DB      KEYEQ   ;LOOK FOR EQUALS SIGN
14A9 CD930F         CALL    VALEXPR ;bscan expr     ;EVALUATE FUNCTION
14AC 2B             DCX     H
14AD CDAB03         CALL    SCANNXT ;bscan ,
14B0 C2EF05         JNZ     ERRASN
14B3 E1             POP     H
14B4 226F03         SHLD    SCANPTR1
14B7 CDD903         CALL    TYPECHK
14BA C20000         JNZ     VALFCTRL
14BD CDD60B         CALL    STRGUNIQ
14C0 EB             XCHG
14C1 229303         SHLD    ACCUMLTR
```

```
                ;  RESTORE PARAMETERS
                ;
                VALFCTRL:
14C4 F1             POP     PSW     ;RESTORE VALUES OF PARAMETERS
14C5 D20000         JNC     VALFCTCR
14C8 E1             POP     H
14C9 CDDC03         CALL    TYPECHKA
14CC CA0000         JZ      VALFCTRS
14CF C1             POP     B
14D0 D1             POP     D
14D1 73             MOV     M,E     ;RESTORE NUMERIC VALUE
14D2 23             INX     H
14D3 72             MOV     M,D
14D4 23             INX     H
14D5 71             MOV     M,C
14D6 23             INX     H
14D7 70             MOV     M,B
14D8 C3C414         JMP     VALFCTRL
                VALFCTRS:
14DB D1             POP     D       ;RESTORE STRING VALUE
14DC EB             XCHG
14DD 228F03         SHLD    STRGTMPP        ;DEALLOCATE TEMPORARY
14E0 EB             XCHG
14E1 0603           MVI     B,TYPESTRG
14E3 CD0000         CALL    COPYVALL
14E6 C3C414         JMP     VALFCTRL

                VALFCTCR:
14E9 2A6F03         LHLD    SCANPTR1        ;COERCE RESULT TO CORRECT TYPE
14EC CDDC03         CALL    TYPECHKA
14EF C2E80B         JNZ     COERCEF
14F2 CDFB0B         CALL    CSTRING ;STRING FUNCTION
14F5 E5             PUSH    H
14F6 2A9303         LHLD    ACCUMLTR
14F9 EB             XCHG
14FA CDC711         CALL    STRGRELT
14FD C38011         JMP     STRGALOU

                ERRAID:
1500 1E1A           MVI     E,ERRNID-ERRN
1502 C3F105         JMP     ERRMSG

                SCANFNN:
1505 CDA303         CALL    SCANNXTV        ;bscan (val)
1508 A7             DB      KEYFN
1509 3EAB           MVI     A,SCANPFLD
150B 326703         STA     SCANPFLG
150E 0620           MVI     B,TYPEDEF
1510 C33B12         JMP     VARSCNDF
```

```
                ;
                ;  GENERATE A NEW CURRENT STRING
                ;
                STRNGEN:
1513 CD0000         CALL    STRGALOC        ;GENERATE A NEW STRING,
                STRSTCDS:
1516 216C03         LXI     H,STRGTMPL      ;SET CURRENT STRING DESCRIPTOR
1519 E5             PUSH    H
151A 77             MOV     M,A
151B 23             INX     H
151C 73             MOV     M,E
151D 23             INX     H
151E 72             MOV     M,D
151F E1             POP     H
1520 C9             RET

                ;
                ;  ALLOCATE STORAGE IN STRING SPACE
                ;
                STRGALOC:
1521 B7             ORA     A       ;ALLOCATE SPACE FOR STRING,
1522 C30000         JMP     STRGALAH        ;SIZE IN A
                STRGALAG:
1525 F1             POP     PSW     ;ENTER FOR SECOND TRY
                STRGALAH:
1526 F5             PUSH    PSW
1527 2A8903         LHLD    STCKBASE
152A EB             XCHG
152B 2A8B03         LHLD    STRGFREE
152E 2F             CMA
152F 4F             MOV     C,A
1530 06FF           MVI     B,0FFH
1532 09             DAD     B
1533 23             INX     H
1534 CDC104         CALL    CMHLLTDE
1537 DA0000         JC      STRGALGC
153A 228B03         SHLD    STRGFREE
153D 23             INX     H
153E EB             XCHG            ;RETURNS: DE=STRING ADDRESS
                POPAFRET:
153F F1             POP     PSW
1540 C9             RET

                STRGALGC:
1541 F1             POP     PSW     ;COLLECT GARBAGE IN STRING SPACE
1542 1E85           MVI     E,ERRNOS-ERRN
1544 CAF105         JZ      ERRMSG
1547 BF             CMP     A
1548 F5             PUSH    PSW
1549 012515         LXI     B,STRGALAG      ;THEN TRY ALLOCATION
154C C5             PUSH    B
```

```
                ;
                ;  COLLECT GARBAGE IN STRING SPACE
                ;
                STRGGBCL:
154D 2AB003         LHLD    STRGBASE        ;MAKE ALL STRINGS UNSAFE
                STRGGBLP:
1550 228B03         SHLD    STRGFREE        ;FIND HIGHEST UNSAFE STRING
1553 210000         LXI     H,0
1556 E5             PUSH    H
1557 2A8903         LHLD    STCKBASE
155A E5             PUSH    H
155B 2A8D03         LHLD    STRGBASE        ;SCAN TEMPORARIES,
155E 23             INX     H
                STRGGBTL:
155F EB             XCHG
1560 2ABF03         LHLD    STRGTMPP
1563 EB             XCHG
1564 CDC104         CALL    CMHLLTOE
1567 015F15         LXI     B,STRGGBTL
156A C20000         JNZ     STRGGBHI
156D 2AB303         LHLD    VARTABLE        ;SCAN REGULAR VARIABLES,
                STRGGBVR:
1570 EB             XCHG
1571 2A8503         LHLD    MATTABLE
1574 EB             XCHG
1575 CDC104         CALL    CMHLLTDE
157B CA0000         JZ      STRGGNAV
157B 7E             MOV     A,M
157C 23             INX     H
157D E60F           ANI     00FH
157F D603           SUI     TYPESTRG
15B1 5F             MOV     E,A
15B2 9F             SBB     A
15B3 57             MOV     D,A
15B4 7E             MOV     A,M
15B5 23             INX     H
15B6 E6B0           ANI     0B0H    ;DEFINITIONS ARE STRINGS
15BB 19             DAD     D
15B9 B3             ORA     E
15BA CD0000         CALL    STRGGBHV
15BD C37015         JMP     STRGGBVR

                STRGGBAL:
1590 C1             POP     B
                STRGGNAV:
1591 EB             XCHG            ;SCAN ARRAY VARIABLES
1592 2AB703         LHLD    FREELIMT
1595 EB             XCHG
1596 CDC104         CALL    CMHLLTDE
1599 CA0000         JZ      STRGGBMV
159C CD0000         CALL    LDRGMM
159F 7B             MOV     A,E
15A0 E5             PUSH    H
15A1 09             DAD     B
15A2 E60F           ANI     00FH
```

```
15A4 FE03       CPI     TYPESTRG
15A6 C29015     JNZ     STRGGBAL
15A9 226F03     SHLD    SCANPTR1
15AC E1         POP     H
15AD 4E         MOV     C,M
15AE 0600       MVI     B,000H
15B0 09         DAD     B
15B1 09         DAD     B
15B2 23         INX     H
            STRGGBAS:
15B3 EB         XCHG            ;LOOK THROUGH ENTIRE ARRAY
15B4 2A6F03     LHLD    SCANPTR1
15B7 EB         XCHG
15B8 CDC104     CALL    CMHLLTDE
15BB CA9115     JZ      STRGGNAV
15BE 01B315     LXI     B,STRGGBAS
            STRGGBHI:
15C1 C5         PUSH    B       ;COMPARE THIS STRING ADDR TO MAX
15C2 AF         XRA     A
            STRGGBHV:
15C3 4E         MOV     C,M     ;LOAD STRING DESCRIPTOR
15C4 23         INX     H
15C5 5E         MOV     E,M
15C6 23         INX     H
15C7 56         MOV     D,M
15C8 23         INX     H
15C9 C0         RNZ             ;NOT A STRING VARIABLE
15CA 79         MOV     A,C
15CB B7         ORA     A       ;CHECK FOR ZERO LENGTH
15CC C8         RZ
15CD 44         MOV     B,H     ;ALREADY SAFE?
15CE 4D         MOV     C,L
15CF 2A8B03     LHLD    STRGFREE
15D2 CDC104     CALL    CMHLLTDE
15D5 60         MOV     H,B
15D6 69         MOV     L,C
15D7 D8         RC
15D8 E1         POP     H       ;COMPARE WITH HIGHEST UNSAFE
15D9 E3         XTHL
15DA CDC104     CALL    CMHLLTDE
15DD E3         XTHL
15DE E5         PUSH    H
15DF 60         MOV     H,B
15E0 69         MOV     L,C
15E1 D0         RNC
15E2 C1         POP     B       ;SAVE NEW HIGHEST UNSAFE ADDR
15E3 F1         POP     PSW
15E4 F1         POP     PSW
15E5 E5         PUSH    H
15E6 D5         PUSH    D
15E7 C5         PUSH    B
15E8 C9         RET
```

```
            STRGGBMV:
15E9 D1         POP     D       ;MAKE HIGHEST UNSAFE SAFE
15EA E1         POP     H
15EB 7D         MOV     A,L
15EC B4         ORA     H
15ED C8         RZ              ;ANY UNSAFE?
15EE 2B         DCX     H       ;LOAD DESCRIPTOR
15EF 46         MOV     B,M
15F0 2B         DCX     H
15F1 4E         MOV     C,M
15F2 E5         PUSH    H
15F3 2B         DCX     H
15F4 6E         MOV     L,M     ;FIND END OF STRING
15F5 2600       MVI     H,000H
15F7 09         DAD     B
15F8 50         MOV     D,B
15F9 59         MOV     E,C
15FA 2B         DCX     H
15FB 44         MOV     B,H
15FC 4D         MOV     C,L
15FD 2A8B03     LHLD    STRGFREE        ;COPY IT TO END OF SAFE AREA
1600 CDCA04     CALL    COPYTEXT
1603 E1         POP     H
1604 71         MOV     M,C
1605 23         INX     H
1606 70         MOV     M,B
1607 60         MOV     H,B
1608 69         MOV     L,C
1609 2B         DCX     H
160A C35015     JMP     STRGGBLP        ;EXTEND SAFE AREA
```

```
                ;
                ;  VARIOUS NUMERIC/STRING CONVERSION FUNCTIONS
                ;

                ;
                ;  FIND LENGTH OF STRING
                ;
                LENFCT:
160D 01CE0D         LXI     B,FLOATA        ;LEN FUNCTION
1610 C5             PUSH    B
                LENFCTC:
1611 CDFB0B         CALL    CSTRING
1614 CDA911         CALL    STRGRELA
1617 3E04           MVI     A,TYPESING
1619 326B03         STA     TYPEFLG
161C 7E             MOV     A,M
161D B7             ORA     A
161E 23             INX     H
161F C9             RET

                ;
                ;  CONVERT CHARACTER TO BYTE
                ;
                ASCFCT:
1620 CD1116         CALL    LENFCTC ;ASC FUNCTION
1623 CA230C         JZ      ERRAFC
1626 4E             MOV     C,M     ;FETCH ADDRESS
1627 23             INX     H
162B 46             MOV     B,M
1629 0A             LDAX    B       ;THEN THE FIRST CHARACTER
162A C3CE0D         JMP     FLOATA

                ;
                ;  CONVERT BYTE TO CHARACTER
                ;
                CHRFCT:
162D 3E01           MVI     A,1     ;CHR$ FUNCTION
162F CD1315         CALL    STRNGEN
1632 CD2F0C         CALL    CBYTE
1635 2A6D03         LHLD    STRGTMPA
1638 73             MOV     M,E
                VALRETST:
1639 C1             POP     B       ;STRING FUNCTION, REMOVE CSINGLE
163A C37D11         JMP     STRGALOT
```

```
                ;
                ;  DECOOE NUMBER FROM STRING
                ;
                VALFCT:
1630 CO1116         CALL    LENFCTC ;VAL FUNCTION
1640 CA0000         JZ      ZEROAC
1643 5F             MOV     E,A
1644 1600           MVI     O,O
1646 4E             MOV     C,M
1647 23             INX     H
1648 46             MOV     B,M
1649 C5             PUSH    B
164A 60             MOV     H,B
164B 69             MOV     L,C
164C 19             DAO     O
164O 46             MOV     B,M
164E 72             MOV     M,D
164F E3             XTHL
1650 C5             PUSH    B
1651 7E             MOV     A,M
1652 COOOOO         CALL    DECODE
1655 C1             POP     B
1656 E1             POP     H
1657 70             MOV     M,B
1658 C9             RET

                ;
                ;  ENCOOE NUMBER IN STRING
                ;
                STRFCT:
1659 CDF40B         CALL    CSINGLE ;STR$ FUNCTION
165C CD4911         CALL    VALSTRGN        ;CREATE STRING FROM NUMBER
165F CDA911         CALL    STRGRELA
1662 013916         LXI     B,VALRETST
1665 C5             PUSH    B
1666 EB             XCHG
                STRGSTOR:
1667 EB             XCHG
1668 7E             MOV     A,M     ;STORE STRING INTO STRING SPACE,
1669 E5             PUSH    H       ;LEAVE DESCRIPTOR IN STRGTMP
166A CD2115         CALL    STRGALOC
166D E1             POP     H
166E CDOOOO         CALL    LDICBMM ;LOAD BUFFER ADDRESS
1671 CD1615         CALL    STRSTCDS
1674 E5             PUSH    H
1675 6F             MOV     L,A
1676 CD1612         CALL    COPYSTRG
                POPDERET:
1679 D1             POP     O
167A C9             RET
```

```
                ;
                ;  CONVERT HEX STRING TO NUMBER
                ;
                HXVFCT:
167B C01116         CALL    LENFCTC ;DO INITIAL PROCESSING
167E CA0000         JZ      ZEROAC
1681 5F             MOV     E,A
1682 4E             MOV     C,M
1683 23             INX     H
1684 46             MOV     B,M
1685 210000         LXI     H,0     ;INITIAL OUTPUT TO ZERO
                HXVFCTL:
1688 0A             LOAX    B       ;FETCH CHARACTER
1689 03             INX     B
168A FE3A           CPI     ":"     ;   VERIFY THAT IT"S  HEX
168C D40000         CNC     HXVFCTCH
168F D2230C         JNC     ERRAFC  ;IF NOT, COMPLAIN
1692 0630           SUI     "0"
1694 DA230C         JC      ERRAFC  ;MUST BE AT LEAST ZERO
1697 29             OAO     H
1698 29             DAD     H       ;INCORPORATE NEW DIGIT
1699 29             OAO     H
169A 29             OAD     H
169B B5             ORA     L
169C 6F             MOV     L,A
169D 1D             DCR     E       ;COUNT OIGITS
169E C28816         JNZ     HXVFCTL
                FLOATHL:
16A1 7C             MOV     A,H     ;CONVERT INTEGER IN HL TO FLOAT
16A2 45             MOV     B,L
16A3 C30000         JMP     FLOATAB

                HXVFCTCH:
16A6 CDBC03         CALL    ALPHACHA        ;CONVERT ANY ALPHA TO UPPER
16A9 00             RNC
16AA D607           SUI     'A-'9-1 ; MOVE ALPHA TO AFTER OIGITS
16AC FE40           CPI     '0+16   ;SET FLAGS CORRECTLY
16AE C9             RET
```

```
                ;
                ;  CONVERT BYTE TO TWO HEX CHARACTERS
                ;
                HEXFCT:
16AF 3E02           MVI     A,2     ;ALLOCATE OUTPUT STRING
16B1 CD1315         CALL    STRNGEN
16B4 3A9603         LOA     FLACCEXP
16B7 CD0000         CALL    FIXAC   ;GET INPUT BYTE
16BA 213916         LXI     H,VALRETST
16BD E5             PUSH    H
16BE 2A6003         LHLO    STRGTMPA
16C1 C00000         CALL    HEXFCTL
                HEXFCTL:
16C4 7B             MOV     A,E     ;CONVERT ONE OIGIT
16C5 07             RLC
16C6 07             RLC
16C7 07             RLC
16C8 07             RLC
16C9 5F             MOV     E,A
16CA E60F           ANI     00FH
16CC FE0A           CPI     10
16CE 3F             CMC                     ;CONVERT TO CHARACTER FORM
16CF CE30           ACI     "0"
16D1 27             DAA
16D2 77             MOV     M,A
16D3 23             INX     H
16O4 C9             RET

                ;
                ;  TRANSLATE STRING TO UPPER CASE
                ;
                UPRFCT:
16D5 CDFB0B         CALL    CSTRING
16D8 2A9303         LHLD    ACCUMLTR        ;GET LENGTH OF OPERAND
16DB E5             PUSH    H
16DC 7E             MOV     A,M
16DD CD1315         CALL    STRNGEN ;ALLOCATE OUTPUT STRING
16E0 D1             POP     D
16E1 CDAD11         CALL    STRGRELO        ;RELEASE INPUT STRING
16E4 CD0000         CALL    LDDCBMM
16E7 2A6D03         LHLD    STRGTMPA
16EA 14             INR     O
                UPRFCTL:
16EB 15             DCR     O       ;TRANSLATE WHILE COPYING
16EC CA3916         JZ      VALRETST        ;OONE
16EF 0A             LDAX    B
16F0 CDBC03         CALL    ALPHACHA        ;CONVERT LOWER TO UPPER
16F3 77             MOV     M,A
16F4 03             INX     B
16F5 23             INX     H
16F6 C3EB16         JMP     UPRFCTL
```

```
                ;
                ;  SUBSTRING FUNCTIONS
                ;
                LFTFCT:
16F9 CD0000         CALL    LEFRIGAR        ;LEFT$ FUNCTION
16FC AF             XRA     A       ;LEFT(X,N)=MID(X,1,N)
                LEFRIGMR:
16FD E3             XTHL
16FE 4F             MOV     C,A     ;C=START-1, B=LEN
                LEFRIGMD:
16FF E5             PUSH    H       ;RESOLVE DESIRED LEN WITH STRING
1700 7E             MOV     A,M
1701 B8             CMP     B
1702 DA0000         JC      LEFRIGMC
1705 78             MOV     A,B
1706 C30000         JMP     LEFRIGMB

                LEFRIGAR:
1709 EB             XCHG            ;INITIAL COMMON PROCESSING
170A CD280C         CALL    VALBYTE2        ;FOR LEFT$, RIGHT$
170D 43             MOV     B,E
170E CDA303         CALL    SCANNXTV        ;bscan (val)
1711 29             DB      ")"
1712 C9             RET

                LEFRIGMC:
1713 0E00           MVI     C,0
                LEFRIGMB:
1715 C5             PUSH    B
1716 CD2115         CALL    STRGALOC        ;ALLOCATE ANSWER STRING
1719 C1             POP     B
171A E1             POP     H
171B E5             PUSH    H
171C 23             INX     H
171D 46             MOV     B,M     ;COMPUTE ADDRESSES FOR COPY
171E 23             INX     H
171F 66             MOV     H,M
1720 68             MOV     L,B     ;(from HL,MB)
1721 0600           MVI     B,0
1723 09             DAD     B
1724 44             MOV     B,H
1725 4D             MOV     C,L
1726 CD1615         CALL    STRSTCDS
1729 6F             MOV     L,A
172A CD1612         CALL    COPYSTRG        ;COPY
172D D1             POP     D
172E CDAD11         CALL    STRGRELD
1731 C37D11         JMP     STRGALOT
```

```
            RIGFCT:
1734 CD0917     CALL    LEFRIGAR        ;RIGHT$ FUNCTION
1737 D1         POP     D
1738 D5         PUSH    D
1739 1A         LDAX    D
173A 90         SUB     B       ;RIGHT(X,N)=MID(X,LEN(X)-N+1,N)
173B C3FD16     JMP     LEFRIGMR

            MIDFCT:
173E EB         XCHG            ;MID$ FUNCTION
173F CD280C     CALL    VALBYTE2        ;SCAN STARTING POSITION
1742 43         MOV     B,E
1743 B7         ORA     A       ;NON-ZERO STARTING POSITION?
1744 CA230C     JZ      ERRAFC
1747 C5         PUSH    B
1748 1EFF       MVI     E,0FFH
174A 7E         MOV     A,M
174B FE29       CPI     ")"
174D C4280C     CNZ     VALBYTE2        ;SCAN OPTIONAL THIRD ARGUMENT
1750 CDA303     CALL    SCANNXTV        ;bscan (val)
1753 29         DB      ")"
1754 F1         POP     PSW     ;COMPUTE STARTING BYTE AND LENGTH
1755 E3         XTHL
1756 01FF16     LXI     B,LEFRIGMD
1759 C5         PUSH    B
175A 3D         DCR     A
175B BE         CMP     M
175C 0600       MVI     B,0     ;START > LENI  =>  LENO=0
175E D0         RNC
175F 4F         MOV     C,A
1760 7E         MOV     A,M
1761 91         SUB     C
1762 BB         CMP     E
1763 47         MOV     B,A
1764 D8         RC              ;LENO = MIN(LENI-START,LENR)
1765 43         MOV     B,E
1766 C9         RET
```

```
                ;
                ;  INDEX OF STRING FUNCTION
                ;
                INSFCT:
1767 E8             XCHG
1768 CDA303         CALL    SCANNXTV        ;bscan (val)
176B 2C             D8      ","
176C CDAA10         CALL    VALPARN2        ;SCAN SECOND ARGUMENT
176F E3             XTHL            ;SHUFFLE RETURN STACK
1770 017008         LXI     8,POPHLRET
1773 C5             PUSH    8
1774 E5             PUSH    H
1775 CD1116         CALL    LENFCTC ;PROCESS  SECOND STRING
1778 E3             XTHL
1779 F5             PUSH    PSW
177A CA0000         JZ      INSFCTXT
177D CDAC11         CALL    STRGRELH        ;WORK ON FIRST STRING
1780 7E             MOV     A,M
1781 C1             POP     8
1782 D1             POP     D
1783 90             SUB     B       ;COMPARE LENGTHS
1784 DA0000         JC      ZEROAC  ;TEST IS LONGER, NO MATCHES
1787 3C             INR     A
1788 4F             MOV     C,A     ;SAVE NUM8ER OF ATTEMPTS
1789 C5             PUSH    B
178A CD0000         CALL    LDICBMM ;GET ADDRESS OF TARGET
178D EB             XCHG
178E 5E             MOV     E,M     ;GET ADDRESS OF MATCHER
178F 23             INX     H
1790 56             MOV     D,M
1791 EB             XCHG
1792 D1             POP     D       ;RECOVER LENGTH, COUNTER
1793 3E01           MVI     A,1
                INSFCTSL:
1795 D5             PUSH    D       ;SAVE LENGTH, COUNTER
1796 F5             PUSH    PSW     ;SAVE POSITION
1797 C5             PUSH    B       ;SAVE ADDRESSES
1798 E5             PUSH    H
1799 5A             MOV     E,D
179A CD5510         CALL    RELOPRSL        ;COMPARE STRINGS
179D E1             POP     H       ;RECOVER ADDRESSES
179E C1             POP     8
                INSFCTXT:
179F D1             POP     D
17A0 7A             MOV     A,D     ;RECOVER POSITION
17A1 D1             POP     D       ;AND LENGTH, COUNTER
17A2 CACE0D         JZ      FLOATA  ;ANSWER FOUND, GIVE IT BACK
17A5 3C             INR     A       ;INCREMENT POSITION
17A6 03             INX     8
17A7 1D             DCR     E       ;COUNT ATTEMPTS
17A8 C29517         JNZ     INSFCTSL        ;KEEP TRYING
17AB C30000         JMP     ZEROAC  ;OR NOMATCH
```

```
                ;
                ;  FUNCTION RETURNING AMOUNT OF REMAINING FREE SPACE
                ;
                FREFCT:
17AE 2A8503         LHLD    MATTA8LE        ;FRE FUNCTION
1781 E8             XCHG
17B2 210000         LXI     H,0
1785 39             DAD     SP
1786 CDD903         CALL    TYPECHK
17B9 C20000         JNZ     FREFCTNS
17BC CDA911         CALL    STRGRELA        ;RETURN 8YTES OF FREE STRNG SPACE
178F CD4D15         CALL    STRGG8CL
17C2 2A8903         LHLD    STCKBASE
17C5 EB             XCHG
17C6 2A8803         LHLD    STRGFREE
                FREFCTNS:
17C9 7D             MOV     A,L
17CA 93             SU8     E
17CB 47             MOV     B,A
17CC 7C             MOV     A,H
17CD 9A             SB8     D
                FLOATAB:
17CE 50             MOV     D,8
17CF 1E00           MVI     E,000H
17D1 216B03         LXI     H,TYPEFLG
17D4 3604           MVI     M,TYPESING
17D6 0690           MVI     B,090H
17D8 C30000         JMP     FLOATINT

                ;
                ;  MEMORY DIDDLING FACILITIES
                ;
                MEMFCT:
17DB CDD903         CALL    TYPECHK ;MEM FUNCTION
17DE CA0000         JZ      MEMFCTC
17E1 CD100C         CALL    CINTEGER
17E4 1A             LDAX    D
17E5 C3CE0D         JMP     FLOATA
                MEMFCTC:
17E8 CD1116         CALL    LENFCTC ;RELEASE ARGUMENT
17EB 2AB103         LHLD    PROGBASE
17EE CAA116         JZ      FLOATHL ;ZERO LENGTH STRING=PROGBASE
17F1 2A9103         LHLD    STRGTLIM
17F4 C3A116         JMP     FLOATHL ;OTHERWISE=UPPER LIMIT

                MEMSTM:
17F7 CDAB03         CALL    SCANNXT ;bscan +        ;MEM STATEMENT
17FA CDA610         CALL    VALPARNS
17FD CD100C         CALL    CINTEGER
1B00 D5             PUSH    D
1801 CDA303         CALL    SCANNXTV        ;bscan (val)
1804 B5             D8      KEYEQ
1B05 CD2C0C         CALL    VALBYTE
1808 D1             POP     D
1809 12             STAX    D
```

```
180A C9         RET
```

```
                ;
                ;  DIRECT I/O FACILITIES
                ;

                PORFCT:
180B CD2F0C         CALL    CBYTE   ;PORT FUNCTION
180E 16DB           MVI     D,OPCINP
1810 CD0000         CALL    INOTGEN
1813 CD9B03         CALL    INOTINS
1816 C3CE0D         JMP     FLOATA

                PORSTM:
1819 CDAB03         CALL    SCANNXT ;bscan +        ;PORT STATEMENT
181C CDA610         CALL    VALPARNS
181F CD2F0C         CALL    CBYTE
1822 D5             PUSH    D
1823 CDA303         CALL    SCANNXTV        ;bscan (val)
1826 B5             DB      KEYEQ
1827 CD2C0C         CALL    VALBYTE
182A D1             POP     D
182B 16D3           MVI     D,OPCOUT
182D CD0000         CALL    INOTGEN
1830 C39B03         JMP     INOTINS

                WAISTM:
1833 CD2C0C         CALL    VALBYTE ;WAIT STATEMENT
1836 D5             PUSH    D
1837 CD280C         CALL    VALBYTE2
183A F5             PUSH    PSW
183B 1E00           MVI     E,0
183D C4280C         CNZ     VALBYTE2
1840 C1             POP     B
1841 4B             MOV     C,E
1842 D1             POP     D
1843 16DB           MVI     D,OPCINP
1845 CD0000         CALL    INOTGEN
                WAISTMIN:
1848 CD4A00         CALL    SYSWAIT ;DO A SYSTEM WAIT
184B CD9B03         CALL    INOTINS ;THEN CHECK DEVICE
184E A9             XRA     C
184F A0             ANA     B
1850 CA4818         JZ      WAISTMIN
1853 C9             RET

                INOTGEN:
1854 E5             PUSH    H       ;GENERATE INPUT/OUTPUT FOLLOWED
1855 219B03         LXI     H,INOTINS       ;BY RETURN
1858 72             MOV     M,D
1859 23             INX     H
185A 73             MOV     M,E
185B 23             INX     H
185C 36C9           MVI     M,OPCRET
185E E1             POP     H
185F C9             RET
```

```
                ;
                ;  CSAVE/CLOAD PROCESSORS
                ;    save filename - save on diskette
                ;    load filename - get from diskette
                ;
                ;  load and save programs from the disk
                ;
1860 8400       d14base     equ     0b400h
1860 8000       fsprom      equ     0b000h
1860 8398       bootstart   equ     fsprom+39bh     ;load image files
1860 B8E0       directorylookup     equ     d14base+4e0h    ;find filename
1860 8796       opens               equ     d14base+396h    ;open stream
1860 B7DC       puts                equ     d14base+3dch    ;put char
1860 882D       closes              equ     d14base+42dh    ;close stream
                ;
                cldstm:
1860 CD0000         call    setfilename     ;parse filename
1863 CDE088         call    directorylookup
1866 D20000         jnc     namenotfound
1869 CD9BB3         call    bootstart
186C CD0000         call    checkprogram
186F CDFF04         call    newload         ;reset program pointers
1872 C31906         jmp     cmndstrt
                namenotfound:
1875 1EE1           mvi     e,errnfi-errn   ;file not saved
1877 C3F105         jmp     errmsg


                csvstm:
187A CD0000         call    setfilename
187D 0602           mvi     b,2             ;write enable
187F CD96B7         call    opens           ;open stream (only one in D14)
1882 D20000         jnc     cannotopen      ; -disk full or other bad stuff
1885 CD0000         call    checkprogram
1888 E5             push    h               ;save end pointer
1889 2A8103         lhld    progbase                ;first address
188C 4D             mov     c,l
188D CDDCB7         call    puts
1890 4C             mov     c,h
1891 CDDCB7         call    puts
1894 0E00           mvi     c,0             ;start address = 0 for no start
1896 CDDCB7         call    puts
1899 CDDC87         call    puts
189C D1             pop     d               ;de has end address+1
                saveloop:
189D 4E             mov     c,m             ;get char
189E 23             inx     h
189F CDDC87         call    puts            ;and send to file
18A2 7C             mov     a,h             ;is this the end?
18A3 BA             cmp     d
18A4 C29D18         jnz     saveloop
18A7 7D             mov     a,l
18AB BB             cmp     e
18A9 C29D18         jnz     saveloop
18AC CD2DB8         call    closes          ;yes
```

```
18AF C31906     jmp     cmndstrt
            cannotopen:
18B2 1E09       mvi     e,errnsl-errn
18B4 C3F105     jmp     errmsg


            ;  setfilename
            ;    returns hl set to a filename string
            ;
            setfilename:
18B7 110000     lxi     d,filename+1
18BA 0600       mvi     b,0
            sfnloop:
18BC 7E         mov     a,m             ;look at char
18BD FE00       cpi     0
18BF CA0000     jz      sfndone
18C2 FE20       cpi     " "
18C4 CA0000     jz      sfndone
18C7 04         inr     b               ;up count
18C8 23         inx     h
18C9 12         stax    d
18CA 13         inx     d
18CB C3BC18     jmp     sfnloop
            sfndone:
18CE 210000     lxi     h,filename
18D1 AF         xra     a               ;is the name non zero
18D2 B0         ora     b
18D3 CAEF05     jz      errasn          ;yes
18D6 77         mov     m,a             ;store count
18D7 C9         ret


            ;  checkprogram
            ;    walk over the program looking for the end
            ;    return last byte+1 in hl
            ;
            checkprogram:
18D8 2A8103     lhld    progbase        ;starts here
            cprogloop:
18DB 7E         mov     a,m             ;pick up line length
18DC 23         inx     h
18DD B6         ora     m
18DE 23         inx     h
18DF CA0000     jz      cprogok         ;if zero then all done
18E2 23         inx     h
18E3 23         inx     h               ;skip line number
            cprogloop2:
18E4 7E         mov     a,m
18E5 B7         ora     a
18E6 23         inx     h
18E7 CADB18     jz      cprogloop       ;zero at the end of the line
18EA C3E418     jmp     cprogloop2
            cprogok:
18ED C9         ret
1929 00     filename:   ds      60
```

```
                ;
                ;  LOGICAL OPERATORS
                ;

                ORNOPR:
192A B7             ORA     A       ;OR OPERATOR
192B C30000         JMP     LOGOPRIC
                ANOOPR:
192E AF             XRA     A       ;AND OPERATOR
                LOGOPRIC:
192F F5             PUSH    PSW
1930 CDF40B         CALL    CSINGLE
1933 CD100C         CALL    CINTEGER
1936 F1             POP     PSW
1937 EB             XCHG
1938 C1             POP     B
1939 E3             XTHL
193A EB             XCHG
193B C00000         CALL    LOACRG
193E F5             PUSH    PSW
193F C0100C         CALL    CINTEGER
1942 F1             POP     PSW
1943 C1             POP     B
1944 79             MOV     A,C
1945 C20000         JNZ     ORNOPRFN
1948 A3             ANA     E
1949 4F             MOV     C,A
194A 78             MOV     A,B
194B A2             ANA     D
194C C30000         JMP     LOGOPRXT        ;RETURN FROM ANO

                ORNOPRFN:
194F B3             ORA     E
1950 4F             MOV     C,A
1951 78             MOV     A,B
1952 B2             ORA     D
                LOGOPRXT:
1953 41             MOV     B,C
1954 C3CE17         JMP     FLOATAB ;RETURN FROM OR

                VALUNOT:
1957 165A           MVI     D,PREDNOT       ;EVALUATE UNARY NOT
1959 CD960F         CALL    VALEXPRL
195C CDF40B         CALL    CSINGLE
195F CD100C         CALL    CINTEGER
1962 7B             MOV     A,E
1963 2F             CMA
1964 4F             MOV     C,A
1965 7A             MOV     A,D
1966 2F             CMA
1967 CD5319         CALL    LOGOPRXT
196A C1             POP     B
196B C3A20F         JMP     VALEXPRC
```

```
                ;
                ;  MOD, MAXIMUM, MINIMUM OPERATORS
                ;

                MODOPR:
196E C1             POP     B       ;MODULO FUNCTION
196F D1             POP     D       ;X MOD Y =
1970 D5             PUSH    D       ;X - INT(X/Y) * Y
1971 C5             PUSH    B
1972 2A9303         LHLD    ACCUMLTR
1975 E5             PUSH    H
1976 2A9503         LHLD    FLACCMSB
1979 E5             PUSH    H
197A CD0000         CALL    FLDIV
197D CD0000         CALL    INTFCT
1980 C1             POP     B
1981 D1             POP     D
1982 CD0000         CALL    FLMUL
1985 C30000         JMP     SUBOPR

                MAXOPR:
1988 C1             POP     B
1989 D1             POP     D
198A CD0000         CALL    FLCMP   ;COMPARE OPERANDS
198D C8             RZ              ;NO DIFFERENCE
198E DA0000         JC      LDACRG  ;REGISTERS LARGER
1991 C30000         JMP     LDRGAC  ;ACCUMULATOR LARGER

                MINOPR:
1994 C1             POP     B
1995 D1             POP     D
1996 CD0000         CALL    FLCMP   ;COMPARE OPERANDS
1999 C8             RZ              ;NO DIFFERENCE
199A D20000         JNC     LDACRG  ;REGISTERS SMALLER
199D C30000         JMP     LDRGAC  ;ACCUMULATOR SMALLER
```

```
                ;
                ;  FLOATING POINT ADD/SUBTRACT ROUTINES
                ;

                FLADDHLF:
19A0 210000         LXI     H,FLHALF
                FLADDM:
19A3 CD0000         CALL    LDRGMM
19A6 C30000         JMP     FLADD

                FLMMMAC:
19A9 CD0000         CALL    LDRGMM  ;COMPUTE MM-AC
19AC C30000         JMP     FLSUB

                SUBOPR:
19AF C1             POP     B
19B0 D1             POP     D
                FLSUB:
19B1 CD0000         CALL    CMACCS  ;SUBTRACT ACC FROM REGISTERS
                FLADD:
19B4 78             MOV     A,B     ;ADD ACCUMULATOR TO REGISTERS
19B5 B7             ORA     A
19B6 CB             RZ
19B7 3A9603         LDA     FLACCEXP
19BA B7             ORA     A
19BB CA0000         JZ      LDACRG
19BE 90             SUB     B
19BF D20000         JNC     FLADDMGC
19C2 2F             CMA                     ;NEED LARGER IN AC, INTERCHANGE
19C3 3C             INR     A
19C4 EB             XCHG
19C5 CD0000         CALL    PUSHAC
19C8 EB             XCHG
19C9 CD0000         CALL    LDACRG
19CC C1             POP     B
19CD D1             POP     D
                FLADDMGC:
19CE FE19           CPI     019H    ;ARE MAGNITUDES ARE COMMENSURATE?
19D0 D0             RNC
19D1 F5             PUSH    PSW
19D2 CD0000         CALL    SIGNIFY
19D5 67             MOV     H,A
19D6 F1             POP     PSW
19D7 CD0000         CALL    SHIFTR0
19DA B4             ORA     H
19DB 219303         LXI     H,ACCUMLTR
19DE F20000         JP      FLADDIFF
19E1 CD0000         CALL    ADDM2CDE
19E4 D20000         JNC     FLROUND
19E7 23             INX     H
19E8 34             INR     M
19E9 CA0000         JZ      ERRAOV
19EC 2E01           MVI     L,001H
19EE CD0000         CALL    SHIFTRLB
19F1 C30000         JMP     FLROUND
```

```
                FLADDIFF:
19F4 AF             XRA     A       ;FIND DIFFERENCE
19F5 90             SUB     B
19F6 47             MOV     B,A
19F7 7E             MOV     A,M
19FB 9B             SBB     E
19F9 5F             MOV     E,A
19FA 23             INX     H
19FB 7E             MOV     A,M
19FC 9A             SBB     D
19FD 57             MOV     D,A
19FE 23             INX     H
19FF 7E             MOV     A,M
1A00 99             SBB     C
1A01 4F             MOV     C,A
                NORMALZI:
1A02 DC0000         CC      CMREGS
                NORMALIZ:
1A05 68             MOV     L,B     ;NORMALIZE REGISTERS
1A06 63             MOV     H,E
1A07 AF             XRA     A
                NORMAL8:
1A0B 47             MOV     B,A     ;NORMALIZE BY BYTES
1A09 79             MOV     A,C
1A0A B7             ORA     A
1A0B C20000         JNZ     NORMAL1
1A0E 4A             MOV     C,D
1A0F 54             MOV     D,H
1A10 65             MOV     H,L
1A11 6F             MOV     L,A
1A12 78             MOV     A,B
1A13 D60B           SUI     008H
1A15 FEE0           CPI     0E0H
1A17 C20B1A         JNZ     NORMAL8
                ZEROAC:
1A1A AF             XRA     A       ;ZERO ACCUMULATOR
                LDACCE:
1A1B 329603         STA     FLACCEXP
1A1E C9             RET

                NORMAL1L:
1A1F 05             DCR     B       ;NORMALIZE BY BITS
1A20 29             DAD     H
1A21 7A             MOV     A,D
1A22 17             RAL
1A23 57             MOV     D,A
1A24 79             MOV     A,C
1A25 8F             ADC     A
1A26 4F             MOV     C,A
                NORMAL1:
1A27 F21F1A         JP      NORMAL1L
1A2A 78             MOV     A,B
1A2B 5C             MOV     E,H
1A2C 45             MOV     B,L
1A2D B7             ORA     A
```

```
1A2E CA0000     JZ      FLROUND
1A31 219603     LXI     H,FLACCEXP
1A34 86         ADD     M
1A35 77         MOV     M,A
1A36 D21A1A     JNC     ZEROAC
1A39 C8         RZ
             FLROUND:
1A3A 78         MOV     A,B     ;ROUND RESULT
             FLROUNDV:
1A3B 219603     LXI     H,FLACCEXP
1A3E B7         ORA     A
1A3F FC0000     CM      INCCDE
1A42 46         MOV     B,M
1A43 23         INX     H
1A44 7E         MOV     A,M
1A45 E680       ANI     080H
1A47 A9         XRA     C
1A48 4F         MOV     C,A
1A49 C30000     JMP     LDACRG

             INCCDE:
1A4C 1C         INR     E       ;INCREMENT CDE
1A4D C0         RNZ
1A4E 14         INR     D
1A4F C0         RNZ
1A50 0C         INR     C
1A51 C0         RNZ
1A52 0E80       MVI     C,080H
1A54 34         INR     M
1A55 C0         RNZ
             ERRAOV:
1A56 1E6D       MVI     E,ERRNOV-ERRN
1A58 C3F105     JMP     ERRMSG

             ADDM2CDE:
1A5B 7E         MOV     A,M     ;ADD MEMORY TO CDE
1A5C 83         ADD     E
1A5D 5F         MOV     E,A
1A5E 23         INX     H
1A5F 7E         MOV     A,M
1A60 8A         ADC     D
1A61 57         MOV     D,A
1A62 23         INX     H
1A63 7E         MOV     A,M
1A64 89         ADC     C
1A65 4F         MOV     C,A
1A66 C9         RET
```

```
            CMREGS:
1A67 219703     LXI     H,FLACCSSV      ;COMPLEMENT SAVED SIGN, CDEB
1A6A 7E         MOV     A,M
1A6B 2F         CMA
1A6C 77         MOV     M,A
1A6D AF         XRA     A
1A6E 6F         MOV     L,A
1A6F 90         SUB     B
1A70 47         MOV     B,A
1A71 7D         MOV     A,L
1A72 9B         SBB     E
1A73 5F         MOV     E,A
1A74 7D         MOV     A,L
1A75 9A         SBB     D
1A76 57         MOV     D,A
1A77 7D         MOV     A,L
1A78 99         SBB     C
1A79 4F         MOV     C,A
1A7A C9         RET

            SHIFTR0:
1A7B 0600       MVI     B,000H
            SHIFTR:
1A7D D608       SUI     008H    ;SHIFT CDEB RIGHT BY A BITS
1A7F DA0000     JC      SHIFTRB
1A82 43         MOV     B,E
1AB3 5A         MOV     E,D
1AB4 51         MOV     D,C
1A85 0E00       MVI     C,000H
1AB7 C37D1A     JMP     SHIFTR
            SHIFTRB:
1A8A C609       ADI     009H
1ABC 6F         MOV     L,A
            SHIFTRBL:
1ABD AF         XRA     A
1A8E 2D         DCR     L
1ABF CB         RZ
1A90 79         MOV     A,C
            SHIFTRLB:
1A91 1F         RAR
1A92 4F         MOV     C,A
1A93 7A         MOV     A,D
1A94 1F         RAR
1A95 57         MOV     D,A
1A96 7B         MDV     A,E
1A97 1F         RAR
1A9B 5F         MOV     E,A
1A99 7B         MOV     A,B
1A9A 1F         RAR
1A9B 47         MOV     B,A
1A9C C38D1A     JMP     SHIFTRBL
```

```
                ;
                ;  FLOATING POINT MULTIPLY ROUTINE
                ;

                MULOPR:
1A9F C1             POP     B
1AA0 D1             POP     D
                FLMUL:
1AA1 CD0000         CALL    SIGNACC ;MULTIPLY REGISTERS BY ACC
1AA4 C8             RZ
1AA5 2E00           MVI     L,000H
1AA7 CD0000         CALL    FLMLDVEX
1AAA 79             MOV     A,C
1AAB 329B03         STA     FLSCR0
1AAE EB             XCHG
1AAF 229C03         SHLD    FLSCR1
1AB2 010000         LXI     B,0
1AB5 50             MOV     D,B
1AB6 59             MOV     E,C
1AB7 21051A         LXI     H,NORMALIZ      ;NORMALIZE ANSWER AFTER
1ABA E5             PUSH    H
1ABB 210000         LXI     H,FLMULLP       ;THREE TIMES THROUGH LOOP
1ABE E5             PUSH    H
1ABF E5             PUSH    H
1AC0 219303         LXI     H,ACCUMLTR
                FLMULLP:
1AC3 7E             MOV     A,M
1AC4 23             INX     H
1AC5 B7             ORA     A
1AC6 CA0000         JZ      FLMULXT
1AC9 E5             PUSH    H
1ACA 2E08           MVI     L,008H
                FLMULLQ:
1ACC 1F             RAR             ;NEXT BIT OF MULTIPLIER
1ACD 67             MOV     H,A
1ACE 79             MOV     A,C
1ACF D20000         JNC     FLMULNA
1AD2 E5             PUSH    H
1AD3 2A9C03         LHLD    FLSCR1  ;BIT ON, ADD MULTIPLICAND
1AD6 19             DAD     D
1AD7 EB             XCHG
1AD8 E1             POP     H
1AD9 3A9B03         LDA     FLSCR0
1ADC 89             ADC     C
                FLMULNA:
1ADD 1F             RAR             ;SHIFT CDEB RIGHT ONE BIT
1ADE 4F             MOV     C,A
1ADF 7A             MOV     A,D
1AE0 1F             RAR
1AE1 57             MOV     D,A
1AE2 7B             MOV     A,E
1AE3 1F             RAR
1AE4 5F             MOV     E,A
1AE5 78             MOV     A,B
1AE6 1F             RAR
```

```
1AE7 47          MOV     B,A
1AE8 2D          DCR     L
1AE9 7C          MOV     A,H
1AEA C2CC1A      JNZ     FLMULLQ
1AED E1          POP     H
1AEE C9          RET

                 FLMULXT:
1AEF 43          MOV     B,E
1AF0 5A          MOV     E,D
1AF1 51          MOV     D,C
1AF2 4F          MOV     C,A
1AF3 C9          RET

                 FLMLDVEX:
1AF4 78          MOV     A,B     ;COMPUTE EXP FOR MULTIPLY/DIVIDE
1AF5 B7          ORA     A
1AF6 CA0000      JZ      FLMLDVEZ
1AF9 7D          MOV     A,L
1AFA 219603      LXI     H,FLACCEXP
1AFD AE          XRA     M
1AFE 80          ADD     B
1AFF 47          MOV     B,A
1B00 1F          RAR
1B01 A8          XRA     B
1B02 78          MOV     A,B
1B03 F20000      JP      FLMLDVEY
1B06 C680        ADI     080H
1B08 77          MOV     M,A
1B09 CA7008      JZ      POPHLRET
1B0C CD0000      CALL    SIGNIFY
1B0F 77          MOV     M,A
1B10 2B          DCX     H
1811 C9          RET

                 EXPRNEXC:
1B12 CD0000      CALL    SIGNACC ;RANGE EXECEEDED FOR EXP FUNCTION
1B15 2F          CMA                     ;RESULT DETERMINED BY SGN(X)
1B16 E1          POP     H
                 FLMLDVEY:
1B17 B7          ORA     A
                 FLMLDVEZ:
1B18 E1          POP     H
1B19 F21A1A      JP      ZEROAC
1B1C C3561A      JMP     ERRAOV
```

```
                ;
                ;  FLOATING POINT DIVIDE ROUTINE
                ;

                FLDIVB10:
1B1F CD0000         CALL    PUSHAC  ;COMPUTE AC/10
1B22 0120B4         LXI     B,0B420H
1B25 110000         LXI     O,00000H
1B28 CD0000         CALL    LDACRG
                DIVOPR:
1B2B C1             POP     B
1B2C D1             POP     O
                FLOIV:
1B2D CD0000         CALL    SIGNACC ;DIVIOE REGISTERS BY ACCUMULATOR
1B30 CA0000         JZ      ERRADO
1B33 2EFF           MVI     L,0FFH
1B35 C0F41A         CALL    FLMLDVEX
1B3B 34             INR     M
1B39 34             INR     M       ;plus 2
1B3A 2B             DCX     H
1B3B 7E             MOV     A,M
1B3C 2F             CMA
1B3D 329D03         STA     FLSCR2
1B40 2B             OCX     H
1B41 7E             MOV     A,M
1B42 2F             CMA
1B43 329C03         STA     FLSCR1
1B46 2B             OCX     H
1B47 7E             MOV     A,M
1B4B 2F             CMA
1B49 329B03         STA     FLSCR0
1B4C 41             MOV     B,C
1B4D EB             XCHG
1B4E AF             XRA     A
1B4F 4F             MOV     C,A
1B50 57             MOV     O,A
1B51 5F             MOV     E,A
1B52 329E03         STA     FLSCR3
                FLDIVLP:
1B55 E5             PUSH    H
1B56 C5             PUSH    B
1B57 37             STC
1B5B 3A9B03         LDA     FLSCR0
1B5B 8D             ADC     L
1B5C 6F             MOV     L,A
1B5D 3A9C03         LDA     FLSCR1
1B60 8C             ADC     H
1B61 67             MOV     H,A
1B62 3A9D03         LOA     FLSCR2
1B65 88             ADC     B
1B66 47             MOV     B,A
1B67 3A9E03         LDA     FLSCR3
1B6A CEFF           ACI     0FFH
1B6C D20000         JNC     FLOIVSF
1B6F 329E03         STA     FLSCR3
```

```
1872 F1          POP     PSW     ;TRIAL SUBTRACT SUCCEEDED,
1B73 F1          POP     PSW     ;THROW AWAY SAVED DIVIDEND
1874 37          STC
1875 C30000      JMP     FLDIVSS
             FLDIVSF:
1B78 C1          POP     B       ;TRIAL SU8TRACT FAILED, RESTORE
1B79 E1          POP     H
             FLDIVSS:
187A 79          MOV     A,C
187B 3C          INR     A
187C 3D          DCR     A
1B7D 1F          RAR
1B7E FA381A      JM      FLROUNDV
1BB1 17          RAL
1882 7B          MOV     A,E
1B83 17          RAL
1B84 5F          MOV     E,A
1B85 7A          MOV     A,D
1B86 17          RAL
1B87 57          MOV     D,A
1BB8 79          MOV     A,C
1889 17          RAL
1B8A 4F          MOV     C,A
1B88 29          DAD     H
1B8C 78          MOV     A,B
1B8D 17          RAL
1B8E 47          MOV     B,A
188F 3A9E03      LDA     FLSCR3
1B92 17          RAL
1B93 329E03      STA     FLSCR3
1B96 79          MOV     A,C
1B97 82          ORA     D
1B98 B3          ORA     E
1B99 C2551B      JNZ     FLDIVLP
1B9C E5          PUSH    H
1B9D 219603      LXI     H,FLACCEXP
1BA0 35          DCR     M
1BA1 E1          POP     H
1BA2 C2551B      JNZ     FLDIVLP
1BA5 C3561A      JMP     ERRAOV

             ERRADO:
1BA8 1E21        MVI     E,ERRNDO-ERRN
18AA C3F105      JMP     ERRMSG
```

```
                ;
                ;  MISCELLANEOUS AUXILIARY ROUTINES
                ;


                ;
                ;  COPY ACCUMULATOR TO STACK
                ;
                PUSHAC:
1BAD EB             XCHG            ;PUSH ACCUMULATOR ONTO STACK
                PUSHAC1:
1BAE 2A9303         LHLD    ACCUMLTR
1BB1 E3             XTHL
1BB2 E5             PUSH    H
1BB3 2A9503         LHLD    FLACCMSB
1BB6 E3             XTHL
1BB7 E5             PUSH    H
1BBB EB             XCHG
1BB9 C9             RET

                ;
                ;  LOAD ACCUMULATOR
                ;
                LDRGACMM:
1BBA CD0000         CALL    LDRGMM  ;LOAD FLOATING ACC AND REGISTERS
                LDACRG:
1BBD EB             XCHG            ;LOAD ACCUMULATOR FROM REGISTERS
1BBE 229303         SHLD    ACCUMLTR
1BC1 60             MOV     H,B
1BC2 69             MOV     L,C
1BC3 229503         SHLD    FLACCMSB
1BC6 EB             XCHG
1BC7 C9             RET

                ;
                ;  LOAD REGISTERS
                ;
                LDRGAC:
1BCB 219303         LXI     H,ACCUMLTR      ;LOAD REGISTERS FROM ACCUMULATOR
                LDRGMM:
1BCB 5E             MOV     E,M     ;LOAD REGISTERS FROM FLOAT NUMBER
1BCC 23             INX     H
                LDDCBMM:
1BCD 56             MOV     D,M     ;LOAD REGISTERS FROM STRING
                LDICBMM:
1BCE 23             INX     H
1BCF 4E             MOV     C,M
1BD0 23             INX     H
1BD1 46             MOV     B,M
                INCHLRET:
1BD2 23             INX     H
1BD3 C9             RET
```

```
                ;
                ;  STORE ACCUMULATOR / COPY A VALUE
                ;
                LDMMAC:
1BD4 119303         LXI     D,ACCUMLTR      ;LOAD MEMORY FROM ACCUMULATOR
                COPYVAL:
1BD7 3A6B03         LDA     TYPEFLG ;COPY VALUE FROM (DE) TO (HL)
1BDA 47             MOV     B,A
                COPYVALL:
1BDB 1A             LDAX    D
1BDC 77             MOV     M,A
1BDD 13             INX     D
1BDE 23             INX     H
1BDF 05             DCR     B
1BE0 C2DB1B         JNZ     COPYVALL
1BE3 C9             RET

                ;
                ;  TURN ON HIGH ORDER MANTISSA BITS OF ACCUMULATOR/REGISTERS
                ;
                SIGNIFY:
1BE4 219503         LXI     H,FLACCMSB      ;SET ON HIGH-ORDER MANTISSA BITS,
1BE7 7E             MOV     A,M     ;AND SAVE SIGN IN FLACCSSV
1BEB 07             RLC
1BE9 37             STC
1BEA 1F             RAR
1BEB 77             MOV     M,A     ;FIRST ACCUMULATOR,
1BEC 3F             CMC
1BED 1F             RAR
1BEE 23             INX     H
1BEF 23             INX     H
1BF0 77             MOV     M,A
1BF1 79             MOV     A,C
1BF2 07             RLC
1BF3 37             STC
1BF4 1F             RAR
1BF5 4F             MOV     C,A     ;THEN REGISTERS
1BF6 1F             RAR
1BF7 AE             XRA     M
1BFB C9             RET
```

```
                ;
                ;  FLOATING POINT COMPARISON:  REGISTERS VS ACCUMULATOR
                ;
                FLCMP:
1BF9 78             MOV     A,B     ;FLOATING COMPARE REGS TO ACC
1BFA B7             ORA     A
1BFB CA0000         JZ      SIGNACC
1BFE 210000         LXI     H,FLCMPXT
1C01 E5             PUSH    H
1C02 CD0000         CALL    SIGNACC
1C05 79             MOV     A,C
1C06 C8             RZ
1C07 219503         LXI     H,FLACCMSB
1C0A AE             XRA     M
1C0B 79             MOV     A,C
1C0C F8             RM
1C0D CD0000         CALL    FLCMPM
1C10 1F             RAR
1C11 A9             XRA     C
1C12 C9             RET

                FLCMPM:
1C13 23             INX     H       ;COMPARE MANTISSAS
1C14 78             MOV     A,B
1C15 BE             CMP     M
1C16 C0             RNZ
1C17 28             DCX     H
1C18 79             MOV     A,C
1C19 BE             CMP     M
1C1A C0             RNZ
1C1B 2B             DCX     H
1C1C 7A             MOV     A,D
1C1D BE             CMP     M
1C1E C0             RNZ
1C1F 28             DCX     H
1C20 7B             MOV     A,E
1C21 96             SUB     M
1C22 C0             RNZ
1C23 E1             POP     H
1C24 E1             POP     H
1C25 C9             RET
```

```
                ;
                ;  COMPUTE INTEGER PART OF ACCUMULATOR
                ;
                FIXAC:
1C26 47             MOV     B,A     ;LOAD REGS WITH FIX(AC)
1C27 4F             MOV     C,A
1C28 57             MOV     D,A
1C29 5F             MOV     E,A
1C2A 87             ORA     A
1C2B C8             RZ
1C2C E5             PUSH    H
1C2D CDC818         CALL    LDRGAC
1C30 CDE418         CALL    SIGNIFY
1C33 AE             XRA     M
1C34 67             MOV     H,A
1C35 FC0000         CM      DECCDE
1C38 3E98           MVI     A,098H
1C3A 90             SU8     8
1C3B CD7B1A         CALL    SHIFTR0
1C3E 7C             MOV     A,H
1C3F 17             RAL
1C40 DC4C1A         CC      INCCDE
1C43 0600           MVI     B,000H
1C45 DC671A         CC      CMREGS
1C48 E1             POP     H
1C49 C9             RET

                DECCDE:
1C4A 1B             DCX     D       ;DECREMENT CDE
1C4B 7A             MOV     A,D
1C4C A3             ANA     E
1C4D 3C             INR     A
1C4E C0             RNZ
1C4F 0D             DCR     C
1C50 C9             RET

                FLMUL810:
1C51 CDC81B         CALL    LDRGAC  ;MULTIPLY CONTENTS OF AC 8Y 10
1C54 78             MOV     A,B
1C55 B7             ORA     A
1C56 C8             RZ
1C57 C602           ADI     002H
1C59 DA561A         JC      ERRAOV
1C5C 47             MOV     B,A
1C5D CDB419         CALL    FLADD   ;AC=AC+4*AC
1C60 219603         LXI     H,FLACCEXP
1C63 34             INR     M       ;AC=2*AC
1C64 C0             RNZ
1C65 C3561A         JMP     ERRAOV

                SIGNACC:
1C68 3A9603         LDA     FLACCEXP        ;FIND SIGN OF ACCUMULATOR
1C6B B7             ORA     A
1C6C C8             RZ
1C6D 3A9503         LDA     FLACCMSB
```

```
1C70 C30000     JMP     SIGNXTND
            FLCMPXT:
1C73 2F         CMA
            SIGNXTND:
1C74 17         RAL
            CMPXT:
1C75 9F         SBB     A
1C76 C0         RNZ
1C77 3C         INR     A
1C78 C9         RET

            CMANSWR:
1C79 210000     LXI     H,CMACCS        ;F(X)=-F(0)
1C7C E3         XTHL
1C7D E9         PCHL

            SGNFCT:
1C7E CD681C     CALL    SIGNACC
            FLOATBYT:
1C81 0688       MVI     B,088H
1C83 110000     LXI     D,0
            FLOATINT:
1C86 219603     LXI     H,FLACCEXP      ;CONVERT INTEGER IN ADE TO FLOAT,
1C89 4F         MOV     C,A
1C8A 70         MOV     M,B     ;EXPONENT ASSUMED IN B
1C8B 0600       MVI     B,000H
1C8D 23         INX     H
1C8E 3680       MVI     M,080H
1C90 17         RAL
1C91 C3021A     JMP     NORMALZI

            ;
            ;  COMPUTE ABSOLUTE VALUE OF ACCUMULATOR
            ;
            ABSFCT:
1C94 CD681C     CALL    SIGNACC ;ABS FUNCTION
1C97 F0         RP
            CMACCS:
1C98 219503     LXI     H,FLACCMSB      ;CHANGE SIGN OF ACCUMULATOR
1C9B 7E         MOV     A,M
1C9C EE80       XRI     080H
1C9E 77         MOV     M,A
1C9F C9         RET

            INTFCT:
1CA0 219603     LXI     H,FLACCEXP      ;INT FUNCTION
1CA3 7E         MOV     A,M
1CA4 FE98       CPI     098H
1CA6 3A9303     LDA     ACCUMLTR
1CA9 D0         RNC
1CAA 7E         MOV     A,M
1CAB CD261C     CALL    FIXAC
1CAE 3698       MVI     M,098H
1CB0 7B         MOV     A,E
1CB1 F5         PUSH    PSW
1CB2 79         MOV     A,C
```

```
1CB3 17          RAL
1CB4 CD021A      CALL    NORMALZI
1CB7 F1          POP     PSW
1CB8 C9          RET
```

```
                ;
                ;  FLOATING POINT DECODE ROUTINE
                ;

                DECODE:
1CB9 FE2D           CPI     "-"     ;DECODE EXTERNAL FORM OF NUMBER
1C8B F5             PUSH    PSW
1C8C CA0000         JZ      DECODEIN
1C8F FE2B           CPI     "+"
1CC1 CA0000         JZ      DECODEIN
1CC4 2B             DCX     H
                DECODEIN:
1CC5 CD1A1A         CALL    ZEROAC
1CC8 47             MOV     B,A
1CC9 57             MDV     D,A
1CCA 5F             MOV     E,A
1CCB 2F             CMA
1CCC 4F             MDV     C,A
                DECODELP:
1CCD CDAB03         CALL    SCANNXT ;bscan ,
1CD0 DA0000         JC      DECDIGIT
1CD3 FE2E           CPI     "."
1CD5 CA0000         JZ      DECDDEPT
1CD8 FE45           CPI     "E"     ;UPPER CASE E
1CDA CA0000         JZ      DECODEXP
1CDD FE65           CPI     "e"     ;LDWER CASE E
1CDF C20000         JNZ     DECDDVAL
                DECODEXP:
1CE2 CDA803         CALL    SCANNXT ;bscan ,
1CE5 E5             PUSH    H
1CE6 210000         LXI     H,DECODEXL
1CE9 E3             XTHL
1CEA 15             DCR     D
1CEB FEAB           CPI     KEYSUB
1CED C8             RZ
1CEE FE2D           CPI     "-"
1CF0 C8             RZ
1CF1 14             INR     D
1CF2 FE2B           CPI     "+"
1CF4 C8             RZ
1CF5 FEAA           CPI     KEYADD
1CF7 C8             RZ
1CF8 F1             PDP     PSW
1CF9 2B             DCX     H
                DECDDEXL:
1CFA CDAB03         CALL    SCANNXT ;bscan ,        ;SCAN EXPDNENT
1CFD D20000         JNC     DECDDEXQ
1D00 7B             MOV     A,E     ;DECDDE EXPDNENT DIGIT
1D01 07             RLC             ;E=10*E+VAL(M)
1D02 07             RLC
1D03 83             ADD     E
1D04 07             RLC
1D05 86             ADD     M
1D06 D630           SUI     "0"
1D08 5F             MOV     E,A
```

```
1D09 C3FA1C     JMP     DECODEXL
            DECODEXQ:
1D0C 14         INR     D
1D0D C20000     JNZ     DECODVAL
1D10 AF         XRA     A
1D11 93         SUB     E
1D12 5F         MOV     E,A
1D13 0C         INR     C
            DECODEPT:
1D14 0C         INR     C       ;DECODE DECIMAL POINT
1D15 CACD1C     JZ      DECODELP
            DECODVAL:
1D18 E5         PUSH    H
1D19 7B         MOV     A,E
1D1A 90         SUB     B
            DECDEXPA:
1D1B F40000     CP      DECMULUP        ;COMBINE MANTISSA, EXPONENT
1D1E F20000     JP      DECDEXAL
1D21 F5         PUSH    PSW
1D22 CD1F1B     CALL    FLDIVB10
1D25 F1         POP     PSW
1D26 3C         INR     A
            DECDEXAL:
1D27 C21B1D     JNZ     DECDEXPA
1D2A D1         POP     D
1D2B F1         POP     PSW
1D2C CC9B1C     CZ      CMACCS
1D2F EB         XCHG
1D30 C9         RET
```

```
                DECMULUP:
1D31 C8             RZ
                FLMLB10C:
1D32 F5             PUSH    PSW
1D33 CD511C         CALL    FLMULB10
1D36 F1             POP     PSW
1D37 3D             DCR     A
1D38 C9             RET

                DECDIGIT:
1D39 D5             PUSH    D       ;DECODE DIGIT OF NUMBER
1D3A 57             MOV     D,A
1D3B 78             MOV     A,B
1D3C 89             ADC     C
1D3D 47             MOV     B,A
1D3E C5             PUSH    B
1D3F E5             PUSH    H
1D40 D5             PUSH    D
1D41 CD511C         CALL    FLMULB10
1D44 F1             POP     PSW
1D45 D630           SUI     "0"
1D47 CD0000         CALL    DECDGADD
1D4A E1             POP     H
1D4B C1             POP     B
1D4C D1             POP     D
1D4D C3CD1C         JMP     DECODELP

                DECDGADD:
1D50 CDAD1B         CALL    PUSHAC
1D53 CD811C         CALL    FLOATBYT
                ADDOPR:
1D56 C1             POP     B
1D57 D1             POP     D
1D58 C3B419         JMP     FLADD
```

```
                ;
                ;  FLOATING POINT ENCODE ROUTINE
                ;
                ERRMSGIN:
1D5B E5             PUSH    H       ;PRINT CUR LINE NUM8ER IN ERROR
1D5C 21D505         LXI     H,MSGIN
1D5F CDAC0D         CALL    PRNTMSG
1D62 E1             POP     H
                PRINTINT:
1D63 E5             PUSH    H       ;PRINT AN INTEGER
1D64 21AB0D         LXI     H,PRNTNUMS
1D67 E3             XTHL
                ENCODEHL:
1D68 EB             XCHG            ;ENCODE AN INTEGER
1D69 AF             XRA     A
1D6A 0698           MVI     8,098H
1D6C CD861C         CALL    FLOATINT
                ENCODE:
1D6F 11F3FF         LXI     D,-13   ;ENCODE AC IN EXTERNAL FORM
1D72 2A8103         LHLD    PROG8ASE
1D75 19             DAD     D       ;CREATE POINTER TO ENCODE BUFFER
1D76 E5             PUSH    H
1D77 CD681C         CALL    SIGNACC
1D7A 3620           MVI     M," "
1D7C F20000         JP      ENCODFRS
1D7F 362D           MVI     M,"-"
                ENCODFRS:
1D81 23             INX     H
1D82 3630           MVI     M,"0"
1D84 CA0000         JZ      ENCODZXT
1D87 E5             PUSH    H
1D88 FC981C         CM      CMACCS
1D8B AF             XRA     A
1D8C F5             PUSH    PSW
1D8D CD0000         CALL    ENCODCMP
                ENCODUPL:
1D90 014391         LXI     8,09143H        ;FORCE NUMBER TO RANGE
1D93 11F84F         LXI     D,04FF8H        ;10**5 <= AC BY MULTIPLICATION
1D96 CDF91B         CALL    FLCMP
1D99 3D             DCR     A
1D9A F20000         JP      ENCODRND
1D9D F1             POP     PSW
1D9E CD321D         CALL    FLMLB10C
1DA1 F5             PUSH    PSW
1DA2 C3901D         JMP     ENCODUPL

                ENCODDNL:
1DA5 CD1F1B         CALL    FLDIVB10        ;FORCE NUMBER TO RANGE
1DA8 F1             POP     PSW     ;AC < 10**6 BY DIVISION
1DA9 3C             INR     A
1DAA F5             PUSH    PSW
1DAB CD0000         CALL    ENCODCMP
                ENCODRND:
1DAE CDA019         CALL    FLADDHLF        ;ROUND UP RESULT
1DB1 3C             INR     A
```

```
1DB2 CD261C     CALL    FIXAC
1DB5 CDBD1B     CALL    LDACRG
1DB8 010602     LXI     8,00206H        ;D.DDDDD
1DBB F1         POP     PSW
1DBC 81         ADD     C
1DBD FA0000     JM      ENCDEXPS
1DC0 FE07       CPI     007H
1DC2 D20000     JNC     ENCDEXPS
1DC5 3C         INR     A
1DC6 47         MOV     8,A
1DC7 3E01       MVI     A,001H
            ENCDEXPS:
1DC9 3D         DCR     A
1DCA E1         POP     H
1DCB F5         PUSH    PSW
1DCC 110000     LXI     D,ENCDCOEF
            ENCODDGL:
1DCF 05         DCR     B
1DD0 362E       MVI     M,"."
1DD2 CCD21B     CZ      INCHLRET
1DD5 C5         PUSH    B
1DD6 E5         PUSH    H
1DD7 D5         PUSH    D
1DD8 CDC81B     CALL    LDRGAC
1DDB E1         POP     H
1DDC 062F       MVI     8,'0-1  ;GENERATE NEXT DIGIT
            ENCODS8L:
1DDE 04         INR     B
1DDF 78         MOV     A,E
1DE0 96         SU8     M
1DE1 5F         MOV     E,A
1DE2 23         INX     H
1DE3 7A         MOV     A,D
1DE4 9E         SBB     M
1DE5 57         MOV     D,A
1DE6 23         INX     H
1DE7 79         MOV     A,C
1DE8 9E         S88     M
1DE9 4F         MOV     C,A
1DEA 2B         DCX     H
1DEB 28         DCX     H
1DEC D2DE1D     JNC     ENCODS8L
1DEF CD5B1A     CALL    ADDM2CDE
1DF2 23         INX     H
1DF3 CDBD1B     CALL    LDACRG
1DF6 EB         XCHG
1DF7 E1         POP     H
1DF8 70         MOV     M,B
1DF9 23         INX     H
1DFA C1         POP     8
1DFB 0D         DCR     C
1DFC C2CF1D     JNZ     ENCODDGL
1DFF 05         DCR     B
1E00 CA0000     JZ      ENCODEXP
            ENCDRTZR:
1E03 2B         DCX     H       ;REMOVE TRAILING ZEROES
```

```
1E04 7E          MOV     A,M
1E05 FE30        CPI     "0"
1E07 CA031E      JZ      ENCDRTZR
1E0A FE2E        CPI     "."     ;REMOVE TRAILING DECIMAL POINT
1E0C C4D21B      CNZ     INCHLRET
                 ENCODEXP:
1E0F F1          POP     PSW     ;ENCODE EXPONENT
1E10 CA0000      JZ      ENCODEXT
1E13 3645        MVI     M,"E"
1E15 23          INX     H
1E16 362B        MVI     M,"+"
1E18 F20000      JP      ENCDEXPP
1E1B 362D        MVI     M,"-"
1E1D 2F          CMA
1E1E 3C          INR     A
                 ENCDEXPP:
1E1F 062F        MVI     B,'0-1
                 ENCDEXPL:
1E21 04          INR     B
1E22 D60A        SUI     10
1E24 D2211E      JNC     ENCDEXPL
1E27 C63A        ADI     '9+1
1E29 23          INX     H
1E2A 70          MOV     M,B
                 ENCODZXT:
1E2B 23          INX     H
1E2C 77          MOV     M,A
1E2D 23          INX     H
                 ENCODEXT:
1E2E 71          MOV     M,C
1E2F E1          POP     H
1E30 C9          RET

                 ENCODCMP:
1E31 017494      LXI     B,09474H        ;10**6
1E34 11F723      LXI     D,023F7H
1E37 CDF91B      CALL    FLCMP
1E3A E1          POP     H
1E3B 3D          DCR     A
1E3C F2A51D      JP      ENCODDNL
1E3F E9          PCHL

                 FLHALF:
1E40 000000      DB      000h, 000h, 000h, 080h  ;1/2
1E43 80
                 ENCDCOEF:
1E44 A08601      db      0a0h, 086h, 001h        ;10**5
1E47 102700      db      010h, 027h, 000h        ;10**4
1E4A E80300      db      0e8h, 003h, 000h        ;10**3
1E4D 640000      db      064h, 000h, 000h        ;10**2
1E50 0A0000      db      00ah, 000h, 000h        ;10**1
1E53 010000      db      001h, 000h, 000h        ;10**0
```

```
                ;
                ;  FLOATING POINT LOGARITHM ROUTINE
                ;

                LOGCOEF:
1E56 03             DB      3
1E57 AA5619         db      0aah, 056h, 019h, 080h
1E5A 80
1E5B F12276         db      0f1h, 022h, 076h, 080h
1E5E 80
1E5F 45AA38         db      045h, 0aah, 038h, 082h
1E62 82

                FLONE:
1E63 000000         db      000h, 000h, 000h, 081h  ;1.0
1E66 81

                LOGFCT:
1E67 CD681C         CALL    SIGNACC ;LOG FUNCTION
1E6A 3D             DCR     A
1E6B FA230C         JM      ERRAFC
1E6E 219603         LXI     H,FLACCEXP
1E71 7E             MOV     A,M
1E72 013580         LXI     B,08035H
1E75 11F304         LXI     D,004F3H
1E78 90             SUB     B
1E79 F5             PUSH    PSW
1E7A 70             MOV     M,B
1E7B D5             PUSH    D
1E7C C5             PUSH    B
1E7D CDB419         CALL    FLADD
1E80 C1             POP     B
1E81 D1             POP     D
1E82 04             INR     B
1E83 CD2D1B         CALL    FLDIV
1E86 21631E         LXI     H,FLONE
1E89 CDA919         CALL    FLMMMAC
1E8C 21561E         LXI     H,LOGCOEF
1E8F CD0000         CALL    FCTPOLY2
1E92 018080         LXI     B,08080H
1E95 110000         LXI     D,00000H
1E98 CDB419         CALL    FLADD
1E9B F1             POP     PSW
1E9C CD501D         CALL    DECDGADD
                FLMULLN2:
1E9F 013180         LXI     B,08031H        ;LN(2)=0.6931472
1EA2 111872         LXI     D,07218H
1EA5 C3A11A         JMP     FLMUL
```

```
                ;
                ;  FLOATING POINT SQUARE ROOT/EXPONENTIATION ROUTINE
                ;

                SQRFCT:
1EA8 CDAD1B         CALL    PUSHAC  ;SQR FUNCTION
1EAB 21401E         LXI     H,FLHALF        ;SQR(X)=X**1/2
1EAE CDBA1B         CALL    LORGACMM
                EXPOPR:
1EB1 C1             POP     B       ;X**Y=EXP(LOG(X)*Y)
1EB2 D1             POP     O
1EB3 CD6B1C         CALL    SIGNACC
1EB6 CA0000         JZ      EXPFCT
1EB9 7B             MOV     A,B
1EBA B7             ORA     A
1EBB CA1B1A         JZ      LOACCE
1EBE D5             PUSH    O
1EBF C5             PUSH    B
1EC0 79             MOV     A,C
1EC1 F67F           ORI     07FH
1EC3 CDC81B         CALL    LDRGAC
1EC6 F20000         JP      EXPEXPOS
1EC9 D5             PUSH    O
1ECA C5             PUSH    B
1ECB CDA01C         CALL    INTFCT
1ECE C1             POP     B
1ECF D1             POP     D
1ED0 F5             PUSH    PSW
1ED1 CDF91B         CALL    FLCMP
1ED4 E1             POP     H
1ED5 7C             MOV     A,H
1ED6 1F             RAR
                EXPEXPOS:
1ED7 E1             POP     H
1EDB 229503         SHLD    FLACCMSB
1ED8 E1             POP     H
1EDC 229303         SHLD    ACCUMLTR
1EDF DC791C         CC      CMANSWR
1EE2 CC9B1C         CZ      CMACCS
1EE5 D5             PUSH    D
1EE6 C5             PUSH    B
1EE7 CD671E         CALL    LOGFCT
1EEA C1             POP     B
1EEB D1             POP     D
1EEC CDA11A         CALL    FLMUL
```

```
                ;
                ;  EXPONENTIAL FUNCTION ROUTINE
                ;

                EXPFCT:
1EEF CDAD18         CALL    PUSHAC  ;EXP FUNCTION
1EF2 013881         LXI     8,08138H        ;LOG(2)E=1.442695
1EF5 113BAA         LXI     D,0AA3BH
1EF8 CDA11A         CALL    FLMUL
1EF8 3A9603         LDA     FLACCEXP
1EFE FE88           CPI     088H
1F00 D21218         JNC     EXPRNEXC
1F03 CDA01C         CALL    INTFCT
1F06 C680           ADI     080H
1F08 C602           ADI     002H
1F0A DA1218         JC      EXPRNEXC
1F0D F5             PUSH    PSW
1F0E 21631E         LXI     H,FLONE
1F11 CDA319         CALL    FLADDM
1F14 CD9F1E         CALL    FLMULLN2
1F17 F1             POP     PSW
1F18 C1             POP     8
1F19 D1             POP     D
1F1A F5             PUSH    PSW
1F1B CDB119         CALL    FLSU8
1F1E CD981C         CALL    CMACCS
1F21 210000         LXI     H,EXPCOEF
1F24 CD0000         CALL    FCTPOLY1
1F27 110000         LXI     D,0
1F2A C1             POP     B
1F2B 4A             MOV     C,D
1F2C C3A11A         JMP     FLMUL

                EXPCOEF:
1F2F 08             DB      8
1F30 402E94         db      040h, 02eh, 094h, 074h
1F33 74
1F34 704F2E         db      070h, 04fh, 02eh, 077h
1F37 77
1F38 6E0288         db      06eh, 002h, 088h, 07ah
1F38 7A
1F3C E6A02A         db      0e6h, 0a0h, 02ah, 07ch
1F3F 7C
1F40 50AAAA         db      050h, 0aah, 0aah, 07eh
1F43 7E
1F44 FFFF7F         db      0ffh, 0ffh, 07fh, 07fh
1F47 7F
1F48 000080         db      000h, 000h, 080h, 081h
1F4B 81
1F4C 000000         db      000h, 000h, 000h, 081h
1F4F 81
```

```
                ;
                ;  FLOATING POINT POLYNOMINAL EVALUATORS
                ;

                FCTPOLY2:
1F50 CDAD1B         CALL    PUSHAC  ;POLYNOMIAL EVALUATOR
1F53 119F1A         LXI     D,MULOPR        ;EVALUATE P(X**2)*X
1F56 D5             PUSH    D
1F57 E5             PUSH    H
1F58 CDCB1B         CALL    LDRGAC
1F5B CDA11A         CALL    FLMUL
1F5E E1             POP     H
                FCTPOLY1:
1F5F CDAD1B         CALL    PUSHAC  ;EVALUATE P(X)
1F62 7E             MOV     A,M
1F63 23             INX     H
1F64 CDBA1B         CALL    LDRGACMM
                FCTPOLYL:
1F67 C1             POP     B
1F68 D1             POP     D
1F69 3D             DCR     A
1F6A C8             RZ
1F6B D5             PUSH    D
1F6C C5             PUSH    B
1F6D F5             PUSH    PSW
1F6E E5             PUSH    H
1F6F CDA11A         CALL    FLMUL
1F72 E1             POP     H
1F73 CDCB1B         CALL    LDRGMM
1F76 E5             PUSH    H
1F77 CDB419         CALL    FLADD
1F7A E1             POP     H
1F7B F1             POP     PSW
1F7C C3671F         JMP     FCTPOLYL
```

```
                ;
                ;  RANDOM NUMBER GENERATOR
                ;


                RNDFCT:
1F7F CD681C         CALL    SIGNACC ;RND FUNCTION
1F82 FA0000         JM      RNDFCTUS        ;<0 - INITIALIZE SEED
1F85 219F03         LXI     H,RNDFCTSD
1F88 CDBA1B         CALL    LDRGACMM
1F8B C8             RZ              ;=0 - PREVIOUS VALUE
1F8C 013598         LXI     B,09835H
1F8F 117A44         LXI     D,0447AH
1F92 CDA11A         CALL    FLMUL   ;>0 - NEXT VALUE
1F95 012868         LXI     B,06828H
1F98 1146B1         LXI     D,0B146H
1F9B CDB419         CALL    FLADD
                RNDFCTUS:
1F9E CDC81B         CALL    LDRGAC  ;CHANGE SEED
1FA1 7B             MOV     A,E
1FA2 59             MOV     E,C
1FA3 4F             MOV     C,A
1FA4 3680           MVI     M,080H
1FA6 2B             DCX     H
1FA7 46             MOV     B,M
1FA8 3680           MVI     M,080H
1FAA CD051A         CALL    NORMALIZ
1FAD 219F03         LXI     H,RNDFCTSD
1FB0 C3D41B         JMP     LDMMAC
```

```
                ;
                ;  FLOATING POINT SINE/COSINE ROUTINES
                ;

                COSFCT:
1FB3 210000         LXI     H,PIOVER2       ;COS FUNCTION
1FB6 CDA319         CALL    FLADDM
                SINFCT:
1FB9 CDAD1B         CALL    PUSHAC  ;SIN FUNCTION
1FBC 014983         LXI     B,08349H        ;Y=X*2*PI
1FBF 11DB0F         LXI     D,00FDBH
1FC2 CDBD1B         CALL    LDACRG
1FC5 C1             POP     B
1FC6 D1             POP     D
1FC7 CD2D1B         CALL    FLDIV
1FCA CDAD1B         CALL    PUSHAC  ;Y=Y MOD 1
1FCD CDA01C         CALL    INTFCT
1FD0 C1             POP     B
1FD1 D1             POP     D
1FD2 CDB119         CALL    FLSUB
1FD5 210000         LXI     H,FLQUART
1FD8 CDA919         CALL    FLMMMAC
1FDB CD681C         CALL    SIGNACC
1FDE 37             STC
1FDF F20000         JP      SINFCTC
1FE2 CDA019         CALL    FLADDHLF
1FE5 CD6B1C         CALL    SIGNACC
1FE8 B7             ORA     A
                SINFCTC:
1FE9 F5             PUSH    PSW
1FEA F49B1C         CP      CMACCS
1FED 210000         LXI     H,FLQUART
1FF0 CDA319         CALL    FLADDM
1FF3 F1             POP     PSW
1FF4 D49B1C         CNC     CMACCS
1FF7 210000         LXI     H,COSCOEF
1FFA C3501F         JMP     FCTPOLY2

                PIOVER2:
1FFD DB0F49         db      0dbh, 00fh, 049h, 081h  ;PI/2
2000 81

                FLQUART:
2001 000000         db      000h, 000h, 000h, 07fh  ;1/4
2004 7F
                COSCOEF:
2005 05             DB      5
2006 BAD71E         db      0bah, 0d7h, 01eh, 086h
2009 86
200A 642699         db      064h, 026h, 099h, 0B7h
200D B7
200E 583423         db      05Bh, 034h, 023h, 087h
2011 87
2012 E05DA5         db      0e0h, 05dh, 0a5h, 086h
2015 B6
```

```
2016 DA0F49      db      0dah, 00fh, 049h, 083h
2019 83
```

```
                ;
                ;  FLOATING POINT TANGENT/ARCTANGENT ROUTINES
                ;

                TANFCT:
201A CDAD1B         CALL    PUSHAC  ;TAN FUNCTION
201D CDB91F         CALL    SINFCT
2020 C1             POP     B       ;TAN(X) = SIN(X)/COS(X)
2021 E1             POP     H
2022 CDAD1B         CALL    PUSHAC
2025 EB             XCHG
2026 CDBD1B         CALL    LDACRG
2029 CDB31F         CALL    COSFCT
202C C32B1B         JMP     DIVOPR

                ATNFCT:
202F CD681C         CALL    SIGNACC
2032 FC791C         CM      CMANSWR
2035 FC9B1C         CM      CMACCS
2038 3A9603         LDA     FLACCEXP
203B FEB1           CPI     0B1H
203D DA0000         JC      ATNFCTC
2040 010081         LXI     B,0B100H
2043 51             MOV     D,C
2044 59             MOV     E,C
2045 CD2D1B         CALL    FLDIV
2048 21A919         LXI     H,FLMMMAC
204B E5             PUSH    H
                ATNFCTC:
204C 210000         LXI     H,ATNCOEF
204F CD501F         CALL    FCTPOLY2
2052 21FD1F         LXI     H,PIOVER2
2055 C9             RET

                ATNCOEF:
2056 09             DB      9
2057 4AD73B         db      04ah, 0d7h, 03bh, 07Bh
205A 7B
205B 026EB4         db      002h, 06eh, 0B4h, 07bh
205E 7B
205F FEC12F         db      0feh, 0c1h, 02fh, 07ch
2062 7C
2063 74319A         db      074h, 031h, 09ah, 07dh
2066 7D
2067 B43D5A         db      084h, 03dh, 05ah, 07dh
206A 7D
206B CB7F91         db      0cBh, 07fh, 091h, 07eh
206E 7E
206F E4BB4C         db      0e4h, 0bbh, 04ch, 07eh
2072 7E
2073 6CAAAA         db      06ch, 0aah, 0aah, 07fh
2076 7F
2077 000000         db      000h, 000h, 000h, 0B1h
207A B1
```

```
                VERSNDAT:
207B 30322F         DB      "02/03/78",0
207E 30332F
2081 373800
                ENDINTRP:
2084 00             DB      0       ;END OF INTERPRETER
```

```
                ;
                ;  INITIALIZATION
                ;

                INITIALZ:

20B5 21FFFF     LXI     H,0FFFFH
2088 227303     SHLD    CURLINE
208B 210000     LXI     H,INITSTCK
208E F9         SPHL
208F 228903     SHLD    STCKBASE
2092 AF         XRA     A
2093 326503     STA     PRINTFLG
2096 CD3805     call    dclr
2099 CD490D     CALL    PRNTCRLF
209C 2100AF     LXI     H,LIMUPPER      ;ADDRESS LAST BYTE
209F 229103     SHLD    STRGTLIM
20A2 11E2FF     LXI     D,-10*3
20A5 19         DAD     D
20A6 228D03     SHLD    STRGBASE
20A9 228B03     SHLD    STRGFREE
20AC 1100FF     LXI     D,-256
20AF 19         DAD     D
20B0 D2F104     JNC     ERRAOM
20B3 E5         PUSH    H
20B4 210080     LXI     H,LIMLOWER      ;ADDRESS FIRST BYTE
20B7 110C00     LXI     D,12
20BA 19         DAD     D
20BB 3600       MVI     M,000H
20BD 23         INX     H
20BE 228103     SHLD    PROGBASE
20C1 E3         XTHL
20C2 D1         POP     D
20C3 F9         SPHL
20C4 228903     SHLD    STCKBASE
20C7 21F3FF     LXI     H,-13
20CA 39         DAD     SP
20CB F9         SPHL
20CC EB         XCHG
20CD CDE504     CALL    SPACECHK
20D0 7B         MOV     A,E
20D1 95         SUB     L
20D2 6F         MOV     L,A
20D3 7A         MOV     A,D
20D4 9C         SBB     H
20D5 67         MOV     H,A
20D6 01F0FF     LXI     B,-16
20D9 09         DAD     B
20DA CD490D     CALL    PRNTCRLF
20DD CD631D     CALL    PRINTINT
20E0 210000     LXI     H,INITMFRE
20E3 CDAC0D     CALL    PRNTMSG
20E6 217B20     LXI     H,VERSNDAT
20E9 CDAC0D     CALL    PRNTMSG
20EC CDF704     CALL    CLEARPGM
20EF 214B06     LXI     H,CMNDRSTR
```

```
20F2 220100     SHLD    SYSINITJ+1
20F5 E9         PCHL


            INITMFRE:
20F6 204259     DB      " BYTES FREE"
20F9 544553
20FC 204652
20FF 4545
2101 0D0A0A     DB      CR,LF,LF
2104 424153     db      "BASIC, Version of ", 0
2107 49432C
210A 205665
210D 727369
2110 6F6E20
2113 6F6620
2116 00
            INITSTSP:
21EF 00         DS      30*2+LINESYZE   ;INITIALIZATION STACK SPACE
            INITSTCK:
2203 00         DS      20


2204 2204       END
```