NIBL -- Tiny Basic for National's SC/MP Kit

complete documentation & annotated source code

by Mark Alexander, National Semiconductor Corp.
Nov. 29, 1976

Introduction

NIBL (National Industrial Basic Language) is a machine-oriented programming language for the SC/MP. It is a language similar to Tiny BASIC, but it also has some unique features. Many of these features, such as a genuinely useful control structure (the PASCAL-influenced DO/UNTIL) and the indirect operator ("@") have been added to the language to allow NIBL to be nearly as flexible as machine language in such applications as medium-speed process control.

By using NIBL, one trades the high execution speed and low memory consumption of machine language for some very tangible advantages: Program readability, modifiability, and reliability, which are truly difficult to achieve in machine language programs.

NIBL programs are interpreted by a large (4K byte) SC/MP program that resides in ROM. The interpreter is broken into two blocks: a program written in an Intermediate (or Interpretive) Language – I. L. for short – which does the actual interpretation; and a collection of SC/MP machine language sub-routines invoked by the I. L. program. The I.L. approach is well-documented in Vol. 1, No. 1 of *Dr. Dobb's Journal of Computer Calisthenics & Orthodontia,* and readers should refer to that issue for a more detailed description of the interpretation process.

In Table 1, the formal grammar for NIBL is given. This is the ultimate authority (other than the interpreter itself) on how legal NIBL statements are formed. The following descriptions of the NIBL statements will refer to portions of the grammar. Table 2 contains a list of the error message produced by the NIBL system. Finally, a listing of the interpreter is given in the Appendix.

History of NIBL

NIBL came into this world as an interpreter for Tiny BASIC, as originally described in the first issue of *Dr. Dobb's Journal.* That program was written by Steve Leininger, who subsequently left before the program was ever assembled or executed. The current version of NIBL is an almost complete re-write of the original interpreter, with changes and additions being made to improve the modularity of the program, to greatly increase execution speed, and to extend the capabilities of the language itself.

The program was developed on the PACE Disk Operating System, and was assembled by a PACE-resident cross-assembler for the SC/MP.

System Requirements

The NIBL interpreter is intended to be a ROM-resident program in the first 4K of the SC/MP address space (although it will run just as well in RAM). The interpreter requires at least 2K bytes of RAM starting at address 1000 (base 16), of which the interpreter uses nearly 300 bytes for stacks, variables, etc., leaving the rest for the user's program. Another 2K bytes of memory may be added to fill up this 4K page, forming what is hereafter referred to as "Page 1".

The SC/MP architecture forces memory to be split into pages of 4K bytes each; therefore, NIBL allows seven such pages to be used for storing programs. NIBL programs in the seven pages are edited separately, but may be linked together during program execution by special NIBL statements described below. The first page, mentioned above, must be RAM since the interpreter uses part of it as temporary storage; the part used to store programs starts at location 111E (base 16).

The other six pages, each of which starts at location n000 (base 16), where n is the page number, may be either RAM or ROM. Page 2 is a special page: it can contain a NIBL program to be executed immediately upon powering up the NIBL system.

The memory organization of NIBL is shown in Figure 1.

Throughout this article, the assumption is made that the user has a teletype with paper tape reader and punch, as with the SC/MP Low Cost Development System. In fact, NIBL was designed to use the LCDS teletype interface, but to be completely independent of the LCDS LCDS firmware. If NIBL is to be run on its own, the system should have the same configuration for the teletype, with the reader relay being operated directly by the SC/MP. At present, paper tape is the only medium for saving NIBL programs, but as soon as the hardware and software for a SC/MP cassette interface become availalbe, NIBL will be able to link to routines for saving and loading programs with ease.

Since the teletype interface is not based on a UART, the terminal baud rate can only be changed by modifying the timed delays in NIBL's I/O routines. NIBL has been run successfully at 1200 baud with a CRT terminal; the listing of the program in the Appendix is for a 110 baud system.

Communicating with NIBL

When the NIBL system is ready to accept input, it prompts at the teletype with a ">" sign. (NIBL is now in "edit mode".) The user then enters a line terminated by a carriage return. There are several special characters that are used to edit lines as they are typed:

Shift/0 (back arrow) cuases the last character typed to be deleted.

Control/U (echoes as " U") causes the entire line to be deleted; NIBL reprompts for a new line.

Entering a line to NIBL without a leading line number causes the line to be executed directly by NIBL. Most NIBL statements, as well as the four program control commands, may be executed in this manner.

A line with a leading number (in the range 0 through 32767) is entered into the NIBL program in the current page. (Make sure that the value of the pseudo-variable PAGE is valid, so that the line isn't lost into non-existent memory.) The NIBL editor sorts the program lines as they are entered into ascending order by line number.

Typing a line number followed by a carriage return deletes that line from the program. Typing a line with the same number as an existing line's causes the new line to replace the old one in the program.

Each of the seven memory pages may contain a different program, separate from the rest. Editing the program in one page will not affect the other pages. To switch editing from one page to another, simply type PAGE = n, where n is the number of the new page.

Variables

There are twenty-six variable names in NIBL: the letters A through Z. They are all 16-bit binary variables, so they can be used to hold addresses as well as signed numeric data. The variables are already pre-declared for the user, and space is allocated for them in RAM when NIBL powers up.

Constants

NIBL allows either decimal or hexadecimal (base 16) constants to appear in expressions. Decimal constants must lie in the range 0

through 32767; the unary minus ("—") is used to obtain negative values. The value —32768 is a valid NIBL integer, but it is not legal as it stands. To represent it, use —32767—1 or #8000 instead.

Hexadecimal constants are denoted by a pound sign ("#") followed by a string of hexadecimal digits (0-9, A-F). NIBL does not check for overrun in hex constants; consequently, only the 4 least significant digits of the nex digit string are kept.

Functions

NIBL provides three built-in functions that may appear in any expression. These are described as follows:

RND (X, Y) returns a pseudo-random integer in the range X through T, inclusive, where X and Y are arbitrary expressions.
T, inclusive, where X and Y are arbitrary expressions. In order for the function to work properly, the value of Y — X should be positive and no greater than 32767.

MOD (X, Y) returns the absolute value of the remainder from X divided by Y (where X and Y are expressions).

TOP (with no arguments) returns the address of the first free byte in the memory page currently being edited or executed. In other words, it is the address of the top of the NIBL program in the current page, plus one.

Pseudo-variables

NIBL has two pseudo-variables in addition to the standard variables. These are STAT and PAGE. Both of these varialbes may appear on either side of an assignment statement.

STAT represents the SC/MP status register. The current value of the status register can be referred to by using STAT in an expression; or an assignment may be made to the status register by executing a statement such as STAT = 4 or STAT = STAT OR #20. When NIBL makes an assignment to the status register in this manner, it clears the interrupt-enable bit of the value before it is actually assigned. Note also that only the lower byte of the value is assigned; the high byte is ignored.

The carry and overflow bits in STAT are meaningless since the NIBL system is continually modifying them. The utility of STAT lies in the fact that 5 of its bits are connected to I/O sense lines on the SC/MP chip.

The pseudo-variable PAGE contains the number of the memory page currently being executed or edited. As indicated in Figure 1, there are seven pages in which NIBL programs may be stored; therefore, PAGE may lie only in the range 1 through 7. If an assignment of a value outside this range is made to PAGE, only the 3 least significant bits of the value are used — and zero is automatically changed to one.

If PAGE is modified while NIBL is in edit mode, all subsequent editing will take place in the new page.

If PAGE is modified by a NIBL program during execution, control will be passed to the first line of the NIBL program in the new page. This transfer would be effected by a statement such as PAGE = 6 or PAGE = PAGE + 1. Thus, several NIBL programs residing in different 4K pages may be linked together as one large program, if need be. This would allow one to write a 28K STAR TREK program in NIBL, a Herculean and indeed foolish task.

Control may also be transferred from one page to another by three other statements: RETURN, NEXT, and UNTIL. Thus, the first part of a subroutine or loop may be in one page, and the second part may be in another (with control being transferred between the two parts by an assignment to PAGE). In these three special cases, NIBL automatically updates the value of PAGE as the statements are executed.

Relational Operators

NIBL provides the standard BASIC relational operators, for comparing the values of integer expressions. The operators are as follows:

=	equal to
<=	less than or equal to
>=	greater than or equal to
<>	not equal to
<	less than
>	greater than

All of these operators produce 1 as a result if the relation is true, and 0 if the relation is false. Note that the relational operators may appear anywhere that an expression is called for in the NIBL grammar, not only in IF statements.

Arithmetic Operators

NIBL provides the four standard arithmetic functions: addition (+), subtraction or unary minus (—), multiplication (*), and division (/). Since only integers are allowed in NIBL, all quotients are truncated (the MOD function can be used to obtain remainders from division). Any overflow or underflow (other than division by zero) is ignored by NIBL; the reasoning behind this is that it may often be necessary to treat NIBL expressions as unsigned values, such as when performing calculations using memory addresses as the operands. Thus the value of 32767 + 1 is —32768 (or in hexadecimal, #7FFF + 1 = #8000, which makes more sense).

Logical Operators

In NIBL, there are three logical operations that may be performed on values: AND, OR, and NOT. The first two are binary operators, and the latter is unary. All three perform bitwise logical operations on 16-bit arguments, producing 16-bit results. AND, OR, and NOT are sufficient to simulate any other logical operation, through various combinations of the operators.

The Indirect Operator

The indirect operator "@" realizes the functions of PEEK and POKE operations in other BASICs, but with somewhat more elegance. The "@" sign followed by an address (whcih can be a constant, variable, or expression in parentheses) denotes the contents of that address in memory. Thus, if memory location 245 (decimal) contains 60, the statement X = @245 would result in the value 60 being assigned to X. The indirect operator may also appear on the left side of an assignment statement. For example, @X=@(Y+10) would result in the memory location pointed to by X being assigned the value of the memory location pointed to by the value Y+10.

Despite this, it is still safest to use plenty of parentheses in expressions to make the intent clear.

Use of the indirect operator is not limited to reading from or writing to memory: it also provides a simple way to communicate with peripheral devices that are interfaced to the SC/MP through memory addresses. Note that the "@" operator can only access memory one byte at a time, and that when an assignment is made to a memory location, only the low order byte of the value is moved to the location; the high order byte is ignored.

The indirect operator can also be used to simulate arrays in NIBL. For example, if we wish to define an M x N matrix of one-byte positive integers, we can access the (I,J)th element of the matrix (assuming that (0,0) is a legal element in the matrix) with the expression @(A+I*N+J). An assignment could be made to that same element by placing the expression on the left side of an assignment statement.

Expressions

Expressions in NIBL are made up of the components described above: variables, constants, function references, pseudo-variables, and operators binding them all together. NIBL expressions are all 16-bit integers. Evaluation of expressions takes place left-to-right, and the order in which operations take place is determined by operator precedence and the presence of parentheses. The order of evaluation can be deduced from the grammar in Table 1; here is a table of operator precedence:

Lowest precedence (applied last): <, >, <=, >=, =, <>
+, —, OR
*, /, AND
Highest precedence (applied first): @, NOT

Program Control Commands

LIST causes the entire program in the current page to be listed. Listing can be halted by hitting any key on the teletype: the BREAK key works best.

LIST <number > causes listing to begin at the given line number (or the nearest one greater than the number), rather than at the first line.

LISTing a program is the method used to save it on paper tape. To accomplish this, type LIST with the punch off, then turn on the punch and hit carriage return. After the program is dumped, type a Shift/0 with teletype on LOCAL so that the last character (a " >") will be deleted when the tape is entered to NIBL at a later time. NIBL will accept a tape made in this fashion at any time during edit mode. The tape reader is enabled at all times by NIBL, and it does not distinguish between the reader and the keyboard when accepting input. Superfluous line-feed and null characters on the tape are echoed but ignored.

RUN causes three actions: first, all variables are zeroed; secondly, all stakes (the FOR, DO, and GOSUB stacks) are cleared; and finally the program in the current page is executed, starting with the first line in sequence.

RUN is not the only way to start program execution: GOTO and GOSUB can slo be used to jump into a program from edit mode. For example, if there is a subroutine at line 1000 that is being tested, typing GOSUB 1000 will cause that routine to be executed, with NIBL returning to edit mode upon encourntering a RETURN statement. When GOTO and GOSUB are used to run a program, the variables and stacks are not cleared.

Hitting any key while a program is being run will cause NIBL to break execution, printing a message and the line number where the break was detected. The BREAK key on the teletype works best for this.

CLEAR causes all variables to be zeroed and the three stacks mentioned above to be cleared. This latter feature of the CLEAR command

is quite useful after a stack nesting error has occurred (for example, if GOSUBS are nested more than eight levels deep).

NEW clears the programs in Page 1, and changes the value of PAGE to 1. This is the form of the command most likely to be used by NIBL novices who do not wish to be confused by the page selection features of NIBL. NEW should be the first thing one types in to NIBL when first powering up.

NEW <number > sets the value of PAGE to the <number>, and clears the program in that page.

Assignment Statements

Already, two different types of assignment statements have been mentioned: assignments to the pseudo-variables STAT and PAGE, and assignments to memory locations with the indirect operator. Another form of the assignment statement is the conventional assignment to a variable (A – Z), e.g. A=A + ! or A = 32 <((4 * I). There are also statements which look like string assignments, but there are not standard BASIC, and are described later in the section on string handling. The word "LET" is optional in front of any assignment statement (leaving it out increases execution speed, unlike most Tiny BASIC systems).

If/then Statement

The IF statements allows conditional execution of one or more statements (as many as can fit on one line). The syntax for the IF statement is:

'IF' Rel-exp 'THEN'? Statement

which indicates that the word THEN is optional, and that any statement (including another IF statement) may follow the conditional expression. If the IF condition is true (i.e. is non-zero), the statement following it (and any others on the line) will be executed; otherwise, control immediately transfers to the next program line. The condition does not need to contain relational operators: a statement such as IF MOD (A,5) THEN.... is perfectly legal. In this example, the statement following the THEN would be executed if A were not divisible by 5.

GOTO, GOSUB, AND RETURN STATEMENTS

The syntax for the GOTO statement is 'GOTO' followed by an expression. The effect of the GOTO statement is to transfer control to the line whose number is indicated by the expression. An error occurs if the specified line does not exist in the current page. Unlike standard BASICs, any arbitrary expression can be used to specify the line number, as well as the usual decimal constant. This allows computed branches to be performed with the same effect as the ON . . . GOTO statement in standard BASIC.

The GOSUB statement is identical to the GOTO statement in form. It too causes a branch to a new line, but it also saves the address of the following statement on a stack. When a RETURN statement is executed, the saved address is popped from the stack, and control returns to that point in the program. Since an actual address, not a line number, is saved on the GOSUB stack, GOSUB statements may appear anywhere on a multiple-statement line.

GOSUBs may be nested up to eight levels deep; an error will occur if an attempt is made to exceed this limit. The error condition does not destroy the previous contents of the stack, so a RETURN statement can be executed (even in edit mode) without an error occurring. However, any modification of the NIBL program will clear the GOSUB stack, so that a subsequent RETURN without a GOSUB will cause an error.

DO AND UNTIL STATEMENTS

The DO and UNTIL statements are useful in writing program loops efficiently, without using misleading GOTO statements. Enclosing a group of zero or more statements between a DO statement and an UNTIL <condition> statement (where <condition> is an arbitrary expression) will cause the statement group to be repeated one or more times until the <condition> becomes true (i.e., non-zero). As an example of the use of the DO and UNTIL statements, we present a program that prints the prime numbers:

```
 10 PRINT 1: PRINT 2
 20 I=3
 30 DO
 40       J=I/2: N=2
 50       DO
 60          N=N+2
 70       UNTIL (MOD(I,N=0) OR (N>J)
 80       IF N>J PRINT I
 90       I=I+2
100 UNTIL 0
```

DO loops may be nested up to eight levels deep, and NIBL acts in the same manner if an overflow occurs as it does with a GOSUB overflow. NIBL also reports an error if an UNTIL statements occurs without a previous DO. A single DO loop may have more than one UNTIL statement as a terminator. For example, if one wished to exit abnormally out of a DO loop and transfer to some appropriate line, it could be done in the following manner:

UNTIL 1: GOTO X

where X is the line number.

Neither the DO nor the UNTIL statement may be executed in edit mode.

FOR AND NEXT STATEMENTS

The NIBL FOR statement is virtually identical to that in standard BASICs; consequently, it is not explained in great detail here.

As in most BASICs, both positive and negative STEPs are allowed in the FOR statement, and a STEP of +1 is assumed if the STEP portion of the statement is omitted. A FOR loop is terminated by a NEXT <variable> statement, and the <variable> must be the same as that referred to in the FOR statement at the beginning of the loop.

FOR loops may be nested four levels deep; NIBL reports an error if this limit is exceeded, or if a NEXT statement occurs without a previous FOR statement. As with the DO and UNTIL statements, FOR and NEXT may not be executed in edit mode.

Perhaps the only differences between the NIBL FOR statement and that of more elaborate BASICs (such as DEC's BASIC-PLUS for the PDP-11) are that a FOR loop is always executed at least once, and that when a NEXT statement is executed, the STEP value is added to the variable before the test is made to determine if the loop should be repeated (rather than after the test).

INPUT STATEMENT

There are two types of INPUT statements in NIBL: numeric input and string input. The form of the first type is 'INPUT' followed by a list of one or more variables. When this statement is executed, NIBL prompts at the teletype with a question mark ("?"). The user responds with a list of expressions separated by commas, and terminated by a carriage return. For example, a legal response to the statement INPUT A,B,C would be #3FA,26,4*27. These three expressions would then be assigned to the variables A, B, and C, respectively. An illegal response (too few arguments or improper expressions) will result in a syntax error. Any extra arguments in the response are ignored.

The second type of INPUT statement allows strings to be input. The form of the statement is 'INPUT' '$' <address>, where <address> is a Factor, syntactically (usually a variable, constant, or expression in parentheses). When this statement is executed, NIBL prompts the user as before, at which point the user enters a line terminated by the usual carriage return. NIBL then stores the line in memory in consecutive locations, beginning at the address specified. Thus, INPUT$ #6000 would cause the input line to be stored starting at location 6000 (base 16); the carriage return would also be stored at the end of the line.

Strings input in this manner can be tested and manipulated by using the "@" operator or the string handling statements described below. They can also be displayed by a PRINT statement.

Neither of the two INPUT statements may be executed in edit mode.

PRINT STATEMENT

The form of the PRINT statement is 'PRINT' or 'PR' followed by a list of print items separated by commas, and optionally terminated by a semicolon, which suppresses an otherwise automatic carriage return after all items in the list are printed.

A print item consists of one of the following:

1. A quoted string, which is printed exactly as it appears (with the quotes removed)
2. An expression, which is evaluated and printed in decimal format, with either a leading space or a minus sign ("-"), and one trailing space
3. A reference to a string in memory, denoted by '$' <address>, where <address> is a Factor as usual. Successive memory locations, starting at the specified address, are printed as ASCII characters, until a carriage return (which is not printed) is encountered.

There is no zone spacing in the PRINT statement, nor does NIBL perform an automatic carriage return/line feed after printing 72 characters. NIBL is not an output-oriented language; fancy formatting has been sacrificed for more useful control structures and data manipulation features. (A subroutine to print a number and skip to the next print zone is trivial to write in NIBL — it takes about two lines of code, with the DO/UNTIL and FOR/NEXT.)

STRING HANDLING STATEMENTS

String handling in NIBL is very minimal and low-level. The string handling features of the INPUT and PRINT statements have already been mentioned; NIBL provides two more statements for manipulating strings.

A statement such as $<address> = "THIS IS A STRING" would cause the quoted string to be stored in memory starting at the specified address (which again is a Factor), with a carriage return being appended to the string.

Another statement allows the programmer to move strings around in memory once they have been created. The form of this statement is '$' <destination> '=' '$' <source>, where both <destination> and <source> are Factors, and are the addresses of strings in memory. This statement causes all the characters in the string pointed to by <source> to be copied one-by-one to the memory pointed to by <destination>, until a carriage return (also copied) is encountered. Overlapping the source and destination addresses can produce disastrous results, such as wiping out the entire contents of the current page. Consequently, a string move can be aborted by hitting the BREAK key on the teletype (but it must be done quickly!).

Note that all strings referred to in these statements, and in the INPUT and PRINT statements, are assumed to lie within a 4K page, and wraparound is a possibility which must be anticipated by the programmer. (Long-time SC/MP programmers will be familiar with this minor problem.)

Using these statements, it should be very easy to develop a set of NIBL subroutines for performing concatenation, comparison, and substring operations on strings.

END STATEMENT

The END statement may appear anywhere in a NIBL program and not necessarily at the end. It causes a message and the current line number to be printed, with NIBL returning to edit mode. The END statement is useful when debugging programs, since it acts as a breakpoint in the program that can be removed easily.

LINK STATEMENT

The LINK statement allows NIBL programs to call SC/MP machine language routines at any address. A statement of the form 'LINK' <address>, where <address> is an arbitrary expression, will cause the NIBL system to call the routine at that address by executing an appropriate XPPC P3 instruction. The user's routine should make sure that it returns by executing another XPPC P3, and that the value of P3 upon entry to the routine is restored before returning. The routine may make use of the fact that P2 is set by NIBL to point to the beginning of the RAM block used to store the variables A through Z, with each variable being stored low byte first, high byte second. Thus, parameters may be passed between NIBL programs and machine language routines through the variables. Both P1 and P2 may be modified by the user's routines; they are automatically restored by the NIBL system upon return. The user should be careful not to modify RAM locations with negative displacements relative to P2, or the locations with displacements greater than 51 relative to P2. These locations are used by the interpreter.

REMARK STATEMENT

A comment can be inserted into a NIBL program by preceding it with the word REM. REM causes the rest of the line to be ignored by NIBL during execution. Remarks are useful in debugging programs or helping other people to understand them, but of course, they take up valuable memory. (Then again, memory is getting cheaper all the time.)

MULTIPLE STATEMENTS ON ONE LINE

A program line may contain more than one statement, if the statements are separated by colons (":"). Using multiple statements on a single line improves the readability of the program by separating it into small blocks, and uses less memroy for storing the program.

It is important to note that an IF statement will cause any statements appearing after it on the line to be ignored if the IF condition turns out to be false. This is the feature that allows a group of statements to be executed conditionally.

A multiple-statement line may be entered without a line number but NIBL will only execute the first statement on the line, ignoring the rest.

POWERING UP

NIBL is capable of executing a program in ROM in Page 2 immediately upon powering up, without the need for the user to give the RUN command at the teletype. When NIBL initializes, it examines Page 2 and makes an educated guess about the possible existence of a legal NIBL program in that page. If NIBL thinks there really is a program there, it starts executing it immediately; thus, if the program halts for some reason, the value of PAGE will be 2. But if NIBL fails to find a legal program in Page 2 initially, it sets the value of PAGE to 1 (the normal case) and prompts at the teletype.

When executing programs, NIBL periodically checks for keyboard interrupt, returning to edit mode if it detects it. Therefore, if a NIBL program is to be executed with the teletype disconnected, the Sense B line of the SC/MP should be set high so that NIBL will not sense an interrupt while running. This would allow a NIBL system to act as a process controller which starts executing immediately upon powering up.

BIOGRAPHICAL NOTE

Mark Alexander, a graduate of the University of California, Santa Cruz, is getting bored with assembly language programming, and wishes someone would save him by making a microprocessor copy of the Burroughs B5500 computer.

TABLE 1: NIBL Grammar

On reading the grammar:

All items in single quotes are actual symbols in NIBL; all other identities are symbols in the grammar. The equals sign "=", means "is defined as"; parentheses are used to group several items together as one item; the exclamation point, "!". means an exclusive-or choice between the items on either side of it; the asterisk, "*", means zero or more occurrences of the item to its left; the plus sign, "+", means one or more repetitions; the question mark, "?", means zero or one occurrences; and the semicolon, ";". marks the end of a definition.

```
NIBL-Line = Immediate-Statement
          ! Program-Line
          ;

Immediate-Statement = (Command ! Statement) Carriage-Return;

Program-Line = (Decimal-Number Statement-List Carriage-Return);

Command = 'NEW'
        ! 'CLEAR'
        ! 'LIST' Decimal-Number ?
        ! 'RUN'
        ;

Statement-List = Statement (':' STATEMENT) *;

Statement = 'LET' ? Left-part '=' Rel-Exp
          ! 'LET' ? '$' Factor '=' (String ! '$' Factor)
          ! 'GO' ('TO' ! 'SUB') Rel-Exp
          ! 'RETURN'
          ! ('PR' ! 'PRINT') Print-List
          ! 'IF' Rel-Expr 'THEN' ? Statement
          ! 'DO'
          ! 'UNTIL' Rel-Exp
          ! 'FOR' Variable '=' Rel-Exp 'TO' Rel-Exp ('STEP' Rel-Exp) ?
          ! 'NEXT' Variable
          ! 'INPUT' (Variable + ! '$' Factor)
          ! 'LINK' Rel-Exp
          ! 'REM' Any-Character-Except-Carriage-Return +
          ! 'END'
          ;

Left-Part = (Variable ! '@' Factor ! 'STAT' ! 'PAGE') ;

Rel-Exp = Expression Relop Expression
        ! Expression
        ;

Relop = '<' ! '<' '=' ! '<' '>' ! '>' ! '>' '=' ! '=' ;

Expression = Expression Adding-Operator term
           ! ('+' ! '-') ? Term
           ;

Adding-Operator = '+' ! '-' ! 'OR' ;

Term = Term Multiplying-Operator Factor
     ! Factor
     ;

Multiplying-Operator = '*' ! '/' ! 'AND' ;

Factor = Variable
       ! Decimal-Number
       ! '(' Rel-Exp ')'
       ! '@' Factor
       ! '$' Hex-Number
       ! 'NOT' Factor
       ! 'MOD' '(' Rel-Exp ',' Rel-Exp ')'
       ! 'RND' '(' Rel-Exp ',' Rel-Exp ')'
       ! 'STAT'
       ! 'TOP'
       ! 'PAGE'
       ;

Variable = 'A' ! 'B' ! 'C' ! ... ! 'Y' ! 'Z' ;

Decimal-Number = Decimal-Digit + ;

Decimal-Digit = '0' ! '1' ! '2' ! ... ! '9' ;

Hex-Number = (Decimal-Digit ! Hex-Digit) + ;

Hex-Digit = 'A' ! 'B' ! 'C' ! 'D' ! 'E' ! 'F' ;

Print-list = Print-Item + ;

Print-Item = (String ! Rel-exp ! '$' Factor) ;

String = '"' Almost-Any-Character '"' ;
```

NOTE: Spaces are not usually significant in NIBL programs, with the following exceptions: spaces cannot appear within key words (such as 'THEN' or 'UNTIL') or within constants. Also, a variable (such as A or Z) must be followed immediately by a non-alphabetic character to distinguish it from a key word.

TABLE 1: NIBL Grammar

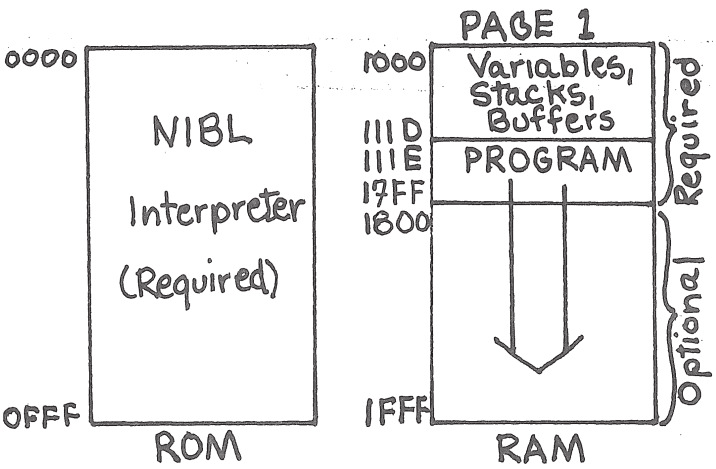
TABLE 2: NIBL error messages

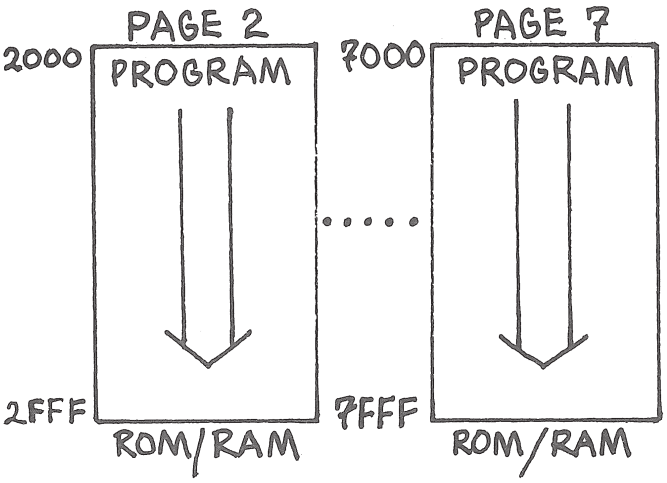
Error messages are of the form:

EEEE ERROR AT LN

where EEEE is one of the error codes below, and LN is the number of of the line in which the error was encountered.

AREA	No more room left for program in current page
CHAR	Character after logical end of statement
DIV0	Division by zero
END"	No ending quote on string
FOR	FOR without NEXT
NEST	Nesting limit exceeded in expression, FOR's, GOSUBs, etc.
NEXT	NEXT without FOR
NOGO	No line number corresponding to GOTO or GOSUB
RTRN	RETURN without previous GOSUB
SNTX	Syntax error
STMT	Statement type used improperly
UNTL	UNTL without DO
VALU	Constant format or value error





CODE FOLLOWS

KILOBAUD – A PRENATEL NAME CHANGE

John Craig, the Editor of Wayne Green's new computer hobby mag, just phoned and told us that Wayne has changed the publication's name – before the first issues comes out – from the initially advertised *"Kilobyte"* to *Kilobaud.* Oh, well . . . we're still waiting for someone to start yet another rag and call it *"Megabyte"* (but with luck, that won't happen).

COMPUTER HOBBYIST CONVENTIONS & TRADE SHOWS

CONVENTIONS ALREADY HELD:

May 2, 1976	Trenton Festival Trenton, NJ Amateur Comp. Group of NJ	1500 people 45 exhibitors
June 11-13, 1976	Midwest Reg. Comp. Conf. Cleveland, OH Midwest Affiliation of Comp. Clubs	1500-2500 people
Aug 28-29, 1976	Personal Computing '76 Atlantic City, NJ S. Counties Amateur Radio Assn. of NJ	4500-5000 people; 103 exhibitors

CONVENTIONS BELIEVED TO BE IN THE WORKS:

Mar 5, 1977 (Saturday) 10 AM - 3 PM	Microprocessor Hobbyists Demo United Good Neighbor Bldg. Renton, WA (Not a convention, but interesting)	Mike & Key Amateur Radio Club Bill Balzarini K7MWC 1518 S. Pearl St. Seattle, WA 98108 (206) 762-7738
Mar 19-20, 1977	Western Personal Computing Show Hyatt House, International Airpt. Los Angeles	Austin Cragg Conference & Exposition Management Co., Box 844 Greenwich, CT 06830 (203) 661-6101
Apr 15-17, 1977	**The First West Coast Computer Faire, Civic Auditorium San Francisco, CA San Francisco, CA [Expecting 7,000-10,000 people, 50 sessions, 200 exhibitors]**	**[co-sponsored by a number of Bay Area hobbyist, professional and educational organizations]**
Apr 31-May 1, 1977	Trenton Computerfest Trenton, NJ	Alan Katz Dept. of Engr., Trenton State Coll., Trenton, NJ 08625 (609) 771-2487
May 7-8, 1977	Eastern Personal Computing Show, Mariott Hotel Philadelphia, PA	Austin Cragg [listed prev.]
Jun 13-16, 1977	**Personal Computing Section National Computer Conference '77 Dallas, TX**	**AFIPS 210 Summit Ave. Montvale, NJ 07645 (201) 391-9810**
Jun 18-19, 1977	New England Personal Comp. Show, J.B. Hynes Aud. Boston, MA	Austin Cragg [listed prev.]
Jun 18-19, 1977	Atlanta Computerfest Atlanta, GA [in conjunction with Hamfest]	? *'73 Magazine* 73 Pine St. Peterborough, NH 03458 (603) 924-3873
Jun, 1977	Midwest Reg. Comp. Conf. Cleveland, OH	Midwest Affiliation of Comp. Clubs, PO Box 83 Brecksville, OH 44141 (216) 732-8458
Jul 29-31, 1977	Northwestern Amateur Radio Convention Seattle Ctr. & Washington Plaza Hotel, Seattle, WA [will include significant microcomputer activities]	ARRL-QCWA-WWDX Club ARRL Conven. Comm. 10352 Sand Point Way NE Seattle, WA 98125
Aug 27-28, 1977	Personal Computing '77 Consumer Trade Fair Atlantic City, NJ [?]	John Dilks, PC'77 503 W. New Jersey Ave. Somers Pt., NJ 08244 (609) 927-6950
Oct 25-28, 1977	(Name unknown at press time) Anaheim Conv. Ctr. Anaheim, CA	*Interface Age* Box 1234 Cerritos, CA 90701 (213) 469-7789
Fall, 1977	(Name unknown at press time) Los Angeles Area [Proposal to hold such a convention has been placed before SCCS Bd. of Directors]	Southern California Computer Society P.O. Box 3123 Los Angeles, CA 90051
???	Technihobby-USA [3 of the 4 listed previously were postponed. Last word was they were considering also postponing the 4th.]	Marketing Ventures, Inc. 5012 Herzel Pl. Beltsville, MD 20705 (301) 937-7177

Note: This list excludes a number of conventions directed towards computer professionals that are expected to have at least nominal activity in the area of personal and hobby computing. Although the '77 NCC is primarily for computer professionals, its Personal Computing Section will be a major activity with a number of significant sessions and events planned for personal computer enthusiasts.

```
                .TITLE  NIBL,'NOV. 27'
0001            .LIST   1

        ;*****************************************************
        ;*                                                   *
        ;*      WE ARE TIED DOWN TO A LANGUAGE WHICH         *
        ;*      MAKES UP IN OBSCURITY WHAT IT LACKS          *
        ;*      IN STYLE.                                    *
        ;*                      -- TOM STOPPARD              *
        ;*                                                   *
        ;*****************************************************

0020    TSTBIT  =       020             ;I.L. INSTRUCTION FLAGS
0040    JMPBIT  =       040
0080    CALBIT  =       080
0001    P1      =       1               ;SC/MP POINTER ASSIGNMENTS
0002    P2      =       2
0003    P3      =       3
FF80    EREG    =       -128            ;THE EXTENSION REGISTER

        ; DISPLACEMENTS FOR RAM VARIABLES USED BY INTERPRETER

FFFF    DOPTR   =       -1              ;DO-STACK POINTER
FFFE    FORPTR  =       -2              ;FOR-STACK POINTER
FFFD    LSTK    =       -3              ;ARITHMETIC STACK POINTER
FFFC    SBRPTR  =       -4              ;GOSUB STACK POINTER
FFFB    PCLOW   =       -5              ;I.L. PROGRAM COUNTER
FFFA    PCHIGH  =       -6
FFF9    PCSTK   =       -7              ;I.L. CALL STACK POINTER
FFF8    LOLINE  =       -8              ;CURRENT LINE NUMBER
FFF7    HILINE  =       -9
FFF6    PAGE    =       -10             ;VALUE OF CURRENT PAGE
FFF5    LISTNG  =       -11             ;LISTING FLAG
FFF4    RUNMOD  =       -12             ;RUN/EDIT FLAG
FFF3    LABLLO  =       -13
FFF2    LABLHI  =       -14
FFF1    P1LOW   =       -15             ;SPACE TO SAVE CURSOR
FFF0    P1HIGH  =       -16
FFEF    LO      =       -17
FFEE    HI      =       -18
FFED    FAILLO  =       -19
FFEC    FAILHI  =       -20
FFEB    NUM     =       -21
FFEA    TEMP    =       -22
FFE9    TEMP2   =       -23
FFE8    TEMP3   =       -24
FFE7    CHRNUM  =       -25
FFE6    RNDF    =       -26
FFE5    RNDX    =       -27             ;SEEDS FOR RANDOM NUMBER
FFE4    RNDY    =       -28

        ; ALLOCATION OF RAM FOR NIBL VARIABLES, STACKS,
        ;  AND LINE BUFFER
0000            .=01000+28
101C    VARS:   .=.+52                  ; NIBL VARIABLES A-Z
1050    AESTK:  .=.+26                  ; ARITHMETIC STACK
106A    SBRSTK: .=.+16                  ;GOSUB STACK
107A    DOSTAK: .=.+16                  ;DO/UNTIL STACK
108A    FORSTK: .=.+28                  ;FOR/NEXT STACK
10A6    PCSTAK: .=.+48                  ;I.L. CALL STACK
10D6    LBUF:   .=.+74                  ;LINE BUFFER
1120    PGM:    .=0                     ;USER'S PROGRAM

                .MACRO  LDPI,P,VAL
                .MLOC   TEMP
                .SET    TEMP,VAL
                LDI     H(TEMP)
                XPAH    P
                LDI     L(TEMP)
                XPAL    P
                .ENDM


        ;*************************************
        ;*      INITIALIZATION OF NIBL       *
        ;*************************************

0000 08         RESET:  NOP
0001            INIT:   LDPI    P2,VARS         ;POINT P2 AT VARIABLES
0007                    LDPI    P1,PGM          ;POINT P1 AT PAGE ONE PROGRAM
000D C4FF               LDI     -1              ;STORE -1 AT START OF PROGRAM
000F C900               ST      0(P1)
0011 C901               ST      1(P1)
0013 C40D               LDI     0D              ;ALSO STORE A DUMMY END-OF-LI
0015 C9FF               ST      -1(P1)
0017 C402               LDI     2               ;POINT P2 AT PAGE 2,
0019 CAF6               ST      PAGE(P2)        ; INITIALLY SET PAGE TO 2
001B 31                 XPAL    P1
001C C420               LDI     020
001E 35                 XPAH    P1
001F B902               DLD     2(P1)           ;CHECK IF THERE IS REALLY
0021 01                 XAE                     ; A PROGRAM IN PAGE 2:
0022 C180               LD      EREG(P1)        ; IF FIRST LINE LENGTH
0024 E40D               XRI     0D              ; POINTS TO CARR. RETURN
0026 9802               JZ      $0              ; AT END OF LINE
0028 BAF6               DLD     PAGE(P2)        ;IF NOT, PAGE = 1
002A C420       $0:     LDI     020
002C 35         $LOOP:  XPAH    P1
002D C4FF               LDI     -1              ;STORE -1 IN 2 CONSECUTIVE
002F C900               ST      (P1)            ; LOCATIONS AT START OF PAGE
0031 C901               ST      1(P1)
0033 C40D               LDI     0D              ;ALSO PUT A DUMMY END-OF-LINE
0035 C9FF               ST      -1(P1)          ; JUST BEFORE TEXT
0037 35                 XPAH    P1              ;UPDATE P1 TO POINT TO
0038 02                 CCL                     ; NEXT PAGE (UNTIL PAGE=8)
0039 F410               ADI     010             ;REPEAT INITIALIZATION
003B E480               XRI     080             ; FOR PAGES 2-7
003D 9804               JZ      $1
003F E480               XRI     080
0041 90E9               JMP     $LOOP
0043 C400       $1:     LDI     0               ;CLEAR SOME FLAGS
0045 CAF4               ST      RUNMOD(P2)
0047 CAF5               ST      LISTNG(P2)
0049 C454               LDI     L(BEGIN)        ;INITIALIZE IL PC SO THAT
004B CAFB               ST      PCLOW(P2)       ; NIBL PROGRAM
004D C40C               LDI     H(BEGIN)        ; IS EXECUTED IMMEDIATELY
004F CAFA               ST      PCHIGH(P2)
0051 C400       CLEAR:  LDI     0
0053 CAEA               ST      TEMP(P2)
0055 01                 XAE
0056 C400       CLEAR1: LDI     0               ;SET ALL VARIABLES
0058 CA80               ST      EREG(P2)        ; TO ZERO
005A AAEA               ILD     TEMP(P2)
005C 01                 XAE
005D C434               LDI
005F 60                 XRE
0060 9CF4               JNZ     CLEAR1
0062 C450               LDI     L(AESTK)        ;INITIALIZE SOME STACKS:
0064 CAFD               ST      LSTK(P2)        ; ARITHMETIC STACK,
0066 C47A               LDI     L(DOSTAK)
0068 CAFF               ST      DOPTR(P2)       ; DO/UNTIL STACK,
006A C46A               LDI     L(SBRSTK)
006C CAFC               ST      SBRPTR(P2)      ; GOSUB STACK,
006E C4A6               LDI     L(PCSTAK)
0070 CAF9               ST      PCSTK(P2)       ; I.L. CALL STACK,
0072 C48A               LDI     L(FORSTK)
0074 CAFE               ST      FORPTR(P2)      ; FOR/NEXT STACK


        ;*************************************
        ;*   INTERMEDIATE LANGUAGE EXECUTOR  *
        ;*************************************

0076 C2FB       EXECIL: LD      PCLOW(P2)       ;SET P3 TO CURRENT
0078 33                 XPAL    P3              ; IL PC.
0079 C2FA               LD      PCHIGH(P2)
007B 37                 XPAH    P3
007C C701       CHEAT:  LD      @1(P3)
007E 01                 XAE                     ;GET NEW IL INSTRUCTION
007F C701               LD      @1(P3)          ; INTO P3 THROUGH
0081 33                 XPAL    P3              ; OBSCURE METHODS
0082 CAFB               ST      PCLOW(P2)       ;SIMULTANEOUSLY, INCREMENT
0084 40                 LDE                     ; THE IL PC BY 2
0085 37                 XPAH    P3
0086 CAFA               ST      PCHIGH(P2)
0088 40                 LDE
0089 D4F0               ANI     0F0             ;CHECK IF IL INSTRUCTION
008B E420               XRI     TSTBIT          ; IS A 'TEST'
008D 9836               JZ      TST
008F E4A0               XRI     CALBIT!TSTBIT   ;CHECK FOR IL CALL
0091 980D               JZ      ILCALL
0093 E4C0               XRI     JMPBIT!CALBIT   ;CHECK FOR IL JUMP
0095 9C06               JNZ     NOJUMP
0097 37                 XPAH    P3              ;*** I.L. JUMP ***
0098 D40F               ANI     0F              ;ALL IT TAKES IS SCRUBBING
009A 37                 XPAH    P3              ; THE JUMP FLAG OFF OF P3
009B 90DF       CHEAT1: JMP     CHEAT
009D 3F         NOJUMP: XPPC    P3              ;MUST BE AN ML SUBROUTINE
009E 90D6               JMP     EXECIL          ; IF NONE OF THE ABOVE


        ;*************************************
        ;*     INTERMEDIATE LANGUAGE CALL    *
        ;*************************************

00A0 C2F9       ILCALL: LD      PCSTK(P2)
00A2 E4D6               XRI     L(LBUF)         ;CHECK FOR STACK OVERFLOW
00A4 9C04               JNZ     ILC1
00A6 C40A               LDI     10
00A8 9063               JMP     E0A
00AA E4D6       ILC1:   XRI     L(LBUF)         ;RESTORE ACCUMULATOR
00AC 33                 XPAL    P3              ;SAVE LOW BYTE OF NEW
00AD CAEA               ST      TEMP(P2)        ; I.L. PC IN TEMP
00AF C410               LDI     H(PCSTAK)       ;POINT P3 AT I.L.
00B1 37                 XPAH    P3              ; SUBROUTINE STACK
00B2 C2FB               LD      PCLOW(P2)       ;SAVE OLD I.L. PC ON STACK
00B4 CF01               ST      @1(P3)
00B6 C2FA               LD      PCHIGH(P2)
00B8 CF01               ST      @1(P3)
00BA C2EA               LD      TEMP(P2)        ;GET LOW BYTE OF NEW
00BC 33                 XPAL    P3              ; I.L. PC INTO P3 LOW
00BD CAF9               ST      PCSTK(P2)       ;UPDATE I.L. STACK POINTER
00BF 40                 LDE                     ;GET HIGH BYTE OF NEW P3
00C0 D40F               ANI     0F              ; I.L. PC INTO P3 HIGH
00C2 37                 XPAH    P3
00C3 90B7               JMP     CHEAT


        ;*************************************
        ;*      I.L. 'TEST' INSTRUCTION      *
        ;*************************************

                .LOCAL
00C5 CAE7       TST:    ST      CHRNUM(P2)      ;CLEAR NUMBER OF CHARS SCANNE
00C7 C501       $SCAN:  LD      @1(P1)          ;SLEW OFF SPACES
00C9 E420               XRI     ' '
00CB 98FA               JZ      $SCAN
00CD C5FF               LD      @-1(P1)         ;REPOSITION CURSOR
00CF C2FA               LD      PCHIGH(P2)      ;POINT P3 AT IL TABLE
00D1 37                 XPAH    P3
00D2 D40F               ANI     0F              ;FAIL ADDRESS <- OLD P3
00D4 CAEC               ST      FAILHI(P2)
00D6 C2FB               LD      PCLOW(P2)
00D8 33                 XPAL    P3
00D9 CAED               ST      FAILLO(P2)
00DB C701       $LOOP:  LD      @1(P3)
00DD 01                 XAE                     ;SAVE CHAR FROM TABLE
00DE BAE7               DLD     CHRNUM(P2)      ;DECREMENT CHAR COUNT
00E0 40                 LDE                     ;GET CHAR BACK
00E1 D47F               ANI     07F             ;SCRUB OFF FLAG (IF ANY)
00E3 E501               XOR     @1(P1)          ;IS CHAR EQUAL TO TEXT CHAR?
00E5 9C07               JNZ     $NEQ            ;NO - END TEST
00E7 40                 LDE                     ;YES - BUT IS IT LAST CHAR?
00E8 94F1               JP      $LOOP           ;IF NOT, CONTINUE TO COMPARE
00EA 9090               JMP     CHEAT           ;IF SO, GET NEXT I.L.
00EC 9088       XO:     JMP     EXECIL          ; INSTRUCTION
00EE C2E7       $NEQ:   LD      CHRNUM(P2)      ;RESTORE P1 TO
00F0 01                 XAE                     ; ORIGINAL VALUE
00F1 C580               LD      @EREG(P1)
00F3 C2ED               LD      FAILLO(P2)      ;LOAD TEST-FAIL ADDRESS
00F5 33                 XPAL    P3              ; INTO P3
00F6 C2EC               LD      FAILHI(P2)
00F8 37                 XPAH    P3
00F9 90A0               JMP     CHEAT1          ;GET NEXT IL INSTRUCTION


        ;*************************************
        ;*       I.L. SUBROUTINE RETURN      *
        ;*************************************
```

```
00FB C410   RTN:    LDI     H(PCSTAK)       ;POINT P3 AT I.L. PC STACK
00FD 37             XPAH    P3
00FE C2F9           LD      PCSTK(P2)
0100 33             XPAL    P3
0101 C7FF           LD      @-1(P3)         ;GET HIGH PART OF OLD PC
0103 01             XAE
0104 C7FF           LD      @-1(P3)         ;GET LOW PART OF OLD PC
0106 33             XPAL    P3
0107 CAF9           ST      PCSTK(P2)       ;UPDATE IL STACK POINTER
0109 40             LDE
010A 37             XPAH    P3              ;P3 NOW HAS OLD IL PC
010B 908E           JMP     CHEAT1
010D 9041   E0A:    JMP     E0

                    ;*************************************
                    ;*    SAVE GOSUB RETURN ADDRESS      *
                    ;*************************************

010F C2FC   SAV:    LD      SBRPTR(P2)
0111 E47A           XRI     L(DOSTAK)       ;CHECK FOR MORE
0113 981C           JZ      SAV2            ; THAN 8 SAVES
0115 AAFC           ILD     SBRPTR(P2)
0117 AAFC           ILD     SBRPTR(P2)
0119 33             XPAL    P3              ;SET P3 TO
011A C410           LDI     H(SBRSTK)       ; SUBROUTINE STACK TOP.
011C 37             XPAH    P3
011D C2F4           LD      RUNMOD(P2)      ;IF IMMEDIATE MODE,
011F 980A           JZ      SAV1            ; SAVE NEGATIVE ADDRESS.
0121 35             XPAH    P1              ;SAVE HIGH PORTION
0122 CBFF           ST      -1(P3)          ; OF CURSOR
0124 35             XPAH    P1
0125 31             XPAL    P1              ;SAVE LOW PORTION
0126 CBFE           ST      -2(P3)          ; OF CURSOR
0128 31             XPAL    P1
0129 90C1           JMP     X0              ;RETURN
012B C4FF   SAV1:   LDI     -1              ;IMMEDIATE MODE
012D CBFF           ST      -1(P3)          ; RETURN ADDRESS IS
012F 90BB           JMP     X0              ; NEGATIVE.
0131 C40A   SAV2:   LDI     10              ;ERROR: MORE THAN
0133 901B           JMP     E0              ; 8 GOSUBS

                    ;*************************************
                    ;*    CHECK STATEMENT FINISHED       *
                    ;*************************************

0135 C501   DONE:   LD      @1(P1)          ;SKIP SPACES
0137 E420           XRI     ' '
0139 98FA           JZ      DONE
013B E42D           XRI     ' ' ! 0D        ;IS IT CARRIAGE RETURN?
013D 9804           JZ      DONE1           ;YES - RETURN
013F E437           XRI     037             ;IS CHAR A ':' ?
0141 9C01           JNZ     DONE2           ;NO - ERROR
0143 3F     DONE1:  XPPC    P3              ;YES - RETURN
0144 C404   DONE2:  LDI     4
0146 9008           JMP     E0

                    ;*************************************
                    ;*      RETURN FROM GOSUB            *
                    ;*************************************

0148 C2FC   RSTR:   LD      SBRPTR(P2)
014A E46A           XRI     L(SBRSTK)       ;CHECK FOR RETURN
014C 9C04           JNZ     RSTR1           ; W/O GOSUB.
014E C409           LDI     9
0150 9043   E0:     JMP     E1              ;GOTO ERROR.
0152 BAFC   RSTR1:  DLD     SBRPTR(P2)
0154 BAFC           DLD     SBRPTR(P2)      ;POP GOSUB STACK,
0156 33             XPAL    P3              ; PUT PTR INTO P3.
0157 C410           LDI     H(SBRSTK)
0159 37             XPAH    P3
015A C301           LD      1(P3)           ;IF ADDRESS NEGATIVE,
015C 9409           JP      RSTR2           ; SUBROUTINE WAS CALLED
015E C402           JS      P3,FIN          ; IN IMMEDIATE MODE,
0165 9085   X1:     JMP     X0              ; SO FINISH UP EXECUTING
0167 35     RSTR2:  XPAH    P1              ;RESTORE CURSOR HIGH
0168 C300           LD      0(P3)
016A 31             XPAL    P1              ;RESTORE CURSOR LOW
016B C401           LDI     1               ;SET RUN MODE
016D CAF4           ST      RUNMOD(P2)
016F 90F4           JMP     X1

                    ;*************************************
                    ;*    TRANSFER TO NEW STATEMENT      *
                    ;*************************************

0171 C2F2   XFER:   LD      LABLHI(P2)      ;CHECK FOR NON-EXISTENT LINE
0173 9404           JP      XFER1
0175 C408           LDI     8
0177 901C           JMP     E1
0179 C401   XFER1:  LDI     1               ;SET RUN MODE TO 1
017B CAF4           ST      RUNMOD(P2)
017D 3F             XPPC    P3

                    ;*************************************
                    ;*    PRINT STRING IN TEXT           *
                    ;*************************************

017E        PRS:    LDPI    P3,PUTC-1       ;POINT P3 AT PUTC ROUTINE
0184 C501           LD      @1(P1)          ;LOAD NEXT CHAR
0186 E422           XRI     '"'             ;IF ", END OF
0188 98DB           JZ      X1              ; STRING
018A E42F           XRI     02F             ;IF CR, ERROR
018C 9805           JZ      PRS1
018E E40D           XRI     0D              ;RESTORE CHAR
0190 3F             XPPC    P3              ;PRINT CHAR
0191 90EB           JMP     PRS             ;GET NEXT CHAR
0193 C407   PRS1:   LDI     7               ;SYNTAX ERROR
0195 9035   E1:     JMP     E2

                    ;*************************************
                    ;*    PRINT NUMBER ON STACK          *
                    ;*************************************

                    ; THIS ROUTINE IS BASED ON DENNIS ALLISON'S BINARY TO DECIMAL
                    ; CONVERSION ROUTINE IN VOL. 1, #1 OF "DR. DOBB'S JOURNAL",
                    ; BUT IS MUCH MORE OBSCURE BECAUSE OF THE STACK MANIPULATION.

                            .LOCAL
0197 C410   PRN:    LDI     H(AESTK)        ;POINT P3 AT A.E. STACK
0199 37             XPAH    P3
019A AAFD           ILD     LSTK(P2)
019C AAFD           ILD     LSTK(P2)
019E 33             XPAL    P3
019F C40A           LDI     10              ;PUT 10 ON STACK (WE'LL BE
01A1 CBFE           ST      -2(P3)          ; DIVIDING BY IT LATER)
01A3 C400           LDI     0
01A5 CBFF           ST      -1(P3)
01A7 C405           LDI     5               ;SET CHRNUM TO POINT TO PLACE
01A9 CAE7           ST      CHRNUM(P2)      ; IN STACK WHERE WE STORE
01AB C4FF           LDI     -1              ; THE CHARACTERS TO PRINT
01AD CB05           ST      5(P3)           ;FIRST CHAR IS A FLAG (-1)
01AF C3FD           LD      -3(P3)          ;CHECK IF NUMBER IS NEGATIVE
01B1 9413           JP      $1
01B3 C42D           LDI     '-'             ;PUT '-' ON STACK, AND NEGATE
01B5 CB04           ST      4(P3)           ; THE NUMBER
01B7 C400           LDI     0
01B9 03             SCL
01BA FBFC           CAD     -4(P3)
01BC CBFC           ST      -4(P3)
01BE C400           LDI     0
01C0 FBFD           CAD     -3(P3)
01C2 CBFD           ST      -3(P3)
01C4 909F           JMP     X1              ;GO DO DIVISION BY 10
01C6 C420   $1:     LDI     ' '             ;IF POSITIVE, PUT ' ' ON
01C8 CB04           ST      4(P3)           ; STACK BEFORE DIVISION
01CA 9099   X4:     JMP     X1
01CC 9057   E2:     JMP     ERR1

                    ; THE DIVISION IS PERFORMED, THEN CONTROL IS TRANSFERRED
                    ; TO PRN1, WHICH FOLLOWS.

01CE AAFD   PRN1:   ILD     LSTK(P2)        ;POINT P1 AT A.E. STACK
01D0 AAFD           ILD     LSTK(P2)
01D2 31             XPAL    P1
01D3 C410           LDI     H(AESTK)
01D5 35             XPAH    P1
01D6 AAE7           ILD     CHRNUM(P2)      ;INCREMENT CHARACTER STACK
01D8 01             XAE                     ; POINTER, PUT IN EX. REG.
01D9 C101           LD      1(P1)           ;GET REMAINDER FROM DIVIDE,
01DB DC30           ORI     '0'
01DD C980           ST      EREG(P1)        ; PUT IT ON THE STACK
01DF C1FD           LD      -3(P1)          ;IS THE QUOTIENT ZERO YET?
01E1 D9FC           OR      -4(P1)
01E3 980A           JZ      $PRNT           ;YES - GO PRINT THE NUMBER
01E5 C40F           LDI     H(PRNUM1)       ;NO - CHANGE THE I.L. PC
01E7 CAFA           ST      PCHIGH(P2)      ; SO THAT DIVIDE IS
01E9 C42F           LDI     L(PRNUM1)       ; PERFORMED AGAIN
01EB CAFB           ST      PCLOW(P2)
01ED 90DB           JMP     X4              ;GO DO DIVISION BY 10 AGAIN
01EF        $PRNT:  LDPI    P3,PUTC-1       ;POINT P3 AT PUTC ROUTINE
01F5 C2F5           LD      LISTNG(P2)      ;IF LISTING, SKIP PRINTING
01F7 9C06           JNZ     $2              ; LEADING SPACE
01F9 C104           LD      4(P1)           ;PRINT EITHER '-'
01FB 3F             XPPC    P3              ; OR LEADING SPACE
01FC C2E7           LD      CHRNUM(P2)      ;GET EX. REG. VALUE BACK
01FE 01             XAE
01FF C580   $2:     LD      @EREG(P1)       ;POINT P3 AT FIRST CHAR
0201 C100           LD      (P1)            ; TO BE PRINTED
0203 3F     $LOOP:  XPPC    P3              ;PRINT THE CHARACTER
0204 C5FF           LD      @-1(P1)         ;GET NEXT CHARACTER
0206 94FB           JP      $LOOP           ;REPEAT UNTIL = -1
0208 C450           LDI     L(AESTK)
020A CAFD           ST      LSTK(P2)        ;CLEAR THE A.E. STACK
020C C2F5           LD      LISTNG(P2)      ;PRINT A TRAILING SPACE
020E 9CBA           JNZ     X4              ; IF NOT LISTING PROGRAM
0210 C420           LDI     ' '
0212 3F             XPPC    P3
0213 90B5           JMP     X4

                    ;*************************************
                    ;*    CARRIAGE RETURN/LINE FEED      *
                    ;*************************************

0215        NLINE:  LDPI    P3,PUTC-1       ;POINT P3 AT PUTC ROUTINE
021B C40D           LDI     0D              ;CARRIAGE RETURN
021D 3F             XPPC    P3
021E C40A           LDI     0A              ;LINE FEED
0220 3F             XPPC    P3
0221 90A7   X5:     JMP     X4

                    ;*************************************
                    ;*    ERROR  ROUTINE                 *
                    ;*************************************

                            .LOCAL
0223 C405   ERR:    LDI     5               ;SYNTAX ERROR
0225 CAEB   ERR1:   ST      NUM(P2)         ;SAVE ERROR #
0227 C2EB   ERR2:   LD      NUM(P2)
0229 CAEA           ST      TEMP(P2)
022B                LDPI    P3,PUTC-1       ;POINT P3 AT PUTC
0231 C40D           LDI     0D              ;PRINT CR/LF
0233 3F             XPPC    P3
0234 C40A           LDI     0A
0236 3F             XPPC    P3
0237                LDPI    P1,MESGS        ;P1 -> ERROR MESSAGES
023D BAEB   $1:     DLD     NUM(P2)         ;IS THIS THE RIGHT MESSAGE?
023F 9806           JZ      $MSG            ;YES - GO PRINT IT
0241 C501   $LOOP:  LD      @1(P1)          ;NO - SCAN THROUGH TO
0243 94FC           JP      $LOOP           ; NEXT MESSAGE
0245 90F6           JMP     $1
0247 C501   $MSG:   LD      @1(P1)          ;GET MESSAGE CHAR
0249 3F             XPPC    P3              ;PRINT IT
024A C1FF           LD      -1(P1)          ;IS MESSAGE DONE?
024C 94F9           JP      $MSG            ;NO - GET NEXT CHAR
024E C2EA           LD      TEMP(P2)        ;WAS THIS A BREAK MESSAGE?
0250 E40E           XRI     14
0252 980D           JZ      $3              ;YES - SKIP PRINTING 'ERROR'
0254                LDPI    P1,MESGS        ;NO - PRINT 'ERROR'
025A C501   $2:     LD      @1(P1)          ;GET CHARACTER
025C 3F             XPPC    P3              ;PRINT IT
025D C1FF           LD      -1(P1)          ;DONE?
```

```
025F 94F9            JP      $2              ;NO - REPEAT LOOP
0261 C2F4  $3:       LD      RUNMOD(P2)      ;DON'T PRINT LINE #
0263 984D            JZ      FIN             ; IF IMMEDIATE MODE
0265 C420            LDI     ' '
0267 3F              XPPC    P3              ;SPACE
0268 C441            LDI     'A'             ;AT
026A 3F              XPPC    P3
026B C454            LDI     'T'
026D 3F              XPPC    P3
026E C410            LDI     H(AESTK)        ;POINT P3 AT A.E.  STACK
0270 37              XPAH    P3
0271 AAFD            ILD     LSTK(P2)
0273 AAFD            ILD     LSTK(P2)
0275 33              XPAL    P3
0276 C2F7            LD      HILINE(P2)      ;GET HIGH BYTE OF LINE #
0278 CBFF            ST      -1(P3)          ;PUT ON STACK
027A C2F8            LD      LOLINE(P2)      ;GET LOW BYTE OF LINE #
027C CBFE            ST      -2(P3)          ;PUT ON STACK
027E C42D            LDI     L(ERRNUM)       ;GO TO PRN
0280 CAFB            ST      PCLOW(P2)
0282 C40E            LDI     H(ERRNUM)
0284 CAFA            ST      PCHIGH(P2)
0286 9099  X5A:      JMP     X5

           ;************************************
           ;*      BREAK, NXT, FIN, & STRT     *
           ;************************************

0288 C40E  BREAK:    LDI     14
028A 9099  E3A:      JMP     ERR1
                                             ;*** NEXT STATEMENT ***
028C C2F4  NXT:      LD      RUNMOD(P2)      ;IF IN IMMED.  MODE,
028E 9822            JZ      FIN             ; STOP EXECUTION
0290 C100            LD      (P1)            ;IF WE HIT END OF FILE,
0292 D480            ANI     080             ; FINISH UP THINGS
0294 9C1C            JNZ     FIN
0296 06              CSA                     ;BREAK IF SOMEONE IS
0297 D420            ANI     020             ; TYPING ON THE CONSOLE
0299 98ED            JZ      BREAK
029B C1FF            LD      -1(P1)          ;GET LAST CHARACTER SCANNED
029D E40D            XRI     0D              ;WAS IT CARRIAGE RETURN?
029F 9C08            JNZ     NXT1            ;YES - SKIP FOLLOWING UPDATES
02A1 C501            LD      @1(P1)          ;GET HIGH BYTE OF NEXT LINE #
02A3 CAF7            ST      HILINE(P2)      ;SAVE IT
02A5 C502            LD      @2(P1)          ;GET LOW BYTE OF LINE #, SKIP
02A7 CAF8            ST      LOLINE(P2)      ; LINE LENGTH BYTE
02A9 C40C  NXT1:     LDI     H(STMT)         ;GO TO 'STMT' IN IL TABLE
02AB CAFA            ST      PCHIGH(P2)
02AD C482            LDI     L(STMT)
02AF CAFB            ST      PCLOW(P2)
02B1 3F              XPPC    P3

02B2 C400  FIN:      LDI     0               ;*** FINISH EXECUTION ***
02B4 CAF4            ST      RUNMOD(P2)      ;CLEAR RUN MODE
02B6 C450            LDI     L(AESTK)        ;CLEAR ARITHMETIC STACK
02B8 CAFD            ST      LSTK(P2)
02BA C418            LDI     L(START)        ;SET IL PC TO GETTING LINES
02BC CAFB            ST      PCLOW(P2)
02BE C40C            LDI     H(START)
02C0 CAFA            ST      PCHIGH(P2)
02C2 C4A6            LDI     L(PCSTAK)
02C4 CAF9            ST      PCSTK(P2)
02C6 90BE            JMP     X5A
                                             ;*** START EXECUTION ***
02C8 AAF4  STRT:     ILD     RUNMOD(P2)      ;RUN MODE = 1
02CA C2E9            LD      TEMP2(P2)       ;POINT CURSOR TO
02CC 35              XPAH    P1              ; START OF NIBL PROGRAM
02CD C2E8            LD      TEMP3(P2)
02CF 31              XPAL    P1
02D0 C46A            LDI     L(SBRSTK)       ;EMPTY SOME STACKS:
02D2 CAFC            ST      SBRPTR(P2)      ; GOSUB STACK,
02D4 C48A            LDI     L(FORSTK)
02D6 CAFE            ST      FORPTR(P2)      ; FOR STACK
02D8 C47A            LDI     L(DOSTAK)
02DA CAFF            ST      DOPTR(P2)       ; & DO/UNTIL STACK
02DC 3F              XPPC    P3              ;RETURN
02DD 90A7  X6:       JMP     X5A
02DF 90A9  E4:       JMP     E3A

           ;************************************
           ;*       LIST NIBL PROGRAM          *
           ;************************************

02E1 C100  LST:      LD      (P1)            ;CHECK FOR END OF FILE
02E3 E480            XRI     080
02E5 9418            JP      LST2
02E7 C410            LDI     H(AESTK)        ;GET LINE NUMBER ONTO STACK
02E9 37              XPAH    P3
02EA AAFD            ILD     LSTK(P2)
02EC AAFD            ILD     LSTK(P2)
02EE 33              XPAL    P3
02EF C501            LD      @1(P1)
02F1 CBFF            ST      -1(P3)
02F3 C501            LD      @1(P1)
02F5 CBFE            ST      -2(P3)
02F7 C501            LD      @1(P1)          ;SKIP OVER LINE LENGTH
02F9 C401            LDI     1
02FB CAF5            ST      LISTNG(P2)      ;SET LISTING FLAG
02FD 90DE            JMP     X6              ;GO PRINT LINE NUMBER
02FF C400  LST2:     LDI     0
0301 CAF5            ST      LISTNG(P2)      ;CLEAR LISTING FLAG
0303 C402            JS      P3,NXT          ;GO TO NXT
030A 90D1  X6A:      JMP     X6
030C 90D1  E5:       JMP     E4
030E       LST3:     LDPI    P3,PUTC-1       ;POINT P3 AT PUTC
0314 06    LST4:     CSA
0315 D420            ANI     020
0317 98E6            JZ      LST2            ;IF TYPING, STOP
0319 C501            LD      @1(P1)          ;GET NEXT CHAR
031B E40D            XRI     0D              ;TEST FOR CR
031D 9805            JZ      LST5
031F E40D            XRI     0D              ;GET CHARACTER
0321 3F              XPPC    P3              ;PRINT CHARACTER
0322 90F0            JMP     LST4
0324 C40D  LST5:     LDI     0D              ;CARRIAGE RETURN
0326 3F              XPPC    P3
0327 C40A            LDI     0A              ;LINE FEED
0329 3F              XPPC    P3
032A 02              CCL
032B C447            LDI     L(LIST3)
032D CAFB            ST      PCLOW(P2)
032F C40C            LDI     H(LIST3)
0331 CAFA            ST      PCHIGH(P2)
0333 90AC            JMP     LST             ;GET NEXT LINE

           ;************************************
           ;*          ADD AND SUBTRACT        *
           ;************************************

0335 C410  ADD:      LDI     H(AESTK)        ;SET P3 TO CURRENT-
0337 37              XPAH    P3              ; STACK LOCATION
0338 BAFD            DLD     LSTK(P2)
033A BAFD            DLD     LSTK(P2)
033C 33              XPAL    P3
033D 02              CCL
033E C3FE            LD      -2(P3)          ;REPLACE TWO TOP ITEMS
0340 F300            ADD     0(P3)           ; ON STACK BY THEIR SUM
0342 CBFE            ST      -2(P3)
0344 C3FF            LD      -1(P3)
0346 F301            ADD     1(P3)
0348 CBFF            ST      -1(P3)
034A 90BE  X7:       JMP     X6A

034C C410  SUB:      LDI     H(AESTK)        ;SET P3 TO CURRENT
034E 37              XPAH    P3              ; STACK LOCATION
034F BAFD            DLD     LSTK(P2)
0351 BAFD            DLD     LSTK(P2)
0353 33              XPAL    P3
0354 03              SCL
0355 C3FE            LD      -2(P3)          ;REPLACE TWO TOP ITEMS
0357 FB00            CAD     0(P3)           ; ON STACK BY THEIR DIFFERENC
0359 CBFE            ST      -2(P3)
035B C3FF            LD      -1(P3)
035D FB01            CAD     1(P3)
035F CBFF            ST      -1(P3)
0361 90A7            JMP     X6A

           ;************************************
           ;*          NEGATE                  *
           ;************************************

0363 C410  NEG:      LDI     H(AESTK)        ;SET P3 TO CURRENT
0365 37              XPAH    P3              ; STACK LOCATION
0366 C2FD            LD      LSTK(P2)
0368 33              XPAL    P3
0369 03              SCL
036A C400            LDI     0
036C FBFE            CAD     -2(P3)          ;NEGATE TOP ITEM ON STACK
036E CBFE            ST      -2(P3)
0370 C400            LDI     0
0372 FBFF            CAD     -1(P3)
0374 CBFF            ST      -1(P3)
0376 90D2  X8:       JMP     X7
0378 9092  E6:       JMP     E5

           ;************************************
           ;*          MULTIPLY                *
           ;************************************

                     .LOCAL
037A C410  MUL:      LDI     H(AESTK)        ;SET P3 TO CURRENT
037C 37              XPAH    P3              ; STACK LOCATION
037D C2FD            LD      LSTK(P2)
037F 33              XPAL    P3              ;DETERMINE SIGN OF PRODUCT,
0380 C3FF            LD      -1(P3)          ; SAVE IN TEMP(P2)
0382 E3FD            XOR     -3(P3)
0384 CAEA            ST      TEMP(P2)
0386 C3FF            LD      -1(P3)          ;CHECK FOR NEGATIVE
0388 940D            JP      $1              ; MULTIPLIER
038A 03              SCL
038B C400            LDI     0               ;IF NEGATIVE,
038D FBFE            CAD     -2(P3)          ; NEGATE
038F CBFE            ST      -2(P3)
0391 C400            LDI     0
0393 FBFF            CAD     -1(P3)
0395 CBFF            ST      -1(P3)
0397 C3FD  $1:       LD      -3(P3)          ;CHECK FOR NEGATIVE
0399 940D            JP      $2              ; MULTIPLICAND
039B 03              SCL
039C C400            LDI     0               ;IF NEGATIVE,
039E FBFC            CAD     -4(P3)          ; NEGATE
03A0 CBFC            ST      -4(P3)
03A2 C400            LDI     0
03A4 FBFD            CAD     -3(P3)
03A6 CBFD            ST      -3(P3)
03A8 C400  $2:       LDI     0               ;CLEAR WORKSPACE
03AA CB00            ST      0(P3)
03AC CB01            ST      1(P3)
03AE CB02            ST      2(P3)
03B0 CB03            ST      3(P3)
03B2 C410            LDI     16              ;SET COUNTER TO 16
03B4 CAEB            ST      NUM(P2)
03B6 C3FF  $LOOP:    LD      -1(P3)          ;ROTATE MULTIPLIER
03B8 1F              RRL                     ; RIGHT ONE BIT
03B9 CBFF            ST      -1(P3)
03BB C3FE            LD      -2(P3)
03BD 1F              RRL
03BE CBFE            ST      -2(P3)
03C0 06              CSA                     ;CHECK FOR CARRY BIT
03C1 9411            JP      $3              ;IF NOT SET, DON'T DO ADD
03C3 02              CCL
03C4 C302            LD      2(P3)           ;ADD MULTIPLICAND
03C6 F3FC            ADD     -4(P3)          ; INTO WORKSPACE
03C8 CB02            ST      2(P3)
03CA C303            LD      3(P3)
03CC F3FD            ADD     -3(P3)
03CE CB03            ST      3(P3)
03D0 9002            JMP     $3
03D2 90A4  E6A:      JMP     E6
03D4 02    $3:       CCL
03D5 C303            LD      3(P3)           ;SHIFT WORKSPACE RIGHT BY 1
03D7 1F              RRL
03D8 CB03            ST      3(P3)
03DA C302            LD      2(P3)
```

```
03DC 1F           RRL
03DD CB02         ST      2(P3)
03DF C301         LD      1(P3)
03E1 1F           RRL
03E2 CB01         ST      1(P3)
03E4 C300         LD      0(P3)
03E6 1F           RRL
03E7 CB00         ST      0(P3)
03E9 BAEB         DLD     NUM(P2)         ;DECREMENT COUNTER
03EB 9CC9         JNZ     $LOOP           ;LOOP IF NOT ZERO
03ED 9002         JMP     $4
03EF 9085   X9:   JMP     X8
03F1 C2EA   $4:   LD      TEMP(P2)        ;CHECK SIGN WORD
03F3 940D         JP      $EXIT           ;IF BIT7 = 1, NEGATE PRODUCT
03F5 03           SCL
03F6 C400         LDI     0
03F8 FB00         CAD     0(P3)
03FA CB00         ST      0(P3)
03FC C400         LDI     0
03FE FB01         CAD     1(P3)
0400 CB01         ST      1(P3)
0402 C300   $EXIT: LD     0(P3)           ;PUT PRODUCT ON TOP
0404 CBFC         ST      -4(P3)          ; OF STACK
0406 C301         LD      1(P3)
0408 CBFD         ST      -3(P3)
040A BAFD         DLD     LSTK(P2)        ;SUBTRACT 2 FROM
040C BAFD         DLD     LSTK(P2)        ; LSTK
040E 90DF         JMP     X9

                  ;************************************
                  ;*            DIVIDE                *
                  ;************************************

                  .LOCAL
0410 C410   DIV:  LDI     H(AESTK)
0412 37           XPAH    P3
0413 C2FD         LD      LSTK(P2)
0415 33           XPAL    P3
0416 C3FF         LD      -1(P3)          ;CHECK FOR DIVISION BY 0
0418 DBFE         OR      -2(P3)
041A 9C04         JNZ     $0
041C C40D         LDI     13
041E 90B2         JMP     E6A
0420 C3FD   $0:   LD      -3(P3)
0422 E3FF         XOR     -1(P3)
0424 CAEA         ST      TEMP(P2)        ;SAVE SIGN OF QUOTIENT
0426 C3FD         LD      -3(P3)          ;IS DIVIDEND POSITIVE?
0428 9411         JP      $POS            ;YES - JUMP
042A C400         LDI     0
042C 03           SCL
042D FBFC         CAD     -4(P3)          ;NO - NEGATE DIVIDEND,
042F CB03         ST      3(P3)           ; STORE IN RIGHT HALF
0431 C400         LDI     0               ; OF 32-BIT ACCUMULATOR
0433 FBFD         CAD     -3(P3)
0435 CB02         ST      2(P3)
0437 900A         JMP     $1
0439 90B4   X9A:  JMP     X9
043B C3FD   $POS: LD      -3(P3)          ;STORE NON-NEGATED DIVIDEND
043D CB02         ST      2(P3)           ; IN 32-BIT ACCUMULATOR
043F C3FC         LD      -4(P3)
0441 CB03         ST      3(P3)
0443 C3FF   $1:   LD      -1(P3)          ;CHECK FOR NEGATIVE DIVISOR
0445 940D         JP      $2
0447 C400         LDI     0               ;NEGATE DIVISOR
0449 03           SCL
044A FBFE         CAD     -2(P3)
044C CBFE         ST      -2(P3)
044E C400         LDI     0
0450 FBFF         CAD     -1(P3)
0452 CBFF         ST      -1(P3)
0454 C400   $2:   LDI     0               ;PUT ZERO IN:
0456 CB01         ST      1(P3)           ;  LEFT HALF OF 32-BIT ACC,
0458 CB00         ST      0(P3)
045A CAEB         ST      NUM(P2)         ;  THE COUNTER, AND
045C CBFD         ST      -3(P3)          ;  IN THE DIVIDEND, NOW USED
045E CBFC         ST      -4(P3)          ;  STORE THE QUOTIENT
0460 02     $LOOP: CCL                    ;BEGIN MAIN DIVIDE LOOP:
0461 C3FC         LD      -4(P3)          ;  SHIFT QUOTIENT LEFT,
0463 F3FC         ADD     -4(P3)
0465 CBFC         ST      -4(P3)
0467 C3FD         LD      -3(P3)
0469 F3FD         ADD     -3(P3)
046B CBFD         ST      -3(P3)
046D 02           CCL                     ;  SHIFT 32-BIT ACC LEFT,
046E C303         LD      3(P3)
0470 F303         ADD     3(P3)
0472 CB03         ST      3(P3)
0474 C302         LD      2(P3)
0476 F302         ADD     2(P3)
0478 CB02         ST      2(P3)
047A C301         LD      1(P3)
047C F301         ADD     1(P3)
047E CB01         ST      1(P3)
0480 C300         LD      (P3)
0482 F300         ADD     (P3)
0484 CB00         ST      (P3)
0486 03           SCL
0487 C301         LD      1(P3)           ;  SUBTRACT DIVISOR INTO
0489 FBFE         CAD     -2(P3)          ;   LEFT HALF OF ACC,
048B CB01         ST      1(P3)
048D C300         LD      (P3)
048F FBFF         CAD     -1(P3)
0491 CB00         ST      (P3)
0493 9411         JP      $ENT1           ;  IF RESULT IS NEGATIVE,
0495 02           CCL                     ;   RESTORE ORIGINAL CONTENTS
0496 C301         LD      1(P3)           ;   OF ACC BY ADDING DIVISOR
0498 F3FE         ADD     -2(P3)
049A CB01         ST      1(P3)
049C C300         LD      (P3)
049E F3FF         ADD     -1(P3)
04A0 CB00         ST      (P3)
04A2 9008         JMP     $3
04A4 9093   X9B:  JMP     X9A
04A6 C3FC   $ENT1: LD     -4(P3)          ;  ELSE IF RESULT POSITIVE,
04A8 DC01         ORI     1               ;   RECORD A 1 IN QUOTIENT
04AA CBFC         ST      -4(P3)          ;   W/O RESTORING THE ACC
04AC AAEB   $3:   ILD     NUM(P2)         ;  INCREMENT THE COUNTER
04AE E410         XRI     16              ;  ARE WE DONE?
04B0 9CAE         JNZ     $LOOP           ;  LOOP IF NOT DONE
04B2 C2EA         LD      TEMP(P2)        ;CHECK THE QUOTIENT'S SIGN,
04B4 940D         JP      $END            ; NEGATING IF NECESSARY
04B6 C400         LDI     0
04B8 03           SCL
04B9 FBFC         CAD     -4(P3)
04BB CBFC         ST      -4(P3)
04BD C400         LDI     0
04BF FBFD         CAD     -3(P3)
04C1 CBFD         ST      -3(P3)
04C3 BAFD   $END: DLD     LSTK(P2)        ;DECREMENT THE STACK POINTER,
04C5 BAFD         DLD     LSTK(P2)
04C7 90DB         JMP     X9B             ; AND EXIT

                  ;************************************
                  ;*          STORE VARIABLE          *
                  ;************************************

04C9 C410   STORE: LDI    H(AESTK)        ;SET P3 TO STACK
04CB 37           XPAH    P3
04CC C2FD         LD      LSTK(P2)
04CE 33           XPAL    P3
04CF C7FD         LD      @-3(P3)         ;GET VARIABLE INDEX
04D1 01           XAE                     ;PUT IN E REG
04D2 C301         LD      1(P3)
04D4 CA80         ST      EREG(P2)        ;STORE LOWER 8 BITS
04D6 02           CCL                     ; INTO VARIABLE
04D7 40           LDE                     ;INCREMENT INDEX
04D8 F401         ADI     1
04DA 01           XAE
04DB C302         LD      2(P3)
04DD CA80         ST      EREG(P2)        ;STORE UPPER 8 BITS
04DF 33           XPAL    P3              ; INTO VARIABLE
04E0 CAFD         ST      LSTK(P2)        ;UPDATE STACK POINTER
04E2 C400   X10:  JS      P3,EXECIL

                  ;************************************
                  ;*     TEST FOR VARIABLE IN TEXT    *
                  ;************************************

04E9 C501   TSTVAR: LD    @1(P1)
04EB E420         XRI     ' '             ;SLEW OFF SPACES
04ED 98FA         JZ      TSTVAR
04EF C1FF         LD      -1(P1)          ;GET CHARACTER IN QUESTION
04F1 03           SCL
04F2 FC5B         CAI     'Z'+1           ;SUBTRACT 'Z'+1
04F4 9405         JP      $FAIL           ;NOT VARIABLE IF POSITIVE
04F6 03           SCL
04F7 FCE6         CAI     'A'-'Z'-1       ;SUBTRACT 'A'
04F9 9412         JP      $MAYBE          ;IF POS, MAY BE VARIABLE
04FB C5FF   $FAIL: LD     @-1(P1)         ;BACKSPACE CURSOR
04FD C2FB         LD      PCLOW(P2)       ;GET TEST-FAIL ADDRESS
04FF 33           XPAL    P3              ; FROM I.L. TABLE, PUT IT
0500 C2FA         LD      PCHIGH(P2)      ; INTO I.L. PROGRAM COUNTER
0502 37           XPAH    P3
0503 C300         LD      (P3)
0505 CAFA         ST      PCHIGH(P2)
0507 C301         LD      1(P3)
0509 CAFB         ST      PCLOW(P2)
050B 90D5         JMP     X10
050D 01     $MAYBE: XAE                   ;SAVE VALUE (0-25)
050E C100         LD      (P1)            ;CHECK FOLLOWING CHAR
0510 03           SCL                     ;MUST NOT BE A LETTER
0511 FC5B         CAI     'Z'+1           ; OTHERWISE WE'D BE LOOKING
0513 9405         JP      $OK             ; AT A KEYWORD, NOT VARIABLE
0515 03           SCL
0516 FCE6         CAI     'A'-'Z'-1
0518 94E1         JP      $FAIL
051A C410   $OK:  LDI     H(AESTK)        ;SET P3 TO CURRENT
051C 37           XPAH    P3              ; STACK LOCATION
051D AAFD         ILD     LSTK(P2)        ;INCR STACK POINTER
051F 33           XPAL    P3
0520 02           CCL                     ;DOUBLE VARIABLE INDEX
0521 40           LDE
0522 70           ADE
0523 CBFF         ST      -1(P3)          ;PUT INDEX ON STACK
0525 C402         LDI     2               ;INCREMENT I.L. PC, SKIPPING
0527 02           CCL                     ; OVER TEST-FAIL ADDRESS
0528 F2FB         ADD     PCLOW(P2)
052A CAFB         ST      PCLOW(P2)
052C C400         LDI     0
052E F2FA         ADD     PCHIGH(P2)
0530 CAFA         ST      PCHIGH(P2)
0532 90AE         JMP     X10

                  ;************************************
                  ;*    IND -- EVALUATE A VARIABLE    *
                  ;************************************

0534 C410   IND:  LDI     H(AESTK)        ;SET P3 TO STACK
0536 37           XPAH    P3
0537 AAFD         ILD     LSTK(P2)
0539 33           XPAL    P3
053A C3FE         LD      -2(P3)          ;GET INDEX OFF TOP
053C 01           XAE                     ;PUT INDEX IN E REG
053D C280         LD      EREG(P2)        ;GET LOWER 8 BITS
053F CBFE         ST      -2(P3)          ;SAVE ON STACK
0541 02           CCL
0542 40           LDE                     ;INCREMENT E REG
0543 F401         ADI     1
0545 01           XAE
0546 C280         LD      EREG(P2)        ;GET UPPER 8 BITS
0548 CBFF         ST      -1(P3)          ;SAVE ON STACK
054A 9096   X11:  JMP     X10

                  ;************************************
                  ;*     RELATIONAL OPERATORS         *
                  ;************************************

054C C401   EQ:   LDI     1               ;EACH RELATIONAL OPERATOR
054E 9012         JMP     CMP             ; LOADS A NUMBER USED LATER
0550 C402   NEQ:  LDI     2               ; AS A CASE SELECTOR, AFTER
0552 900E         JMP     CMP             ; THE TWO OPERANDS ARE COM-
0554 C403   LSS:  LDI     3               ; PARED.  BASED ON THE COM-
0556 900A         JMP     CMP             ; PARISON, FLAGS ARE SET THAT
0558 C404   LEQ:  LDI     4               ; ARE EQUIVALENT TO THOSE SET
055A 9006         JMP     CMP             ; BY THE 'CMP' INSTRUCTION IN
```

```
055C C405   GTR:    LDI     5               ; THE PDP-11.  THESE PSEUDO-
055E 9002           JMP     CMP             ; FLAGS ARE USED TO DETERMINE
0560 C406   GEQ:    LDI     6               ; WHETHER THE PARTICULAR
                                            ; RELATION IS SATISFIED OR NO
0562 CAEB   CMP:    ST      NUM(P2)
0564 C410           LDI     H(AESTK)        ;SET P3 -> ARITH STACK
0566 37             XPAH    P3
0567 BAFD           DLD     LSTK(P2)
0569 BAFD           DLD     LSTK(P2)
056B 33             XPAL    P3
056C 03             SCL
056D C3FE           LD      -2(P3)          ;SUBTRACT THE TWO OPERANDS,
056F FB00           CAD     (P3)            ; STORING RESULT IN LO & HI
0571 CAEF           ST      LO(P2)
0573 C3FF           LD      -1(P3)
0575 FB01           CAD     1(P3)
0577 CAEE           ST      HI(P2)
0579 E3FF           XOR     -1(P3)          ;OVERFLOW OCCURS IF SIGNS OF
057B 01             XAE                     ; RESULT AND 1ST OPERAND
057C C3FF           LD      -1(P3)          ; DIFFER, AND SIGNS OF THE
057E E301           XOR     1(P3)           ; TWO OPERANDS DIFFER
0580 50             ANE                     ;BIT 7 EQUIVALENT TO V FLAG
0581 E2EE           XOR     HI(P2)          ;BIT 7 EQUIVALENT TO N XOR V
0583 CAEA           ST      TEMP(P2)        ;STORE IN TEMP
0585 C2EE           LD      HI(P2)          ;DETERMINE IF RESULT WAS ZERO
0587 DAEF           OR      LO(P2)
0589 9802           JZ      SETZ            ;IF RESULT=0, SET Z FLAG
058B C480           LDI     080             ; ELSE CLEAR Z FLAG
058D E480   SETZ:   XRI     080
058F 01             XAE                     ;BIT 7 OF EX = Z FLAG

0590 BAEB           DLD     NUM(P2)         ;TEST FOR =
0592 9C05           JNZ     NEQ1
0594 40             LDE                     ; EQUAL IF Z = 1
0595 902B           JMP     CMP1
0597 90B1   X12:    JMP     X11
0599 BAEB   NEQ1:   DLD     NUM(P2)         ;TEST FOR <>
059B 9C05           JNZ     LSS1
059D 40             LDE                     ; NOT EQUAL IF Z = 0
059E E480           XRI     080
05A0 9020           JMP     CMP1
05A2 BAEB   LSS1:   DLD     NUM(P2)         ;TEST FOR <
05A4 9C04           JNZ     LEQ1
05A6 C2EA           LD      TEMP(P2)        ; LESS THAN IF (N XOR V)=1
05A8 9018           JMP     CMP1
05AA BAEB   LEQ1:   DLD     NUM(P2)         ;TEST FOR <=
05AC 9C05           JNZ     GTR1
05AE 40             LDE                     ; LESS THAN OR EQUAL
05AF DAEA           OR      TEMP(P2)        ;  IF (Z OR (N XOR V))=1
05B1 900F           JMP     CMP1
05B3 BAEB   GTR1:   DLD     NUM(P2)         ;TEST FOR >
05B5 9C07           JNZ     GEQ1
05B7 40             LDE                     ; GREATER THAN
05B8 DAEA           OR      TEMP(P2)        ;  IF (Z OR (N XOR V))=0
05BA E480           XRI     080
05BC 9004           JMP     CMP1
05BE C2EA   GEQ1:   LD      TEMP(P2)        ;GREATER THAN OR EQUAL
05C0 E480           XRI     080             ; IF (N XOR V)=0
05C2 9404   CMP1:   JP      FALSE           ;IS RELATION SATISFIED?
05C4 C401           LDI     1               ;YES - PUSH 1 ON STACK
05C6 9002           JMP     CMP2
05C8 C400   FALSE:  LDI     0               ;NO - PUSH 0 ON STACK
05CA CBFE   CMP2:   ST      -2(P3)
05CC C400           LDI     0
05CE CBFF           ST      -1(P3)
05D0 C400           JS      P3,RTN          ;DO AN I.L.  RETURN
05D7 90BE           JMP     X12

                ;*************************************
                ;*   IF STATEMENT TEST FOR ZERO      *
                ;*************************************

05D9 C2EF   CMPR:   LD      LO(P2)          ;GET LOW & HI BYTES OF EXPR.
05DB DAEE           OR      HI(P2)          ;TEST IF EXPRESSION IS ZERO
05DD 9802           JZ      FAIL            ;YES - IT IS
05DF 90B6           JMP     X12             ;NO - IT ISN'T SO CONTINUE
05E1 C501   FAIL:   LD      @1(P1)          ;SKIP TO NEXT LINE IN PROGRAM
05E3 E40D           XRI     0D              ; (I.E.  TIL NEXT CR)
05E5 9CFA           JNZ     FAIL
05E7 C402           JS      P3,NXT          ;CALL NXT AND RETURN
05EE 90A7   X12A:   JMP     X12

                ;*************************************
                ;*        AND, OR, & NOT           *
                ;*************************************

                .LOCAL
05F0 C401   ANDOP:  LDI     1               ;EACH OPERATION HAS ITS
05F2 9006           JMP     $1              ; OWN CASE SELECTOR.
05F4 C402   OROP:   LDI     2
05F6 9002           JMP     $1
05F8 C403   NOTOP:  LDI     3
05FA CAEB   $1:     ST      NUM(P2)
05FC C410           LDI     H(AESTK)        ;SET P3 -> ARITH.  STACK
05FE 37             XPAH    P3
05FF BAFD           DLD     LSTK(P2)
0601 BAFD           DLD     LSTK(P2)
0603 33             XPAL    P3
0604 BAEB           DLD     NUM(P2)         ;TEST FOR 'AND'
0606 9C0E           JNZ     $OR
0608 C301           LD      1(P3)           ;REPLACE TWO TOP ITEMS ON
060A D3FF           AND     -1(P3)          ; STACK BY THEIR 'AND'
060C CBFF           ST      -1(P3)
060E C300           LD      0(P3)
0610 D3FE           AND     -2(P3)
0612 CBFE           ST      -2(P3)
0614 90D8           JMP     X12A
0616 BAEB   $OR:    DLD     NUM(P2)         ;TEST FOR 'OR'
0618 9C0E           JNZ     $NOT
061A C301           LD      1(P3)           ;REPLACE TWO TOP ITEMS ON
061C DBFF           OR      -1(P3)          ; STACK BY THEIR 'OR'
061E CBFF           ST      -1(P3)
0620 C300           LD      0(P3)
0622 DBFE           OR      -2(P3)
0624 CBFE           ST      -2(P3)
0626 90C6           JMP     X12A
0628 C701   $NOT:   LD      @1(P3)          ;'NOT' OPERATION
062A E4FF           XRI     0FF
062C CBFF           ST      -1(P3)          ;REPLACE TOP ITEM ON STACK
062E C701           LD      @1(P3)          ; BY ITS ONE'S COMPLEMENT
0630 E4FF           XRI     0FF
0632 CBFF           ST      -1(P3)
0634 33             XPAL    P3
0635 CAFD           ST      LSTK(P2)        ;STACK POINTER FIXUP
0637 90B5   X12B:   JMP     X12A

                ;*************************************
                ;*     EXCHANGE CURSOR WITH RAM      *
                ;*************************************

0639 C2F1   XCHGP1: LD      P1LOW(P2)       ;THIS ROUTINE IS HANDY WHEN
063B 31             XPAL    P1              ; EXECUTING AN 'INPUT' STMT
063C CAF1           ST      P1LOW(P2)       ; IT EXCHANGES THE CURRENT
063E C2F0           LD      P1HIGH(P2)      ; TEXT CURSOR WITH ONE SAVED
0640 35             XPAH    P1              ; IN RAM
0641 CAF0           ST      P1HIGH(P2)
0643 3F             XPPC    P3

                ;*************************************
                ;*         CHECK RUN MODE            *
                ;*************************************

0644 C2F4   CKMODE: LD      RUNMOD(P2)      ;THIS ROUTINE CAUSES AN ERROR
0646 9801           JZ      CK1             ; IF CURRENTLY IN EDIT MODE
0648 3F             XPPC    P3
0649 C403   CK1:    LDI     3
064B CAEB   E8:     ST      NUM(P2)         ;ERROR IF RUN MODE = 0
064D C402           JS      P3,ERR2         ;MINOR KLUGE

                ;*************************************
                ;*     GET HEXADECIMAL NUMBER        *
                ;*************************************

                .LOCAL
0654 AAFD   HEX:    ILD     LSTK(P2)        ;POINT P3 AT ARITH STACK
0656 AAFD           ILD     LSTK(P2)
0658 33             XPAL    P3
0659 C410           LDI     H(AESTK)
065B 37             XPAH    P3
065C C400           LDI     0               ;NUMBER INITIALLY ZERO
065E CBFF           ST      -1(P3)          ;PUT IT ON STACK
0660 CBFE           ST      -2(P3)
0662 CAEB           ST      NUM(P2)         ;ZERO NUMBER OF DIGITS
0664 C501   $SKIP:  LD      @1(P1)          ;SKIP ANY SPACES
0666 E420           XRI     ' '
0668 98FA           JZ      $SKIP
066A C5FF           LD      @-1(P1)
066C C100   $LOOP:  LD      (P1)            ;GET A CHARACTER
066E 03             SCL
066F FC3A           CAI     '9'+1           ;CHECK FOR A NUMERIC CHAR
0671 9409           JP      $LETR
0673 03             SCL
0674 FCF6           CAI     '0'-'9'-1       ;IF NUMERIC, SHIFT NUMBER
0676 9413           JP      $ENTER          ; AND ADD NEW HEX DIGIT
0678 9032           JMP     $END
067A 90BB   X12C:   JMP     X12B
067C 03     $LETR:  SCL                     ;CHECK FOR HEX LETTER
067D FC0D           CAI     'G'-'9'-1
067F 942B           JP      $END
0681 03             SCL
0682 FCFA           CAI     'A'-'G'
0684 9402           JP      $OK
0686 9024           JMP     $END
0688 02     $OK:    CCL                     ;ADD 10 TO GET TRUE VALUE
0689 F40A           ADI     10              ; OF LETTER
068B 01     $ENTER: XAE                     ;NEW DIGIT IN EX REG
068C C404           LDI     4               ;SET SHIFT COUNTER
068E CAEA           ST      TEMP(P2)
0690 CAEB           ST      NUM(P2)         ;DIGIT COUNT IS NON-ZERO
0692 C3FE   $SHIFT: LD      -2(P3)          ;SHIFT NUMBER LEFT BY 4
0694 02             CCL
0695 F3FE           ADD     -2(P3)
0697 CBFE           ST      -2(P3)
0699 C3FF           LD      -1(P3)
069B F3FF           ADD     -1(P3)
069D CBFF           ST      -1(P3)
069F BAEA           DLD     TEMP(P2)
06A1 9CEF           JNZ     $SHIFT
06A3 C3FE           LD      -2(P3)          ;ADD NEW DIGIT
06A5 58             ORE                     ; INTO NUMBER
06A6 CBFE           ST      -2(P3)
06A8 C501           LD      @1(P1)          ;ADVANCE THE CURSOR
06AA 90C0           JMP     $LOOP           ;GET NEXT CHAR
06AC C2EB   $END:   LD      NUM(P2)         ;CHECK IF THERE WERE
06AE 9C87           JNZ     X12B            ; MORE THAN 0 CHARACTERS
06B0 C405           LDI     5               ;ERROR IF THERE WERE NONE
06B2 9097   E8B:    JMP     E8

                ;*************************************
                ;*     TEST FOR NUMBER IN TEXT       *
                ;*************************************

                ; THIS ROUTINE TESTS FOR A NUMBER IN THE TEXT.  IF NO
                ; NUMBER IS FOUND, I.L. CONTROL PASSES TO THE ADDRESS
                ; INDICATED IN THE 'TSTN' INSTRUCTION.  OTHERWISE, THE
                ; NUMBER IS SCANNED AND PUT ON THE ARITHMETIC STACK,
                ; WITH I.L. CONTROL PASSING TO THE NEXT INSTRUCTION.

                .LOCAL
06B4 C501   TSTNUM: LD      @1(P1)
06B6 E420           XRI     ' '             ;SKIP OVER ANY SPACES
06B8 98FA           JZ      TSTNUM
06BA C5FF           LD      @-1(P1)         ;GET FIRST CHAR
06BC 03             SCL                     ;TEST FOR DIGIT
06BD FC3A           CAI     '9'+1
06BF 9405           JP      $ABORT
06C1 03             SCL
06C2 FCF6           CAI     '0'-'9'-1
06C4 9421           JP      $1
06C6 C2FB   $ABORT: LD      PCLOW(P2)       ;GET TEST-FAIL ADDRESS
06C8 33             XPAL    P3              ; FROM I.L. TABLE
06C9 C2FA           LD      PCHIGH(P2)
06CB 37             XPAH    P3
```

```
06CC C300          LD      (P3)            ;PUT TEST-FAIL ADDRESS
06CE CAFA          ST      PCHIGH(P2)      ; INTO I.L. PC
06D0 C301          LD      1(P3)
06D2 CAFB          ST      PCLOW(P2)
06D4 90A4          JMP     X12C
06D6 C402  $RET:   LDI     2               ;SKIP OVER ONE IL INSTRUCTION
06D8 02            CCL                     ; IF NUMBER IS DONE
06D9 F2FB          ADD     PCLOW(P2)
06DB CAFB          ST      PCLOW(P2)
06DD C400          LDI     0
06DF F2FA          ADD     PCHIGH(P2)
06E1 CAFA          ST      PCHIGH(P2)
06E3 9095  X13:    JMP     X12C
06E5 90CB  E8A:    JMP     E8B
06E7 01    $1:     XAE                     ;SAVE DIGIT IN EX REG
06E8 C410          LDI     H(AESTK)        ;POINT P3 AT AE STACK
06EA 37            XPAH    P3
06EB AAFD          ILD     LSTK(P2)
06ED AAFD          ILD     LSTK(P2)
06EF 33            XPAL    P3
06F0 C400          LDI     0
06F2 CBFF          ST      -1(P3)
06F4 40            LDE
06F5 CBFE          ST      -2(P3)
06F7 C501  $LOOP:  LD      @1(P1)          ;GET NEXT CHAR
06F9 C100          LD      (P1)
06FB 03            SCL                     ;TEST IF IT IS DIGIT
06FC FC3A          CAI     '9'+1
06FE 94D6          JP      $RET            ;RETURN IF IT ISN'T
0700 03            SCL
0701 FCF6          CAI     '0'-'9'-1
0703 9402          JP      $2
0705 90CF          JMP     $RET
0707 01    $2:     XAE                     ;SAVE DIGIT
0708 C3FF          LD      -1(P3)          ;PUT RESULT IN SCRATCH SPACE
070A CB01          ST      1(P3)
070C C3FE          LD      -2(P3)
070E CB00          ST      (P3)
0710 C402          LDI     2
0712 CAEA          ST      TEMP(P2)        ;MULTIPLY RESULT BY 10
0714 02    $SHIFT: CCL                     ;FIRST MULTIPLY BY 4
0715 C3FE          LD      -2(P3)
0717 F3FE          ADD     -2(P3)
0719 CBFE          ST      -2(P3)
071B C3FF          LD      -1(P3)
071D F3FF          ADD     -1(P3)
071F CBFF          ST      -1(P3)
0721 BAEA          DLD     TEMP(P2)
0723 9CEF          JNZ     $SHIFT
0725 02            CCL                     ;THEN ADD OLD RESULT,
0726 C3FE          LD      -2(P3)          ; SO WE HAVE RESULT * 5
0728 F300          ADD     (P3)
072A CBFE          ST      -2(P3)
072C C3FF          LD      -1(P3)
072E F301          ADD     1(P3)
0730 CBFF          ST      -1(P3)
0732 02            CCL                     ;THEN MULTIPLY BY TWO
0733 C3FE          LD      -2(P3)
0735 F3FE          ADD     -2(P3)
0737 CBFE          ST      -2(P3)
0739 C3FF          LD      -1(P3)
073B F3FF          ADD     -1(P3)
073D CBFF          ST      -1(P3)
073F 02            CCL                     ;THEN ADD IN NEW DIGIT
0740 40            LDE
0741 F3FE          ADD     -2(P3)
0743 CBFE          ST      -2(P3)
0745 C400          LDI     0
0747 F3FF          ADD     -1(P3)
0749 CBFF          ST      -1(P3)
074B 94AA          JP      $LOOP           ;REPEAT IF NO OVERFLOW
074D C406          LDI     6
074F 9094  E9:     JMP     E8A             ;ELSE REPORT ERROR
0751 9090  X14:    JMP     X13

                   ;*************************************
                   ;*    GET LINE FROM TELETYPE         *
                   ;*************************************

                   .LOCAL
0753       GETL:   LDPI    P1,LBUF         ;SET P1 TO LBUF
0759 C400          LDI     0               ;CLEAR NO. OF CHAR
075B CAE7          ST      CHRNUM(P2)
075D               LDPI    P3,PUTC-1       ;POINT P3 AT PUTC ROUTINE
0763 C2F4          LD      RUNMOD(P2)      ;PRINT '? ' IF RUNNING
0765 9808          JZ      $0              ; (I.E. DURING 'INPUT')
0767 C43F          LDI     '?'
0769 3F            XPPC    P3
076A C420          LDI     ' '
076C 3F            XPPC    P3
076D 9003          JMP     $1
076F C43E  $0:     LDI     '>'             ;OTHERWISE PRINT '>'
0771 3F            XPPC    P3
0772 C40F  $1:     JS      P3,GECO         ;GET CHARACTER
0779 C4BD          LDI     L(PUTC)-1       ;POINT P3 AT PUTC AGAIN
077B 33            XPAL    P3
077C 40            LDE                     ;GET TYPED CHAR
077D 98F3          JZ      $1              ;IGNORE NULLS
077F E40A          XRI     0A              ;IGNORE LINE FEED
0781 98EF          JZ      $1
0783 40            LDE
0784 E40D          XRI     0D              ;CHECK FOR CR
0786 9850          JZ      $CR
0788 40            LDE
0789 E45F          XRI     'O'+010         ;CHECK FOR SHIFT/O
078B 9841          JZ      $RUB
078D 40            LDE                     ;CHECK FOR CTRL/H
078E E408          XRI     8
0790 9836          JZ      $XH
0792 40            LDE
0793 E415          XRI     015             ;CHECK FOR CTRL/U
0795 980F          JZ      $XU
0797 40            LDE
0798 E403          XRI     3               ;CHECK FOR CTRL/C
079A 9C1A          JNZ     $ENTER
079C C45E          LDI     '^'             ;ECHO CONTROL/C AS ^C
079E 3F            XPPC    P3
079F C443          LDI     'C'
07A1 3F            XPPC    P3
07A2 C40E          LDI     14              ;CAUSE A BREAK
07A4 90A9          JMP     E9
07A6 C45E  $XU:    LDI     '^'             ;ECHO CONTROL/U AS ^U
07A8 3F            XPPC    P3
07A9 C455          LDI     'U'
07AB 3F            XPPC    P3
07AC C40D          LDI     0D              ;PRINT CR/LF
07AE 3F            XPPC    P3
07AF C40A          LDI     0A
07B1 3F            XPPC    P3
07B2 909F  $2:     JMP     GETL            ;GO GET ANOTHER LINE
07B4 909B  X15:    JMP     X14
07B6 40    $ENTER: LDE
07B7 CD01          ST      @1(P1)          ;PUT CHAR IN LBUF
07B9 AAE7          ILD     CHRNUM(P2)      ;INCREMENT CHRNUM
07BB E448          XRI     72              ;IF=72, LINE FULL
07BD 9CB3          JNZ     $1
07BF C40D          LDI     0D
07C1 01            XAE                     ;SAVE CARRIAGE RET
07C2 40            LDE
07C3 3F            XPPC    P3              ;PRINT IT
07C4 9012          JMP     $CR             ;STORE IT IN LBUF
07C6 9087  E10:    JMP     E9
07C8 C420  $XH:    LDI     ' '             ;BLANK OUT THE CHARACTER
07CA 3F            XPPC    P3
07CB C408          LDI     8               ;PRINT ANOTHER BACKSPACE
07CD 3F            XPPC    P3
07CE C2E7  $RUB:   LD      CHRNUM(P2)
07D0 98A0          JZ      $1
07D2 BAE7          DLD     CHRNUM(P2)      ;ONE LESS CHAR
07D4 C5FF          LD      @-1(P1)         ;BACKSPACE CURSOR
07D6 909A          JMP     $1
07D8 40    $CR:    LDE
07D9 CD01          ST      @1(P1)          ;STORE CR IN LBUF
07DB C40A          LDI     0A              ;PRINT LINE FEED
07DD 3F            XPPC    P3
07DE C410          LDI     H(LBUF)         ;SET P1 TO BEGIN-
07E0 35            XPAH    P1              ; NING OF LBUF
07E1 C4D6          LDI     L(LBUF)
07E3 31            XPAL    P1
07E4 90CE  X16:    JMP     X15

                   ;*************************************
                   ;*      EVAL -- GET MEMORY CONTENTS  *
                   ;*************************************

                   ; THIS ROUTINE IMPLEMENTS THE '@' OPERATOR IN EXPRESSIONS

07E6 C410  EVAL:   LDI     H(AESTK)
07E8 37            XPAH    P3
07E9 C2FD          LD      LSTK(P2)
07EB 33            XPAL    P3              ;P3 -> ARITH STACK
07EC C3FF          LD      -1(P3)          ;GET ADDR OFF STACK,
07EE 35            XPAH    P1              ; AND INTO P1,
07EF 01            XAE                     ; SAVING OLD P1 IN EX & LO
07F0 C3FE          LD      -2(P3)
07F2 31            XPAL    P1
07F3 CAEF          ST      LO(P2)
07F5 C100          LD      0(P1)           ;GET MEMORY CONTENTS,
07F7 CBFE          ST      -2(P3)          ; SHOVE ONTO STACK
07F9 C400          LDI     0
07FB CBFF          ST      -1(P3)          ;HIGH ORDER 8 BITS ZEROED
07FD C2EF          LD      LO(P2)
07FF 31            XPAL    P1              ;RESTORE ORIGINAL P1
0800 40            LDE
0801 35            XPAH    P1
0802 90B0          JMP     X15

                   ;*************************************
                   ;*    MOVE -- STORE INTO MEMORY      *
                   ;*************************************

                   ; THIS ROUTINE IMPLEMENTS THE STATEMENT:
                   ;     '@' FACTOR '=' REL-EXP

0804 C410  MOVE:   LDI     H(AESTK)
0806 37            XPAH    P3
0807 C2FD          LD      LSTK(P2)
0809 33            XPAL    P3              ;P3 -> ARITH STACK
080A C7FE          LD      @-2(P3)         ;GET BYTE TO BE MOVED
080C 01            XAE
080D C7FF          LD      @-1(P3)         ;NOW GET ADDRESS INTO P3
080F CAEA          ST      TEMP(P2)
0811 C7FF          LD      @-1(P3)
0813 33            XPAL    P3
0814 CAFD          ST      LSTK(P2)        ;STACK PTR UPDATED NOW
0816 C2EA          LD      TEMP(P2)
0818 37            XPAH    P3
0819 40            LDE
081A CB00          ST      0(P3)           ;MOVE THE BYTE INTO MEMORY
081C 90C6  X17:    JMP     X16
081E 90A6  E11:    JMP     E10

                   ;*************************************
                   ;*          TEXT EDITOR              *
                   ;*************************************

                   ;INPUTS TO THIS ROUTINE: POINTER TO LINE BUFFER IN P1LOW &
                   ; P1HIGH.  P1 POINTS TO THE INSERTION POINT IN THE TEXT.
                   ; THE A.E. STACK HAS THE LINE NUMBER ON IT (STACK POINTER
                   ; IS ALREADY POPPED).

                   ;EACH LINE IN THE NIBL TEXT IS STORED IN THE FOLLOWING
                   ; FORMAT: TWO BYTES CONTAINING THE LINE NUMBER (IN BINARY,
                   ; HIGH ORDER BYTE FIRST), THEN ONE BYTE CONTAINING THE
                   ; LENGTH OF THE LINE, AND FINALLY THE LINE ITSELF FOLLOWED
                   ; BY A CARRIAGE RETURN.  THE LAST LINE IN THE TEXT IS
                   ; FOLLOWED BY TWO CONSECUTIVE BYTES OF X'FF.

                   .LOCAL
0820 C410  INSRT:  LDI     H(AESTK)        ;POINT P3 AT AE STACK,
0822 37            XPAH    P3              ; WHICH HAS THE LINE #
0823 C2FD          LD      LSTK(P2)        ; ON IT
0825 33            XPAL    P3
0826 C301          LD      1(P3)           ;SAVE NEW LINE'S NUMBER
0828 CAF7          ST      HILINE(P2)
082A C300          LD      0(P3)
```

```
082C CAF8          ST    LOLINE(P2)
082E C2F1          LD    P1LOW(P2)      ;PUT POINTER TO LBUF INTO P3
0830 33            XPAL  P3
0831 C2F0          LD    P1HIGH(P2)
0833 37            XPAH  P3
0834 C404          LDI   4              ;INITIALLY LENGTH OF NEW LINE
0836 CAE7          ST    CHRNUM(P2)     ; = 4.  ADD 1 TO LENGTH FOR
0838 C701    $1:   LD    @1(P3)         ; EACH CHAR IN LINE UP TO, BU
083A E40D          XRI   0D             ; NOT INCLUDING, CARR. RETURN
083C 9804          JZ    $2
083E AAE7          ILD   CHRNUM(P2)
0840 90F6          JMP   $1
0842 C2E7    $2:   LD    CHRNUM(P2)     ;IF LENGTH STILL 4, WE'LL DEL
0844 E404          XRI   4              ; A LINE, SO SET LENGTH = 0
0846 9C02          JNZ   $3
0848 CAE7          ST    CHRNUM(P2)
084A C2E7    $3:   LD    CHRNUM(P2)     ;PUT NEW LINE LENGTH IN EX
084C 01            XAE
084D C2F2          LD    LABLHI(P2)     ;IS NEW LINE REPLACING OLD?
084F 9406          JP    $4             ;YES - DO REPLACE
0851 D47F          ANI   07F            ;NO - WE'LL INSERT LINE HERE,
0853 CAF2          ST    LABLHI(P2)     ; WHERE FNDLBL GOT US.
0855 9018          JMP   $MOVE          ;BUT FIRST MAKE ROOM
0857 C503    $4:   LD    @3(P1)         ;SKIP LINE # AND LENGTH
0859 40            LDE                  ;EX, NOW HOLDING NEW LINE
085A 02            CCL                  ; LENGTH, WILL SOON HOLD
085B F4FC          ADI   -4             ; DISPLACEMENT OF LINES
085D 01            XAE                  ; TO BE MOVED
085E C501    $5:   LD    @1(P1)         ;SUBTRACT 1 FROM DISPLACEMENT
0860 E40D          XRI   0D             ; FOR EACH CHAR IN LINE BEING
0862 980B          JZ    $MOVE          ; REPLACED
0864 40            LDE
0865 02            CCL
0866 F4FF          ADI   -1
0868 01            XAE
0869 90F3          JMP   $5
086B 90AF    X19:  JMP   X17
086D 90AF    E12:  JMP   E11
086F 40      $MOVE: LDE                 ;IF DISPLACEMENT AND LENGTH
0870 DAE7          OR    CHRNUM(P2)     ; OF NEW LINE ARE 0, RETURN
0872 98F7          JZ    X19
0874 C47A          LDI   L(DOSTAK)      ;CLEAR SOME STACKS
0876 CAFF          ST    DOPTR(P2)
0878 C46A          LDI   L(SBRSTK)
087A CAFC          ST    SBRPTR(P2)
087C C48A          LDI   L(FORSTK)
087E CAFE          ST    FORPTR(P2)
0880 40            LDE
0881 9860          JZ    $ADD           ;DON'T NEED TO MOVE LINES
0883 9410          JP    $UP            ;SKIP IF DISPLACEMENT POSITIV
0885 C100    $DOWN: LD    0(P1)          ;NEGATIVE DISPLACEMENT:
0887 C980          ST    EREG(P1)       ;DO;
0889 C501          LD    @1(P1)         ;    M(P1+DISP) = M(P1);
088B 94F8          JP    $DOWN          ;    P1 = P1+1;
088D C100          LD    0(P1)          ;UNTIL M(P1)<0 & M(P1-1)<0;
088F 94F4          JP    $DOWN
0891 C980          ST    EREG(P1)       ;M(P1+DISP) = M(P1);
0893 904E          JMP   $ADD
0895 C1FE    $UP:  LD    -2(P1)         ;POSITIVE DISPLACEMENT:
0897 CAEA          ST    TEMP(P2)       ;FLAG BEGINNING OF MOVE WITH
0899 C4FF          LDI   -1             ; A -1 FOLLOWED BY 80, WHICH
089B C9FE          ST    -2(P1)         ; CAN NEVER APPEAR IN A
089D C450          LDI   80             ; NIBL TEXT
089F C9FF          ST    -1(P1)
08A1 C501    $UP1: LD    @1(P1)         ;ADVANCE P1 TO END OF TEXT
08A3 94FC          JP    $UP1
08A5 C100          LD    0(P1)
08A7 94F8          JP    $UP1
08A9 35            XPAH  P1             ;SAVE P1 IN LO, HI
08AA CAEE          ST    HI(P2)
08AC 35            XPAH  P1
08AD 31            XPAL  P1
08AE CAEF          ST    LO(P2)
08B0 31            XPAL  P1
08B1 C2EF          LD    LO(P2)         ;ADD DISPLACEMENT TO
08B3 02            CCL                  ; VALUE OF P1, TO CHECK
08B4 70            ADE                  ; WHETHER WE'RE OUT OF
08B5 C400          LDI   0              ; RAM FOR USER'S PROGRAM
08B7 F2EE          ADD   HI(P2)
08B9 E2EE          XOR   HI(P2)
08BB D4F0          ANI   0F0
08BD 9803          JZ    $UP2
08BF C400          LDI   0              ;IF OUT OF RAM, CHANGE
08C1 01            XAE                  ; DISPLACEMENT TO ZERO
08C2 C4FF    $UP2: LDI   -1
08C4 C980    $UP3: ST    EREG(P1)       ;MOVE TEXT UP UNTIL WE REACH
08C6 C5FF          LD    @-1(P1)        ; THE FLAGS SET ABOVE
08C8 94FA          JP    $UP3
08CA C101          LD    1(P1)
08CC E450          XRI   80
08CE 9804          JZ    $UP4
08D0 C100          LD    0(P1)
08D2 90F0          JMP   $UP3
08D4 C2EA    $UP4: LD    TEMP(P2)       ;RESTORE THE FLAGGED LOCATION
08D6 C900          ST    0(P1)          ; TO THEIR ORIGINAL VALUES
08D8 C40D          LDI   0D
08DA C901          ST    1(P1)
08DC 40            LDE                  ;IF DISPLACEMENT = 0, WE'RE
08DD 9C04          JNZ   $ADD           ; OUT OF RAM, SO REPORT ERROR
08DF C402          LDI   2
08E1 908A    E12A: JMP   E12
08E3 C2E7    $ADD: LD    CHRNUM(P2)     ;INSERT NEW LINE
08E5 9884    X19A: JZ    X19            ; UNLESS LENGTH IS ZERO
08E7 C2F1          LD    P1LOW(P2)      ;POINT P1 AT LINE BUFFER
08E9 31            XPAL  P1
08EA C2F0          LD    P1HIGH(P2)
08EC 35            XPAH  P1
08ED C2F3          LD    LABLLO(P2)     ;POINT P3 AT INSERTION PLACE
08EF 33            XPAL  P3
08F0 C2F2          LD    LABLHI(P2)
08F2 37            XPAH  P3
08F3 C2F7          LD    HILINE(P2)     ;PUT LINE NUMBER INTO TEXT
08F5 CF01          ST    @1(P3)
08F7 C2F8          LD    LOLINE(P2)
08F9 CF01          ST    @1(P3)
08FB C2E7          LD    CHRNUM(P2)     ;STORE LINE LENGTH IN TEXT
08FD CF01          ST    @1(P3)
08FF C501    $ADD1: LD    @1(P1)         ;PUT REST OF CHARS
0901 CF01          ST    @1(P3)         ; (INCLUDING CR) INTO TEXT
0903 E40D          XRI   0D
0905 9CF8          JNZ   $ADD1
0907 90DC          JMP   X19A           ;RETURN
0909 C400    X20:  JS    P3,EXECIL
090B 90CF    E13:  JMP   E12A

                   ;*************************************
                   ;*       POP ARITHMETIC STACK        *
                   ;*************************************

090D BAFD    POPAE: DLD    LSTK(P2)      ;THIS ROUTINE POP THE A.E.
090F BAFD           DLD    LSTK(P2)      ; STACK, AND PUTS THE RESULT
0911 33             XPAL   P3            ; INTO LO(P2) AND HI(P2)
0912 C410           LDI    H(AESTK)
0914 37             XPAH   P3
0915 C300           LD     (P3)
0917 CAEF           ST     LO(P2)
0919 C301           LD     1(P3)
091B CAEE           ST     HI(P2)
091D 90E5           JMP    X20

                   ;*************************************
                   ;*              UNTIL                *
                   ;*************************************

                   .LOCAL
091F C2FF    UNTIL: LD     DOPTR(P2)     ;CHECK FOR DO-STACK UNDERFLOW
0921 01             XAE
0922 40             LDE
0923 E47A           XRI    L(DOSTAK)
0925 9C04           JNZ    $1
0927 C40F           LDI    15
0929 90E0           JMP    E13
092B C2EF    $1:    LD     LO(P2)        ;CHECK FOR EXPRESSION = 0
092D DAEE           OR     HI(P2)
092F 9806           JZ     $REDO         ;IF ZERO, REPEAT DO-LOOP
0931 BAFF           DLD    DOPTR(P2)     ;ELSE POP SAVE STACK
0933 BAFF           DLD    DOPTR(P2)
0935 90CD           JMP    X20           ;CONTINUE TO NEXT STMT
0937 40      $REDO: LDE                  ;POINT P3 AT DO-STACK
0938 33             XPAL   P3
0939 C410           LDI    H(DOSTAK)
093B 37             XPAH   P3
093C C3FF           LD     -1(P3)        ;LOAD P1 FROM DO STACK
093E 35             XPAH   P1
093F C3FE           LD     -2(P3)
0941 31             XPAL   P1            ;CURSOR NOW POINTS TO FIRST
0942 90C0           JMP    X20           ; STATEMENT OF DO-LOOP

                   ;*************************************
                   ;*     STORE INTO STATUS REGISTER    *
                   ;*************************************

                   ; THIS ROUTINE IMPLEMENTS THE STATEMENT:
                   ;     'STAT' '=' REL-EXP

0944 C2EF    MOVESR: LD     LO(P2)        ;LOW BYTE GOES TO STATUS
0946 D4F7           ANI    0F7           ; BUT WITH IEN BIT CLEARED
0948 07             CAS
0949 90B9    X21:   JMP    X20
094B 90BE    E14:   JMP    E13

                   ;*************************************
                   ;*          STAT FUNCTION            *
                   ;*************************************

094D C410    STATUS: LDI    H(AESTK)
094F 37             XPAH   P3            ;POINT P3 AT AE STACK
0950 AAFD           ILD    LSTK(P2)
0952 AAFD           ILD    LSTK(P2)
0954 33             XPAL   P3
0955 06             CSA
0956 CBFE           ST     -2(P3)        ;STATUS REG IS LOW BYTE
0958 C400           LDI    0
095A CBFF           ST     -1(P3)        ;ZERO IS HIGH BYTE
095C 90EB           JMP    X21

                   ;*************************************
                   ;*    MACHINE LANGUAGE SUBROUTINE    *
                   ;*************************************

                   ; THIS ROUTINE IMPLEMENTS THE 'LINK' STATEMENT

095E C2EE    CALLML: LD     HI(P2)        ;GET HIGH BYTE OF ADDRESS
0960 37             XPAH   P3
0961 C2EF           LD     LO(P2)        ;GET LOW BYTE
0963 33             XPAL   P3            ;P3 -> USER'S ROUTINE
0964 C7FF           LD     @-1(P3)       ;CORRECT P3
0966 3F             XPPC   P3            ;CALL ROUTINE (PRAY IT WORKS)
0967                LDPI   P2,VARS       ;RESTORE RAM POINTER
0969 90DA           JMP    X21           ;RETURN

                   ;*************************************
                   ;*        SAVE DO LOOP ADDRESS       *
                   ;*************************************

                   ; THIS ROUTINE IMPLEMENTS THE 'DO' STATEMENT.

                   .LOCAL
096B C2FF    SAVEDO: LD     DOPTR(P2)     ;CHECK FOR STACK OVERFLOW
096D E48A           XRI    L(FORSTK)
096F 9C04           JNZ    $1
0971 C40A           LDI    10
0973 90D2    E15:   JMP    E14
0975 AAFF    $1:    ILD    DOPTR(P2)
0977 AAFF           ILD    DOPTR(P2)
0979 33             XPAL   P3
097A C410           LDI    H(DOSTAK)
097C 37             XPAH   P3            ;P3 -> TOP OF DO STACK
097D 35             XPAH   P1            ;SAVE CURSOR ON THE STACK
097E CBFF           ST     -1(P3)
0980 35             XPAH   P1
0981 31             XPAL   P1
0982 CBFE           ST     -2(P3)
0984 31             XPAL   P1
```

```
09BE 90BE   X22:    JMP     X21

            ;*************************************
            ;*        TOP OF RAM FUNCTION        *
            ;*************************************

                    .LOCAL
0990 C2E9   TOP:    LD      TEMP2(P2)       ;SET P3 TO POINT TO
0992 37             XPAH    P3              ; START OF NIBL TEXT
0993 C2E8           LD      TEMP3(P2)
0995 33             XPAL    P3
0996 C300   $0:     LD      (P3)            ;HAVE WE HIT END OF TEXT?
0998 9402           JP      $1              ;NO - SKIP TO NEXT LINE
099A 9007           JMP     $2              ;YES - PUT CURSOR ON STACK
099C C302   $1:     LD      2(P3)           ;GET LENGTH OF LINE
099E 01             XAE
099F C780           LD      @EREG(P3)       ;SKIP TO NEXT LINE
09A1 90F3           JMP     $0              ;GO CHECK FOR EOF
09A3 C702   $2:     LD      @2(P3)          ;P3 := P3 + 2
09A5 AAFD           ILD     LSTK(P2)        ;SET P3 TO STACK, SAVING
09A7 AAFD           ILD     LSTK(P2)        ; OLD P3 (WHICH CONTAINS TOP)
09A9 33             XPAL    P3              ; ON IT SOMEHOW
09AA 01             XAE
09AB C410           LDI     H(AESTK)
09AD 37             XPAH    P3
09AE CBFF           ST      -1(P3)
09B0 40             LDE
09B1 CBFE           ST      -2(P3)
09B3 90D9           JMP     X22

            ;*************************************
            ;*       SKIP TO NEXT NIBL LINE      *
            ;*************************************

09B5 C501   IGNORE: LD      @1(P1)          ;SCAN TIL WE'RE PAST
09B7 E40D           XRI     0D              ; CARRIAGE RETURN
09B9 9CFA           JNZ     IGNORE
09BB 3F             XPPC    P3              ;YES - RETURN

            ;*************************************
            ;*          MODULO FUNCTION          *
            ;*************************************

09BC C2FD   MODULO: LD      LSTK(P2)        ;THIS ROUTINE MUST BE
09BE 33             XPAL    P3              ; IMMEDIATELY AFTER A
09BF C410           LDI     H(AESTK)        ; DIVIDE TO WORK CORRECTLY
09C1 37             XPAH    P3
09C2 C303           LD      3(P3)           ;GET LOW BYTE OF REMAINDER
09C4 CBFE           ST      -2(P3)          ;PUT ON STACK
09C6 C302           LD      2(P3)           ;GET HIGH BYTE OF REMAINDER
09C8 CBFF           ST      -1(P3)          ;PUT ON STACK
09CA 90C2   X23:    JMP     X22
09CC 90AE   E16:    JMP     E15

            ;*************************************
            ;*          RANDOM FUNCTION          *
            ;*************************************

                    .LOCAL
09CE C408   RANDOM: LDI     8               ;LOOP COUNTER FOR MULTIPLY
09D0 CAEB           ST      NUM(P2)
09D2 C2E5           LD      RNDX(P2)
09D4 01             XAE
09D5 C2E4           LD      RNDY(P2)
09D7 CAE9           ST      TEMP2(P2)
09D9 C2E5   $LOOP:  LD      RNDX(P2)        ;MULTIPLY THE SEEDS BY 9
09DB 02             CCL
09DC 70             ADE
09DD 01             XAE
09DE C2E4           LD      RNDY(P2)
09E0 02             CCL
09E1 F2E9           ADD     TEMP2(P2)
09E3 CAE4           ST      RNDY(P2)
09E5 BAEB           DLD     NUM(P2)
09E7 9CF0           JNZ     $LOOP
09E9 40             LDE                     ;ADD 7 TO SEEDS
09EA 02             CCL
09EB F407           ADI     7
09ED 01             XAE
09EE C2E4           LD      RNDY(P2)
09F0 02             CCL
09F1 F407           ADI     7
09F3 1E             RR
09F4 CAE4           ST      RNDY(P2)
09F6 AAE6           ILD     RNDF(P2)        ;HAVE WE GONE THROUGH
09F8 9803           JZ      $1              ; 256 GENERATIONS?
09FA 40             LDE                     ;IF SO, SKIP GENERATING
09FB CAE5           ST      RNDX(P2)        ; THE NEW RNDX
09FD C2FD   $1:     LD      LSTK(P2)        ;START MESSING WITH THE STACK
09FF 33             XPAL    P3
0A00 C410           LDI     H(AESTK)
0A02 37             XPAH    P3
0A03 C401           LDI     1               ;FIRST PUT 1 ON STACK
0A05 CB00           ST      (P3)
0A07 C400           LDI     0
0A09 CB01           ST      1(P3)
0A0B C3FE           LD      -2(P3)          ;PUT EXPR2 ON STACK
0A0D CB02           ST      2(P3)
0A0F C3FF           LD      -1(P3)
0A11 CB03           ST      3(P3)
0A13 C3FC           LD      -4(P3)          ;PUT EXPR1 ON STACK
0A15 CB04           ST      4(P3)
0A17 C3FD           LD      -3(P3)
0A19 CB05           ST      5(P3)
0A1B C2E4           LD      RNDY(P2)        ;PUT RANDOM # ON STACK
0A1D CBFE           ST      -2(P3)
0A1F C2E5           LD      RNDX(P2)
0A21 E4FF           XRI     0FF
0A23 D47F           ANI     07F
0A25 CBFF           ST      -1(P3)
0A27 C706           LD      @6(P3)          ;ADD 6 TO STACK POINTER
0A29 33             XPAL    P3
0A2A CAFD           ST      LSTK(P2)
0A2C 909C   X24:    JMP     X23
0A2E 909C   E16A:   JMP     E16

            ;*************************************
            ;*    PUSH 1 ON ARITHMETIC STACK     *
            ;*************************************

0A30 AAFD   LIT1:   ILD     LSTK(P2)
0A32 AAFD           ILD     LSTK(P2)
0A34 33             XPAL    P3
0A35 C410           LDI     H(AESTK)
0A37 37             XPAH    P3
0A38 C400           LDI     0
0A3A CBFF           ST      -1(P3)
0A3C C401           LDI     1
0A3E CBFE           ST      -2(P3)
0A40 90EA           JMP     X24

            ;*************************************
            ;*      FOR-LOOP INITIALIZATION      *
            ;*************************************

                    .LOCAL
0A42 C2FE   SAVFOR: LD      FORPTR(P2)      ;CHECK FOR FOR STACK
0A44 E4A6           XRI     L(PCSTAK)       ; OVERFLOW
0A46 9C04           JNZ     $1
0A48 C40A           LDI     10
0A4A 90E2   E17:    JMP     E16A
0A4C E4A6   $1:     XRI     L(PCSTAK)
0A4E 31             XPAL    P1              ;POINT P1 AT FOR STACK
0A4F CAF1           ST      P1LOW(P2)       ; SAVING OLD P1
0A51 C410           LDI     H(FORSTK)
0A53 35             XPAH    P1
0A54 CAF0           ST      P1HIGH(P2)
0A56 C2FD           LD      LSTK(P2)        ;POINT P3 AT AE STACK
0A58 33             XPAL    P3
0A59 C410           LDI     H(AESTK)
0A5B 37             XPAH    P3
0A5C C3F9           LD      -7(P3)          ;GET VARIABLE INDEX
0A5E CD01           ST      @1(P1)          ;SAVE ON FOR-STACK
0A60 C3FC           LD      -4(P3)          ;GET L(LIMIT)
0A62 CD01           ST      @1(P1)          ;SAVE
0A64 C3FD           LD      -3(P3)          ;GET H(LIMIT)*
0A66 CD01           ST      @1(P1)          ;SAVE
0A68 C3FE           LD      -2(P3)          ;GET L(STEP)
0A6A CD01           ST      @1(P1)          ;SAVE
0A6C C3FF           LD      -1(P3)          ;GET H(STEP)
0A6E CD01           ST      @1(P1)          ;SAVE
0A70 C2F1           LD      P1LOW(P2)       ;GET L(P1)
0A72 CD01           ST      @1(P1)          ;SAVE
0A74 C2F0           LD      P1HIGH(P2)      ;GET H(P1)
0A76 CD01           ST      @1(P1)          ;SAVE
0A78 35             XPAH    P1              ;RESTORE OLD P1
0A79 C2F1           LD      P1LOW(P2)
0A7B 31             XPAL    P1
0A7C CAFE           ST      FORPTR(P2)      ;UPDATE FOR STACK PTR
0A7E C7FC           LD      @-4(P3)
0A80 33             XPAL    P3
0A81 CAFD           ST      LSTK(P2)        ;UPDATE AE STACK PTR
0A83 90A7   X25:    JMP     X24

            ;*************************************
            ;*    FIRST PART OF 'NEXT VAR'       *
            ;*************************************

                    .LOCAL
0A85 C2FE   NEXTV:  LD      FORPTR(P2)      ;POINT P1 AT FOR STACK,
0A87 E48A           XRI     L(FORSTK)       ; CHECKING FOR UNDERFLOW
0A89 9C04           JNZ     $1
0A8B C40B           LDI     11              ;REPORT ERROR
0A8D 90BB           JMP     E17
0A8F E48A   $1:     XRI     L(FORSTK)
0A91 31             XPAL    P1
0A92 CAF1           ST      P1LOW(P2)       ;SAVE OLD P1
0A94 C410           LDI     H(FORSTK)
0A96 35             XPAH    P1
0A97 CAF0           ST      P1HIGH(P2)
0A99 C2FD           LD      LSTK(P2)        ;POINT P3 AT AE STACK
0A9B 33             XPAL    P3
0A9C C410           LDI     H(AESTK)
0A9E 37             XPAH    P3
0A9F C7FF           LD      @-1(P3)         ;GET VARIABLE INDEX
0AA1 E1F9           XOR     -7(P1)          ;COMPARE WITH INDEX
0AA3 9804           JZ      $10             ; ON FOR STACK: ERROR
0AA5 C40C           LDI     12              ; IF NOT EQUAL
0AA7 90A1   E18:    JMP     E17
0AA9 E1F9   $10:    XOR     -7(P1)          ;RESTORE INDEX
0AAB 01             XAE                     ;SAVE IN EREG
0AAC C280           LD      EREG(P2)        ;GET L(VARIABLE)
0AAE 02             CCL
0AAF F1FC           ADD     -4(P1)          ;ADD L(STEP)
0AB1 CA80           ST      EREG(P2)        ;STORE IN VARIABLE
0AB3 CB00           ST      (P3)            ; AND ON STACK
0AB5 C601           LD      @1(P2)          ;INCREMENT RAM PTR
0AB7 C280           LD      EREG(P2)        ;GET H(VARIABLE)
0AB9 F1FD           ADD     -3(P1)          ;ADD H(STEP)
0ABB CA80           ST      EREG(P2)        ;STORE IN VARIABLE
0ABD CB01           ST      1(P3)           ; AND ON STACK
0ABF C6FF           LD      @-1(P2)         ;RESTORE RAM POINTER
0AC1 C1FA           LD      -6(P1)          ;GET L(LIMIT)
0AC3 CB02           ST      2(P3)           ;PUT ON STACK
0AC5 C1FB           LD      -5(P1)          ;GET H(LIMIT)
0AC7 CB03           ST      3(P3)           ;PUT ON STACK
0AC9 C1FD           LD      -3(P1)          ;GET H(STEP)
0ACB 9410           JP      $2              ;IF NEGATIVE, INVERT
0ACD C404           LDI     4               ; ITEMS ON A.E. STACK
0ACF CAEB           ST      NUM(P2)         ;NUM = LOOP COUNTER
0AD1 C701   $LOOP:  LD      @1(P3)          ;GET BYTE FROM STACK
0AD3 E4FF           XRI     0FF             ;INVERT IT
0AD5 CBFF           ST      -1(P3)          ;PUT BACK ON STACK
0AD7 BAEB           DLD     NUM(P2)         ;DO UNTIL NUM = 0
0AD9 9CF6           JNZ     $LOOP
0ADB 9002           JMP     $3
0ADD C704   $2:     LD      @4(P3)          ;UPDATE AE STACK POINTER
0ADF 33     $3:     XPAL    P3
0AE0 CAFD           ST      LSTK(P2)
0AE2 C2F1           LD      P1LOW(P2)       ;RESTORE OLD P1
0AE4 31             XPAL    P1
0AE5 C2F0           LD      P1HIGH(P2)
0AE7 35             XPAH    P1
0AE8 9099   X26:    JMP     X25
```

```
                ;************************************
                ;*     SECOND PART OF 'NEXT VAR'    *
                ;************************************

0AEA C2EF   NEXTV1: LD      LO(P2)          ;IS FOR-LOOP OVER WITH?
0AEC 9808           JZ      $REDO           ;NO - REPEAT LOOP
0AEE C2FE           LD      FORPTR(P2)      ;YES - POP FOR-STACK
0AF0 02             CCL
0AF1 F4F9           ADI     -7
0AF3 CAFE           ST      FORPTR(P2)
0AF5 3F             XPPC    P3              ;RETURN TO I.L. INTERPRETER
0AF6 C2FE   $REDO:  LD      FORPTR(P2)      ;POINT P3 AT FOR STACK
0AF8 33             XPAL    P3
0AF9 C410           LDI     H(FORSTK)
0AFB 37             XPAH    P3
0AFC C3FF           LD      -1(P3)          ;GET OLD P1 OFF STACK
0AFE 35             XPAH    P1
0AFF C3FE           LD      -2(P3)
0B01 31             XPAL    P1
0B02 90E4           JMP     X26
0B04 90A1   E19:    JMP     E18
                ;************************************
                ;*     PRINT MEMORY AS STRING       *
                ;************************************

                ; THIS ROUTINE IMPLEMENTS THE STATEMENT:
                ;     'PRINT' '$' FACTOR

                .LOCAL
0B06 C2EE   PSTRNG: LD      HI(P2)          ;POINT P1 AT STRING TO PRINT
0B08 35             XPAH    P1
0B09 C2EF           LD      LO(P2)
0B0B 31             XPAL    P1
0B0C                LDPI    P3,PUTC-1       ;POINT P3 AT PUTC ROUTINE
0B12 C501   $1:     LD      @1(P1)          ;GET A CHARACTER
0B14 E40D           XRI     0D              ;IS IT A CARRIAGE RETURN?
0B16 98D0           JZ      X26             ;YES - WE'RE DONE
0B18 E40D           XRI     0D              ;NO - PRINT THE CHARACTER
0B1A 3F             XPPC    P3
0B1B 06             CSA                     ;MAKE SURE NO ONE IS
0B1C D420           ANI     020             ; TYPING ON THE TTY
0B1E 9CF2           JNZ     $1              ; BEFORE REPEATING LOOP
0B20 90C6           JMP     X26


                ;************************************
                ;*        INPUT A STRING            *
                ;************************************

                ; THIS ROUTINE IMPLEMENTS THE STATEMENT:
                ;     'INPUT' '$' FACTOR

0B22 C2EE   ISTRNG: LD      HI(P2)          ;GET ADDRESS TO STORE THE
0B24 37             XPAH    P3              ; STRING, PUT IT INTO P3
0B25 C2EF           LD      LO(P2)
0B27 33             XPAL    P3
0B28 C501   $2:     LD      @1(P1)          ;GET A BYTE FROM LINE BUFFER
0B2A CF01           ST      @1(P3)          ;PUT IT IN SPECIFIED LOCATION
0B2C E40D           XRI     0D              ;DO UNTIL CHAR = CARR. RETURN
0B2E 9CF8           JNZ     $2
0B30 90B6   X27:    JMP     X26


                ;************************************
                ;*   STRING CONSTANT ASSIGNMENT     *
                ;************************************

                ; THIS ROUTINE IMPLEMENTS THE STATEMENT:
                ;     '$' FACTOR '=' STRING

                .LOCAL
0B32 C2EF   PUTSTR: LD      LO(P2)          ;GET ADDRESS TO STORE STRING,
0B34 33             XPAL    P3              ; PUT IT INTO P3
0B35 C2EE           LD      HI(P2)
0B37 37             XPAH    P3
0B38 C501   $LOOP:  LD      @1(P1)          ;GET A BYTE FROM STRING
0B3A E422           XRI     '"'             ;CHECK FOR END OF STRING
0B3C 980E           JZ      $END
0B3E E42F           XRI     '"' ! 0D        ;MAKE SURE THERE'S NO CR
0B40 9C04           JNZ     $1
0B42 C407           LDI     7
0B44 90BE           JMP     E19             ;ERROR IF CARRIAGE RETURN
0B46 E40D   $1:     XRI     0D              ;RESTORE CHARACTER
0B48 CF01           ST      @1(P3)          ;PUT IN SPECIFIED LOCATION
0B4A 90EC           JMP     $LOOP           ;GET NEXT CHARACTER
0B4C C40D   $END:   LDI     0D              ;APPEND CARRIAGE RETURN
0B4E CB00           ST      (P3)            ; TO STRING
0B50 90DE           JMP     X27


                ;************************************
                ;*           MOVE STRING            *
                ;************************************

                ; THIS ROUTINE IMPLEMENTS THE STATEMENT:
                ;     '$' FACTOR '=' '$' FACTOR

                .LOCAL
0B52 C2FD   MOVSTR: LD      LSTK(P2)        ;POINT P3 AT A.E. STACK
0B54 33             XPAL    P3
0B55 C410           LDI     H(AESTK)
0B57 37             XPAH    P3
0B58 C7FF           LD      @-1(P3)         ;GET ADDRESS OF SOURCE STRING
0B5A 35             XPAH    P1              ; INTO P1
0B5B C7FF           LD      @-1(P3)
0B5D 31             XPAL    P1
0B5E C7FF           LD      @-1(P3)         ;GET ADDRESS OF DESTINATION
0B60 01             XAE                     ; STRING INTO P3
0B61 C7FF           LD      @-1(P3)
0B63 33             XPAL    P3
0B64 CAFD           ST      LSTK(P2)        ;UPDATE STACK POINTER
0B66 40             LDE
0B67 37             XPAH    P3
0B68 C501   $LOOP:  LD      @1(P1)          ;GET A SOURCE CHARACTER
0B6A CF01           ST      @1(P3)          ;SEND IT TO DESTINATION
0B6C E40D           XRI     0D              ;REPEAT UNTIL CARRIAGE RET.
0B6E 98C0           JZ      X27
0B70 06             CSA                     ; OR KEYBOARD INTERRUPT
0B71 D420           ANI     020
0B73 9CF3           JNZ     $LOOP

0B75 90B9           JMP     X27


                ;************************************
                ;*     PUT PAGE NUMBER ON STACK     *
                ;************************************

0B77 AAFD   PUTPGE: ILD     LSTK(P2)
0B79 AAFD           ILD     LSTK(P2)
0B7B 33             XPAL    P3
0B7C C410           LDI     H(AESTK)
0B7E 37             XPAH    P3
0B7F C2F6           LD      PAGE(P2)
0B81 CBFE           ST      -2(P3)
0B83 C400           LDI     0
0B85 CBFF           ST      -1(P3)
0B87 90A7           JMP     X27


                ;************************************
                ;*         ASSIGN NEW PAGE          *
                ;************************************

                .LOCAL
0B89 C2EF   NUPAGE: LD      LO(P2)          ;GET PAGE # FROM STACK,
0B8B D407           ANI     7               ; GET THE LOW 3 BITS
0B8D 9C02           JNZ     $0              ;PAGE 0 BECOMES PAGE 1
0B8F C401           LDI     1
0B91 CAF6   $0:     ST      PAGE(P2)
0B93 3F             XPPC    P3              ;RETURN


                ;************************************
                ;*         FIND START OF PAGE       *
                ;************************************

                ; THIS ROUTINE COMPUTES THE START OF THE CURRENT TEXT PAGE,
                ; STORING THE ADDRESS IN TEMP2(P2) [THE HIGH BYTE], AND
                ; TEMP3(P2) [THE LOW BYTE].

0B94 C2F6   FNDPGE: LD      PAGE(P2)
0B96 E401           XRI     1               ;SPECIAL CASE IS PAGE 1, BUT
0B98 9C09           JNZ     $1              ; OTHERS ARE CONVENTIONAL
0B9A C411           LDI     H(PGM)          ;PAGE 1 STARTS AT 'PGM'
0B9C CAE9           ST      TEMP2(P2)
0B9E C420           LDI     L(PGM)
0BA0 CAE8           ST      TEMP3(P2)
0BA2 3F             XPPC    P3              ;RETURN
0BA3 E401   $1:     XRI     1               ;RESTORE PAGE #
0BA5 01             XAE                     ;SAVE IT
0BA6 C404           LDI     4               ;LOOP COUNTER = 4
0BA8 CAEB           ST      NUM(P2)
0BAA 40     $LOOP:  LDE                     ;MULTIPLY PAGE# BY 16
0BAB 02             CCL
0BAC 70             ADE
0BAD 01             XAE
0BAE BAEB           DLD     NUM(P2)
0BB0 9CF8           JNZ     $LOOP
0BB2 40             LDE
0BB3 CAE9           ST      TEMP2(P2)       ;TEMP2 HAS HIGH BYTE
0BB5 C402           LDI     2               ; OF ADDRESS NOW
0BB7 CAE8           ST      TEMP3(P2)       ;LOW BYTE IS ALWAYS 2
0BB9 3F             XPPC    P3
                ;************************************
                ;*     MOVE CURSOR TO NEW PAGE      *
                ;************************************

0BBA C2E9   CHPAGE: LD      TEMP2(P2)       ;PUT START OF PAGE
0BBC 35             XPAH    P1              ; INTO P1.  THIS ROUTINE
0BBD C2E8           LD      TEMP3(P2)       ; MUST BE CALLED RIGHT
0BBF 31             XPAL    P1              ; AFTER 'FNDPGE'
0BC0 3F             XPPC    P3              ;RETURN


                ;************************************
                ;*     DETERMINE CURRENT PAGE       *
                ;************************************

0BC1 35     DETPGE: XPAH    P1              ;CURRENT PAGE IS HIGH
0BC2 01             XAE                     ; PART OF CURSOR DIVIDED
0BC3 40             LDE                     ; BY 16
0BC4 35             XPAH    P1
0BC5 40             LDE
0BC6 1C             SR
0BC7 1C             SR
0BC8 1C             SR
0BC9 1C             SR
0BCA CAF6           ST      PAGE(P2)
0BCC 3F             XPPC    P3              ;RETURN


                ;************************************
                ;*        CLEAR CURRENT PAGE        *
                ;************************************

0BCD C2E9   NEWPGM: LD      TEMP2(P2)       ;POINT P1 AT CURRENT PAGE
0BCF 35             XPAH    P1
0BD0 C2E8           LD      TEMP3(P2)
0BD2 31             XPAL    P1
0BD3 C40D           LDI     0D              ;PUT DUMMY END-OF-LINE
0BD5 C9FF           ST      -1(P1)          ; JUST BEFORE TEXT
0BD7 C4FF           LDI     -1              ;PUT -1 AT START OF TEXT
0BD9 C900           ST      (P1)
0BDB C901           ST      1(P1)
0BDD 3F             XPPC    P3              ;RETURN


                ;************************************
                ;*     FIND LINE NUMBER IN TEXT     *
                ;************************************

                ;INPUTS: THE START OF THE CURRENT PAGE IN TEMP2 AND TEMP3,
                ;        THE LINE NUMBER TO LOOK FOR IN LO AND HI.
                ;OUPUTS: THE ADDRESS OF THE FIRST LINE IN THE NIBL TEXT
                ;        WHOSE LINE NUMBER IS GREATER THAN OR EQUAL TO THE
                ;        NUMBER IN HI AND LO, RETURNED IN ADRLO AND ADRHI.

                .LOCAL
0BDE C2E9   FNDLBL: LD      TEMP2(P2)       ;POINT P1 AT START OF TEXT
0BE0 35             XPAH    P1
```

```
0BE1 C2E8          LD      TEMP3(P2)
0BE3 31            XPAL    P1
0BE4 C100   $1:    LD      (P1)            ;HAVE WE HIT END OF TEXT?
0BE6 E4FF          XRI     OFF
0BE8 9412          JP      $2              ;YES - STOP LOOKING
0BEA 03            SCL                     ;NO - COMPARE LINE NUMBERS
0BEB C101          LD      1(P1)           ; BY SUBTRACTING
0BED FAEF          CAD     LO(P2)
0BEF C100          LD      0(P1)
0BF1 FAEE          CAD     HI(P2)          ;IS TEXT LINE # >= LINE #?
0BF3 9407          JP      $2              ;YES - STOP LOOKING
0BF5 C102          LD      2(P1)           ;NO - TRY NEXT LINE IN TEXT
0BF7 01            XAE
0BF8 C580          LD      @EREG(P1)       ; SKIP LENGTH OF LINE
0BFA 90E8          JMP     $1
0BFC 31     $2:    XPAL    P1              ;SAVE ADDRESS OF FOUND LINE
0BFD CAF3          ST      LABLLO(P2)      ; IN LABLHI AND LABLLO
0BFF 31            XPAL    P1
0C00 35            XPAH    P1
0C01 CAF2          ST      LABLHI(P2)
0C03 35            XPAH    P1
0C04 C2EF          LD      LO(P2)          ;WAS THERE AN EXACT MATCH?
0C06 E101          XOR     1(P1)
0C08 9C07          JNZ     $3
0C0A C2EE          LD      HI(P2)
0C0C E100          XOR     0(P1)
0C0E 9C01          JNZ     $3              ;NO - FLAG THE ADDRESS
0C10 3F            XPPC    P3              ;YES - RETURN NORMALLY
0C11 C2F2   $3:    LD      LABLHI(P2)      ;SET SIGN BIT OF HIGH PART
0C13 DC80          ORI     080             ; OF ADDRESS TO INDICATE
0C15 CAF2          ST      LABLHI(P2)      ; INEXACT MATCH OF LINE #'S
0C17 3F            XPPC    P3
                   PAGE    ' I. L. MACROS'

                ;************************************
                ;*          I. L. MACROS            *
                ;************************************

                   .LOCAL

     2000  $TSTBIT =       TSTBIT*256
     8000  $CALBIT =       CALBIT*256
     4000  $JMPBIT =       JMPBIT*256

                   .MACRO  TST,FAIL,A,B
                   .DBYTE  $TSTBIT!FAIL
                   .IF     #=2
                   .BYTE   'A'!080
                   .ELSE
                   .ASCII  'A'
                   .BYTE   'B'!080
                   .ENDIF
                   .ENDM

                   .MACRO  TSTCR,FAIL
                   .DBYTE  $TSTBIT!FAIL
                   .BYTE   0D!080
                   .ENDM

                   .MACRO  TSTV,FAIL
                   .ADDR   TSTVAR
                   .DBYTE  FAIL
                   .ENDM

                   .MACRO  TSTN,FAIL
                   .ADDR   TSTNUM
                   .DBYTE  FAIL
                   .ENDM

                   .MACRO  JUMP,ADR
                   .DBYTE  $JMPBIT!ADR
                   .ENDM

                   .MACRO  CALL,ADR
                   .DBYTE  $CALBIT!ADR
                   .ENDM

                   .MACRO  DO
                   .MLOC   I
                   .SET    I,1
                   .DO     #
                   .ADDR   #I
                   .SET    I,I+1
                   .ENDDO
                   .ENDM

                    .PAGE   ' I. L. TABLE'

                ;************************************
                ;*          I. L. TABLE             *
                ;************************************

0C18        START:  DO      NLINE
0C1A        PROMPT: DO      GETL
0C1C                TSTCR   PRMPT1
0C1F                JUMP    PROMPT
0C21        PRMPT1: TSTN    LIST
0C25                DO      FNDPGE,XCHGP1,POPAE,FNDLBL,INSRT
0C2F                JUMP    PROMPT

0C31        LIST:   TST     RUN,'LIS','T'
0C37                DO      FNDPGE
0C39                TSTN    LIST1
0C3D                DO      POPAE,FNDLBL
0C41                JUMP    LIST2
0C43        LIST1:  DO      CHPAGE
0C45        LIST2:  DO      LST
0C47        LIST3:  CALL    PRNUM
0C49                DO      LST3
0C4B                JUMP    START

0C4D        RUN:    TST     CLR,'RU','N'
0C52                DO      DONE
0C54        BEGIN:  DO      FNDPGE,CHPAGE,STRT,NXT

0C5C        CLR:    TST     NEW,'CLEA','R'
0C63                DO      DONE,CLEAR,NXT

0C69        NEW:    TST     STMT,'NE','W'
0C6E                TSTN    DFAULT

0C72                JUMP    NEW1
0C74        DFAULT: DO      LIT1
0C76        NEW1:   DO      DONE,POPAE,NUPAGE,FNDPGE,NEWPGM,NXT

0C82        STMT:   TST     LET,'LE','T'
0C87        LET:    TSTV    AT
0C8B                TST     SYNTAX,'='
0C8E                CALL    RELEXP
0C90                DO      STORE,DONE,NXT
0C96        AT:     TST     IF,'@'
0C99                CALL    FACTOR
0C9B                TST     SYNTAX,'='
0C9E                CALL    RELEXP
0CA0                DO      MOVE,DONE,NXT

0CA6        IF:     TST     UNT,'I','F'
0CAA                CALL    RELEXP
0CAC                TST     IF1,'THE','N'
0CB2        IF1:    DO      POPAE,CMPR
0CB6                JUMP    STMT
0CB8        UNT:    TST     DO,'UNTI','L'
0CBF                DO      CKMODE
0CC1                CALL    RELEXP
0CC3                DO      DONE,POPAE,UNTIL,DETPGE,NXT

0CCD        DO:     TST     GOTO,'D','O'
0CD1                DO      CKMODE,DONE,SAVEDO,NXT

0CD9        GOTO:   TST     RETURN,'G','O'
0CDD                TST     GOSUB,'T','O'
0CE1                CALL    RELEXP
0CE3                DO      DONE
0CE5                JUMP    GO1
0CE7        GOSUB:  TST     SYNTAX,'SU','B'
0CEC                CALL    RELEXP
0CEE                DO      DONE,SAV
0CF2        GO1:    DO      FNDPGE,POPAE,FNDLBL,XFER,NXT

0CFC        RETURN: TST     NEXT,'RETUR','N'
0D04                DO      DONE,RSTR,DETPGE,NXT

0D0C        NEXT:   TST     FOR,'NEX','T'
0D12                DO      CKMODE
0D14                TSTV    SYNTAX
0D18                DO      DONE,NEXTV
0D1C                CALL    GTROP
0D1E                DO      POPAE,NEXTV1,DETPGE,NXT

0D26        FOR:    TST     STAT,'FO','R'
0D2B                DO      CKMODE
0D2D                TSTV    SYNTAX
0D31                TST     SYNTAX,'='
0D34                CALL    RELEXP
0D36                TST     SYNTAX,'T','O'
0D3A                CALL    RELEXP
0D3C                TST     FOR1,'STE','P'
0D42                CALL    RELEXP
0D44                JUMP    FOR2
0D46        FOR1:   DO      LIT1
0D48        FOR2:   DO      DONE,SAVFOR,STORE,NXT

0D50        STAT:   TST     PGE,'STA','T'
0D56                TST     SYNTAX,'='
0D59                CALL    RELEXP
0D5B                DO      POPAE,MOVESR
0D5F                DO      DONE,NXT

0D63        PGE:    TST     DOLLAR,'PAG','E'
0D69                TST     SYNTAX,'='
0D6C                CALL    RELEXP
0D6E                DO      DONE,POPAE,NUPAGE,FNDPGE,CHPAGE,NXT

0D7A        DOLLAR: TST     PRINT,'$'
0D7D                CALL    FACTOR
0D7F                TST     SYNTAX,'='
0D82                TST     DOLR1,'"'
0D85                DO      POPAE,PUTSTR
0D89                JUMP    DOLR2
0D8B        DOLR1:  TST     SYNTAX,'$'
0D8E                CALL    FACTOR
0D90                DO      XCHGP1,MOVSTR,XCHGP1
0D96        DOLR2:  DO      DONE,NXT

0D9A        PRINT:  TST     INPUT,'P','R'
0D9E                TST     PR1,'IN','T'
0DA3        PR1:    TST     PR2,'"'
0DA6                DO      PRS
0DA8                JUMP    COMMA
0DAA        PR2:    TST     PR3,'$'
0DAD                CALL    FACTOR
0DAF                DO      XCHGP1,POPAE,PSTRNG,XCHGP1
0DB7                JUMP    COMMA
0DB9        PR3:    CALL    RELEXP
0DBB                CALL    PRNUM
0DBD        COMMA:  TST     PR4,','
0DC0                JUMP    PR1
0DC2        PR4:    TST     PR5,';'
0DC5                JUMP    PR6
0DC7        PR5:    DO      NLINE
0DC9        PR6:    DO      DONE,NXT

0DCD        INPUT:  TST     END,'INPU','T'
0DD4                DO      CKMODE
0DD6                TSTV    IN2
0DDA                DO      XCHGP1,GETL
0DDE        IN1:    CALL    RELEXP
0DE0                DO      STORE,XCHGP1
0DE4                TST     IN3,',',
0DE7                TSTV    SYNTAX
0DEB                DO      XCHGP1
0DED                TST     SYNTAX,',',
0DF0                JUMP    IN1
0DF2        IN2:    TST     SYNTAX,'$'
0DF5                CALL    FACTOR
0DF7                DO      XCHGP1,GETL,POPAE,ISTRNG,XCHGP1
0E01        IN3:    DO      DONE,NXT

0E05        END:    TST     ML,'EN','D'
0E0A                DO      DONE,BREAK
```

```
0E0E        ML:     TST     REM,'LIN','K'
0E14                CALL    RELEXP
0E16                DO      DONE,XCHGP1,POPAE,CALLML,XCHGP1,NXT

0E22        REM:    TST     SYNTAX,'RE','M'
0E27                DO      IGNORE,NXT
0E2B        SYNTAX: DO      ERR
0E2D        ERRNUM: CALL    PRNUM
0E2F                DO      FIN

            ; NOTE: EACH RELATIONAL OPERATOR (EQ, LEQ, ETC.) DOES AN
            ; AUTOMATIC 'RTN' (THIS SAVES VALUABE BYTES!)

0E31        RELEXP: CALL    EXPR
0E33                TST     REL1,'='
0E36                CALL    EXPR
0E38                DO      EQ
0E3A        REL1:   TST     REL4,'<'
0E3D                TST     REL2,'='
0E40                CALL    EXPR
0E42                DO      LEQ
0E44        REL2:   TST     REL3,'>'
0E47                CALL    EXPR
0E49                DO      NEQ
0E4B        REL3:   CALL    EXPR
0E4D                DO      LSS
0E4F        REL4:   TST     RETEXP,'>'
0E52                TST     REL5,'='
0E55                CALL    EXPR
0E57                DO      GEQ
0E59        REL5:   CALL    EXPR
0E5B        GTROP:  DO      GTR

0E5D        EXPR:   TST     EX1,'-'
0E60                CALL    TERM
0E62                DO      NEG
0E64                JUMP    EX3
0E66        EX1:    TST     EX2,'+'
0E69        EX2:    CALL    TERM
0E6B        EX3:    TST     EX4,'+'
0E6E                CALL    TERM
0E70                DO      ADD
0E72                JUMP    EX3
0E74        EX4:    TST     EX5,'-'
0E77                CALL    TERM
0E79                DO      SUB
0E7B                JUMP    EX3
0E7D        EX5:    TST     RETEXP,'O','R'
0E81                CALL    TERM
0E83                DO      OROP
0E85                JUMP    EX3
0E87        RETEXP: DO      RTN

0E89        TERM:   CALL    FACTOR
0E8B        T1:     TST     T2,'*'
0E8E                CALL    FACTOR
0E90                DO      MUL
0E92                JUMP    T1
0E94        T2:     TST     T3,'/'
0E97                CALL    FACTOR
0E99                DO      DIV
0E9B                JUMP    T1
0E9D        T3:     TST     RETEXP,'AN','D'
0EA2                CALL    FACTOR
0EA4                DO      ANDOP
0EA6                JUMP    T1

0EA8        FACTOR: TSTV    F1
0EAC                DO      IND,RTN
0EB0        F1:     TSTN    F2
0EB4                DO      RTN
0EB6        F2:     TST     F3,'#'
0EB9                DO      HEX,RTN
0EBD        F3:     TST     F4,'('
0EC0                CALL    RELEXP
0EC2                TST     SYNTAX,')'
0EC5                DO      RTN
0EC7        F4:     TST     F5,'@'
0ECA                CALL    FACTOR
0ECC                DO      EVAL,RTN
0ED0        F5:     TST     F6,'NO','T'
0ED5                CALL    FACTOR
0ED7                DO      NOTOP,RTN
0EDB        F6:     TST     F7,'STA','T'
0EE1                DO      STATUS,RTN
0EE5        F7:     TST     F8,'TO','P'
0EEA                DO      FNDPGE,TOP,RTN
0EF0        F8:     TST     F9,'MO','D'
0EF5                CALL    DOUBLE
0EF7                DO      DIV,MODULO,RTN
0EFD        F9:     TST     F10,'RN','D'
0F02                CALL    DOUBLE
0F04                DO      RANDOM,SUB,ADD,DIV,MODULO,ADD,RTN
0F12        F10:    TST     SYNTAX,'PAG','E'
0F18                DO      PUTPGE,RTN

0F1C        DOUBLE: TST     SYNTAX,'('
0F1F                CALL    RELEXP
0F21                TST     SYNTAX,','
0F24                CALL    RELEXP
0F26                TST     SYNTAX,')'
0F29                DO      RTN

0F2B        PRNUM:  DO      XCHGP1,PRN
0F2F        PRNUM1: DO      DIV,PRN1,XCHGP1,RTN
                    PAGE    'ERROR MESSAGES'

            ;*************************************
            ;*          ERROR MESSAGES           *
            ;*************************************

                    .MACRO  MESSAGE,A,B
                    .ASCII  'A'
                    .BYTE   'B'!080
                    .ENDM

0F37        MESGS:  MESSAGE ' ERRO','R'     ; 1
0F3D                MESSAGE 'ARE','A'       ; 2
0F41                MESSAGE 'STM','T'       ; 3
0F45                MESSAGE 'CHA','R'       ; 4
0F49                MESSAGE 'SNT','X'       ; 5
0F4D                MESSAGE 'VAL','U'       ; 6
0F51                MESSAGE 'END','"'       ; 7
0F55                MESSAGE 'NOG','O'       ; 8
0F59                MESSAGE 'RTR','N'       ; 9
0F5D                MESSAGE 'NES','T'       ; 10
0F61                MESSAGE 'NEX','T'       ; 11
0F65                MESSAGE 'FO','R'        ; 12
0F68                MESSAGE 'DIV','O'       ; 13
0F6C                MESSAGE 'BR','K'        ; 14
0F6F                MESSAGE 'UNT','L'       ; 15
                    PAGE    ' TELETYPE ROUTINES'

            ;*************************************
            ;*     GET CHARACTER AND ECHO IT     *
            ;*************************************
            ;
                    .LOCAL
0F73 C408   GECO:   LDI     8               ;SET COUNT = 8
0F75 CAEB           ST      NUM(P2)
0F77 06             CSA                     ;SET READER RELAY
0F78 DC02           ORI     2
0F7A 07             CAS
0F7B 06     $1:     CSA                     ;WAIT FOR START BIT
0F7C D420           ANI     020
0F7E 9CFB           JNZ     $1              ;NOT FOUND
0F80 C457           LDI     87              ;DELAY 1/2 BIT TIME
0F82 8F04           DLY     4
0F84 06             CSA                     ;IS START BIT STILL THERE?
0F85 D420           ANI     020
0F87 9CF2           JNZ     $1              ;NO
0F89 06             CSA                     ;SEND START BIT
0F8A D4FD           ANI     %2              ;RESET READER RELAY
0F8C DC01           ORI     1
0F8E 07             CAS
0F8F C485   $2:     LDI     133             ;DELAY 1 BIT TIME
0F91 8F08           DLY     8
0F93 06             CSA                     ;GET BIT (SENSEB)
0F94 D420           ANI     020
0F96 9804           JZ      $3
0F98 C401           LDI     1
0F9A 9004           JMP     $4
0F9C C400   $3:     LDI     0
0F9E 9C00           JNZ     $4
0FA0 CAEA   $4:     ST      TEMP(P2)        ;SAVE BIT VALUE (0 OR 1)
0FA2 1F             RRL                     ;ROTATE INTO LINK
0FA3 01             XAE
0FA4 1D             SRL                     ;SHIFT INTO CHARACTER
0FA5 01             XAE                     ;RETURN CHAR TO E
0FA6 06             CSA                     ;ECHO BIT TO OUTPUT
0FA7 DC01           ORI     1
0FA9 E2EA           XOR     TEMP(P2)
0FAB 07             CAS
0FAC BAEB           DLD     NUM(P2)         ;DECREMENT BIT COUNT
0FAE 9CDF           JNZ     $2              ;LOOP UNTIL 0
0FB0 06             CSA                     ;SET STOP BIT
0FB1 D4FE           ANI     0FE
0FB3 07             CAS
0FB4 8F08           DLY     8               ;DELAY APPROX. 2 BIT TIMES
0FB6 40             LDE                     ;AC HAS INPUT CHARACTER
0FB7 D47F           ANI     07F
0FB9 01             XAE
0FBA 40             LDE
0FBB 3F             XPPC    P3              ;RETURN
0FBC 90B5           JMP     GECO
            ;*************************************
            ;*     PRINT CHARACTER AT TTY        *
            ;*************************************

0FBE 01     PUTC:   XAE
0FBF C4FF           LDI     255
0FC1 8F17           DLY     23
0FC3 06             CSA                     ;SET OUTPUT BIT TO LOGIC 0
0FC4 DC01           ORI     1               ; FOR START BIT. (NOTE INVERS
0FC6 07             CAS
0FC7 C409           LDI     9               ;INITIALIZE BIT COUNT
0FC9 CAE8           ST      TEMP3(P2)
0FCB C48A   PUTC1:  LDI     138             ;DELAY 1 BIT TIME
0FCD 8F08           DLY     8
0FCF BAE8           DLD     TEMP3(P2)       ;DECREMENT BIT COUNT.
0FD1 9810           JZ      PUTC2
0FD3 40             LDE                     ;PREPARE NEXT BIT
0FD4 D401           ANI     1
0FD6 CAE9           ST      TEMP2(P2)
0FD8 01             XAE                     ;SHIFT DATA RIGHT 1 BIT
0FD9 1C             SR
0FDA 01             XAE
0FDB 06             CSA                     ;SET UP OUTPUT BIT
0FDC DC01           ORI     1
0FDE E2E9           XOR     TEMP2(P2)
0FE0 07             CAS                     ;PUT BIT TO TTY
0FE1 90E8           JMP     PUTC1
0FE3 06     PUTC2:  CSA                     ;SET STOP BIT
0FE4 D4FE           ANI     0FE
0FE6 07             CAS
0FE7 3F             XPPC    P3              ;RETURN
0FE8 90D4           JMP     PUTC
0000                .END    0
```

ADD	0335	AESTK	1050	ANDOP	05F0	AT	0C96
BEGIN	0C54	BREAK	0288	CALBIT	0080	CALLML	0963
CHEAT	007C	CHEAT1	009B	CHPAGE	0BBA	CHRNUM	FFE7
CK1	0649	CKMODE	0644	CLEAR	0051	CLEAR1	0056
CLR	0C5C	CMP	0562	CMP1	05C2	CMP2	05CA
CMPR	05D9	COMMA	0DBD	DETPGE	0BC1	DFAULT	0C74
DIV	0410	DO	0CCD	DOLLAR	0D7A	DOLR1	0D8B
DOLR2	0D96	DONE	0135	DONE1	0143	DONE2	0144
DOPTR	FFFF	DOSTAK	107A	DOUBLE	0F1C	E0	0150
E0A	010D	E1	0195	E10	07C6	E11	081E
E12	086D	E12A	08E1	E13	0910	E14	0950
E15	097C	E16	09CC	E16A	0A2E	E17	0A4A
E18	0AA7	E19	0B04	E2	01CC	E3A	028A
E4	02DF	E5	030C	E6	0378	E6A	03D2
E8	064B	E8A	06E5	E8B	06B2	E9	074F
END	0E05	EQ	054C	EREG	FF80	ERR	0223
ERR1	0225	ERR2	0227	ERRNUM	0E2D	EVAL	07E6
EX1	0E66	EX2	0E69	EX3	0E6B	EX4	0E74
EX5	0E7D	EXECIL	0076	EXPR	0E5D	F1	0EB0
F10	0F12	F2	0EB6	F3	0EBD	F4	0EC7
F5	0ED0	F6	0EDB	F7	0EE5	F8	0EF0
F9	0EFD	FACTOR	0EA8	FAIL	05E1	FAILHI	FFEC

```
FAILLO FFED     FALSE  05C8     FIN    02B2     FNDLBL 0BDE
FNDPGE 0B94     FOR    0D26     FOR1   0D46     FOR2   0D48
FORPTR FFFE     FORSTK 108A     GECO   0F73     GEQ    0560
GEQ1   05BE     GETL   0753     GO1    0CF2     GOSUB  0CE7
GOTO   0CD9     GTR    055C     GTR1   05B3     GTROP  0E5B
HEX    0654     HI     FFEE     HILINE FFF7     IF     0CA6
IF1    0CB2     IGNORE 09B5     ILC1   00AA     ILCALL 00A0
IN1    0DDE     IN2    0DF2     IN3    0E01     IND    0534
INPUT  0DCD     INSRT  0820     ISTRNG 0B22     JMPBIT 0040
LABLHI FFF2     LABLLO FFF3     LBUF   10D6     LEQ    0558
LEQ1   05AA     LET    0C87     LIST   0C31     LIST1  0C43
LIST2  0C45     LIST3  0C47     LISTNG FFF5     LIT1   0A30
LO     FFEF     LOLINE FFF8     LSS    0554     LSS1   05A2
LST    02E1     LST2   02FF     LST3   030E     LST4   0314
LST5   0324     LSTK   FFFD     MESGS  0F37     ML     0E0E
MODULO 09BC     MOVE   0804     MOVESR 0949     MOVSTR 0B52
MUL    037A     NEG    0363     NEQ    0550     NEQ1   0599
NEW    0C69     NEW1   0C76     NEWPGM 0BCD     NEXT   0D0C
NEXTV  0A85     NEXTV1 0AEA     NLINE  0215     NOJUMP 009D
NOTOP  05F8     NUM    FFEB     NUPAGE 0B89     NXT    028C
NXT1   02A9     OROP   05F4     P1     0001     P1HIGH FFF0
P1LOW  FFF1     P2     0002     P3     0003     PAGE   FFF6
PCHIGH FFFA     PCLOW  FFFB     PCSTAK 10A6     PCSTK  FFF9
PGE    0D63     PGM    1120     POPAE  0912     PR1    0DA3
PR2    0DAA     PR3    0DB9     PR4    0DC2     PR5    0DC7
PR6    0DC9     PRINT  0D9A     PRMPT1 0C21     PRN    0197
PRN1   01CE     PRNUM  0F2B     PRNUM1 0F2F     PROMPT 0C1A
PRS    017E     PRS1   0193     PSTRNG 0B06     PUTC   0FBE
PUTC1  0FCB     PUTC2  0FE3     PUTPGE 0B77     PUTSTR 0B32
RANDOM 09CE     REL1   0E3A     REL2   0E44     REL3   0E4B
REL4   0E4F     REL5   0E59     RELEXP 0E31     REM    0E22
RETEXP 0E87     RETURN 0CFC     RNDF   FFE6     RNDX   FFE5
RNDY   FFE4     RSTR   0148     RSTR1  0152     RSTR2  0167
RTN    00FB     RUN    0C4D     RUNMOD FFF4     SAV    010F
SAV1   012B     SAV2   0131     SAVEDO 0974     SAVFOR 0A42
SBRPTR FFFC     SBRSTK 106A     SETZ   058D     START  0C18
STAT   0D50     STATUS 0952     STMT   0C82     STORE  04C9
STRT   02C8     SUB    034C     SYNTAX 0E2B     T1     0E8B
T2     0E94     T3     0E9D     TEMP   FFEA     TEMP2  FFE9
TEMP3  FFE8     TERM   0E89     TOP    0990     TST    00C5
TSTBIT 0020     TSTNUM 06B4     TSTVAR 04E9     UNT    0CB8
UNTIL  0924     VARS   101C     X0     00EC     X1     0165
X10    04E2     X11    054A     X12    0597     X12A   05EE
X12B   0637     X12C   067A     X13    06E3     X14    0751
X15    07B4     X16    07E4     X17    081C     X19    086B
X19A   08E5     X20    0909     X21    094E     X22    098E
X23    09CA     X24    0A2C     X25    0A83     X26    0AE8
X27    0B30     X4     01CA     X5     0221     X5A    0286
X6     02DD     X6A    030A     X7     034A     X8     0376
X9     03EF     X9A    0439     X9B    04A4     XCHGP1 0639
XFER   0171     XFER1  0179     ZZ0001 101C     ZZ0002 1120
ZZ0003 0FBD     ZZ0004 0FBD     ZZ0005 0FBD     ZZ0006 0FBD
ZZ0007 0F37     ZZ0008 0F37     ZZ0009 0FBD     ZZ000A 10D6
ZZ000B 0FBD     ZZ000C 101C     ZZ000D 0FBD     ZZ000E 0002
ZZ000F 0002     ZZ0010 0006     ZZ0011 0002     ZZ0012 0003
ZZ0013 0002     ZZ0014 0002     ZZ0015 0002     ZZ0016 0002
ZZ0017 0005     ZZ0018 0004     ZZ0019 0002     ZZ001A 0007
ZZ001B 0004     ZZ001C 0004     ZZ001D 0003     ZZ001E 0002
ZZ001F 0006     ZZ0020 0005     ZZ0021 0002     ZZ0022 0003
ZZ0023 0006     ZZ0024 0005     ZZ0025 0002     ZZ0026 0003
ZZ0027 0005     ZZ0028 0002     ZZ0029 0002     ZZ002A 0005
ZZ002B 0003     ZZ002C 0003     ZZ002D 0007     ZZ002E 0003
ZZ002F 0004     ZZ0030 0003     ZZ0031 0002     ZZ0032 0005
ZZ0033 0002     ZZ0034 0003     ZZ0035 0002     ZZ0036 0003
ZZ0037 0003     ZZ0038 0002     ZZ0039 0006     ZZ003A 0003
ZZ003B 0003     ZZ003C 0007     ZZ003D 0003     ZZ003E 0002
ZZ003F 0002     ZZ0040 0002     ZZ0041 0002     ZZ0042 0002
ZZ0043 0002     ZZ0044 0002     ZZ0045 0002     ZZ0046 0002
ZZ0047 0002     ZZ0048 0002     ZZ0049 0002     ZZ004A 0002
ZZ004B 0002     ZZ004C 0002     ZZ004D 0002     ZZ004E 0003
ZZ004F 0002     ZZ0050 0003     ZZ0051 0002     ZZ0052 0003
ZZ0053 0003     ZZ0054 0003     ZZ0055 0004     ZZ0056 0004
ZZ0057 0008     ZZ0058 0003     ZZ0059 0002     ZZ005A 0003
ZZ005B 0005     $0     002A     $0     0420     $0     076F
$0     0996     $0     0B91     $1     0043     $1     01C6
$1     023D     $1     0397     $1     0443     $1     05FA
$1     06E7     $1     0772     $1     0838     $1     0930
$1     097E     $1     099C     $1     09FD     $1     0A4C
$1     0A8F     $1     0B12     $1     0B46     $1     0BA3
$1     0BE4     $1     0F7B     $10    0AA9     $2     01FF
$2     025A     $2     03A8     $2     0454     $2     0707
$2     07B2 *   $2     0842     $2     09A3     $2     0ADD
$2     0B28     $2     0BFC     $2     0F8F     $3     0261
$3     03D4     $3     04AC     $3     084A     $3     0ADF
$3     0C11     $3     0F9C     $4     03F1     $4     0857
$4     0FA0     $5     085E     $ABOR  06C6     $ADD   08E3
$ADD1  08FF     $CALB  8000     $CR    07D8     $DOWN  0885
$END   04C3     $END   06AC     $END   0B4C     $ENT1  04A6
$ENTE  068B     $ENTE  07B6     $EXIT  0402     $FAIL  04FB
$JMPB  4000     $LETR  067C     $LOOP  002C     $LOOP  00DB
$LOOP  0203     $LOOP  0241     $LOOP  03B6     $LOOP  0460
$LOOP  066C     $LOOP  06F7     $LOOP  09D9     $LOOP  0AD1
$LOOP  0B38     $LOOP  0B68     $LOOP  0BAA     $MAYB  050D
$MOVE  086F     $MSG   0247     $NEQ   00EE     $NOT   0628
$OK    051A     $OK    0688     $OR    0616     $POS   043B
$PRNT  01EF     $REDO  093C     $REDO  0AF6     $RET   06D6
$RUB   07CE     $SCAN  00C7     $SHIF  0692     $SHIF  0714
$SKIP  0664     $TSTB  2000     $UP    0895     $UP1   08A1
$UP2   08C2     $UP3   08C4     $UP4   08D4     $XH    07C8
$XU    07A6

NO ERROR LINES
SOURCE CHECKSUM = 33FE
INPUT FILE     1:NIBL2.SRC
```

6502 STRING OUTPUT, REVISITED

Dear Mr. Warren, Oct. 6, 1976

In *DDJ*, Vol. 1, No. 8 (p. 33), Mr. Espinosa proposed the exchange of "handy" subroutines to save bytes in space-limited systems. He also presented an example, an ASCII string output subroutine for the 6502 microprocessor. I would like to submit a revised version of Mr. Espinosa's subroutine. I have done extensive work on 6502's with OSI's Model 400 microcomputer. During this time I have learned several byte saving programming "tricks" which I would like to pass on by illustration. Through a few simple changes I was able to reduce the length from 40 to 2B (hex) bytes. The result is a subroutine which works the same and saves a few more bytes. The program demonstrates a few simple "tricks":

- Preservation of the Y index register on the stack (3 bytes saved)
- Replace JMP instruction (with ranges less than 128 bt bytes) with forced relative branches. This permits easier relocation of a generalized subroutine so it may be used elsewhere in memory.
- Make use of TYA instruction rather than saving the Y index in a memory location and then adding it in later (5 bytes saved).
- Test the carry flag condition and increment the high order byte if set rather than adding 00 (2 bytes saved).
- Try to avoid dead space inside programs, and non-sero page data storage (i.e. locations 0433 to 043F) (12 bytes saved).

Sincereiy,
Marcel Meier

8850 S. Spring Valley Dr.
Chagrin Falls, OH 44022

```
                        ; STRINGOUT: REVISED VERSION
                        ; ORIGINAL BY C. ESPINOSA
                        ; REVISIONS BY M. MEIER
                        ; 10/5/76
                        ;
                              ORG $400
                        AKEEP EQU $42A
                        OUT   EQU $FEEE
                        LO    EQU $EE
                        HI    EQU $FF
                        LO    DPZ
                        HI    DPZ
                        ;
0400  8D  2A  04  BEGIN STA AKEEP     SAVE AC
0403  68              PLA             GET RETURN ADDRESS
0404  85  EE          STA LO
0406  68              PLA
0407  85  FF          STA HI
0409  98              TYA             PUT Y INDEX ON STACK
040A  48              PHA
040B  A0  01          LDY #01         SET UP INDEX POINTER
040D  B1  EE      NEXT  LDA (LO),Y    GET NEXT CHARACTER
040F  F0  07          BEQ EXIT        DONE IF NULL CHARACTER
0411  C8              INY
0412  20  EE  FF      JSR OUT         OUTPUT CHARACTER
0415  18              CLC             FORCED LOOP WITH
0416  90  F5          BCC NEXT        A RELATIVE BRANCH
0418  98          EXIT  TYA           GET STRING LENGTH
0419  38              SEC
041A  65  EE          ADC LO          ADD RETURN ADDRESS TO
041C  85  EE          STA LO          OFFSET
041E  90  02          BCC NOCAR       IF CARRY, INCREMENT HI
0420  E6  FF          INC HI
0422  68          NOCAR PLA           RESTORE Y FROM STACK
0423  A8              TAY
0424  AD  2A  04      LDA AKEEP       RESTORE AC
0427  6C  EE  00      JMP (LO)        RETURN TO INSTRUCTION AFTER NULL
```

IN-GROUP HUMOR FOR DINOSAUR USERS

We recently heard of some new instructions proposed for some of the maxi computers of industry and business:

BRANCH & BOMB
BRANCH & HANG
PUNCH OPERATOR
BACKSPACE & EJECT DISC
BACKSPACE & PUNCH DISC

Oh well; we *said* it was in-group humor.

TSC LIVES! THEY *DO* HAVE A PHONE NUMBER

Technical Systems Consultants, Box 2574, West Lafayette, IN 47906, peddles some interesting, low-cost micro software. Several people have asked us if TSC is OK to deal with, stating that they were unable to locate a phone number or street address. We wish to emphatically state that they *are* real; they *are* reputable; and they *do* have a phone: (317) 742-7509.