TSC 6800 Micro Basic Plus

SL68-19

TECHNICAL SYSTEMS CONSULTANTS

TSC 6800 Micro Basic Plus

TSC Technical Systems Consultants
Box 2574 W. Lafayette IN 47906 TSC

MICRO BASIC PLUS

I. INTRODUCTION:

This version of BASIC is a subset of the statements and commands usually available on large machines. The purpose of this manual is not to teach BASIC but simply to demonstrate the syntax and sample usage of MICRO BASIC PLUS. Particular attention should be paid to Appendix C which shows how to adapt this program to your particular system.

As in all TSC software, a great effort has been put forth in testing to eliminate "bugs" in the code. This however is no guarantee of perfect code. If a suspected bug is spotted, please jot down the circumstances involved and send it to us. We will do our best to send out errata sheets with all patches to owners of MICRO BASIC PLUS if necessary.

II. GENERAL INFORMATION:

A. The initial starting address is hex 0100. To restart after returning to monitor program, address hex 0103 should be used. This is set up automatically If MIKBUG is being used.

B. The prompt character is "!".

C. Line numbers must be between 0 and 9999 (4 digits maximum). Imbedded spaces are not permitted.

D. Numbers in arithmetic expressions must be between -99999 and +99999. If a larger number is entered, the least significant 5 digits are the only ones used.

E. Spaces are not permitted internal to numbers or keywords but may be used freely elsewhere.

F. All keywords (PRINT, GOTO, etc.) must be followed by a space or non alphabetic character.

G. Expressions are evaluated left to right with all operator precedence being equal. Parenthesis should be used to group sub-expressions. The allowed operators are +, -, *, /, and ^. There are several functions available also. ^ is used for exponentiation.

H. Variables are the 26 letters "A" through "Z". Variables may be DIMENSIONED either single (maximum = 98) or double (maximum = 98 x 98).

I. Multiple statements per line are permitted using a ":" as the separator.

J. Calculator mode of operation is permitted by typing a statement without a line number. MICRO BASIC PLUS will immediately perform the operation. Example:

```
PRINT 4*7
```

will print the answer 28 and then return with the prompt.

III. EDITING FEATURES:

A. Lines may be entered in any sequence. The interpreter automatically puts them in ascending order. It is recommended that multiples of 10 be used so if insertions are necessary they can be easily done.

B. Line numbers should begin in column 1.

C. To delete an existing line simply type that line number followed by a carriage return.

D. Backspacing is done using "control H".

E. To delete the current line being entered, type "control X".

F. Lines may be inserted, deleted, or added at anytime.

G. Line lengths are limited to 72 characters. If this is exceeded the line entered is thrown away and a new prompt will be issued.

IV. COMMANDS:

A. SCRATCH is used to delete the current users program from memory as well as clear all variables. Normally used without a line number but may appear in program with suicidal results.

B. RUN is used to start executing the users program with the lowest numbered line. May be used with a line number as well.

C. MONITOR is used to return to your monitor system.

D. LIST is used to list the users program. Several forms exist:

1. LIST c.r. - Lists the entire program
2. LIST X c.r. - Lists line X.
3. LIST X, Y c.r. - Lists Y lines starting at line X.
4. LIST X, c.r. - Lists entire program starting at line X.

E. BREAK: The "BREAK" key is used any time a BASIC program is running or a program is being listed and you wish it to stop. Hitting the "BREAK" key will cause current operation to halt and the prompt to be issued.

V. ASSIGNMENT STATEMENTS:

A. LET

1. Form:

```
LET (variable) = (expression)
```

2. Examples:

```
10 LET A = 200
20 LET B - C*62
```

3. The word "LET" is optional.

   Example:

```
30 D = 25 + A/B
```

B. READ and DATA

1. DATA statements contain a list of expressions or constants separated by commas and must be entered all on the same line. Each DATA statement "executed" becomes the current DATA statement, thus allowing several different DATA statements throughout the program.

2. READ is used to assign variables the values in a DATA statement. The first READ causes the first value of the current DATA statement to be assigned to the variable of the READ statement. The second READ gets the second value, etc.

3. If all data of the current DATA statement has been read, the next READ statement will go back and read the first value of that DATA statement.

4. Example:

```
10 DATA 2, 10, 12, -65/3, 42 + A

20 READ X, Y, Z
```

this results in X = 2, Y = 10, Z = 12. The next READ would cause the value of -65/3 to be assigned.

C. RESTORE

1. Used in conjunction with READ and DATA statements. When a RESTORE statement is executed, it causes the "pointer" which is pointing to the next piece of data in a DATA statement to move (be restored) to the first value of that data statement. May be thought of as restoring the "pointer" to its original position.

2. Example:

```
DATA 2, 4, 6, 8

READ X, Y

RESTOR

READ A
```

This results in X = 2, Y = 4 and A = 2 due to the RESTOR statement.

D. INPUT

1. The INPUT statement allows data entry during program execution.

2. Form:

```
INPUT "(optional string)", (variable), (variable)
```

3. The string portion of INPUT will type out the string on the terminal before issuing the prompt.

4. The INPUT prompt is a question mark, signifying BASIC is ready to accept input.

5. As many strings and variables may be used on one INPUT as desired.

6. If more than one value is to be input after the "?", the values should be separated by a comma.

7. The number of values entered must exactly equal the number of variables of the INPUT statement. If too few are entered another "?" will be output. If too many are entered, the excess will be ignored.

8. After the last value is Input, a "carriage return" should be entered. This terminates the input.

9. Only constants may be entered.

10. If a mistake is made on an entry a "control X" may be typed to delete that particular entry and a "?" will be output. This can only be done before the comma or carriage return is entered and only deletes the last value entered.

11. Examples:

```
10 INPUT A

20 INPUT "NUMBER", X

30 INPUT B, C, D
```

When line 20 is executed, the word NUMBER will be printed on the terminal followed by a "?". If 25 is then typed, X will be assigned the value 25.

12. The INPUT statement may also be used to stop the program but not ask for any values.

    Example:

    ```
    50 INPUT "STOP"
    ```

    This causes STOP to be printed, no "?" will be issued. To restart execution, a carriage return must be entered.

VI. OUTPUT STATEMENT

A. PRINT

1. Form:

   ```
   PRINT (list)
   ```

2. The (list) may be a list of variables, constants, or expressions in which case these values will be output to the terminal.

3. The (list) may also contain strings of alphanumeric characters enclosed in quotes ("). In this case the string would be output to the terminal.

4. The (list) may be blank in which case a blank line will be output, (skip a line).

5. Formatting Output:

   a. There are 9 print zones available per line, each being 8 columns wide.

   b. To make use of the print zones, items in the print list should be followed by a comma. When this is done, the next item to be printed will start in the next available zone. If 2 successive commas are used, a print zone will be skipped. If an alphanumeric string is output and extends into part of a following zone, the comma will cause the next printed item to start in the next unoccupied zone.

c. Semicolons may be used instead of commas. The semicolon does not cause the next item to be in the next available zone but instead it will be printed in the next available column (no spacing).

d. Two output formatting functions are also permitted, TAB and SPC. See function description for their use.

6. Examples:

```
10 PRINT "THE ANSWER IS"; A

20 PRINT "X = "; X, "Y = ";Y

30 PRINT A, B, C,, D

40 PRINT 2*(R+S), 62*4, A
```

VII. SUBSCRIPTED VARIABLES:

A. GENERAL INFORMATION

1. Subscripted variables should be thought of as arrays, vectors, matrices, or a variable with several values (memory locations).

2. All arrays may be either one or two dimensions.

3. The lowest subscript value is 0.

4. The maximum value is 98.

B. DIMENSION statement.

1. All subscripted variables must first appear in a DIMENSION statement. (DIM). It is good practice to put all DIM statements at the start of the program.

2. DIM is used to set the maximum size of an array.

3. Only constants can be used in DIM statements.

4. Examples:

```
10 DIM A(8), B(6,6)

20 DIM X(20,4)

30 DIM X(5), Y(l0), Z(98)
```

5. When using subscripted variables they should have the form:

   X(expression) or X(expression, expression)

   where X is the variable and the expression can be any valid expression including other subscripted variables. If the value of the subscript exceeds the value for which that variable was DIMENSIONED, an error will result.

   Examples:

```
A(3)

B(6+R, S(16))

Z(5, A(B))
```

VIII. TRANSFER OF CONTROL STATEMENTS

A. GOTO

1. Form:

   GOTO (line no.)

2. The line number may be represented as a variable, constant, or expression.

3. GOTO causes transfer of control to the line specified.

4. If used on multiple statements per line it should be the last statement.

5. Examples:

```
10 GOTO 100

20 GOTO 200 + B
```

B. GOSUB

1. Form:

```
GOSUB (line no.)
```

2. The line number may be represented as a variable, constant, or expression.

3. If used on multiple statements per line it should be the last statement.

4. Examples:

```
35 GOSUB 200

40 GOSUB 102 + B
```

5. Subroutines may be nested as deep as the stack will permit.

C. RETURN

1. Used to return from a subroutine

2. Returns to next line numbered statement following the calling GOSUB.

D. ON statement

1. Used with GOTO or GOSUB

2. Forms:

```
ON (expression) G0T0 (expression),..., (expression)

ON (expression) GOSUB(expression),..., (expression)
```

3. The value of the expression after ON is used to determine which of the expressions following the GO- should be evaluated to form the destination line number. The first expression is selected on a value of 1, the second for 2, etc.

4. The maximum number of expressions is 9.

5. If the value is less than 1 or greater than the number of expressions provided, the last one listed will be used.

6. Examples:

```
ON A GOTO 100, 200, 300
```

If A = I control will be transfered to line 100; if A = 2, 200, etc.

IX. CONDITIONAL STATEMENT

A. IF-THEN

1. Form:

```
IF X1 OP X2 THEN ST
```

where X1 and X2 can be constants, variables, or expressions and ST is any MICRO BASIC PLUS statement. OP is a comparison operator (see below).

2. Transfer of control is conditional depending on the result of the comparison of XI and X2. If the comparison is true, the statement following the THEN is executed. If the comparison is false, the statement following the THEN is ignored.

3. THEN is optional.

4. Comparison operators are the following:

SYMBOL	EXAMPLE	MEANING
=	A=B	A equals B
<	A<B	A is less than B
>	A>B	A is greater than B
<=	A<=B	A is less than or equal to B
>=	A>=B	A is greater than or equal to B
<>	A<>B	A is not equal to B

5. Examples:

```
10 IF A<B THEN PRINT "YES"

15 IF 2*C <= D+5 LET C = 5

20 IF A<B IF C<D PRINT "NO"

25 IF 12>X + (2*A) THEN 200
```

The last example Is used to GOTO line 200 (GOTO is not needed).

X. PROGRAM LOOPS

A. FOR and NEXT

1. Form:

```
FOR C = C1 TO C2 STEP C3
```

where C is the control or index variable, Cl is its initial value, C2 is its final value, and C3 is the increment size.

2. The index variable can not be a DIMENSIONED variable.

3. STEP is optional and if left off the value of C3 is assumed to be +1.

4. STEP may be positive for forward counting or negative for backwards counting.

5. All FOR-NEXT loops are executed at least once.

6. Loops may be nested as deep as memory will permit.

7. While nesting loops, no index variable should be used more than once.

8. Loops may be exited at any time.

9. Loops may be reentered if not previously indexed out.

10. NEXT is used to close the loops and should state the index variable of that loop.

11. Examples:

```
10 FOR A = 1 TO 10

20 NEXT A

50 FOR I = D*2 TO 100 + 3 STEP 2

60 NEXT I
```

12. If expressions are used for Cl, C2, and C3, they will be evaluated each time through the loop.

XI. MISCELLANEOUS STATEMENTS

A. REMARK

1. Used to insert remarks into programs.

2. Skipped during execution.

3. Example:

```
10 REMARK TEST 1

20 REM THIS IS A REMARK.
```

B. END

1. Used to terminate a MICRO BASIC PLUS program.

C. EXTERNAL

2. Used to execute machine code subroutines.

3. See Appendix D for details of its use.

XII. FUNCTIONS:

A. ARITHMETIC FUNCTIONS

1. SGN has the form:

```
SGN(X)
```

where X may be any arithmetic expression. This function returns a value of +1 for positive arguments, 0 if X is zero, and -1 for negative arguments.

2. ABS returns the absolute value of its argument. It has the form:

```
ABS (X)
```

where X is any expression

3. RND should be treated as a variable rather than a function since it has no argument. Whenever RND appears in an expression it will be replaced by a random number between 0 and 99.

4. Examples:

```
LET A = SGN(100-B)

B = ABS(R*100/C)

R = 65 + RND
```

B. OUTPUT FORMATTING FUNCTIONS.

1. TAB is used to move to a desired print column. It has the form:

```
TAB (X)
```

where X can be any expression. If the value of the argument is less than or equal to the column presently in, the TAB will be ignored.

2. SPC is used to output a specified number of spaces. It has the form:

```
SPC (X)
```

where X is any expression.

3. Examples:

```
10 PRINT TAB(6); A
```

prints the value of A starting in column 6.

```
20 PRINT X; SPC(5); Y
```

prints 5 spaces between the values of X and Y.

```
30 PRINT TAB(A+B); "*"; SPC(l0); X
```

XIII. OTHER INFORMATION:

A. All keywords may be written using the first 3 letters. (PRINT = PRI, INPUT = INP, etc.)

B. Some syntax checking is performed by MICRO BASIC PLUS during initial line entry.

C. When using the exponentiation operator (^) only 2 digits are allowed for the exponent (largest exponent is 99).

D. Keep in mind that large dimensioned variables eat up memory quickly. For example, to dimension A as A(98, 98) requires 29405 bytes of storage! To determine the amount of memory used, use the following formula:

```
Number of bytes = 3 * ((1st dimension + 1) *

                  (2nd dimension + 1)] + 2
```

APPENDIX A

ERROR CODES FOR MICRO BASIC PLUS

ERROR NUMBER	MEANING
10	Unrecognizable keyword
14	Illegal variable
16	No line number referenced by GOTO or GOSUB
20	Expression syntax, unbalanced parens, or dimension error
21	Expression expected but not found
22	Divided by zero
23	Arithmetic overflow
24	Expression too complex
31	Syntax error in PRINT statement
32	Missing closing quote in printed string
40	Bad DIM statement
45	Syntax error in INPUT statement
51	Syntax error in READ statement
62	Syntax error in IF statement
73	RETURN with no GOSUB
81	Error with FOR-NEXT
90	Memory overflow
99	"BREAK" detected

APPENDIX B

DUMPING AND LOADING PROCEDURES

I. DUMPING THE PROGRAM

After entering your MICRO BASIC PLUS program it is usually desirable to dump it to paper or cassette tape. If using Motorola's MIKBUG the procedure is extremely simple. First, from BASIC, enter the command MON to return to the monitor. MICRO BASIC PLUS has already done all the work of setting the punch limits. All that is necessary once in MIKBUG is to type "P" after turning on your recording device. For other systems, see Appendix C.

II. LOADING THE PROGRAM

While in MICRO BASIC PLUS type MON to return to MIKBUG. Prepare to load your cassette or paper tape as usual. Type "L" (MIKBUG's load function). When complete, type "G" and BASIC will return with the prompt. A quick LIST will verify your load. MICRO BASIC PLUS should always be reentered at location hex 103 to avoid clearing memory.

APPENDIX C

ADAPTING MICRO BASIC PLUS

I. This section is primarily intended for those who own systems not based around Motorola's MIKBUG, and hopefully gives enough information for adaptation. MICRO BASIC PLUS has been assembled for MIKBUG systems containing 8K of memory. If a different amount is available (as little as 4K may be used) the "memory end" should be adjusted accordingly as stated in part 11 below. (If EXT will not be used and a 4K system is owned, set memory end (locations O10F - 0110) to OF and FF respectively).

II. MEMORY END is stored in locations OIOF and 0110. It is now set to 1EFF which requires an 8K system. If your system is of different size, this number should be adjusted accordingly. BASIC will not run correctly if this is not set up for your system. Space should also be allowed for a stack (= 128 BYTES) + any I/0 patches if MIKBUG is not being used.

III. BREAK is presently referenced at location 010C. It jumps to an internal break routine at location 0452. This routine monitors MIKBUG's PIA for activity such that hitting the "BREAK" key during program execution or listing will immediately return to the main BASIC loop and respond with the prompt.

If using an ACIA this could be written to look for a special character, for example control C, before kicking out.

IV. OUTEEE is a jump to the output routine in MIKBUG (character in accumulator A, other registers undisturbed), and is at location 0106. If MIKBUG is not used, this should be patched to vector to your routine.

V. INCH is a jump to the input routine in MIKBUG and is at location 0109. Patch this if a different routine is used.

VI. COLD START should be done from location 0100 hex. Warm start is automatically setup and stored in MIKBUG's P.C. (A048 and A049). This is set up at location 01B3.

VII. STACK is initialized at 0186 and its top is set to A07F in MIKBUG's RAM. If different stroage is allocated for the stack, allow at least 128 BYTES. *IMPORTANT - at location 0943 the bottom of the stack is referenced. If the stack is moved this reference should be changed accordingly!

VIII. PUNCH LIMIT for dumping the source are set up in MICRO BASIC PLUS at locations 01C3 and 01C8., If MIKBUG is not used, these should be changed accordingly.

IX. PROMPT CHARACTER is stored at location 01D4. This may be changed if desired.

X. BACKSPACE CODE is stored at location 02D4. This may be changed.

XI. CANCEL CODE is at locations 02E3 and 07C2. These may be changed if both are changed identically.

XII. MON returns to MIKBUG. If a different monitor is used, the entry address at location 015F should be changed to that of the monitor used.

XIII. MEMORY ASSIGNMENT

0000-0003	Random number locations (must not all be 00)
00B0-00FD	Undimensioned variable storage
0100	START entry point
0103	RESTART entry point
0106	JUMP to OUTPUT CHARACTER
0109	JUMP to INPUT CHARACTER
010C	JUMP to BREAK routine
010F-0110	MEMORY END pointer
015F-0160	Monitor program entry point address
01B7-01B8	Stack address
01C3-01C4	Low punch limit address
01C8-01C9	High punch limit address
01D4	Prompt character (!)
02D4	Backspace code (control-H)
02E3	Line cancel code (control-X)
07C2	Line cancel code (control-X)
0D4D-0D4E	Pointer to end of user's source program
0D4F	Start of users source program
0FFF	Actual end of memory (4K system)
1EFF	Suggested MEMORY END (8K system)
1F00	Suggested EXT address (8K system)
1FFF	Actual end of memory (8K system)

For MIKBUG users:

A000	Stack end
A002-A003	Low punch limit
A004-A005	High punch limit
A048-A049	MIKBUG PC
A07F	Stack beginning
E0E3	MIKBUG entry point
E1AC	INPUT routine
E1D1	OUTPUT routine

APPENDIX D

THE EXTERNAL STATEMENT

The EXTERNAL (EXT) statement is internally set up to do a "JSR" to location 1F00. This can be found in BASIC at location 0701 and should be changed according to memory organization used. It Is important that all EXT routines exist beyond the address set up as the end of memory.

At first glance EXT seems limiting since only one address can be jumped to. This is not the case however. All non dimensioned variable are stored In fixed locations requiring three bytes each starting at location 00B0. (A = 0080, B = 0083, C = 00B6, etc.). They are stored as packed BCD with the least significant digits in the highest address (L.S.D. of A are in 0082). With this in mind, a variable can be chosen as a reference such that upon execution of EXT that variable can be read from memory and used as an offset or index in A "jump table". Using this method, one can have many, program selected, EXTERNAL routines available All EXTERNAL routines should end with an "RTS". Be sure to adjust "memory end" as required if using this feature of MICRO BASIC PLUS.

```
*
* PATCHES TO TSC MICRO BASIC PLUS TO ALLOW OPERATION WITH
* SWTBUG (SWTPC) OR MINIBUG (MOTOROLA) MONITORS.
*
* COPYRIGHT (C) 1978 BY
* TECHNICAL SYSTEMS CONSULTANTS, INC.
* P. O.  BOX 2574
* WEST LAFAYETTE, INDIANA  47906
* (317) 423-5465
*
* THE DIFFICULTY IN RUNNING TSC MICRO BASIC PLUS ON OTHER
* THAN MIKBUG MONITORS IS IN THE BREAK ROUTINE CALLED
* "INTBRK" AT $0452.  THIS ROUTINE WAS WRITTEN FOR A PIA
* CONTROL PORT RATHER THAN AN ACIA SERIAL PORT.  A NEW BREAK
* ROUTINE IS SUPPLIED BELOW.  UNFORTUNATELY, THIS ROUTINE
* IS TOO LARGE TO FIT IN THE SPACE OCCUPIED BY "INTBRK".
* FOR THIS REASON, YOU MUST PLACE THE SUPPLIED BREAK ROUTINE
* ELSEWHERE IN MEMORY.  WE RECOMMEND THAT YOU LOWER THE
* MEMORY END ADDRESS YOU PATCHED IN AT $010F BY 25 (DECIMAL)
* TO ALLOW ROOM FOR THIS ROUTINE TO BE ENTERED IN MEMORY.
* WHEREVER YOU PLACE THE ROUTINE (IT IS RELOCATABLE AND
* CAN BE PLACED ANYWHERE IN RAM), BE SURE TO PATCH THE
* ADDRESS OF WHERE THE NEW BREAK ROUTINE STARTS INTO
* MICRO BASIC AT $010D IN PLACE OF THE 0452 ALREADY
* THERE.  THE OLD "INTBRK" ROUTINE MAY BE LEFT IN MEMORY
* AND WILL BE IGNORED.
*
* AN EQUATE IS USED BELOW TO SET THE BASE ADDRESS OF
* THE SERIAL I O PORT (ACIA).  IF YOU ARE USING SWTBUG,
* THIS ADDRESS IS $8004 (THIS IS WHAT IS USED IN THE
* ASSEMBLY BELOW).  IF YOU ARE USING MINIBUG, THIS
* ADDRESS IS $8008, AND IT WILL BE NECESSARY TO CHANGE
* THE TWO REFERENCES TO ACIA AND ACIA+1 IN THE OBJECT
* CODE GENERATED BELOW.
*
****  REMEMBER TO PUT THE ADDRESS OF "EXTBRK" AT $010D

8004            ACIA    EQU    $8004     SETUP BASE ADDRESS OF ACIA

1FE6                    ORG    $1FE6     ORG AT END OF 8K SYSTEM

1FE6 36         EXTBRK  PSH A
1FE7 B6 80 04           LDA A  ACIA
1FEA 44                 LSR A
1FEB 25 02              BCS    BREAK2
1FED 32         BREAK1  PUL A
1FEE 39                 RTS
1FEF B6 80 05   BREAK2  LDA A  ACIA+1
1FF2 84 7F              AND A  #$7F
1FF4 81 03              CMP A  #3
1FF6 26 F5              BNE    BREAK1
1FF8 86 99              LDA A  #$99
1FFA 7E 04 61           JMP    $0461
                        END
     NO ERROR(S) DETECTED
```

APPENDIX E

INSTRUCTION SUMMARY

COMMANDS	STATEMENTS		FUNCTIONS
RUN	LET	GOTO	ABS
LIST	READ	GOSUB	SGN
SCRATCH	DATA	ON-GOTO	RND
MONITOR	RESTORE	ON_GOSUB	TAB
BREAK	INPUT	RETURN	SPC
	PRINT	FOR	
	REM	NEXT	
	END	IF-THEN	
	DIM	EXTERNAL	

MATH OPERATORS	RELATIONAL OPERATORS
- (unary) Minus	= Equal
- (unary) Plus	< Less than
* Multiplication	> Greater than
/ Division	<= Less than or equal
^ Exponentiation	>= Greater than or equal
+ Addition	<> Not equal
– Subtraction	

Line Numbers	- 0 to 9999
Constants	- 99999 to +99999
Variables	- single letters, A to Z, may be subscripted
Backspace	- control H
Line cancel	- control X

APPENDIX F

SAMPLE PROGRAMS

```
10 REM BASIC PLUS 'SWITCH'
12 REM THE OBJECT OF SWITCH IS TO REARRANGE A
14 REM RANDOM SEQUENCE TO NUMERICAL ORDER, LEFT TO RIGHT.
16 REM THIS IS DONE BY 'SWITCH'ING A PARTIAL
18 REM SEQUENCE STARTING FROM THE LEFT. FOR EXAMPLE
20 REM SWITCH 3 WOULD REVERSE THE SEQUENCE OF THE FIRST
22 REM THREE NUMBERS FROM THE LEFT.
25 DIM M(9)
30 FOR 1=1 TO 9 : M(I)=10-I : NEXT I
40 FOR 1=1 TO 10
50 A=RND/12+1
60 K=M(A) : M(A)=M(1) : M(1)=K
70 NEXT I
80 PRINT "THE SEQUENCE IS ":T=0
90 GOSUB 220
100 INPUT "    SWITCH HOW MANY ",D
110 IF D>0 IF D<10 G8TO 120
115 GOTO 100
120 E=1:T=T+1
130 IF D<=E GOTO 150
140 F=M(E):M(E)=M(D):M(D)=F : D=D-1 : E=E+1 : GOTO 130
150 FOR 1=1 TO 9
160 IF M(I)<>I GOTO 90
170 NEXT I
175 GOSUB 220
180 PRI:PRINT "YOU WIN IN ";T;" MOVES"
190 PRI:INPUT "WANT TO PLAY AGAIN (YES=1) ",T
200 IF T=1 GOTO 30
210 END
220 FOR I=1 TO 9:PRI M(I);:NEXT I:RET
```

```
!
!LIST
10 REM TEST OF RANDOM NUMBER DISTRIBUTION
15 DIM X(9)
20 GOSUB 1000
30 INPUT *NUMBER OF TIMES ",A
40 FOR B=0 TO 9: X(B)=0: NEXT B
50 FOR B=1 TO ABS(A)
60 C=RND/10: X(C)=X(C)+l
70 NEXT B
80 GOSUB 1000
90 PRINT TAB(10);"NUMBER";TAB(20);"TIMES"
100 PRINT TAB(10);"-------";TAB(20);"-----":PRI
110 FOR 1=0 TO 9:PRI TAB(12);I;TAB(21);X(I)
120 NEXT I
130 GOSUB 1000
135 R=0
140 FOR J=0 TO 9: R=R+(J*X(J)): NEXT J
150 PRINT "AVERAGE = ";R/A;" ";R-(R/A*A)
155 Z=2
160 IF R/A<4 LET Z=1
170 IF R/A>4 THEN Z=3
180 GOSUB 1000
190 ON Z GOSUB 300,400,500
200 END
300 PRINT "AVERAGE IS LOW":RETURN
400 PRI "AVERAGE IS OK!!": RET 500 PRIN *AVERAGE IS HIGH":RET
1000 PRI:PRI: RET

|RUN

NUMBER OF TIMES ? 1000

NUMBER TIMES
  0     101
  1     97
  2     110
  3     102
  4     93
  5     96
  6     100
  7     103
  8     97
  9     101

AVERAGE = 4.481
AVERAGE IS OK!!

!
```

```
               * MICRO BASIC PLUS SOURCE LISTING
               *
               * MICRO BASIC PLUS
               * COPYRIGHT (C) 1976 BY
               *
               * TECHNICAL SYSTEMS CONSULTANTS
               * BOX 2574
               * W. LAFAYETTE INDIANA 47906
               *
               *

               * EQUATES
A07F           STACK   EQU    $A07F
8004           PIAADR  EQU    $8004
A002           PFILBG  EQU    $A002
A004           PFILEN  EQU    $A004
1F00           EXTERN  EQU    $1F00
E0E3           MONITR  EQU    $E0E3
A048           MONPC   EQU    $A048
A000           STKBOT  EQU    $A000

               * TEMPORARY STORAGE

0000           RNDM    RMB    4
0004           BUFPNT  RMB    2
0006           FORSTK  RMB    2
0008           DIMPNT  RMB    2
000A           XTEMP3  RMB    2
000C           DATAST  RMB    2
000E           DATAPT  RMB    2
0010           TRYVAL  RMB    2
0012           CRFLAG  RMB    1
0013           QMFLAG  RMB    1
0014           ROWVAR  RMB    1
0015           ROWCON  RMB    1
0016           COLCON  RMB    1
0017           TABFLG  RMB    1
0018           DIMFLG  RMB    1
0019           RUNFLG  RMB    1
001A           DATAFL  RMB    1
001B           SUBCNT  RMB    1
001C           LETFLG  RMB    1
001D           FLDCNT  RMB    1
001E           NXPNTR  RMB    2
0020           XTEMP   RMB    2
0022           XSAVE   RMB    2
0024           XSAVE2  RMB    2
0026           NUMCNT  RMB    1
0027           NEGFLG  RMB    1
0028           NOEXFL  RMB    1
0029           EXTRA   RMB    2
002B           COUNT   RMB    1
002C           STKCNT  RMB    1
002D           AUXCNT  RMB    1
002E           SIGN    RMB    1
002F           AXSIGN  RMB    1
0030           OVFLBF  RMB    1
0031           XTEMP2  RMB    2
0033           XTEMP4  RMB    2
0035           XTEMP5  RMB    2
0037           CPX1    RMB    2
0039           CPX2    RMB    2
003B           STKEND  RMB    3
003E           CHRCNT  RMB    1
003F           OPSTAK  RMB    32
005F           AC      RMB    3
0062           NUMBER  RMB    3
0065           AX      RMB    3
```

```
0068           BUFFER  RMB    72

               * LABLE TABLE

00B0           LBLTBL  RMB    78
00FE           STKTOP  RMB    2

               * CONSTANTS

0008           BACKSP  EQU    $8
0018           DELCOD  EQU    $18
0021           PRMPTC  EQU    $21

0100                   ORG    $0100

               * MAIN PROGRAM

0100 7E 01 A6  START   JMP    MICBAS    JMP TO BEGIN
0103 7E 01 B0  RESTRT  JMP    FILBUF

               * EXTERNAL I-O ROUTINES

0106 7E E1 D1  OUTEEE  JMP    $E1D1
0109 BD E1 AC  INCH    JSR    $E1AC
010C 7E 04 52  BREAK   JMP    INTBRK
010F 1E FF     MEMEND  FDB    $1EFF

               * KEYWORD AND JUMP TABLE

0111 50        KEYTBL  FCC    ;PRI;
0112 52 49
0114 04 A6             FDB    PRINT

0116 49                FCC    ;INP;
0117 4E 50
0119 07 98             FDB    INPUT

011B 49                FCC    ;IF ;
011C 46 20
011E 08 B2             FDB    IF

0120 4C                FCC    ;LET;
0121 45 54
0123 07 72     LETADR  FDB    LET

0125 46                FCC    ;FOR;
0126 4F 52
0128 09 76             FDB    FOR

012A 4E                FCC    ;NEX;
012B 45 58
012D 09 9D             FDB    NEXT

012F 47                FCC    ;GOT;
0130 4F 54
0132 07 81             FDB    GOTO

0134 47                FCC    ;GOS;
0135 4F 53
0137 09 2B             FDB    GOSUB

0139 4F                FCC    ;ON ;
013A 4E 20
013C 08 76             FDB    ONGOTO

013E 52                FCC    ;RET;
013F 45 54
0141 09 53             FDB    RETURN
```

```
0143 52                FCC    ;REA;
0144 45 41
0146 08 26             FDB    READ

0148 44                FCC    ;DAT;
0149 41 54
014B 08 17             FDB    DATA

014D 52                FCC    ;RES;
014E 45 53
0150 08 6C             FDB    RESTOR

0152 44                FCC    ;DIM;
0153 49 4D
0155 06 71             FDB    DIM

0157 45                FCC    ;EXT;
0158 58 54
015A 07 01             FDB    EXTRNL

015C 4D                FCC    ;MON;
015D 4F 4E
015F E0 E3             FDB    MONITR

0161 45                FCC    ;END;
0162 4E 44
0164 01 B0             FDB    FILBUF

0166 52                FCC    ;REM;
0167 45 4D
0169 07 04             FDB    RUNEXC

016B 52                FCC    ;RUN;
016C 55 4E
016E 07 5F             FDB    RUN

0170 4C                FCC    ;LIS;
0171 49 53
0173 03 EC             FDB    LIST

0175 53                FCC    ;SCR;
0176 43 52
0178 01 A6             FDB    MICBAS
017A 00                FCB    0

017B 52        FCTTBL  FCC    ;RND;
017C 4E 44
017E 0A C0             FDB    EVAL88

0180 41                FCC    ;ABS;
0181 42 53
0183 0A BC             FDB    EVAL85

0185 53                FCC    ;SGN;
0186 47 4E
0188 0A B4             FDB    EVAL86
018A 00                FCB    0

               * INITIALIZATION

018B CE 01 00  CLRBEG  LDX    #START
018E DF 0A             STX    XTEMP3    SAVE X
0190 CE 00 0C  CLRBG2  LDX    #DATAST   SET START
0193 20 08             BRA    CLEAR     GO CLEAR

0195 FE 01 0F  CLREND  LDX    MEMEND    SET END
0198 DF 0A             STX    XTEMP3    SAVE
```

```
019A FE 0D 4D           LDX    ENDSTR
019D 4F         CLEAR   CLR A             CLEAR ACC.
019E A7 00      CLEAR2  STA A  0,X        CLEAR BYTE
01A0 08                 INX               BUMP THE POINTER
01A1 9C 0A              CPX    XTEMP3     DONE?
01A3 26 F9              BNE    CLEAR2
01A5 39                 RTS               RETURN

01A6 8D E3      MICBAS  BSR    CLRBEG     GO CLEAR
01A8 CE 0D 4F           LDX    #STORSP
01AB FF 0D 4D           STX    ENDSTR     SET END STORAGE:
01AE 8D E5              BSR    CLREND     GO CLEAR

                * GET LINE INTO INPUT BUFFER

01B0 CE 01 03   FILBUF  LDX    #RESTRT
01B3 FF A0 48           STX    MONPC      SET UP RETURN POINTER
01B6 8E A0 7F           LDS    #STACK
01B9 CE 00 68           LDX    #BUFFER
01BC DF 0A              STX    XTEMP3     SAVE BOUND
01BE 8D D0              BSR    CLRBG2
01C0 CE 0D 4D           LDX    #ENDSTR    SET PUHCH LIMITS
01C3 FF A0 02           STX    PFILBG
01C6 EE 00              LDX    0,X        SET END
01C8 FF A0 04           STX    PFILEN
01CB DF 08              STX    DIMPNT
01CD CE 00 68           LDX    #BUFFER    POINT TO BUFFER
01D0 BD 02 EA           JSR    PCRLF      OUT A CR & LF
01D3 86 21              LDA A  #PRMPTC
01D5 BD 04 4C           JSR    OUTCH      OUTPUT PROMPT
01D8 BD 02 D0   FILBU2  JSR    INCHAR     GET A CHARACTER
01DB 27 D3              BEQ    FILBUF
01DD A7 00              STA A  0,X        SAVE CHAR.
01DF 81 0D              CMP A  #$0D       IS IT A C.R. ?
01E1 27 08              BEQ    FILBU6
01E3 08                 INX               BUMP THE POINTER
01E4 8C 00 B0           CPX    #BUFFER+72
01E7 26 EF              BNE    FILBU2     END OF BUFFER?
01E9 20 C5              BRA    FILBUF
01EB CE 00 68   FILBU6  LDX    #BUFFER    RESET POINTER
01EE BD 03 31           JSR    BCDCO1     LINE NO. CONV.
01F1 DF 31              STX    XTEMP2     SAVE POINTER
01F3 BD 03 7B           JSR    FNDKEY     CHECK KEY WORD
01F6 4D                 TST A
01F7 26 1A              BNE    FILBU8     IF NONZERO THEN OK
01F9 DE 04              LDX    BUFPNT     POINT TO BUFFER
01FB A6 00              LDA A  0,X        GET CHARACTER
01FD 81 0D              CMP A  #$D        IS IT A C.R.?
01FF 26 08              BNE    FILBU7
0201 D6 28              LDA B  NOEXFL     DIR. EXECUTION?
0203 27 AB              BEQ    FILBUF
0205 97 12              STA A  CRFLAG     SET FLAG
0207 20 0A              BRA    FILBU8     IT IS OK
0209 BD 07 45   FILBU7  JSR    TSTLET     LET?
020C 27 05              BEQ    FILBU8
020E 86 10      FILB75  LDA A  #$10
0210 7E 04 61           JMP    MISTAK     REPORT ERROR #0
0213 96 3E      FILBU8  LDA A  CHRCNT     GET CHAR. COUNT
0215 90 26              SUB A  NUMCNT     SUB LINE # DIGITS
0217 97 3E              STA A  CHRCNT     SAVE
0219 D6 28              LDA B  NOEXFL     DIRECT EXECUTE ?
021B 26 06              BNE    STUFLN     IF NOT GO PUT LINE
021D BD 02 EA           JSR    PCRLF      OUTPUT C.R. L.F.
0220 7E 07 41           JMP    RUNEX4     GO TO ROUTINE

                * PUT LINE IN PROGRAM STORAGE

0223 FE 01 0F   STUFLN  LDX    MEMEND
```

```
0226 DF 37              STX    CPX1
0228 DE 31              LDX    XTEMP2    SET POINTER
022A DF 04              STX    BUFPNT    SAVE POINTER
022C BD 02 A5           JSR    FNDLIN    GO FIND LINE IN STORE
022F DF 22              STX    XSAVE     SAVE POINTER
0231 5D                 TST B            DID WE FIND IT?
0232 26 20              BNE    INSERT    IF NOT GO INSERT


               * REPLACE EXISTING LINE WITH NEW ONE

0234 5C        REPLAC   INC B            INC THE COUNTER
0235 A6 00              LDA A  0,X       GET A CHARACTER
0237 08                 INX              BUMP THE POINTER
0238 81 0D              CMP A  #$D       IS IT A C.R,?
023A 26 F8              BNE    REPLAC
023C F7 02 4C  REPLA4   STA B  OFSET2+1  SETUP OFFSET
023F 86 FF              LDA A  #$FF      GET COUNT
0241 50                 NEG B            2'S COMP. IT
0242 8D 46              BSR    ADJEND    GO FIX END PNTR
0244 DE 22              LDX    XSAVE     RESTORE THE POINTER
0246 BC 0D 4D  REPLA5   CPX    ENDSTR    END OF STORAGE?
0249 27 07              BEQ    REPLA6
024B A6 00     OFSET2   LDA A  0,X
024D A7 00              STA A  0,X       MOVE A CHARACTER
024F 08                 INX              BUMP THE POINTER
0250 20 F4              BRA    REPLA5    REPEAT
0252 DE 22     REPLA6   LDX    XSAVE     RESTORE THE POINTER

               * INSERT A LINE INTO PROGRAM STORAGE

0254 96 12     INSERT   LDA A  CRFLAG    LONE C.R. ?
0256 26 2F              BNE    INSER6
0258 FE 0D 4D           LDX    ENDSTR
025B D6 3E              LDA B  CHRCNT    GET CHAR. COUNT
025D CB 02              ADD B  #2        BIAS FOR LINE NUM.
025F F7 02 6C           STA B  OFFSET+1  SETUP OFFSET
0262 8D 26              BSR    ADJEND    FIX END PNTR
0264 9C 22     INSER2   CPX    XSAVE     DONE?
0266 27 07              BEQ    INSER3
0268 09                 DEX              DEC THE POINTER
0269 A6 00              LDA A  0,X       GET A CHAR,
026B A7 00     OFFSET   STA A  0,X
026D 20 F5              BRA    INSER2    MOVE IT
026F 09        INSER3   DEX
0270 BD 06 68           JSR    PUTLB2    PUT LAB
0273 08                 INX              BUMP THE POINTER
0274 08                 INX
0275 DF 22     INSER4   STX    XSAVE     SAVE POINTER
0277 DE 04              LDX    BUFPNT
0279 A6 00              LDA A  0,X       GET CHAR*
027B 08                 INX              BUMP THE POINTER
027C DF 04              STX    BUFPNT    SAVE
027E DE 22              LDX    XSAVE     RESTOR PNTR
0280 08                 INX
0281 A7 00              STA A  0,X       SAVE IT
0283 81 0D              CMP A  #$D       IS IT A C.R.?
0285 26 EE              BNE    INSER4
0287 7E 01 B0  INSER6   JMP    FILBUF    60 TO MAIN LOOP

               * ADJUST THE END OF PROGRAM POINTER

028A FB 0D 4E  ADJEND   ADD B  ENDSTR+1
028D B9 0D 4D           ADC A  ENDSTR    ADD IN VALUE
0290 D7 3A              STA B  CPX2+1
0292 97 39              STA A  CPX2      SET END POINTER
0294 BD 0C B3           JSR    CMPX1
0297 24 07              BCC    ADJEN2
```

```
0299 F7 0D 4E          STA B  ENDSTR+1
029C B7 0D 4D          STA A  ENDSTR    SAVE NEW POINTER
029F 39                RTS              RETURN
02A0 86 90     ADJEN2  LDA A  #$90      SET ERROR
02A2 7E 04 61          JMP    MISTAK

               * TRY TO FIND LINE

02A5 96 64     FNDLIN  LDA A  NUMBER+2
02A7 D6 63             LDA B  NUMBER+1
02A9 CE 0D 4F  FINDLN  LDX    #STORSP   SETUP POINTER
02AC BC 0D 4D  FINDL1  CPX    ENDSTR    END OF STORAGE?
02AF 26 02             BNE    FINDL4
02B1 5C        FINDL2  INC B
02B2 39                RTS              RETURN
02B3 E1 00     FINDL4  CMP B  0,X       CHECK M.S. DIGITS
02B5 22 0A             BHI    FINDL6
02B7 26 F8             BNE    FINDL2
02B9 A1 01             CMP A  1,X       CHECK L.S, DIGITS
02BB 22 04             BHI    FINDL6
02BD 26 F2             BNE    FINDL2
02BF 5F                CLR B            CEAR FLAG
02C0 39                RTS              RETURN
02C1 8D 03     FINDL6  BSR    FNDCRT    GO FIND C.R,
02C3 08                INX              BUMP THE POINTER
02C4 20 E6             BRA    FINDL1    REPEAT

               * FIND A C,R, IN STORAGE

02C6 36        FNDCRT  PSH A            SAVE A
02C7 86 0D             LDA A  #$D
02C9 08        FNDVAL  INX              BUMP THE POINTER
02CA A1 00             CMP A  0,X       TEST FOR C.R.
02CC 26 FB             BNE    FNDVAL
02CE 32                PUL A            RESTORE A
02CF 39                RTS              RETURN

               * INPUT

02D0 BD 01 09  INCHAR  JSR    INCH      GET THE CHAR.
02D3 81 08             CMP A  #BACKSP   IS IT A BACKSPACE?
02D5 26 0B             BNE    INCHR2
02D7 8C 00 68          CPX    #BUFFER   BEGINNING OF BUF ?
02DA 27 0D             BEQ    INCHR4
02DC 09                DEX              BACKUP ONE POS.
02DD 7A 00 3E          DEC    CHRCNT    DEC CHAR. COUNT
02E0 20 EE             BRA    INCHAR
02E2 81 18     INCHR2  CMP A  #DELCOD   DELETE LINE ?
02E4 27 03             BEQ    INCHR4
02E6 7C 00 3E          INC    CHRCNT
02E9 39        INCHR4  RTS              RETURN

               * PRINT CARRIAGE RETURN & LINEFEED

02EA DF 22     PCRLF   STX    XSAVE     SAVE X REG
02EC CE 03 01          LDX    #CRLFST   POINT TO STRING
02EF A6 00     PDATA1  LDA A  0,X       GET CHAR
02F1 81 04             CMP A  #4        IS IT 4?
02F3 27 06             BEQ    PCRLF2
02F5 BD 04 4C          JSR    OUTCH     OUTPUT CHAR
02F8 08                INX              BUMP THE POINTER
02F9 20 F4             BRA    PDATA1    REPEAT
02FB DE 22     PCRLF2  LDX    XSAVE     RESTORE X REG
02FD 7F 00 1D          CLR    FLDCNT    ZERO FIELD COUNT
0300 39                RTS              RETURN

0301 0D        CRLFST  FCB    $D,$A,0,0,0,0,4
0302 0A 00
```

```
0304 00 00
0306 00 04

               * TEST FOR STATEMENT TERMINATOR

0308 81 0D     TSTTRM  CMP A  #$D       C,R,?
030A 27 02             BEQ    TSTTR2
030C 81 3A             CMP A  #':       COLON?
030E 39        TSTTR2  RTS              RETURN

               * CLEAR NUMBER THROUGH NUMBER+2

030F BD 0B 51  UPSCLR  JSR    STAKUP
0312 4F        CLRNUM  CLR A
0313 97 62             STA A  NUMBER
0315 97 63             STA A  NUMBER+1
0317 97 64             STA A  NUMBER+2
0319 39                RTS

               * CONVERT NUMBER TO PACKED BCD

031A 8D F6     BCDCON  BSR    CLRNUM    CLEAR NUMBER
031C 97 28             STA A  NOEXFL
031E 97 27             STA A  NEGFLG
0320 97 26             STA A  NUMCNT
0322 BD 03 68          JSR    SKIPSP    SKIP SPACES
0325 81 2B             CMP A  #'+       IS IT A +?
0327 27 07             BEQ    BCDC01
0329 81 2D             CMP A  #'-       IS IT A - ?
032B 26 04             BNE    BCDCO1
032D 73 00 27          COM    NEGFLG    SET FLAG
0330 08        BCDC01  INX
0331 BD 0C E3  BCDCO1  JSR    CLASS     GET A DIGIT
0334 C1 03             CMP B  #3        IS IT A NUMBER?
0336 27 05             BEQ    BCDCO2
0338 96 27             LDA A  NEGFLG
033A 7E 0B EA          JMP    FIXSIN    GO FIX UP THE SIGN
033D 08        BCDCO2  INX              BUMP THE POINTER
033E 97 28             STA A  NOEXFL    SET NO EXEC FLU
0340 84 0F             AND A  #$0F      MASK OFF ASCII
0342 C6 04             LDA B  #4        SET COUNTER
0344 78 00 64  BCDCO4  ASL    NUMBER+2
0347 79 00 63          ROL    NUMBER+1
034A 79 00 62          ROL    NUMBER    SHIFT PREV. OVER
034D 5A                DEC B            DEC THE COUNTER
034E 26 F4             BNE    BCDCO4
0350 9B 64             ADD A  NUMBER+2
0352 97 64             STA A  NUMBER+2  SAVE NEW VALUE
0354 7C 00 26          INC    NUMCNT    INC NUMBER CNTR
0357 20 D8             BRA    BCDCO1

               * FIND NEXT BLOCK

0359 DE 04     NXTBLK  LDX    BUFPNT    RESTORE POINTER
035B A6 00     NXTBL4  LDA A  0,X       GET A CHAR.
035D 81 20             CMP A  #'        IS IT A SPACE?
035F 27 07             BEQ    SKIPSP
0361 08                INX              BUMP THE POINTER
0362 20 F7             BRA    NXTBL4    REPEAT

               * CONVERT AND SKIP

0364 8D B4     CONSKP  BSR    BCDCON
0366 09                DEX

               * SKIP ALL SPACES

0367 08        SKPSP0  INX
```

```
0368 A6 00      SKIPSP  LDA A  0,X        GET CHR FROM BUF
036A 81 20              CMP A  #$20       IS IT A SPACE?
036C 27 F9              BEQ    SKPSP0
036E 39         SKIPS4  RTS               RETURN

                * FIND NEXT BLOCK NOT EXPECTING A SPACE

036F DE 04      NXTSPC  LDX    BUFPNT     SET POINTER
0371 BD 0C E3   NXTSP4  JSR    CLASS      GO CLASSIFY
0374 C1 02              CMP B  #2         IS IT A LETTER?
0376 26 F0              BNE    SKIPSP
0378 08                 INX               BUMP THE POINTER
0379 20 F6              BRA    NXTSP4

                * FIND KEY WORD IF POSSIBLE

037B BD 03 68   FNDKEY  JSR    SKIPSP     SKIP SPACES
037E DF 04              STX    BUFPNT     SAVE THE POINTER
0380 DF 22              STX    XSAVE
0382 CE 01 11           LDX    #KEYTBL    POINT TO KEY WORDS
0385 C6 05      FNDKE2  LDA B  #5
0387 A1 00      FNDKE4  CMP A  0,X        TEST THE CHARACTER
0389 26 12              BNE    FNDKE6
038B DF 0A              STX    XTEMP3     SAVE POINTER
038D DE 22              LDX    XSAVE
038F 08                 INX               BUMP POINTER
0390 A6 00              LDA A  0,X        GET CHAR.
0392 DF 22              STX    XSAVE
0394 DE 0A              LDX    XTEMP3     REST. PNTR.
0396 08                 INX
0397 5A                 DEC B
0398 C1 02              CMP B  #2
039A 26 EB              BNE    FNDKE4     IF NOT DONE REPEAT
039C 39         FNDKE5  RTS               RETURN
039D 08         FNDKE6  INX               BUMP THE COUNTER
039E 5A                 DEC B
039F 26 FC              BNE    FNDKE6
03A1 A6 00              LDA A  0,X        GET CHARACTER
03A3 27 F7              BEQ    FNDKE5     IF ZERO, END OF LIST
03A5 DF 0A              STX    XTEMP3     SAVE POINTER
03A7 DE 04              LDX    BUFPNT
03A9 DF 22              STX    XSAVE
03AB A6 00              LDA A  0,X        GET NEW CHAR.
03AD DE 0A              LDX    XTEMP3     RESTORE POINTER
03AF 20 D4              BRA    FNDKE2     REPEAT


                * OUTPUT A NUMBER FROM PACKED BCD BYTES

03B1 CE 00 62   OUTBCD  LDX    #NUMBER    SET POINTER
03B4 C6 02      OUTBCI  LDA B  #2         SET COUNTER
03B6 0C                 CLC
03B7 A6 00              LDA A  0,X        GET A WORD
03B9 2A 19              BPL    OUTBC4     IF NOT NEG JMP AHEAD
03BB 86 2D              LDA A  #'-
03BD BD 04 4C           JSR    OUTCH      OUTPUT A
03C0 7C 00 1D           INC    FLDCNT
03C3 20 0F              BRA    OUTBC4
03C5 A6 00      OUTBC2  LDA A  0,X        GET DIGITS
03C7 85 F0              BIT A  #$F0       MASK
03C9 25 02              BCS    OUTBC3
03CB 27 07              BEQ    OUTBC4     JMP IF ZEROES
03CD BD 04 44   OUTBC3  JSR    OUTHL      OUTPUT A DIGIT
03D0 7C 00 1D           INC    FLDCNT
03D3 0D                 SEC
03D4 A6 00      OUTBC4  LDA A  0,X        GET A DIGIT
03D6 C5 FF              BIT B  #$FF       LAST DIGIT?
03D8 27 06              BEQ    OUTBC6
```

```
03DA 85 0F              BIT A  #$0F      MASK
03DC 25 02              BCS    OUTBC6
03DE 27 07              BEQ    OUTBC8    JMP IF ZEROES
03E0 BD 04 48  OUTBC6   JSR    OUTHR     OUTPUT A DIGIT
03E3 7C 00 1D           INC    FLDCNT
03E6 0D                 SEC
03E7 08        OUTBC8   INX              BUMP THE POINTER
03E8 5A                 DEC B            DEC THE COUNTER
03E9 2A DA              BPL    OUTBC2    REPEAT IF NOT DONE
03EB 39                 RTS              RETURN

               * LIST USERS PROGRAM

03EC BD 03 6F  LIST     JSR    NXTSPC    FIND NEXT
03EF 81 0D              CMP A  #$D
03F1 27 25              BEQ    LIST3
03F3 BD 03 1A           JSR    BCDCON    GET LINE NUM
03F6 DF 04              STX    BUFPNT    SAVE POINTER
03F8 BD 02 A5           JSR    FNDLIN    FIND LINE
03FB DF 22              STX    XSAVE     SAVE IT
03FD BD 03 6F           JSR    NXTSPC
0400 81 0D              CMP A  #$D       C.R.?
0402 26 05              BNE    LIST1
0404 7C 00 1B           INC    SUBCNT    SET TO 1
0407 20 0B              BRA    LIST2
0409 08        LIST1    INX              BUMP THE POINTER
040A BD 03 68           JSR    SKIPSP
040D BD 03 1A           JSR    BCDCON    GET COUNT
0410 96 64              LDA A  NUMBER+2
0412 97 1B              STA A  SUBCNT    SAVE IT
0414 DE 22     LIST2    LDX    XSAVE     POINT TO LINE
0416 20 03              BRA    LIST4
0418 CE 0D 4F  LIST3    LDX    #STORSP   SET POINTER
041B BC 0D 4D  LIST4    CPX    ENDSTR    END OF STORAGE?
041E 27 21              BEQ    LIST8
0420 BD 02 EA           JSR    PCRLF     OUTPUT A
0423 C6 01              LDA B  #1        SETUP COUNTER
0425 0C                 CLC
0426 8D 9D              BSR    OUTBC2    OUT LINE NUMBER
0428 A6 00     LIST5    LDA A  0,X       GET A CHARACTER
042A 81 0D              CMP A  #$D       IS IT A C.R.?
042C 27 05              BEQ    LIST6
042E 8D 1C              BSR    OUTCH     OUTPUT CHARACTER
0430 08                 INX              BUMP THE POINTER
0431 20 F5              BRA    LIST5     REPEAT
0433 08        LIST6    INX              BUMP THE POINTER
0434 96 1B              LDA A  SUBCNT    GET COUNT
0436 27 E3              BEQ    LIST4
0438 8B 99              ADD A  #$99      DEC THE COUNT
043A 19                 DAA
043B 27 04              BEQ    LIST8
043D 97 1B              STA A  SUBCNT    SAVE
043F 20 DA              BRA    LIST4
0441 7E 01 B0  LIST8    JMP    FILBUF

0444 44        OUTHL    LSR A
0445 44                 LSR A
0446 44                 LSR A
0447 44                 LSR A            MOVE TO BOTTOM
0448 84 0F     OUTHR    AND A  #$0F      MASK
044A 8B 30              ADD A  #$30      BIAS
044C BD 01 0C  OUTCH    JSR    BREAK     CHECK FOR BREAK
044F 7E 01 06           JMP    OUTEEE    GO PRINT

               * INTERNAL BREAK ROUTINE

0452 36        INTBRK   PSH A
0453 B6 80 04           LDA A  PIAADR    CHECK
```

```
0456 2A 02              BPL    BREAK2
0458 32                 PUL A             GET CHAR
0459 39                 RTS               RETURN
045A B6 80 04  BREAK2   LDA A  PIAADR
045D 2A FB              BPL    BREAK2
045F 86 99              LDA A  #$99       SET ERROR

               * OUTPUT ERROR MESSAGE

0461 36        MISTAK   PSH A             SAVE A
0462 BD 02 EA           JSR    PCRLF      OUTPUT A CR & LF
0465 CE 04 98  MISTA1   LDX    #ERRSTR    POINT TO ERROR STRING
0468 BD 02 EF           JSR    PDATA1     OUTPUT IT
046B 32                 PUL A             RESTORE A
046C 36                 PSH A             SAVE A
046D BD 04 44           JSR    OUTHL      OUTPUT DIGIT
0470 32        MISTA2   PUL A             RESTORE A
0471 BD 04 48           JSR    OUTHR      OUT 1'S DIGIT
0474 D6 19              LDA B  RUNFLG     RUNNING?
0476 26 03              BNE    RUNER1
0478 7E 01 B0  MISTA4   JMP    FILBUF
047B CE 04 A1  RUNER1   LDX    #ERSTR2    POINT TO STRING
047E BD 02 EF           JSR    PDATA1     OUTPUT IT
0481 DE 04              LDX    BUFPNT     SET POINTER
0483 09        RUNER2   DEX               DEC THE POINTER
0484 8C 0D 4F           CPX    #STORSP    BEGINNING?
0487 27 07              BEQ    RUNER4
0489 A6 00              LDA A  0,X        GET CHAR
048B 81 0D              CMP A  #$D        C.R.?
048D 26 F4              BNE    RUNER2
048F 08                 INX               BUMP THE POINTER
0490 C6 01     RUNER4   LDA B  #1
0492 0C                 CLC
0493 BD 03 C5           JSR    OUTBC2     OUT LINE NUM.
0496 20 E0              BRA    MISTA4
0498 07        ERRSTR   FCB    7
0499 45                 FCC    ;ERROR #;
049A 52 52
049C 4F 52
049E 20 23
04A0 04                 FCB    4

04A1 20        ERSTR2   FCC    ; AT ;
04A2 41 54
04A4 20
04A5 04                 FCB    4

               * PRINT ROUTINE

04A6 BD 03 6F  PRINT    JSR    NXTSPC     FIND NEXT BLOCK
04A9 BD 03 08  PRINT0   JSR    TSTTRM
04AC 26 03              BNE    FIELD1
04AE 7E 05 3C           JMP    PRINT8
04B1 7F 00 12  FIELD1   CLR    CRFLAG
04B4 81 2C              CMP A  #',        IS IT A ","
04B6 26 20              BNE    PRINT2
04B8 D6 1D              LDA B  FLDCNT     GET COUNT
04BA 86 20     FIELD2   LDA A  #'         SPACE
04BC BD 04 4C           JSR    OUTCH      OUTPUT A SPACE
04BF 5C                 INC B
04C0 C5 07              BIT B  #7         END OF FIELD?
04C2 26 F6              BNE    FIELD2
04C4 C1 47              CMP B  #$47       END OF LINE?
04C6 22 04              BHI    FIELD3
04C8 D7 1D              STA B  FLDCNT     SAVE FIELD INFO
04CA 20 03              BRA    PRINT1
04CC BD 02 EA  FIELD3   JSR    PCRLF      OUT A C.R. & L.F.
04CF 7C 00 12  PRINT1   INC    CRFLAG     SET FLAG
```

```
04D2 08                 INX              BUMP THE POINTER
04D3 BD 03 68           JSR    SKIPSP
04D6 20 D1              BRA    PRINT0
04D8 81 3B      PRINT2  CMP A  #';       IS IT A ";"
04DA 27 F3              BEQ    PRINT1
04DC 81 22              CMP A  #'"       IS IT A QUOTE?
04DE 26 05              BNE    PRINT4
04E0 08                 INX              BUMP THE POINTER
04E1 8D 64              BSR    PSTRNG    OUTPUT STRING
04E3 20 49              BRA    PRINT6
04E5 7F 00 17   PRINT4  CLR    TABFLG    CLEAR FLAG
04E8 81 54              CMP A  #'T       IS IT A T?
04EA 26 06              BNE    PRIN45
04EC 97 17              STA A  TABFLG    SET FLAG
04EE 86 41              LDA A  #'A
04F0 20 06              BRA    PRIN47
04F2 81 53      PRIN45  CMP A  #'S       IS IT A S?
04F4 26 2E              BNE    PRIN55
04F6 86 50              LDA A  #'P
04F8 A1 01      PRIN47  CMP A  1,X
04FA 26 28              BNE    PRIN55
04FC BD 03 71           JSR    NXTSP4    FIND NEXT
04FF BD 0A 26           JSR    EXPR      EVALUATE
0502 BD 06 1E           JSR    BINCON    CONVERT
0505 D6 64              LDA B  NUMBER+2
0507 27 25              BEQ    PRINT6
0509 96 17              LDA A  TABFLG    CHECK FLAG
050B 27 07              BEQ    PRINT5
050D 5A                 DEC B
050E D1 1D              CMP B  FLDCNT    CHECK COUNT
0510 23 1C              BLS    PRINT6
0512 20 02              BRA    PRIN51
0514 DB 1D      PRINT5  ADD B  FLDCNT
0516 86 20      PRIN51  LDA A  #'        SPACE
0518 BD 04 4C           JSR    OUTCH     OUTPUT SPACE
051B 7C 00 1D           INC    FLDCNT    BUMP COUNTER
051E D1 1D              CMP B  FLDCNT
0520 26 F4              BNE    PRIN51    REPEAT
0522 20 0A      PRIN52  BRA    PRINT6
0524 BD 0A 26   PRIN55  JSR    EXPR      EVAL EXPRESSION
0527 DF 22              STX    XSAVE     SAVE POINTER
0529 BD 03 B1           JSR    OUTBCD    OUTPUT VALUE
052C DE 22              LDX    XSAVE     RESTORE
052E BD 0C DE   PRINT6  JSR    SKYCLS
0531 5A                 DEC B
0532 26 03              BNE    PRINT7    CHECK FOR ERROR
0534 7E 04 A9           JMP    PRINT0
0537 86 31      PRINT7  LDA A  #$31
0539 7E 04 61           JMP    MISTAK
053C 7D 00 12   PRINT8  TST    CRFLAG    C.R. ?
053F 26 03              BNE    PRINT9
0541 BD 02 EA           JSR    PCRLF     OUTPUT C.R. L.F
0544 7E 07 04   PRINT9  JMP    RUNEXC

                * PRINT STRING ROUTINE

0547 A6 00      PSTRNG  LDA A  0,X       GET A CHAR.
0549 81 22              CMP A  #'"       IS I T A QUOTE?
054B 27 0E              BEQ    PSTRN4
054D BD 03 08           JSR    TSTTRM    IS IT A C.R.?
0550 27 0D              BEQ    PSTRN8
0552 BD 04 4C           JSR    OUTCH     OUTPUT CHARACTER
0555 7C 00 1D           INC    FLDCNT    BUMP FIELD CNT
0558 08                 INX              BUMP THE POINTER
0559 20 EC              BRA    PSTRNG    REPEAT
055B 08         PSTRN4  INX
055C 7E 03 68           JMP    SKIPSP
055F 86 32      PSTRN8  LDA A  #$32
```

```
0561 7E 04 61          JMP    MISTAK    REPORT ERROR

               * FIND LABLE ROUTINE

0564 DF 04     FNDVAR  STX    BUFPNT    SAVE POINTER
0566 BD 0C E5          JSR    CLASS1    GO CLASSIFY CHAR.
0569 C1 02             CMP B  #2        CHECK FOR LETTER
056B 26 2F             BNE    FNDL25    ERROR
056D 7F 00 20          CLR    XTEMP
0570 16                TAB              SAVE LABLE
0571 48                ASL A            MULT IT BY 2
0572 1B                ABA              ADD IT
0573 80 13             SUB A  #$13
0575 97 21             STA A  XTEMP+1
0577 DE 20             LDX    XTEMP     POINT TO IT
0579 39                RTS              RETURN

               * FIND DIMENSIONED VARIABLE

057A A6 00     FNDLB0  LDA A  0,X
057C 08        FNDLBL  INX              ADVANCE POINTER
057D 7F 00 18          CLR    DIMFLG
0580 8D E2             BSR    FNDVAR    GO FIND VAR.
0582 5F                CLR B
0583 A6 00             LDA A  0,X       GET CHAR.
0585 81 0A             CMP A  #$0A      CHECK FOR 1 DIM
0587 27 06             BEQ    FNDLB2
0589 81 0B             CMP A  #$0B      CHECK IF 2 DIM
058B 27 01             BEQ    FNDLB1
058D 39                RTS
058E 5C        FNDLB1  INC B            SET FLAG-2 DIM
058F A6 01     FNDLB2  LDA A  1,X       SET POINTER
0591 36                PSH A
0592 A6 02             LDA A  2,X
0594 36                PSH A
0595 37                PSH B            SAVE B
0596 BD 03 6F          JSR    NXTSPC    FIND NEXT
0599 33                PUL B
059A 81 28             CMP A  #'(       IS IT A PAREN?
059C 26 71     FNDL25  BNE    FNDLB9
059E 5D                TST B
059F 27 13             BEQ    FNDLB3
05A1 08                INX
05A2 BD 0A 29          JSR    EXPRO     GO EVALUATE
05A5 96 64             LDA A  NUMBER+2  GET RESULT
05A7 36                PSH A            SAVE IT
05A8 BD 0B 62          JSR    STAKDN    RESTORE
05AB BD 03 6F          JSR    NXTSPC    FIND NEXT
05AE 81 2C             CMP A  #',       IS IT A COMMA?
05B0 26 5D             BNE    FNDLB9
05B2 20 02             BRA    FNDLB4
05B4 4F        FNDLB3  CLR A
05B5 36                PSH A            SET ROWV
05B6 4C        FNDLB4  INC A
05B7 97 18             STA A  DIMFLG    SET FLAG
05B9 08                INX
05BA BD 0A 29          JSR    EXPRO
05BD 08                INX
05BE DF 04             STX    BUFPNT    SAVE POINTER
05C0 32                PUL A
05C1 97 14             STA A  ROWVAR    SAVE
05C3 32                PUL A
05C4 97 21             STA A  XTEMP+1   SAVE
05C6 32                PUL A
05C7 97 20             STA A  XTEMP     SAVE
05C9 DE 20             LDX    XTEMP     SET POINTER
05CB A6 00             LDA A  0,X       GET CHAR
05CD 97 16             STA A  COLCON    SAVE IT
```

```
05CF 08                INX              BUMP THE POINTER
05D0 08                INX
05D1 DF 20             STX    XTEMP
05D3 BD 03 0F          JSR    UPSCLR
05D6 96 14             LDA A  ROWVAR    GET VAR.
05D8 DE 20             LDX    XTEMP
05DA 09                DEX              DEC POINTER
05DB A1 00             CMP A  0,X       CHECK
05DD 22 30             BHI    FNDLB9
05DF 97 64             STA A  NUMBER+2
05E1 BD 03 0F          JSR    UPSCLR    PUSH STACK
05E4 96 16             LDA A  COLCON    GET CONST,
05E6 91 5E             CMP A  AC-1      CHECK
05E8 27 02             BEQ    FNDL45
05EA 23 23             BLS    FNDLB9    ERROR!
05EC 8B 01     FNDL45  ADD A  #1
05EE 19                DAA              BIAS IT
05EF 97 64             STA A  NUMBER+2
05F1 BD 0B F4          JSR    MULT      GO MULTIPLY
05F4 BD 0B CA          JSR    ADD       GO ADD
05F7 BD 06 14  FNDLB5  JSR    TIMTHR

               * ROUTINE TO ADD VALUE TO X-REG.

05FA 96 20     ADDX    LDA A  XTEMP     GET M.S.BYTE
05FC D6 21             LDA B  XTEMP+1
05FE DB 64             ADD B  NUMBER+2
0600 99 63             ADC A  NUMBER+1
0602 97 20             STA A  XTEMP     SAVE SUM
0604 D7 21             STA B  XTEMP+1
0606 BD 0B 62          JSR    STAKDN
0609 DE 20             LDX    XTEMP     SET POINTER
060B 7F 00 18          CLR    DIMFLG    RESTORE FLAG
060E 39                RTS              RETURN

060F 86 14     FNDLB9  LDA A  #$14      SET ERROR
0611 7E 04 61          JMP    MISTAK    GO REPORT

               * ROUTINE TO MULTIPLY BY 3

0614 BD 03 0F  TIMTHR  JSR    UPSCLR
0617 86 03             LDA A  #$3       SET MULTIPLIER
0619 97 64             STA A  NUMBER+2
061B BD 0B F4          JSR    MULT      GO MULTIPLY

               * BCD TO BINARY CONVERT.

061E 96 64     BINCON  LDA A  NUMBER+2  GET LS BYTE
0620 36                PSH A            SAVE
0621 96 63             LDA A  NUMBER+1
0623 36                PSH A            SAVE:
0624 5F                CLR B
0625 D7 63             STA B  NUMBER+1
0627 D7 64             STA B  NUMBER+2  INITIALIZE
0629 96 62             LDA A  NUMBER
062B 8D 12             BSR    ADSHF1    ADD AND SHIFT
062D 32                PUL A
062E 36                PSH A
062F 8D 0A             BSR    ADSHF0    GO ADD IN AND SHIFT
0631 32                PUL A            GET MS BYTE AGAIN
0632 8D 0B             BSR    ADSHF1    GO ADD IN AND SHIFT
0634 32                PUL A            GET LS BYTE
0635 36                PSH A
0636 8D 03             BSR    ADSHF0
0638 32                PUL A
0639 20 1D             BRA    ADDIN     G0 ADD IN ONES
063B 44        ADSHF0  LSR A
063C 44                LSR A
```

```
063D 44                 LSR A
063E 44                 LSR A             MOVE TO LS HALF
063F 8D 17     ADSHF1   BSR    ADDIN      GO ADD IN
0641 D6 63              LDA B  NUMBER+1
0643 48                 ASL A
0644 59                 ROL B             MULT BY 2
0645 37                 PSH B
0646 36                 PSH A             SAVE
0647 48                 ASL A
0648 59                 ROL B
0649 48                 ASL A
064A 59                 ROL B             MULT BY 4, =*8
064B 97 64              STA A  NUMBER+2
064D 32                 PUL A
064E D7 63              STA B  NUMBER+1
0650 8D 08              BSR    ADDIN1     GO ADD IN
0652 32                 PUL A
0653 9B 63              ADD A  NUMBER+1
0655 97 63              STA A  NUMBER+1   MULTIPLY BY TEN
0657 39                 RTS
0658 84 0F     ADDIN    AND A  #$0F       MASK
065A 9B 64     ADDIN1   ADD A  NUMBER+2
065C 97 64              STA A  NUMBER+2
065E 24 03              BCC    ADDIN2     CHECK FOR CARRY
0660 7C 00 63           INC    NUMBER+1
0663 39        ADDIN2   RTS

               * PUT LABLE ROUTINE

0664 96 62     PUTLBL   LDA A  NUMBER
0666 A7 00              STA A  0,X        PUT M.S. BYTE
0668 96 62     PUTLB2   LDA A  NUMBER     +1
066A A7 01              STA A  1,X        PUT NEXT
066C 96 64              LDA A  NUMBER+2
066E A7 02              STA A  2,X        PUT L.S. BYTE
0670 39                 RTS               RETURN

               * DIMENSION

0671 DE 06     DIM      LDX    FORSTK     SET BOUNDS
0673 DF 37              STX    CPX1
0675 BD 03 6F           JSR    NXTSPC
0678 BD 03 68  DIMN     JSR    SKIPSP     CLASSIFY
067B BD 05 64           JSR    FNDVAR
067E DF 0A              STX    XTEMP3     SAVE IT
0680 BD 03 6F           JSR    NXTSPC     GET TO NEXT
0683 81 28              CMP A  #'(        IS IT A PARENT
0685 26 20              BNE    DIM9
0687 08        DIM01    INX               BUMP THE POINTER
0688 BD 03 64           JSR    CONSKP     CONVERT DIM
068B 81 29              CMP A  #')        IS IT A PAREN
068D 26 05              BNE    DIM1
068F 4F                 CLR A
0690 5F                 CLR B
0691 36                 PSH A             SAVE IT
0692 20 18              BRA    DIM2
0694 81 2C     DIM1     CMP A  #',        COMMA?
0696 26 0F              BNE    DIM9       ERROR!
0698 96 64              LDA A  NUMBER+2
069A 27 0B              BEQ    DIM9
069C 36                 PSH A             SAVE
069D 08                 INX               BUMP THE POINTER
069E BD 03 64           JSR    CONSKP     CONVERT
06A1 C6 01              LDA B  #1
06A3 81 29              CMP A  #')        PAREN?
06A5 27 05              BEQ    DIM2
06A7 86 40     DIM9     LDA A  #$40       SET ERROR
06A9 7E 04 61           JMP    MISTAK     REPORT
```

```
06AC 96 64     DIM2    LDA A  NUMBER+2
06AE 27 F7             BEQ    DIM9
06B0 36                PSH A               SAVE
06B1 DF 04             STX    BUFPNT       SAVE POINTER
06B3 DE 0A             LDX    XTEMP3       SET X
06B5 86 0A             LDA A  #$0A
06B7 1B                ABA                 SET MARKER
06B8 A7 00             STA A  0,X          SAVE IT
06BA 96 08             LDA A  DIMPNT       GET POINTER
06BC A7 01             STA A  1,X          SAVE IT
06BE 96 09             LDA A  DIMPNT+1
06C0 A7 02             STA A  2,X
06C2 DE 08             LDX    DIMPNT       SET POINTER
06C4 32                PUL A
06C5 A7 00             STA A  0,X          SAVE 1ST DIM
06C7 08                INX                 BUMP THE POINTER
06C8 33                PUL B
06C9 E7 00             STA B  0,X          SAVE 2ND DIM
06CB 08                INX
06CC DF 20             STX    XTEMP        SAVE POINTER
06CE 8B 01             ADD A  #1
06D0 19                DAA                 BIAS
06D1 36                PSH A
06D2 17                TBA
06D3 8B 01             ADD A  #1           BIAS
06D5 19                DAA                 ADJUST
06D6 16                TAB                 SAVE
06D7 BD 03 12          JSR    CLRNUM       CLEAR STORAGE
06DA D7 64             STA B  NUMBER+2
06DC BD 03 0F          JSR    UPSCLR       GO CLEAR
06DF 32                PUL A
06E0 97 64             STA A  NUMBER+2
06E2 BD 0B F4          JSR    MULT         MULTIPLY
06E5 BD 05 F7          JSR    FNDLB5       GO FIX X
06E8 BD 0C B1          JSR    CMPX         TEST BOUNDS
06EB 23 03             BLS    DIM5
06ED 7E 02 A0          JMP    ADJEN2
06F0 DF 08     DIM5    STX    DIMPNT       SAVE RESULT
06F2 DE 04             LDX    BUFPNT       RESTORE F'NTR
06F4 08                INX
06F5 BD 03 68          JSR    SKIPSP       SKIP SPACES
06F8 BD 03 08          JSR    TSTTRM
06FB 27 07             BEQ    RUNEXC
06FD 08                INX                 BUMP THE POINTER
06FE 7E 06 78          JMP    DIMN

               * EXTERNAL ROUTINE JUMP

0701 BD 1F 00  EXTRNL  JSR    EXTERN       GO TO IT

               * RUN EXECUTIVE

0704 4F        RUNEXC  CLR A
0705 97 12             STA A  CRFLAG
0707 97 1C             STA A  LETFLG
0709 97 18             STA A  DIMFLG
070B 97 2C             STA A  STKCNT
070D 96 19             LDA A  RUNFLG       RUN MODE?
070F 26 03             BNE    RUNEX0
0711 7E 01 B0  RUNEXA  JMP    FILBUF
0714 DE 04     RUNEX0  LDX    BUFPNT       SET POINTER
0716 86 0D     RUNE05  LDA A  #$D
0718 C6 3A             LDA B  #':          SETUP TERMINATORS
071A A1 00     RUNEX1  CMP A  0,X          C.R. ?
071C 27 07             BEQ    RUNEX2
071E E1 00             CMP B  0,X          IS IT A ':' ?
0720 27 0A             BEQ    RUNE27
0722 08                INX                 BUMP THE POINTER
```

```
0723 20 F5              BRA    RUNEX1    REPEAT
0725 08         RUNEX2  INX
0726 BC 0D 4D  RUNE22  CPX    ENDSTR    END OF STORAGE?
0729 27 E6              BEQ    RUNEXA
072B 08         RUNE25  INX              BUMP THE POINTER
072C 08         RUNE27  INX
072D BD 01 0C           JSR    BREAK     GO CHECK BREAK
0730 BD 03 7B  RUNEX3  JSR    FNDKEY    FIND KEY WORD
0733 4D                 TST A
0734 26 0B              BNE    RUNEX4
0736 DE 04              LDX    BUFPNT    SET POINTER
0738 8D 0B              BSR    TSTLET
073A 27 05              BEQ    RUNEX4
073C 86 10              LDA A  #$10
073E 7E 04 61  RUNE35  JMP    MISTAK
0741 EE 00      RUNEX4  LDX    0,X
0743 6E 00              JMP    0,X       GO TO ROUTINE

                * TEST FOR IMPLIED LET

0745 BD 0C E3  TSTLET  JSR    CLASS     CHECK CHAR.
0748 C1 02              CMP B  #2        LETTER?
074A 26 12              BNE    TSTLE2
074C 08                 INX              BUMP THE POINTER
074D BD 03 68           JSR    SKIPSP    SKIP SPACES
0750 81 3D              CMP A  #'=       EQUALS?
0752 27 04              BEQ    TSTLE1
0754 81 28              CMP A  #'(       LEFT PARENT
0756 26 06              BNE    TSTLE2
0758 CE 01 23  TSTLE1  LDX    #LETADR   SET POINTER
075B 97 1C              STA A  LETFLG    SET FLAG
075D 5F                 CLR B
075E 39         TSTLE2  RTS

                * RUN ROUTINE

075F BD 01 8B  RUN     JSR    CLRBEG
0762 BD 01 95           JSR    CLREND
0765 FE 01 0F           LDX    MEMEND
0768 DF 06              STX    FORSTK
076A CE 0D 4F           LDX    #STORSP   SET POINTER
076D 7C 00 19           INC    RUNFLG
0770 20 B4              BRA    RUNE22

                * LET ROUTINE

0772 DE 04      LET     LDX    BUFPNT
0774 96 1C              LDA A  LETFLG    TEST FLAG
0776 26 03              BNE    LET2
0778 BD 03 59           JSR    NXTBLK    FIND NEXT
077B BD 09 65  LET2    JSR    EXPEQU
077E 7E 07 04           JMP    RUNEXC

                * GOTO ROUTINE

0781 BD 03 6F  GOTO    JSR    NXTSPC    FIND BLOCK
0784 BD 0A 26  GOTO1   JSR    EXPR      GO EVALUATE
0787 BD 02 A5  GOTO2   JSR    FNDLIN    GO FIND LINE
078A 5D         GOTO3   TST B            FIND?
078B 27 05              BEQ    GOTO5
078D 86 16              LDA A  #$16      SET ERROR
078F 7E 04 61  GOTO4   JMP    MISTAK    REPORT
0792 5C         GOTO5   INC B
0793 D7 19              STA B  RUNFLG    SET RUN FLAG
0795 7E 07 26           JMP    RUNE22

                * INPUT ROUTINE
```

```
0798 BD 03 6F  INPUT   JSR    NXTSPC    FIND NEXT
079B 7F 00 13  INPUT0  CLR    QMFLAG    CLEAR FLAG
079E BD 03 68  INPUT1  JSR    SKIPSP    SKIP SPACES
07A1 81 22             CMP A  #'"       IS IT A QUOTE?
07A3 26 06             BNE    INPUT2
07A5 08                INX              BUMP THE POINTER
07A6 BD 05 47          JSR    PSTRNG    OUTPUT STRING
07A9 20 3B             BRA    INPUT6
07AB BD 05 7C  INPUT2  JSR    FNDLBL    FIND LABLE
07AE DF 33             STX    XTEMP4    SAVE POINTER
07B0 CE 00 68  INPUT3  LDX    #BUFFER   SET POINTER
07B3 96 13             LDA A  QMFLAG    TEST FLAG
07B5 26 07             BNE    INPUT4
07B7 86 3F             LDA A  #'?
07B9 97 13             STA A  QMFLAG    SET FLAG
07BB BD 04 4C          JSR    OUTCH     OUT A ?
07BE BD 01 09  INPUT4  JSR    INCH      GET A DIGIT
07C1 81 18             CMP A  #DELCOD   DELETE?
07C3 26 05             BNE    INPU45
07C5 7F 00 13          CLR    QMFLAG
07C8 20 E6             BRA    INPUT3
07CA A7 00     INPU45  STA A  0,X       SAVE IT
07CC 08                INX
07CD 81 2C             CMP A  #',       1S IT COMMA?
07CF 27 09             BEQ    INPUT5
07D1 81 0D             CMP A  #$D       IS IT A C.R.?
07D3 26 E9             BNE    INPUT4
07D5 97 12             STA A  CRFLAG    SET FLAG
07D7 BD 02 EA          JSR    PCRLF     OUTPUT A CR & LF
07DA CE 00 68  INPUT5  LDX    #BUFFER   SET POINTER
07DD BD 03 1A          JSR    BCDCON    GO CNVRT NUM.
07E0 DE 33             LDX    XTEMP4
07E2 8D 2D             BSR    LABLS2
07E4 DF 04             STX    BUFPNT    SAVE POINTER
07E6 81 2C     INPUT6  CMP A  #',       IS IT A COMMA?
07E8 26 07             BNE    INPUT7
07EA 08                INX
07EB 96 12             LDA A  CRFLAG    TEST FLAG
07ED 27 AF             BEQ    INPUT1
07EF 20 AA             BRA    INPUT0
07F1 BD 03 08  INPUT7  JSR    TSTTRM
07F4 26 13             BNE    INPUT9
07F6 96 12     INPU72  LDA A  CRFLAG    TEST FLAG
07F8 27 03             BEQ    INPUT8
07FA 7E 07 04  INPU75  JMP    RUNEXC
07FD BD 01 09  INPUT8  JSR    INCH      GET CHAR.
0800 81 0D             CMP A  #$D       C.R.?
0802 26 F9             BNE    INPUT8
0804 BD 02 EA          JSR    PCRLF
0807 20 F1             BRA    INPU75
0809 86 45     INPUT9  LDA A  #$45
080B 7E 04 61          JMP    MISTAK    REPORT ERROR


               * GET AND PUT LABLE

080E BD 05 7C  LABLES  JSR    FNDLBL    GO FIND IT
0811 BD 06 64  LABLS2  JSR    PUTLBL    GO PUT IT
0814 7E 03 6F          JMP    NXTSPC    GET TO NEXT SET


               * DATA ROUTINE

0817 96 19     DATA    LDA A  RUNFLG    RUNNING?
0819 27 49             BEQ    READ6
081B BD 03 6F          JSR    NXTSPC    FIND NEXT
081E 97 1A             STA A  DATAFL    SET DATA FLAG
0820 DF 0C             STX    DATAST    SET POINTER
```

```
0822 DF 0E              STX    DATAPT
0824 20 3E              BRA    READ6      RETURN


               * READ DATA ROUTINE

0826 96 19     READ     LDA A  RUNFLG     RUNNING?
0828 27 3A              BEQ    READ6
082A 96 1A              LDA A  DATAFL     CHECK FLAG
082C 27 39              BEQ    READ8
082E BD 03 59           JSR    NXTBLK     GET NEXT
0831 BD 03 68  READ2    JSR    SKIPSP     GO CLASSIFY
0834 BD 05 7C           JSR    FNDLBL
0837 DF 33              STX    XTEMP4
0839 DE 04              LDX    BUFPNT
083B DF 35              STX    XTEMP5     SAVE IT
083D DE 0E              LDX    DATAPT     GET DATA PNTR
083F BD 0A 26           JSR    EXPR       GET DATA
0842 A6 00              LDA A  0,X        GET CHAR.
0844 BD 03 08           JSR    TSTTRM     TEST IT
0847 26 04              BNE    READ25
0849 DE 0C              LDX    DATAST     SET POINTER
084B 20 01              BRA    READ3
084D 08        READ25   INX               BUMP THE POINTER
084E DF 0E     READ3    STX    DATAPT
0850 DE 35              LDX    XTEMP5
0852 DF 04              STX    BUFPNT
0854 DE 33              LDX    XTEMP4
0856 8D B9              BSR    LABLS2
0858 81 2C              CMP A  #',        IS IT A COMMA?
085A 26 03              BNE    READ4
085C 08                 INX
085D 20 D2              BRA    READ2      REPEAT
085F BD 03 08  READ4    JSR    TSTTRM
0862 26 03              BNE    READ8      ERROR
0864 7E 07 04  READ6    JMP    RUNEXC     RETURN
0867 86 51     READ8    LDA A  #$51
0869 7E 04 61           JMP    MISTAK

               * RESTORE DATA STRING

086C DF 22     RESTOR   STX    XSAVE      SAVE POINTER
086E DE 0C              LDX    DATAST
0870 DF 0E              STX    DATAPT     FIX DATA PNTR
0872 DE 22              LDX    XSAVE      RESTORE POINTER
0874 20 EE              BRA    READ6

               * ON GOTO ROUTINE
0876 BD 03 59  ONGOTO   JSR    NXTBLK     FIND NEXT BLOCK
0879 BD 0A 26           JSR    EXPR       EVAL. EXPR.
087C 96 64              LDA A  NUMBER+2
087E 84 0F              AND A  #$0F       MASK L.S. DIGIT
0880 36                 PSH A             SAVE A
0881 7F 00 12           CLR    CRFLAG
0884 08                 INX               BUMP THE POINTER
0885 08                 INX
0886 A6 00              LDA A  0,X        GET CHAR
0888 81 54              CMP A  #'T        IS IT A "T"?
088A 27 02              BEQ    ONGOT0
088C 97 12              STA A  CRFLAG     SET FLAG
088E BD 03 5B  ONGOT0   JSR    NXTBL4     GET NEXT
0891 DF 22              STX    XSAVE      SAVE X
0893 32                 PUL A             RESTORE A
0894 4A        ONGOT1   DEC A
0895 27 11              BEQ    ONGOT4
0897 E6 00     ONGOT2   LDA B  0,X        GET A CHAR,
0899 08                 INX               BUMP THE POINTER
089A C1 2C              CMP B  #',        IS IT A COMMA?
```

```
089C 26 04             BNE    ONGOT3
089E DF 22             STX    XSAVE      SAVE THE POINTER
08A0 20 F2             BRA    ONGOT1     REPEAT
08A2 C1 0D     ONGOT3  CMP B  #$D        C^R^ ?
08A4 26 F1             BNE    ONGOT2
08A6 DE 22             LDX    XSAVE      RESTORE POINTER
08A8 D6 12     ONGOT4  LDA B  CRFLAG     CHECK FLAG
08AA 27 03             BEQ    ONGOT6
08AC 7E 09 32          JMP    GOSUB2
08AF 7E 07 84  ONGOT6  JMP    GOTO1

               * ROUTINE

08B2 BD 03 6F  IF      JSR    NXTSPC     FIND NEXT
08B5 BD 0A 26          JSR    EXPR       EUAL EXPR
08B8 A6 00             LDA A  0,X        GET CHAR
08BA 8D 63             BSR    CLSREL     REL OPERATOR?
08BC 26 5C             BNE    IF9        ERROR!
08BE 36                PSH A             SAVE A
08BF A6 01             LDA A  1,X        GET CHAR
08C1 8D 5C             BSR    CLSREL     REL OP?
08C3 32                PUL A             RESTORE A
08C4 26 04             BNE    IF1
08C6 E6 01             LDA B  1,X
08C8 1B                ABA               FORM REL CODE
08C9 08                INX               BUMP THE POINTER
08CA 08        IF1     INX
08CB 36                PSH A             SAVE A
08CC BD 0A 26          JSR    EXPR       EVAL EXPR
08CF 32                PUL A
08D0 84 0F             AND A  #$0F       MASK
08D2 80 09             SUB A  #9         BIAS IT
08D4 2B 44             BMI    IF9        ERROR?
08D6 48                ASL A             TIMES FOUR
08D7 48                ASL A
08D8 B7 08 E2          STA A  OFSET3+1
08DB BD 0B C4          JSR    SUB        GO COMPARE
08DE BD 0C BE          JSR    ZCHK       SET CC REG
08E1 20 FE     OFSET3  BRA    *
08E3 2F 18     BRATBL  BLE    IF4        BRANCH TABLE
08E5 20 30             BRA    IF8
08E7 26 14             BNE    IF4
08E9 20 2C             BRA    IF8
08EB 2C 10             BGE    IF4
08ED 20 28             BRA    IF8
08EF 2D 0C             BLT    IF4
08F1 20 24             BRA    IF8
08F3 27 08             BEQ    IF4
08F5 20 20             BRA    IF8
08F7 2E 04             BGT    IF4
08F9 20 1C             BRA    IF8
08FB 20 1D             BRA    IF9        ERROR!
08FD DE 04     IF4     LDX    BUFPNT     SET POINTER
08FF A6 00             LDA A  0,X        GET CHAR
0901 81 54             CMP A  #'T        IS IT A "T"?
0903 26 0F             BNE    IF6
0905 BD 03 6F          JSR    NXTSPC
0908 DF 04             STX    BUFPNT     SAVE POINTER
090A BD 0C E5          JSR    CLASS1     GO CLASSIFY
090D C1 03             CMP B  #3         IS IT A NUMBER?
090F 26 03             BNE    IF6
0911 7E 07 84          JMP    GOTO1      GO TO GOTO
0914 7E 07 30  IF6     JMP    RUNEX3
0917 7E 07 04  IF8     JMP    RUNEXC     GO PROCESS CMND
091A 86 62     IF9     LDA A  #$62       SET ERROR
091C 7E 04 61          JMP    MISTAK

               * CLASSIFY RELATIONAL OPERATION
```

```
091F 81 3B      CLSREL  CMP A  #$3B
0921 23 06              BLS    CLSRE5
0923 81 3E              CMP A  #$3E      CHECK CHAR
0925 22 02              BHI    CLSRE5
0927 5F                 CLR B            CLEAR FLAG
0928 39                 RTS              RETURN
0929 5C         CLSRE5  INC B            SET FLAG
092A 39                 RTS              RETURN

                * GOSUB ROUTINE

092B D6 19      GOSUB   LDA B  RUNFLG
092D 27 E8              BEQ    IF8
092F BD 03 6F           JSR    NXTSPC    FIND NEXT
0932 7C 00 1B   GOSUB2  INC    SUBCNT
0935 BD 0A 26           JSR    EXPR      EVALUATE EXPR
0938 09                 DEX
0939 BD 02 C6           JSR    FNDCRT    FIND C.R.
093C 08                 INX              BUMP THE POINTER
093D A6 00              LDA A  0,X       GET LINE NO
093F 36                 PSH A
0940 A6 01              LDA A  1,X
0942 36                 PSH A            SAVE AS RET. ADD.
0943 9F 37              STS    CPX1      SAVE SP
0945 CE A0 23           LDX    #STKBOT+35
0948 BD 0C B1           JSR    CMPX      CHECK BOUNDS
094B 23 03              BLS    GOSUB4
094D 7E 02 A0           JMP    ADJEN2    RPT OVFL
0950 7E 07 87   GOSUB4  JMP    GOTO2

                * RETURN ROUTINE

0953 86 73      RETURN  LDA A  #$73
0955 7A 00 1B           DEC    SUBCNT    DEC COUNTER
0958 2A 03              BPL    RETUR2
095A 7E 04 61           JMP    MISTAK    ERROR!
095D 32         RETUR2  PUL A            GET RET. ADD.
095E 33                 PUL B
095F BD 02 A9           JSR    FINDLN    GO FIND LINE
0962 7E 07 8A           JMP    GOTO3

                * EXPRESSION EQUATE

0965 BD 05 7A   EXPEQU  JSR    FNDLB0    FIND LABLE
0968 DF 33              STX    XTEMP4    SAVE
096A BD 03 6F           JSR    NXTSPC
096D 08                 INX
096E BD 0A 26           JSR    EXPR      GO EVALUATE
0971 DE 33              LDX    XTEMP4    GET POINTER
0973 7E 06 64           JMP    PUTLBL    INSTALL

                * FOR ROUTINE

0976 BD 03 59   FOR     JSR    NXTBLK    FIND NEXT
0979 36                 PSH A
097A 8D E9              BSR    EXPEQU
097C DE 08              LDX    DIMPNT
097E DF 37              STX    CPX1
0980 DE 06              LDX    FORSTK
0982 32                 PUL A
0983 A7 00              STA A  0,X
0985 96 05              LDA A  BUFPNT+1
0987 09                 DEX              DEC THE POINTER
0988 A7 00              STA A  0,X
098A 96 04              LDA A  BUFPNT    SET UP INDEX
098C 09                 DEX
098D A7 00              STA A  0,X
```

```
098F 09                    DEX
0990 BD 0C B1              JSR    CMPX       CHECK FOR OVFLW
0993 22 03                 BHI    FOR5
0995 7E 02 A0              JMP    ADJEN2
0998 DF 06       FOR5      STX    FORSTK     SAVE POINTER
099A 7E 07 04              JMP    RUNEXC

                 * NEXT ROUTINE

099D BD 03 59    NEXT      JSR    NXTBLK     FIND NEXT
09A0 DF 1E                 STX    NXPNTR
09A2 DE 06                 LDX    FORSTK     SET POINTER
09A4 BC 01 0F    NEXT1     CPX    MEMEND     OVFLW?
09A7 26 04                 BNE    NEXT2
09A9 DE 04                 LDX    BUFPNT     RESTORE PNTR
09AB 20 74                 BRA    NEXT9      ERROR!
09AD 08          NEXT2     INX               FIXUP POINTER
09AE 08                    INX
09AF 08                    INX
09B0 A1 00                 CMP A  0,X        CHECK
09B2 26 F0                 BNE    NEXT1
09B4 09                    DEX               FIX POINTER
09B5 09                    DEX
09B6 09                    DEX
09B7 DF 06                 STX    FORSTK
09B9 08                    INX
09BA EE 00                 LDX    0,X
09BC DF 04                 STX    BUFPNT     SAVE IT
09BE BD 05 7C              JSR    FNDLBL     FIND LABLE
09C1 DF 33                 STX    XTEMP4     SAVE IT
09C3 BD 03 6F              JSR    NXTSPC     FIND NEXT
09C6 BD 0A 26              JSR    EXPR       EVALUATE
09C9 BD 0B 51              JSR    STAKUP
09CC DE 33                 LDX    XTEMP4     RESTORE PNTR
09CE BD 0B 44              JSR    GETVAL     GET LABLE VALUE
09D1 DE 04                 LDX    BUFPNT
09D3 A6 00                 LDA A  0,X        GET CHAR
09D5 81 53                 CMP A  #'S        IS IT STEP?
09D7 27 08                 BEQ    NEXT4
09D9 BD 03 0F              JSR    UPSCLR
09DC 4C                    INC A
09DD 97 64                 STA A  NUMBER+2
09DF 20 0A                 BRA    NEXT5
09E1 BD 03 71    NEXT4     JSR    NXTSP4
09E4 BD 0A 26              JSR    EXPR
09E7 96 62                 LDA A  NUMBER
09E9 97 1C                 STA A  LETFLG     SHOW NEG.
09EB BD 0B CA    NEXT5     JSR    ADD        GO ADD IN STEP
09EE CE 00 10              LDX    #TRYVAL    SET POINTER
09F1 BD 06 64              JSR    PUTLBL     SAVE LABLE
09F4 BD 0B C4              JSR    SUB        COMPARE
09F7 BD 0C BE              JSR    ZCHK       SET CC REG
09FA D6 1C                 LDA B  LETFLG     CHK FLAG
09FC 2B 05                 BMI    NEXT6
09FE 06                    TAP               SET CC
09FF 2C 12                 BGE    NEXT8
0A01 20 03                 BRA    NEXT7
0A03 06          NEXT6     TAP               SET CC
0A04 2F 0D                 BLE    NEXT8
0A06 DE 06       NEXT7     LDX    FORSTK
0A08 08                    INX               FIXUP PNTR
0A09 08                    INX
0A0A 08                    INX
0A0B DF 06                 STX    FORSTK     SAVE IT
0A0D DE 1E                 LDX    NXPNTR
0A0F DF 04                 STX    BUFPNT     SAVE
0A11 20 0B                 BRA    NEXT85
0A13 CE 00 10    NEXT8     LDX    #TRYVAL
```

```
0A16 BD 0B 44          JSR    GETVAL
0A19 DE 33             LDX    XTEMP4
0A1B BD 06 64          JSR    PUTLBL
0A1E 7E 07 04  NEXT85  JMP    RUNEXC
0A21 86 81     NEXT9   LDA A  #$81      SET ERROR
0A23 7E 04 61  NEXTIO  JMP    MISTAK

               * EXPRESSION HANDLER

0A26 7F 00 2C  EXPR    CLR    STKCNT    SET COUNT = 0
0A29 96 2C     EXPRO   LDA A  STKCNT
0A2B 97 2D             STA A  AUXCNT
0A2D 8D 04             BSR    EVAL
0A2F 4D                TST A            CHECK FOR ERROR
0A30 26 F1             BNE    NEXTIO
0A32 39        EXPR1   RTS              RETURN
               *
               **EVAL
               * EVALUATE AN ALGEBRAIC STRING
               *
0A33 9F FE     EVAL    STS    STKTOP    SAVE SP TOP
0A35 BD 0C DE  EVA0A   JSR    SKYCLS
0A38 DF 04             STX    BUFPNT
0A3A C1 01             CMP B  #1        SEE IF EMPTY EXPRESSION
0A3C 26 04             BNE    EVAL0
0A3E 86 21             LDA A  #$21
0A40 20 4A             BRA    EVAL3
0A42 54        EVAL0   LSR B            SET UP
0A43 C1 03             CMP B  #3        CHECK FOR UNARY + OR -
0A45 26 03             BNE    EVAL1
0A47 BD 03 0F          JSR    UPSCLR
0A4A DE 04     EVAL1   LDX    BUFPNT
0A4C BD 0C DE  EVAL1A  JSR    SKYCLS    GET NEXT CHAR
0A4F DF 04             STX    BUFPNT
0A51 C1 04             CMP B  #4        CHECK FOR OPERATORS
0A53 23 02             BLS    EVAL1Z
0A55 C6 05             LDA B  #5        SET UP
0A57 58        EVAL1Z  ASL B
0A58 F7 0A 5C          STA B  OFFREL+1  SET UP BRANCH
0A5B 20 FE     OFFREL  BRA    *
0A5D 20 2B             BRA    EVAL2     ERROR
0A5F 20 1B             BRA    EVAL4     TERMINATOR
0A61 20 38             BRA    EVAL8     LETTER
0A63 20 2C             BRA    EVAL7     NUMBER
0A65 20 04             BRA    EVAL1C    RIGHT PAREN
0A67 36                PSH A            SAVE
0A68 08                INX
0A69 20 CA             BRA    EVA0A     AGAIN
0A6B 30        EVAL1C  TSX              GET SP
0A6C 09                DEX              ADJUST
0A6D D6 18             LDA B  DIMFLG
0A6F 9C FE             CPX    STKTOP    CHECK FOR EMPTY
0A71 27 06             BEQ    EVAL1E
0A73 32                PUL A
0A74 5F                CLR B
0A75 81 28             CMP A  #'(       CHECK FOR L PAREN ON STACK
0A77 27 F2             BEQ    EVAL1C    IF SO, OK
0A79 5D        EVAL1E  TST B            CHECK FOR ALRIGHT
0A7A 27 0E             BEQ    EVAL2     IF NOT SET, ERROR
0A7C 4F        EVAL4   CLR A
0A7D D6 2C             LDA B  STKCNT    GET STACK STKCNT
0A7F 5A                DEC B            CHECK OP STACK
0A80 D1 2D             CMP B  AUXCNT
0A82 26 06             BNE    EVAL2     IF NOT EMPTY, ERROR
0A84 30                TSX
0A85 09                DEX              ALIGN
0A86 9C FE             CPX    STKTOP    CHECK OPERATOR STACK
0A88 27 04             BEQ    EVAL3A    IF NOT EMPTY ERROR
```

```
0A8A 86 20      EVAL2   LDA A  #$20       SET ERROR NUMBER
0A8C 9E FE      EVAL3   LDS    STKTOP     GET SP
0A8E DE 04      EVAL3A  LDX    BUFPNT     SET POINTER
0A90 39                 RTS
0A91 BD 0B 51   EVAL7   JSR    STAKUP     SHIFT OP STACK UP
0A94 DE 04              LDX    BUFPNT
0A96 BD 03 1A           JSR    BCDCON     GET OPERAND
0A99 20 59              BRA    EVAL12
0A9B A6 01      EVAL8   LDA A  1,X        GET NEXT CHAR
0A9D BD 0C E5           JSR    CLASS1     GO CLASSIFY
0AA0 C1 02              CMP B  #2         CHECK FOR LETTER
0AA2 26 28              BNE    EVAL9      IF NOT, VARIABLE
0AA4 A6 00              LDA A  0,X        GET CHAR BACK
0AA6 DF 22              STX    XSAVE      SET FOR ENTRY TO FIMDKEY
0AA8 CE 01 7B           LDX    #FCTTBL
0AAB BD 03 85           JSR    FNDKE2     GO CHECK FUNCTION
0AAE 4D                 TST A             CHECK SUCCESS
0AAF 27 CB              BEQ    EVAL4
0AB1 7E 07 41           JMP    RUNEX4     GO SERVICE
0AB4 86 3F      EVAL86  LDA A  #'?        GET STGNUM OPERATOR
0AB6 36         EVAL87  PSH A             PUT ON STACK
0AB7 DE 22              LDX    XSAVE
0AB9 7E 0A 35           JMP    EVA0A
0ABC 86 40      EVAL85  LDA A  #'@        GET ABS OPERATOR
0ABE 20 F6              BRA    EVAL87
0AC0 BD 03 0F   EVAL88  JSR    UPSCLR     MOVE STACK UP
0AC3 BD 0D 2A           JSR    RANDOM     COMPUTE RANDOM #
0AC6 97 64              STA A  NUMBER+2
0AC8 DE 22      EVAL89  LDX    XSAVE      RESTORE POINTER
0ACA 20 28              BRA    EVAL12
0ACC D6 FE      EVAL9   LDA B  STKTOP
0ACE 37                 PSH B
0ACF D6 FF              LDA B  STKTOP+1
0AD1 37                 PSH B
0AD2 D6 2D              LDA B  AUXCNT     GET COUNTER
0AD4 37                 PSH B             SAVE
0AD5 D6 18              LDA B  DIMFLG     GET FLAG
0AD7 37                 PSH B             SAVE
0AD8 BD 05 7A           JSR    FNDLB0     FIND VARIABLE STORAGE
0ADB 33                 PUL B             GET FLAG
0ADC D7 18              STA B  DIMFLG     RESTORE
0ADE 33                 PUL B             GET COUNTER
0ADF D7 2D              STA B  AUXCNT     RESTORE
0AE1 33                 PUL B
0AE2 D7 FF              STA B  STKTOP+1
0AE4 33                 PUL B
0AE5 D7 FE              STA B  STKTOP
0AE7 BD 0B 51           JSR    STAKUP
0AEA DE 20              LDX    XTEMP
0AEC BD 0B 44           JSR    GETVAL     MOVE VALUE TO NUMBER
0AEF 20 05              BRA    EVA12A
0AF1 DE 04      EVA11C  LDX    BUFPNT     RESTORE POINTER
0AF3 08                 INX
0AF4 DF 04      EVAL12  STX    BUFPNT     SAVE POINTER
0AF6 30         EVA12A  TSX
0AF7 09                 DEX
0AF8 9C FE              CPX    STKTOP     CHECK OPERATOR STACK
0AFA 27 37              BEQ    EVAL10     IF EMPTY, DON'T OPERATE
0AFC 32                 PUL A
0AFD 36                 PSH A             PUT BACK
0AFE 81 28              CMP A  #'(        CHECK FOR LEFT PAREM
0B00 27 31              BEQ    EVAL10     IF SO, DON'T OPERATE
0B02 BD 0C E5           JSR    CLASS1     GO CLASSYFY
0B05 37                 PSH B
0B06 54                 LSR B             SET UP ID
0B07 96 2C              LDA A  STKCNT     GET COUNT
0B09 4A                 DEC A
0B0A C1 04              CMP B  #4         CHECK FOR ABS OR SON
```

```
0B0C 27 04              BEQ    EVA12C    IF SO, GO AHEAD
0B0E 91 2D              CMP A  AUXCNT    OTHERWISE CHECK FOR 2 OPERANDS
0B10 27 21              BEQ    EVAL10    IF NOT, ABORT
0B12 81 09     EVA12C   CMP A  #9        CHECK OVERFLOW
0B14 23 04              BLS    EVA12D    OK
0B16 86 24              LDA A  #$24      SET ERROR
0B18 20 16              BRA    EVAL19
0B1A 32        EVA12D   PUL A            GET CLASSIFICATION
0B1B 33                 PUL B            GET OPERATOR
0B1C 80 06              SUB A  #6        REMOVE BIAS
0B1E 48                 ASL A            #2
0B1F B7 0B 26           STA A  OPOFF+1   SET UP JMP
0B22 CE 0B 36           LDX    #OPTBL    POINT
0B25 EE 00     OPOFF    LDX    0,X
0B27 AD 00              JSR    0,X       GO OPERATE
0B29 BD 0C BE           JSR    ZCHK      CHECK RESULT
0B2C 28 C8              BVC    EVA12A    IF NO OVFL, GO OPERATE AGAIN
0B2E 86 23     EVAL18   LDA A  #$23      SET ERROR NUMBER
0B30 7E 0A 8C  EVAL19   JMP    EVAL3
0B33 7E 0A 4A  EVAL10   JMP    EVAL1
0B36 0B CA     OPTBL    FDB    ADD
0B38 0B C4              FDB    SUB
0B3A 0C 82              FDB    SIGNUM
0B3C 0B BC              FDB    ABSVAL
0B3E 0B F4              FDB    MULT
0B40 0C 15              FDB    DIVIDE
0B42 0C 94              FDB    EXPON
               *
               ** GET VALUE
               * MOVE 3 BYTES POINTED TO BY X TO NUMBER
               *
0B44 A6 00     GETVAL   LDA A  0,X       GET VALUE
0B46 97 62              STA A  NUMBER    STORE
0B48 A6 01              LDA A  1,X
0B4A 97 63              STA A  NUMBER+1
0B4C A6 02              LDA A  2,X
0B4E 97 64              STA A  NUMBER+2
0B50 39                 RTS
               *
               *
               ** STACKUP
               * ROLL OPERATIONAL STACK UPWARD
               *
0B51 CE 00 3B  STAKUP   LDX    #STKEND   POINT TO END
0B54 E6 03     STAKU2   LDA B  3,X
0B56 E7 00              STA B  0,X       MOVE
0B58 08                 INX
0B59 8C 00 62           CPX    #NUMBER   SEE IF DONE
0B5C 26 F6              BNE    STAKU2
0B5E 7C 00 2C           INC    STKCNT
0B61 39                 RTS
               *
               *
               ** STACKDOWN
               * ROLL OPERATIONAL STACK DOWNWARD
               *
0B62 CE 00 64  STAKDN   LDX    #AX-1     POINT TO STORE
0B65 E6 00     STAKD1   LDA B  0,X
0B67 E7 03              STA B  3,X
0B69 09                 DEX
0B6A 8C 00 3A           CPX    #STKEND-1 SEE IF DONE
0B6D 26 F6              BNE    STAKD1
0B6F 7A 00 2C           DEC    STKCNT
0B72 39                 RTS
               *
               *
               ** UADD
               * UNSIGNED ADD OF AX TO NUMBER
```

```
                 *
0B73 0C          UADD    CLC                ZERO THE CARRY
0B74 CE 00 64    UADD1   LDX    #NUMBER+2 POINT TO STORE
0B77 A6 00       UADD2   LDA A  0,X         GET ADDEND
0B79 A9 03               ADC A  3,X         ADD IN AUGEND
0B7B 19                  DAA
0B7C A7 00               STA A  0,X         SAVE
0B7E 09                  DEX
0B7F 8C 00 61            CPX    #NUMBER-1 SEE IF DONE
0B82 26 F3               BNE    UADD2
0B84 37          UADD22  PSH B
0B85 C6 02               LDA B  #$02        SET FOR OVFL
0B87 85 F0               BIT A  #$F0        AND AGAIN
0B89 26 01               BNE    UADD25
0B8B 5F                  CLR B              RESET OFVL
0B8C DA 30       UADD25  ORA B  OVFLBF
0B8E D7 30               STA B  OVFLBF      SET OVFL IF NECESSARY
0B90 17                  TBA
0B91 33                  PUL B
0B92 39          UADD3   RTS
                 *
                 *
                 **USUB
                 * UNSIGNED SUBTRACT OF AX FROM NUMBER
                 *
0B93 8D 03       USUB    BSR    TENCOM      GO TEN'S COMPLEMENT
0B95 0D                  SEC                FIX UP
0B96 20 DC               BRA    UADD1       GO ADD
                 *
                 *
                 **TENCOM
                 * UNSIGNED TEN'S COMPLEMENT OF AX (ALMOST)
                 *
0B98 CE 00 67    TENCOM  LDX    #AX+2       POINT TO AX
0B9B 86 99       TENCO1  LDA A  #$99
0B9D A0 00               SUB A  0,X         SUBTRACT FROM 99
0B9F A7 00               STA A  0,X         SAVE
0BA1 09                  DEX
0BA2 8C 00 64            CPX    #AX-1
0BA5 26 F4               BNE    TENCO1
0BA7 84 0F               AND A  #$0F        RESET SIGN
0BA9 A7 01               STA A  1,X         STORE
0BAB 39                  RTS
                 *
                 *
                 ** SET SIN
                 * CALCULATE RESULT SIGN
                 *
0BAC 7F 00 30    SETSIN  CLR    OVFLBF      CLEAR OVFL INDICATOR
0BAF 96 65       SETSI0  LDA A  AX          GET SIGN
0BB1 16                  TAB                SAVE
0BB2 C4 0F               AND B  #$0F        RESET SIGN
0BB4 D7 65               STA B  AX          PUT BACK
0BB6 97 2F               STA A  AXSIGN      SAVE SIGN
0BB8 98 62               EOR A  NUMBER      FORM NEW SIGN
0BBA 97 2E               STA A  SIGN        SAVE
0BBC D6 62       ABSVAL  LDA B  NUMBER      GET MS BYTE
0BBE C4 0F               AND B  #$0F        RESET SIGN
0BC0 D7 62               STA B  NUMBER      PUT BACK
0BC2 4D                  TST A              TEST NEW SIGN
0BC3 39                  RTS
                 *
                 *
                 **
                 * SUBTRACT AX FROM NUMBER
                 *
0BC4 96 62       SUB     LDA A  NUMBER      GET MS BYTE
0BC6 88 F0               EOR A  #$F0        CHANGE SIGN
```

```
0BC8 97 62              STA A  NUMBER     PUT BACK
                * GO INTO ADD
                *
                *
                * ADD
                * ADD AX TO NUMBER
                *
0BCA 8D 58      ADD     BSR    RELAY
0BCC 8D DE              BSR    SETSIN     GO CALCULATE SIGN
0BCE 2A 0A              BPL    ADD0       USE EITHER SIGN
0BD0 8D C1              BSR    USUB       OTHERWISE SUBTRACT
0BD2 06                 TAP               SET CCR
0BD3 28 09              BVC    ADD1       CHECK OVERFLOW
0BD5 73 00 2F           COM    AXSIGN     CHANGE FOR AX SMALLER
0BD8 20 0B              BRA    ADD15
0BDA 8D 97      ADD0    BSR    UADD       GO ADD
0BDC 20 0A              BRA    ADD2       GO FIX SIGN
0BDE 8D 44      ADD1    BSR    RELAY      COPY NUMBER TO AX
0BE0 BD 03 0F           JSR    UPSCLR     RESTORE
0BE3 8D AE              BSR    USUB       GO NEGATE
0BE5 7F 00 30   ADD15   CLR    OVFLBF
0BE8 96 2F      ADD2    LDA A  AXSIGN     GET OLD SIGN
                *
                *
                ** FIXSIN
                * SET THE SIGN ON THE RESULT
                *
0BEA 84 F0      FIXSIN  AND A  #$F0       MASK
0BEC C6 0F              LDA B  #$0F       SET MASK
0BEE D4 62              AND B  NUMBER     RESET SIGN
0BF0 1B                 ABA               TACK ON SIGN
0BF1 97 62              STA A  NUMBER     PUT BACK
0BF3 39         FIX2    RTS
                *
                *
                ** MULT
                * MULTIPLY AC BY AX
                *
0BF4 8D 2E      MULT    BSR    RELAY      MOVE STACK
0BF6 8D B4              BSR    SETSIN     GO CALC. SIGNS
0BF8 BD 03 0F   MULT0   JSR    UPSCLR     MOVE STACK UP
0BFB C6 05              LDA B  #5         SET COUNTER
0BFD 96 5F      MULT1   LDA A  AC         GET MS BYTE OF AC
0BFF 27 08              BEQ    MULT3      IF ZERO , LOOP
0C01 BD 0B 73   MULT2   JSR    UADD       ADD IN AX
0C04 7A 00 5F           DEC    AC         ONCE DONE
0C07 26 F8              BNE    MULT2
0C09 5A         MULT3   DEC B             ONCE DONE
0C0A 27 3D              BEQ    MULT4      CHECK IF ALL DONE
0C0C 8D 4A              BSR    ACLEFT     SHIFT AC LEFT
0C0E 96 62              LDA A  NUMBER
0C10 BD 0B 84           JSR    UADD22
0C13 20 E8              BRA    MULT1
                *
                *
                ** DIVIDE
                * DIVIDE AC-NUMBER BY AX
                *
0C15 8D 0D      DIVIDE  BSR    RELAY
0C17 CE 00 65           LDX    #AX
0C1A BD 0C C1           JSR    ZCHK1      GO CHECK IF AX=O
0C1D 26 08              BNE    DIVID1     IF NOT, OK
0C1F 86 22      DIVID0  LDA A  #$22       SET ERROR
0C21 7E 0A 8C           JMP    EVAL3
0C24 7E 0B 62   RELAY   JMP    STAKDN     RELAY TO STACK DOWN
0C27 BD 0B AC   DIVID1  JSR    SETSIN     CALC, SIGNS
0C2A BD 0B 51           JSR    STAKUP     PUSH BACK
0C2D 8D 29              BSR    ACLEFT     SHIFT DOWN
```

```
0C2F 6F 02              CLR    2,X
0C31 6F 03              CLR    3,X       ZERO OUT NUMBER
0C33 C6 05              LDA B  #5        SET LOOP COUNT
0C35 8D 21     DIVID2   BSR    ACLEFT    MOVE AC DOWN
0C37 BD 0B 98  DIVI2A   JSR    TENCOM    TAKE 10'S COMP
0C3A 8D 2E     DIVID3   BSR    DADD      GO SPECIAL ADD
0C3C 85 F0              BIT A  #$F0      CHECK FOR OVERFLOW
0C3E 26 13              BNE    DIVID4
0C40 BD 0B 98           JSR    TENCOM    IF SO, RESTORE AX
0C43 0C                 CLC
0C44 8D 25              BSR    DADD1     ADD BACK IN
0C46 5A                 DEC B            ONE PASS MADE
0C47 26 EC              BNE    DIVID2
0C49 96 2E     MULT4    LDA A  SIGN      GET THE SIGN
0C4B 8D 9D              BSR    FIXSIN    GO FIX UP THE SIGN
0C4D CE 00 5E           LDX    #AC-1     POINT TO AC
0C50 7E 0B 65           JMP    STAKD1    MOVE STACK BACK
0C53 7C 00 64  DIVID4   INC    NUMBER+2  ADD ONE IN
0C56 20 E2              BRA    DIVID3    GO DO AGAIN
               *
               *
               ** ACLEFT
               * SHIFT AC-NUMBER LEFT 4 BITS
               *
0C58 86 04     ACLEFT   LDA A  #4        SET FOR 4 BITS
0C5A CE 00 64  ACLEF1   LDX    #AX-1     POINT X
0C5D 0C                 CLC
0C5E 69 00     ACLEF2   ROL    0,X       ROTATE
0C60 09                 DEX
0C61 8C 00 5E           CPX    #AC-1     CHECK IF DONE
0C64 26 F8              BNE    ACLEF2
0C66 4A                 DEC A            CHECK FOR DONE
0C67 26 F1              BNE    ACLEF1
0C69 39                 RTS
               *
               *
               ** DADD
               * ADD AX TO A C
               *
0C6A 0D        DADD     SEC
0C6B CE 00 61  DADD1    LDX    #AC+2
0C6E 96 5F              LDA A  AC        GET MS BYTE
0C70 84 0F              AND A  #$0F      RESET SIGN
0C72 97 5F              STA A  AC        STORE BACK
0C74 A6 00     DADD2    LDA A  0,X       GET ADDEND
0C76 A9 06              ADC A  6,X       ADD IN
0C78 19                 DAA
0C79 A7 00              STA A  0,X       SAVE
0C7B 09                 DEX
0C7C 8C 00 5E           CPX    #AC-1     SEE IF DONE
0C7F 26 F3              BNE    DADD2
0C81 39                 RTS
               *
               ** SIGNUM
               * CALCULATE SIGNUM FUNCTION
               *
0C82 8D 3A     SIGNUM   BSR    ZCHK      GO CHECK = O
0C84 27 0B              BEQ    SIGNU2    IF SOY RESULT =0
0C86 D6 62              LDA B  NUMBER    OTHERWISE GET SIGN
0C88 8D 07     SIGNU1   BSR    SIGNU2    GO CLEAR
0C8A 7C 00 64           INC    NUMBER+2  MAKE = I
0C8D 17                 TBA              SET FOR FIXSIN
0C8E 7E 0B EA           JMP    FIXSIN    GO SET THE SIGN
0C91 7E 03 12  SIGNU2   JMP    CLRNUM
               *
               *
               ** EXPON
               * CALCULATE EXPONENTIATION
```

```
                 * ONLY POSITIVE EXPONENTS UP TO 99 ALLOWED
                 *
0C94 8D 8E       EXPON   BSR    RELAY      MOVE OPERANDS DOWN
0C96 5F                  CLR B
0C97 D7 30               STA B  OVFLBF     CLEAR OVER FLOW
0C99 96 67               LDA A  AX+2       GET EXPONENT
0C9B 27 EB               BEQ    SIGNU1     IF O, GO MAKE RESULT +1
0C9D BD 0B 51            JSR    STAKUP     GET TWO COPIES
0CA0 8D 82               BSR    RELAY      MOVE DOWN
0CA2 8B 99       EXPON1  ADD A  #$99       DECREMENT
0CA4 19                  DAA
0CA5 27 16               BEQ    CMPX2      WHEN 0 ALL DONE
0CA7 36                  PSH A             SAVE EXP
0CA8 BD 0B AF            JSR    SETSI0     GO FIX SIGNS
0CAB BD 0B F8            JSR    MULT0      GO MULTIPLY
0CAE 32                  PUL A             GET EXPONENT
0CAF 20 F1               BRA    EXPON1     LOOP
                 *
                 *
                 ** CMPX
                 * FULL COMPARE ON X
                 * COMPARES X WITH CONTENTS OF CPX1
                 *
0CB1 DF 39       CMPX    STX    CPX2       SAVE
0CB3 96 39       CMPX1   LDA A  CPX2       GET MS BYTE
0CB5 91 37               CMP A  CPX1       COMPARE
0CB7 26 04               BNE    CMPX2      IF NOT EQUAL, DONE
0CB9 D6 3A               LDA B  CPX2+1     GET LS BYTE
0CBB D1 38               CMP B  CPX1+1     COMPARE
0CBD 39          CMPX2   RTS               DOME
                 *
                 *
                 ** ZCHK
                 * CHECK OPERAND FOR EQUAL TO 0
                 *
0CBE CE 00 62    ZCHK    LDX    #NUMBER
0CC1 5F          ZCHK1   CLR B
0CC2 6D 02               TST    2,X
0CC4 26 0E               BNE    ZCHK2
0CC6 6D 01               TST    1,X
0CC8 26 0A               BNE    ZCHK2
0CCA A6 00               LDA A  0,X        GET MS BYTE
0CCC 84 0F               AND A  #$0F
0CCE 26 04               BNE    ZCHK2      CHECK FOR 0
0CD0 A7 00               STA A  0,X        RESET SIGN BITS
0CD2 C6 04               LDA B  #4
0CD4 A6 00       ZCHK2   LDA A  0,X        GET MS BYTE
0CD6 46                  ROR A             MOVE A SIGN BIT TO N
0CD7 84 08               AND A  #8         MASK N BIT
0CD9 1B                  ABA               MERGE Z AND N
0CDA 9A 30               ORA A  OVFLBF     ADD IN V
0CDC 06                  TAP               SET CCR
0CDD 39                  RTS
                 *
                 *
                 **
0CDE BD 03 68    SKYCLS  JSR    SKIPSP
0CE1 20 02               BRA    CLASS1
                 *
                 *
                 **CLASS
                 *CLASSIFY A CHARACTER IN THE A ACCUMULATOR
                 *CLASSIFICATION RETURNED IN B
                 *  0 ERROR
                 *  1 TERMINATOR
                 *  2 LETTER
                 *  3 NUMBER
                 *  4 )
```

```
                *  5 (
                *  6 +
                *  7 -
                *  8 SGN
                *  9 ABS
                * 10 *
                * 11 /
                * 12 ~
0CE3 A6 00      CLASS   LDA A  0,X        GET CHAR
0CE5 C6 01      CLASS1  LDA B  #1         SET UP
0CE7 81 0D              CMP A  #$D        CHECK FOR CR
0CE9 27 17              BEQ    CLAS25
0CEB 5A                 DEC B
0CEC 36                 PSH A             SAVE CHAR
0CED 80 28      CLAS2B  SUB A  #'(        REMOVE BIAS
0CEF 2B 10              BMI    CLASS2     CHECK ILLEGAL
0CF1 81 18              CMP A  #'@-'(     CHECK LIMIT
0CF3 23 0E              BLS    CLASS3     NOT LETTER
0CF5 81 32              CMP A  #'Z-'(     CHECK FOR LETTER
0CF7 23 06              BLS    CLAS1B
0CF9 81 36              CMP A  #'^-'(     CHECK FOR ILLEGAL
0CFB 26 04              BNE    CLASS2
0CFD C6 0A              LDA B  #10        FIX UP
0CFF CB 02      CLAS1B  ADD B  #02
0D01 32         CLASS2  PUL A             RESTORE CHARACTER
0D02 39         CLAS25  RTS               DONE
0D03 DF 24      CLASS3  STX    XSAVE2     SAVE X REG
0D05 CE 0D 11           LDX    #CLSTBL    POINT TO TABLE
0D08 B7 0D 0C           STA A  CLSOFF+1   SET BIAS
0D0B E6 00      CLSOFF  LDA B  0,X        GET CLASSIFICATION
0D0D DE 24              LDX    XSAVE2     RESTORE X REG,
0D0F 20 F0              BRA    CLASS2
0D11 05         CLSTBL  FCB    5,4,10,6,1,7,0,11,3,3,3,3
0D12 04 0A
0D14 06 01
0D16 07 00
0D18 0B 03
0D1A 03 03
0D1C 03
0D1D 03                 FCB    3,3,3,3,3,3,1,1,1,1,1,8,9
0D1E 03 03
0D20 03 03
0D22 03 01
0D24 01 01
0D26 01 01
0D28 08 09
                *
                *
                * RANDOM GENERATOR
                *
0D2A C6 08      RANDOM  LDA B  #8         SET COUNTER
0D2C CE 00 00           LDX    #RNDM
0D2F A6 03      RPT     LDA A  3,X        GET M.S. BYTE OF RANDOM NO.
0D31 48                 ASL A             SHIFT IT LEFT THREE:
0D32 48                 ASL A             TIMES TO GET BIT 28
0D33 48                 ASL A             IN LINE WITH BIT 31
0D34 A8 03              EOR A  3,X        XOR A WITH RANDOM NO
0D36 48                 ASL A             PUT BIT 28.XOR31 IN
0D37 48                 ASL A             CARRY BY SHIFTING LEFT
0D38 69 00              ROL    0,X        ROTATE ALL FOUR BYTES OF
0D3A 69 01              ROL    1,X        THE RANDOM NO, ROTATING
0D3C 69 02              ROL    2,X        THE CARRY INTO THE LSB
0D3E 69 03              ROL    3,X        THE MSB IS LOST
0D40 5A                 DEC B             DECREMENT THE COUNTER
0D41 26 EC              BNE    RPT        IF ITS NOT O, GO REPEAT
0D43 A6 00              LDA A  0,X        PUT RANDOM # IN A
0D45 81 9F              CMP A  #$9F       CHECK IN RANGE
0D47 22 E1              BHI    RANDOM     IN NOT GET ANOTHER
```

```
 0D49 8B 00              ADD A  #0        SET HALF CARRY
 0D4B 19                 DAA
 0D4C 39                 RTS
 0D4D           ENDSTR   RMB    2
 0D4F           STORSP   EQU    *

 1F00                    ORG    EXTERN
 1F00 39                 RTS
                         END

NO ERROR(S) DETECTED

   SYMBOL TABLE:

ABSVAL 0BBC   AC     005F   ACLEF1 0C5A   ACLEF2 0C5E   ACLEFT 0C58
ADD    0BCA   ADD0   0BDA   ADD1   0BDE   ADD15  0BE5   ADD2   0BE8
ADDIN  0658   ADDIN1 065A   ADDIN2 0663   ADDX   05FA   ADJEN2 02A0
ADJEND 028A   ADSHF0 063B   ADSHF1 063F   AUXCNT 002D   AX     0065
AXSIGN 002F   BACKSP 0008   BCDC01 0330   BCDCO1 0331   BCDCO2 033D
BCDCO4 0344   BCDCON 031A   BINCON 061E   BRATBL 08E3   BREAK  010C
BREAK2 045A   BUFFER 0068   BUFPNT 0004   CHRCNT 003E   CLAS1B 0CFF
CLAS25 0D02   CLAS2B 0CED   CLASS  0CE3   CLASS1 0CE5   CLASS2 0D01
CLASS3 0D03   CLEAR  019D   CLEAR2 019E   CLRBEG 018B   CLRBG2 0190
CLREND 0195   CLRNUM 0312   CLSOFF 0D0B   CLSRE5 0929   CLSREL 091F
CLSTBL 0D11   CMPX   0CB1   CMPX1  0CB3   CMPX2  0CBD   COLCON 0016
CONSKP 0364   COUNT  002B   CPX1   0037   CPX2   0039   CRFLAG 0012
CRLFST 0301   DADD   0C6A   DADD1  0C6B   DADD2  0C74   DATA   0817
DATAFL 001A   DATAPT 000E   DATAST 000C   DELCOD 0018   DIM    0671
DIM01  0687   DIM1   0694   DIM2   06AC   DIM5   06F0   DIM9   06A7
DIMFLG 0018   DIMN   0678   DIMPNT 0008   DIVI2A 0C37   DIVID0 0C1F
DIVID1 0C27   DIVID2 0C35   DIVID3 0C3A   DIVID4 0C53   DIVIDE 0C15
ENDSTR 0D4D   ERRSTR 0498   ERSTR2 04A1   EVA0A  0A35   EVA11C 0AF1
EVA12A 0AF6   EVA12C 0B12   EVA12D 0B1A   EVAL   0A33   EVAL0  0A42
EVAL1  0A4A   EVAL10 0B33   EVAL12 0AF4   EVAL18 0B2E   EVAL19 0B30
EVAL1A 0A4C   EVAL1C 0A6B   EVAL1E 0A79   EVAL1Z 0A57   EVAL2  0A8A
EVAL3  0A8C   EVAL3A 0A8E   EVAL4  0A7C   EVAL7  0A91   EVAL8  0A9B
EVAL85 0ABC   EVAL86 0AB4   EVAL87 0AB6   EVAL88 0AC0   EVAL89 0AC8
EVAL9  0ACC   EXPEQU 0965   EXPON  0C94   EXPON1 0CA2   EXPR   0A26
EXPR1  0A32   EXPRO  0A29   EXTERN 1F00   EXTRA  0029   EXTRNL 0701
FCTTBL 017B   FIELD1 04B1   FIELD2 04BA   FIELD3 04CC   FILB75 020E
FILBU2 01D8   FILBU6 01EB   FILBU7 0209   FILBU8 0213   FILBUF 01B0
FINDL1 02AC   FINDL2 02B1   FINDL4 02B3   FINDL6 02C1   FINDLN 02A9
FIX2   0BF3   FIXSIN 0BEA   FLDCNT 001D   FNDCRT 02C6   FNDKE2 0385
FNDKE4 0387   FNDKE5 039C   FNDKE6 039D   FNDKEY 037B   FNDL25 059C
FNDL45 05EC   FNDLB0 057A   FNDLB1 058E   FNDLB2 058F   FNDLB3 05B4
FNDLB4 05B6   FNDLB5 05F7   FNDLB9 060F   FNDLBL 057C   FNDLIN 02A5
FNDVAL 02C9   FNDVAR 0564   FOR    0976   FOR5   0998   FORSTK 0006
GETVAL 0B44   GOSUB  092B   GOSUB2 0932   GOSUB4 0950   GOTO   0781
GOTO1  0784   GOTO2  0787   GOTO3  078A   GOTO4  078F   GOTO5  0792
IF     08B2   IF1    08CA   IF4    08FD   IF6    0914   IF8    0917
IF9    091A   INCH   0109   INCHAR 02D0   INCHR2 02E2   INCHR4 02E9
INPU45 07CA   INPU72 07F6   INPU75 07FA   INPUT  0798   INPUT0 079B
INPUT1 079E   INPUT2 07AB   INPUT3 07B0   INPUT4 07BE   INPUT5 07DA
INPUT6 07E6   INPUT7 07F1   INPUT8 07FD   INPUT9 0809   INSER2 0264
INSER3 026F   INSER4 0275   INSER6 0287   INSERT 0254   INTBRK 0452
KEYTBL 0111   LABLES 080E   LABLS2 0811   LBLTBL 00B0   LET    0772
LET2   077B   LETADR 0123   LETFLG 001C   LIST   03EC   LIST1  0409
LIST2  0414   LIST3  0418   LIST4  041B   LIST5  0428   LIST6  0433
LIST8  0441   MEMEND 010F   MICBAS 01A6   MISTA1 0465   MISTA2 0470
MISTA4 0478   MISTAK 0461   MONITR E0E3   MONPC  A048   MULT   0BF4
MULT0  0BF8   MULT1  0BFD   MULT2  0C01   MULT3  0C09   MULT4  0C49
NEGFLG 0027   NEXT   099D   NEXT1  09A4   NEXT2  09AD   NEXT4  09E1
NEXT5  09EB   NEXT6  0A03   NEXT7  0A06   NEXT8  0A13   NEXT85 0A1E
NEXT9  0A21   NEXTIO 0A23   NOEXFL 0028   NUMBER 0062   NUMCNT 0026
NXPNTR 001E   NXTBL4 035B   NXTBLK 0359   NXTSP4 0371   NXTSPC 036F
OFFREL 0A5B   OFFSET 026B   OFSET2 024B   OFSET3 08E1   ONGOT0 088E
ONGOT1 0894   ONGOT2 0897   ONGOT3 08A2   ONGOT4 08A8   ONGOT6 08AF
ONGOTO 0876   OPOFF  0B25   OPSTAK 003F   OPTBL  0B36   OUTBC2 03C5
```

```
OUTBC3 03CD   OUTBC4 03D4   OUTBC6 03E0   OUTBC8 03E7   OUTBCD 03B1
OUTBCI 03B4   OUTCH  044C   OUTEEE 0106   OUTHL  0444   OUTHR  0448
OVFLBF 0030   PCRLF  02EA   PCRLF2 02FB   PDATA1 02EF   PFILBG A002
PFILEN A004   PIAADR 8004   PRIN45 04F2   PRIN47 04F8   PRIN51 0516
PRIN52 0522   PRIN55 0524   PRINT  04A6   PRINT0 04A9   PRINT1 04CF
PRINT2 04D8   PRINT4 04E5   PRINT5 0514   PRINT6 052E   PRINT7 0537
PRINT8 053C   PRINT9 0544   PRMPTC 0021   PSTRN4 055B   PSTRN8 055F
PSTRNG 0547   PUTLB2 0668   PUTLBL 0664   QMFLAG 0013   RANDOM 0D2A
READ   0826   READ2  0831   READ25 084D   READ3  084E   READ4  085F
READ6  0864   READ8  0867   RELAY  0C24   REPLA4 023C   REPLA5 0246
REPLA6 0252   REPLAC 0234   RESTOR 086C   RESTRT 0103   RETUR2 095D
RETURN 0953   RNDM   0000   ROWCON 0015   ROWVAR 0014   RPT    0D2F
RUN    075F   RUNE05 0716   RUNE22 0726   RUNE25 072B   RUNE27 072C
RUNE35 073E   RUNER1 047B   RUNER2 0483   RUNER4 0490   RUNEX0 0714
RUNEX1 071A   RUNEX2 0725   RUNEX3 0730   RUNEX4 0741   RUNEXA 0711
RUNEXC 0704   RUNFLG 0019   SETSI0 0BAF   SETSIN 0BAC   SIGN   002E
SIGNU1 0C88   SIGNU2 0C91   SIGNUM 0C82   SKIPS4 036E   SKIPSP 0368
SKPSP0 0367   SKYCLS 0CDE   STACK  A07F   STAKD1 0B65   STAKDN 0B62
STAKU2 0B54   STAKUP 0B51   START  0100   STKBOT A000   STKCNT 002C
STKEND 003B   STKTOP 00FE   STORSP 0D4F   STUFLN 0223   SUB    0BC4
SUBCNT 001B   TABFLG 0017   TENCO1 0B9B   TENCOM 0B98   TIMTHR 0614
TRYVAL 0010   TSTLE1 0758   TSTLE2 075E   TSTLET 0745   TSTTR2 030E
TSTTRM 0308   UADD   0B73   UADD1  0B74   UADD2  0B77   UADD22 0B84
UADD25 0B8C   UADD3  0B92   UPSCLR 030F   USUB   0B93   XSAVE  0022
XSAVE2 0024   XTEMP  0020   XTEMP2 0031   XTEMP3 000A   XTEMP4 0033
XTEMP5 0035   ZCHK   0CBE   ZCHK1  0CC1   ZCHK2  0CD4
```

OBJECT CODE:

```
S1 13 0100 7E 01 A6 7E 01 B0 7E E1 D1 BD E1 AC 7E 04 52 1E 2B
S1 13 0110 FF 50 52 49 04 A6 49 4E 50 07 98 49 46 20 08 B2 58
S1 13 0120 4C 45 54 07 72 46 4F 52 09 76 4E 45 58 09 9D 47 2F
S1 13 0130 4F 54 07 81 47 4F 53 09 2B 4F 4E 20 08 76 52 45 A1
S1 13 0140 54 09 53 52 45 41 08 26 44 41 54 08 17 52 45 53 13
S1 13 0150 08 6C 44 49 4D 06 71 45 58 54 07 01 4D 4F 4E E0 13
S1 13 0160 E3 45 4E 44 01 B0 52 45 4D 07 04 52 55 4E 07 5F D6
S1 13 0170 4C 49 53 03 EC 53 43 52 01 A6 00 52 4E 44 0A C0 67
S1 13 0180 41 42 53 0A BC 53 47 4E 0A B4 00 CE 01 00 DF 0A 71
S1 13 0190 CE 00 0C 20 08 FE 01 0F DF 0A FE 0D 4D 4F A7 00 14
S1 13 01A0 08 9C 0A 26 F9 39 8D E3 CE 0D 4F FF 0D 4D 8D E5 E0
S1 13 01B0 CE 01 03 FF A0 48 8E A0 7F CE 00 68 DF 0A 8D D0 59
S1 13 01C0 CE 0D 4D FF A0 02 EE 00 FF A0 04 DF 08 CE 00 68 B4
S1 13 01D0 BD 02 EA 86 21 BD 04 4C BD 02 D0 27 D3 A7 00 81 0D
S1 13 01E0 0D 27 08 08 8C 00 B0 26 EF 20 C5 CE 00 68 BD 03 9B
S1 13 01F0 31 DF 31 BD 03 7B 4D 26 1A DE 04 A6 00 81 0D 26 B6
S1 13 0200 08 D6 28 27 AB 97 12 20 0A BD 07 45 27 05 86 10 74
S1 13 0210 7E 04 61 96 3E 90 26 97 3E D6 28 26 06 BD 02 EA C5
S1 13 0220 7E 07 41 FE 01 0F DF 37 DE 31 DF 04 BD 02 A5 DF AB
S1 13 0230 22 5D 26 20 5C A6 00 08 81 0D 26 F8 F7 02 4C 86 74
S1 13 0240 FF 50 8D 46 DE 22 BC 0D 4D 27 07 A6 00 A7 00 08 EF
S1 13 0250 20 F4 DE 22 96 12 26 2F FE 0D 4D D6 3E CB 02 F7 59
S1 13 0260 02 6C 8D 26 9C 22 27 07 09 A6 00 A7 00 20 F5 09 09
S1 13 0270 BD 06 68 08 08 DF 22 DE 04 A6 00 08 DF 04 DE 22 CB
S1 13 0280 08 A7 00 81 0D 26 EE 7E 01 B0 FB 0D 4E B9 0D 4D 81
S1 13 0290 D7 3A 97 39 BD 0C B3 24 07 F7 0D 4E B7 0D 4D 39 36
S1 13 02A0 86 90 7E 04 61 96 64 D6 63 CE 0D 4F BC 0D 4D 26 B8
S1 13 02B0 02 5C 39 E1 00 22 0A 26 F8 A1 01 22 04 26 F2 5F 39
S1 13 02C0 39 8D 03 08 20 E6 36 86 0D 08 A1 00 26 FB 32 39 55
S1 13 02D0 BD 01 09 81 08 26 0B 8C 00 68 27 0D 09 7A 00 3E B0
S1 13 02E0 20 EE 81 18 27 03 7C 00 3E 39 DF 22 CE 03 01 A6 CD
S1 13 02F0 00 81 04 27 06 BD 04 4C 08 20 F4 DE 22 7F 00 1D 83
S1 13 0300 39 0D 0A 00 00 00 00 04 81 0D 27 02 81 3A 39 BD 2D
S1 13 0310 0B 51 4F 97 62 97 63 97 64 39 8D F6 97 28 97 27 07
S1 13 0320 97 26 BD 03 68 81 2B 27 07 81 2D 26 04 73 00 27 98
S1 13 0330 08 BD 0C E3 C1 03 27 05 96 27 7E 0B EA 08 97 28 1E
S1 13 0340 84 0F C6 04 78 00 64 79 00 63 79 00 62 5A 26 F4 45
S1 13 0350 9B 64 97 64 7C 00 26 20 D8 DE 04 A6 00 81 20 27 B5
S1 13 0360 07 08 20 F7 8D B4 09 08 A6 00 81 20 27 F9 39 DE 93
S1 13 0370 04 BD 0C E3 C1 02 26 F0 08 20 F6 BD 03 68 DF 04 C7
S1 13 0380 DF 22 CE 01 11 C6 05 A1 00 26 12 DF 0A DE 22 08 F3
S1 13 0390 A6 00 DF 22 DE 0A 08 5A C1 02 26 EB 39 08 5A 26 D3
S1 13 03A0 FC A6 00 27 F7 DF 0A DE 04 DF 22 A6 00 DE 0A 20 0F
S1 13 03B0 D4 CE 00 62 C6 02 0C A6 00 2A 19 86 2D BD 04 4C B8
S1 13 03C0 7C 00 1D 20 0F A6 00 85 F0 25 02 27 07 BD 04 44 EC
S1 13 03D0 7C 00 1D 0D A6 00 C5 FF 27 06 85 0F 25 02 27 07 F3
S1 13 03E0 BD 04 48 7C 00 1D 0D 08 5A 2A DA 39 BD 03 6F 81 0B
S1 13 03F0 0D 27 25 BD 03 1A DF 04 BD 02 A5 DF 22 BD 03 6F 4F
S1 13 0400 81 0D 26 05 7C 00 1B 20 0B 08 BD 03 68 BD 03 1A 63
S1 13 0410 96 64 97 1B DE 22 20 03 CE 0D 4F BC 0D 4D 27 21 81
S1 13 0420 BD 02 EA C6 01 0C 8D 9D A6 00 81 0D 27 05 8D 1C 19
S1 13 0430 08 20 F5 08 96 1B 27 E3 8B 99 19 27 04 97 1B 20 9E
S1 13 0440 DA 7E 01 B0 44 44 44 44 84 0F 8B 30 BD 01 0C 7E F9
S1 13 0450 01 06 36 B6 80 04 2A 02 32 39 B6 80 04 2A FB 86 A5
S1 13 0460 99 36 BD 02 EA CE 04 98 BD 02 EF 32 36 BD 04 44 8B
S1 13 0470 32 BD 04 48 D6 19 26 03 7E 01 B0 CE 04 A1 BD 02 C4
```

```
S1 13 0480 EF DE 04 09 8C 0D 4F 27 07 A6 00 81 0D 26 F4 08 22
S1 13 0490 C6 01 0C BD 03 C5 20 E0 07 45 52 52 4F 52 20 23 2C
S1 13 04A0 04 20 41 54 20 04 BD 03 6F BD 03 08 26 03 7E 05 C8
S1 13 04B0 3C 7F 00 12 81 2C 26 20 D6 1D 86 20 BD 04 4C 5C 76
S1 13 04C0 C5 07 26 F6 C1 47 22 04 D7 1D 20 03 BD 02 EA 7C D6
S1 13 04D0 00 12 08 BD 03 68 20 D1 81 3B 27 F3 81 22 26 05 41
S1 13 04E0 08 8D 64 20 49 7F 00 17 81 54 26 06 97 17 86 41 9A
S1 13 04F0 20 06 81 53 26 2E 86 50 A1 01 26 28 BD 03 71 BD F6
S1 13 0500 0A 26 BD 06 1E D6 64 27 25 96 17 27 07 5A D1 1D 2D
S1 13 0510 23 1C 20 02 DB 1D 86 20 BD 04 4C 7C 00 1D D1 1D 44
S1 13 0520 26 F4 20 0A BD 0A 26 DF 22 BD 03 B1 DE 22 BD 0C 5B
S1 13 0530 DE 5A 26 03 7E 04 A9 86 31 7E 04 61 7D 00 12 26 DC
S1 13 0540 03 BD 02 EA 7E 07 04 A6 00 81 22 27 0E BD 03 08 2C
S1 13 0550 27 0D BD 04 4C 7C 00 1D 08 20 EC 08 7E 03 68 86 32
S1 13 0560 32 7E 04 61 DF 04 BD 0C E5 C1 02 26 2F 7F 00 20 2A
S1 13 0570 16 48 1B 80 13 97 21 DE 20 39 A6 00 08 7F 00 18 37
S1 13 0580 8D E2 5F A6 00 81 0A 27 06 81 0B 27 01 39 5C A6 4C
S1 13 0590 01 36 A6 02 36 37 BD 03 6F 33 81 28 26 71 5D 27 E5
S1 13 05A0 13 08 BD 0A 29 96 64 36 BD 0B 62 BD 03 6F 81 2C 06
S1 13 05B0 26 5D 20 02 4F 36 4C 97 18 08 BD 0A 29 08 DF 04 2F
S1 13 05C0 32 97 14 32 97 21 32 97 20 DE 20 A6 00 97 16 08 1E
S1 13 05D0 08 DF 20 BD 03 0F 96 14 DE 20 09 A1 00 22 30 97 06
S1 13 05E0 64 BD 03 0F 96 16 91 5E 27 02 23 23 8B 01 19 97 8E
S1 13 05F0 64 BD 0B F4 BD 0B CA BD 06 14 96 20 D6 21 DB 64 82
S1 13 0600 99 63 97 20 D7 21 BD 0B 62 DE 20 7F 00 18 39 86 BD
S1 13 0610 14 7E 04 61 BD 03 0F 86 03 97 64 BD 0B F4 96 64 D6
S1 13 0620 36 96 63 36 5F D7 63 D7 64 96 62 8D 12 32 36 8D 01
S1 13 0630 0A 32 8D 0B 32 36 8D 03 32 20 1D 44 44 44 44 8D DE
S1 13 0640 17 D6 63 48 59 37 36 48 59 48 59 97 64 32 D7 63 9F
S1 13 0650 8D 08 32 9B 63 97 63 39 84 0F 9B 64 97 64 24 03 EA
S1 13 0660 7C 00 63 39 96 62 A7 00 96 63 A7 01 96 64 A7 02 8B
S1 13 0670 39 DE 06 DF 37 BD 03 6F BD 03 68 BD 05 64 DF 0A DD
S1 13 0680 BD 03 6F 81 28 26 20 08 BD 03 64 81 29 26 05 4F F8
S1 13 0690 5F 36 20 18 81 2C 26 0F 96 64 27 0B 36 08 BD 03 7D
S1 13 06A0 64 C6 01 81 29 27 05 86 40 7E 04 61 96 64 27 F7 84
S1 13 06B0 36 DF 04 DE 0A 86 0A 1B A7 00 96 08 A7 01 96 09 FE
S1 13 06C0 A7 02 DE 08 32 A7 00 08 33 E7 00 08 DF 20 8B 01 09
S1 13 06D0 19 36 17 8B 01 19 16 BD 03 12 D7 64 BD 03 0F 32 E7
S1 13 06E0 97 64 BD 0B F4 BD 05 F7 BD 0C B1 23 03 7E 02 A0 D6
S1 13 06F0 DF 08 DE 04 08 BD 03 68 BD 03 08 27 07 08 7E 06 7B
S1 13 0700 78 BD 1F 00 4F 97 12 97 1C 97 18 97 2C 96 19 26 9F
S1 13 0710 03 7E 01 B0 DE 04 86 0D C6 3A A1 00 27 07 E1 00 7E
S1 13 0720 27 0A 08 20 F5 08 BC 0D 4D 27 E6 08 08 BD 01 0C 72
S1 13 0730 BD 03 7B 4D 26 0B DE 04 8D 0B 27 05 86 10 7E 04 3E
S1 13 0740 61 EE 00 6E 00 BD 0C E3 C1 02 26 12 08 BD 03 68 11
S1 13 0750 81 3D 27 04 81 28 26 06 CE 01 23 97 1C 5F 39 BD DD
S1 13 0760 01 8B BD 01 95 FE 01 0F DF 06 CE 0D 4F 7C 00 19 F4
S1 13 0770 20 B4 DE 04 96 1C 26 03 BD 03 59 BD 09 65 7E 07 1B
S1 13 0780 04 BD 03 6F BD 0A 26 BD 02 A5 5D 27 05 86 16 7E 3E
S1 13 0790 04 61 5C D7 19 7E 07 26 BD 03 6F 7F 00 13 BD 03 78
S1 13 07A0 68 81 22 26 06 08 BD 05 47 20 3B BD 05 7C DF 33 52
S1 13 07B0 CE 00 68 96 13 26 07 86 3F 97 13 BD 04 4C BD 01 EF
S1 13 07C0 09 81 18 26 05 7F 00 13 20 E6 A7 00 08 81 2C 27 3D
S1 13 07D0 09 81 0D 26 E9 97 12 BD 02 EA CE 00 68 BD 03 1A 0D
S1 13 07E0 DE 33 8D 2D DF 04 81 2C 26 07 08 96 12 27 AF 20 D7
S1 13 07F0 AA BD 03 08 26 13 96 12 27 03 7E 07 04 BD 01 09 28
S1 13 0800 81 0D 26 F9 BD 02 EA 20 F1 86 45 7E 04 61 BD 05 0D
S1 13 0810 7C BD 06 64 7E 03 6F 96 19 27 49 BD 03 6F 97 1A 42
S1 13 0820 DF 0C DF 0E 20 3E 96 19 27 3A 96 1A 27 39 BD 03 AE
```

```
S1 13 0830 59 BD 03 68 BD 05 7C DF 33 DE 04 DF 35 DE 0E BD 44
S1 13 0840 0A 26 A6 00 BD 03 08 26 04 DE 0C 20 01 08 DF 0E DC
S1 13 0850 DE 35 DF 04 DE 33 8D B9 81 2C 26 03 08 20 D2 BD BA
S1 13 0860 03 08 26 03 7E 07 04 86 51 7E 04 61 DF 22 DE 0C 22
S1 13 0870 DF 0E DE 22 20 EE BD 03 59 BD 0A 26 96 64 84 0F E6
S1 13 0880 36 7F 00 12 08 08 A6 00 81 54 27 02 97 12 BD 03 80
S1 13 0890 5B DF 22 32 4A 27 11 E6 00 08 C1 2C 26 04 DF 22 3E
S1 13 08A0 20 F2 C1 0D 26 F1 DE 22 D6 12 27 03 7E 09 32 7E 04
S1 13 08B0 07 84 BD 03 6F BD 0A 26 A6 00 8D 63 26 5C 36 A6 99
S1 13 08C0 01 8D 5C 32 26 04 E6 01 1B 08 08 36 BD 0A 26 32 77
S1 13 08D0 84 0F 80 09 2B 44 48 48 B7 08 E2 BD 0B C4 BD 0C 03
S1 13 08E0 BE 20 FE 2F 18 20 30 26 14 20 2C 2C 10 20 28 2D 5A
S1 13 08F0 0C 20 24 27 08 20 20 2E 04 20 1C 20 1D DE 04 A6 02
S1 13 0900 00 81 54 26 0F BD 03 6F DF 04 BD 0C E5 C1 03 26 2F
S1 13 0910 03 7E 07 84 7E 07 30 7E 07 04 86 62 7E 04 61 81 3D
S1 13 0920 3B 23 06 81 3E 22 02 5F 39 5C 39 D6 19 27 E8 BD 94
S1 13 0930 03 6F 7C 00 1B BD 0A 26 09 BD 02 C6 08 A6 00 36 4B
S1 13 0940 A6 01 36 9F 37 CE A0 23 BD 0C B1 23 03 7E 02 A0 9F
S1 13 0950 7E 07 87 86 73 7A 00 1B 2A 03 7E 04 61 32 33 BD C7
S1 13 0960 02 A9 7E 07 8A BD 05 7A DF 33 BD 03 6F 08 BD 0A 7D
S1 13 0970 26 DE 33 7E 06 64 BD 03 59 36 8D E9 DE 08 DF 37 93
S1 13 0980 DE 06 32 A7 00 96 05 09 A7 00 96 04 09 A7 00 09 08
S1 13 0990 BD 0C B1 22 03 7E 02 A0 DF 06 7E 07 04 BD 03 59 0D
S1 13 09A0 DF 1E DE 06 BC 01 0F 26 04 DE 04 20 74 08 08 08 DE
S1 13 09B0 A1 00 26 F0 09 09 09 DF 06 08 EE 00 DF 04 BD 05 E1
S1 13 09C0 7C DF 33 BD 03 6F BD 0A 26 BD 0B 51 DE 33 BD 0B 87
S1 13 09D0 44 DE 04 A6 00 81 53 27 08 BD 03 0F 4C 97 64 20 0E
S1 13 09E0 0A BD 03 71 BD 0A 26 96 62 97 1C BD 0B CA CE 00 D0
S1 13 09F0 10 BD 06 64 BD 0B C4 BD 0C BE D6 1C 2B 05 06 2C 55
S1 13 0A00 12 20 03 06 2F 0D DE 06 08 08 08 DF 06 DE 1E DF AF
S1 13 0A10 04 20 0B CE 00 10 BD 0B 44 DE 33 BD 06 64 7E 07 FC
S1 13 0A20 04 86 81 7E 04 61 7F 00 2C 96 2C 97 2D 8D 04 4D C5
S1 13 0A30 26 F1 39 9F FE BD 0C DE DF 04 C1 01 26 04 86 21 A8
S1 13 0A40 20 4A 54 C1 03 26 03 BD 03 0F DE 04 BD 0C DE DF C0
S1 13 0A50 04 C1 04 23 02 C6 05 58 F7 0A 5C 20 FE 20 2B 20 9B
S1 13 0A60 1B 20 38 20 2C 20 04 36 08 20 CA 30 09 D6 18 9C B4
S1 13 0A70 FE 27 06 32 5F 81 28 27 78 5D 27 0E 4F D6 2C 5A 31
S1 13 0A80 D1 2D 26 06 30 09 9C FE 27 04 86 20 9E FE DE 04 16
S1 13 0A90 39 BD 0B 51 DE 04 BD 03 1A 20 59 A6 01 BD 0C E5 76
S1 13 0AA0 C1 02 26 28 A6 00 DF 22 CE 01 7B BD 03 85 4D 27 87
S1 13 0AB0 CB 7E 07 41 86 3F 36 DE 22 7E 0A 35 86 40 20 F6 0D
S1 13 0AC0 BD 03 0F BD 0D 2A 97 64 DE 22 20 28 D6 FE 37 D6 3B
S1 13 0AD0 FF 37 D6 2D 37 D6 18 37 BD 05 7A 33 D7 18 33 D7 15
S1 13 0AE0 2D 33 D7 FF 33 D7 FE BD 0B 51 DE 20 BD 0B 44 20 81
S1 13 0AF0 05 DE 04 08 DF 04 30 09 9C FE 27 37 32 36 81 28 DE
S1 13 0B00 27 31 BD 0C E5 37 54 96 2C 4A C1 04 27 04 91 2D 96
S1 13 0B10 27 21 81 09 23 04 86 24 20 16 32 33 80 06 48 B7 0E
S1 13 0B20 0B 26 CE 0B 36 EE 00 AD 00 BD 0C BE 28 C8 86 23 C6
S1 13 0B30 7E 0A 8C 7E 0A 4A 0B CA 0B C4 0C 82 0B BC 0B F4 D3
S1 13 0B40 0C 15 0C 94 A6 00 97 62 A6 01 97 63 A6 02 97 64 FD
S1 13 0B50 39 CE 00 3B E6 03 E7 00 08 8C 00 62 26 F6 7C 00 F1
S1 13 0B60 2C 39 CE 00 64 E6 00 E7 03 09 8C 00 3A 26 F6 7A B5
S1 13 0B70 00 2C 39 0C CE 00 64 A6 00 A9 03 19 A7 00 09 8C 27
S1 13 0B80 00 61 26 F3 37 C6 02 85 F0 26 01 5F DA 30 D7 30 DC
S1 13 0B90 17 33 39 8D 03 0D 20 DC CE 00 67 86 99 A0 00 A7 9A
S1 13 0BA0 00 09 8C 00 64 26 F4 84 0F A7 01 39 7F 00 30 96 75
S1 13 0BB0 65 16 C4 0F D7 65 97 2F 98 62 97 2E D6 62 C4 0F 17
S1 13 0BC0 D7 62 4D 39 96 62 88 F0 97 62 8D 58 8D DE 2A 0A 75
S1 13 0BD0 8D C1 06 28 09 73 00 2F 20 0B 8D 97 20 0A 8D 44 A0
```

```
S1 13 0BE0 BD 03 0F 8D AE 7F 00 30 96 2F 84 F0 C6 0F D4 62 04
S1 13 0BF0 1B 97 62 39 8D 2E 8D B4 BD 03 0F C6 05 96 5F 27 F2
S1 13 0C00 08 BD 0B 73 7A 00 5F 26 F8 5A 27 3D 8D 4A 96 62 19
S1 13 0C10 BD 0B 84 20 E8 8D 0D CE 00 65 BD 0C C1 26 08 86 71
S1 13 0C20 22 7E 0A 8C 7E 0B 62 BD 0B AC BD 0B 51 8D 29 6F ED
S1 13 0C30 02 6F 03 C6 05 8D 21 BD 0B 98 8D 2E 85 F0 26 13 FA
S1 13 0C40 BD 0B 98 0C 8D 25 5A 26 EC 96 2E 8D 9D CE 00 5E FC
S1 13 0C50 7E 0B 65 7C 00 64 20 E2 86 04 CE 00 64 0C 69 00 8F
S1 13 0C60 09 8C 00 5E 26 F8 4A 26 F1 39 0D CE 00 61 96 5F A4
S1 13 0C70 84 0F 97 5F A6 00 A9 06 19 A7 00 09 8C 00 5E 26 B9
S1 13 0C80 F3 39 8D 3A 27 0B D6 62 8D 07 7C 00 64 17 7E 0B EF
S1 13 0C90 EA 7E 03 12 8D 8E 5F D7 30 96 67 27 EB BD 0B 51 2A
S1 13 0CA0 8D 82 8B 99 19 27 16 36 BD 0B AF BD 0B F8 32 20 F8
S1 13 0CB0 F1 DF 39 96 39 91 37 26 04 D6 3A D1 38 39 CE 00 46
S1 13 0CC0 62 5F 6D 02 26 0E 6D 01 26 0A A6 00 84 0F 26 04 BB
S1 13 0CD0 A7 00 C6 04 A6 00 46 84 08 1B 9A 30 06 39 BD 03 43
S1 13 0GE0 68 20 02 A6 00 C6 01 81 0D 27 17 5A 36 80 28 2B DA
S1 13 0CF0 10 81 18 23 0E 81 32 23 06 81 36 26 04 C6 0A CB BE
S1 13 0D00 02 32 39 DF 24 CE 0D 11 B7 0D 0C E6 00 DE 24 20 AB
S1 13 0D10 F0 05 04 0A 06 01 07 00 0B 03 03 03 03 03 03 03 9E
S1 13 0D20 03 03 03 01 01 01 01 01 08 09 C6 08 CE 00 00 A6 5E
S1 13 0D30 03 48 48 48 A8 03 48 48 69 00 69 01 69 02 69 03 EF
S1 10 0D40 5A 26 EC A6 00 81 9F 22 E1 8B 00 19 39 90
S1 04 1F00 39 A3
S9
```