1974

Development of a method for checking syntax in a conversational computing system

N W. Bennett
University of Wollongong

Development of a method for checking syntax in a conversational computing system

N.W.Bennett B.Sc., F.A.C.S.

Ph.D. Thesis

1974

8080457

ABSTRACT

Three systems in the field of conversational computing for small machines have been developed. The main theme has been the exploitation of the full duplex operation of a typewriter to validate the syntax of all expressions and statements entered by the user. A high degree of interaction between the user and his program has been a secondary theme.

The first of these systems, called ACTIV-8, was implemented on a D.E.C. PDP-8 computer. This system explored how a computer program might be used to validate the syntax of characters typed by the user. The language implemented by ACTIV-8 was a very limited subset of FORTRAN, and once the concept of echo-checking was developed, its implementation was straightforward.

The second system is known as ACL-NOVA and was implemented on a D.G.C. NOVA computer. This system was developed by the author and is being used on several installations.

The last system has the generic name of XB and has been implemented on the D.E.C. PDP-11 computer. Three types of expression have been provided in a language structure designed to demonstrate the syntax and recovery techniques. A concept of a "set of allowable next characters" is developed. A broader view of the validity of expressions is explored. Very flexible means are available for the user to interact with the system and his program.

CONTENTS

INTRODUCTION

The first system in this project was a simple conversational computing facility called ACTIV-8 developed for a PDP-8 computer in 1968. A feature of ACTIV-8 was a dynamic syntax checking scheme which exploited the properties of a full duplex typewriter. ACTIV-8 was readily accepted and a second system called ACL was developed, based on the experience gained from the earlier system and developing the theme of dynamic syntax checking. The ACL system was implemented on a NOVA computer by Dr. P.L.Sanger, an officer with the Australian Atomic Energy Commission after the candidate changed employment from the A.A.E.C. to Compunet Ltd. in August, 1969. The Director of the A.A.E.C. has acknowledged that the candidate was the originator of the ACL-NOVA project and of the major concepts behind it.

A paper describing these two systems has been published and this is reproduced as Appendix B. Although these two systems have been superceded by a third system, called XB, they were essential steps in the development of the XB system and the candidate requests that they be considered as part of this thesis.

The last system has been developed solely by the candidate whilst employed with Compunet Ltd. The XB system is being described here for the first time. The theme of syntax checking has been developed to include real, logical and string expressions.

A mechanism to exercise the syntax algorithms has been developed. Initially, this was intended solely to validate the routines used to check the syntax, but has been retained in the final system to fill an advisory role. A user may explore all the possible paths available at any place in expressions or statements.

Although the XB system has been mainly developed to explore and demonstrate the syntax checking, it is a powerful conversational system in its own right. A guide for users of the system has been included as Appendix A. A paper tape copy of the XB system is provided as part of this thesis. The candidate would be pleased to demonstrate this system, but if this is not possible, Appendix E gives instructions for its use on a 16K PDP-11. The body of the thesis builds upon the users guide and the demonstration of the system and it is concerned only with the broad philosophy behind the development and implementation of the XB system.

The XB system consists of over eight thousand statements and the most relevant part is reproduced as Appendix G. A set of sample flowcharts illustrating the techniques used is given in Appendix F. A cross assembler and simulator for the PDP-11 on a Univac 1108 has also been written in conjunction with the XB project and is described in Appendix D.

This document has been prepared using the UCC FASTEXT

system which is described in Appendix C.

HISTORICAL PERSPECTIVE

The I.B.M. 1620 was one of the standard machines of a decade ago. By modern standards this machine is very slow and cumbersome; it had a 20 microsecond cycle time per character and took about 500 microseconds to add two ten digit fields together. At the same time, a great virtue of the 1620 was its slowness and therefore, accessability. One felt one could stop the machine and inspect the value of variables; the time spent in doing so would not greatly inconvenience the next user. At the A.A.E.C. time was booked in lots of half an hour.

After this, machines became faster, operating systems were used to speed the flow of jobs and the machine became remote from the user. Now there is a trend to use the larger, faster machines in a conversational mode in order to regain the advantages which existed a decade ago with slow machines. Rather than have just one user stopping and starting a whole machine, several users are able to have this ability via timesharing and typewriters.

The three systems in this Ph.D. project are part of this revolution; an attempt to regain the conversational facilities which existed in a crude form 10 years ago.

A second trend over recent years has been the use of machines in an "impedance matching" mode, or, to have

different machines linked together with each performing the tasks to which it is best suited (in particular, input and output tasks). A classic example is the UCCompunet computer network (Ref. 1). The central machine is a Univac 1108 which has a high internal speed and also a lot of hardware devoted to performing sophisticated instructions in as few machine cycles as possible; for example, a floating point multiply is performed in 3 cycles. All input and output to slow peripherals such as card readers and printers are controlled by a PDP-8 which is a fast mini-computer with a minimal instruction set. At present 14 card readers and printers are attached to this PDP-8 (2 on site and 12 remote). Cards are read into the PDP-8 and buffered into large blocks. These are sent into the 1108 at drum transfer speeds; several hundred thousand characters per second. Similarly, output from the 1108 is sent to the PDP-8 and then to printers at much lower speed.

Another example in the same network is the FASBAC system (Ref. 2). Here 3 machines are arranged in a heirarchy. A PDP-8 computer is used to receive and send characters from and to low speed asynchronous terminals (110, 134.5 and 300 baud). The PDP-8 is also used to automatically echo any character it may receive from a keyboard back to its matching teleprinter. Whenever a complete line is accumulated it is transmitted into a larger, more sophisticated PDP-9 computer. In this regard

the PDP-8 acts as a pseudo card reader to the PDP-9.

The PDP-9 system is capable of building files, editing files and performing character oriented (vs. arithmetic) tasks. This document was formatted by the FASTEXT system on the PDP-9 computer (Ref. 2, Chapter 5).

A complete program may be built in the PDP-9 as a file and sent accross to the 1108 for processing.

Thus there is a heirarchy:-

a) The PDP-8 handles characters and builds lines.

b) The PDP-9 handles lines and builds files.

c) The 1108 handles files.

This approach of treating input and output in as simple a fashion as possible is usually carried over to single processor conversational systems. In this regard, the three systems in this project are a radical departure from the norm. The XB system treats input from the keyboard as a most important event; one requiring high priority and perhaps a lot of computing. The behaviour of the system could not be duplicated with a simple machine to receive and echo characters and a more powerful machine to perform arithmetic and logic.

It will be demonstrated that this approach taken in the XB system is worthwhile; that although the techniques employed are inefficient, they are effective.

MAIN THEME AND OUTLINE OF RECOVERY PROBLEM

The major theme of this project has been to explore the possibilities of utilizing the full duplex behaviour of a typewriter communicating with a computer, mainly to check syntax.

Firstly, the term full duplex has several usages and its meaning in this context should be clarified. On a teletypewriter attached to a D.E.C. PDP-11 the keyboard and teleprinter are two distinct devices. When a character is keyed, the character is transmitted (actually the ASCII representation) to the computer and an indication given in the machine. The character is not printed automatically by the action of striking the keyboard; it is the responsibility of the program in the computer to do this.

Some machines, such as the I.B.M. golf-ball typewriters are half-duplex; each character keyed by the user is typed locally without reference to the computer system. Others, such as the Memorex 1280 are switch selectable between half-duplex and full-duplex.

This degree of freedom of the program leads to the possibility of checking the incoming character for validity in context and, in particular, syntax. If a character is entered which cannot possibly be correct (such as two decimal points in a number) the simplest way of indicating the error is to not echo the character onto the teleprinter.

In this way all syntax errors can be prevented. This

leads to two savings. Firstly, no messages relating to errors of syntax are needed (a saving of space) and, secondly, interpretation of statements may assume that no errors of syntax are possible (an increase of speed). Each character is checked dynamically for syntax as it is keyed, rather than considering a complete line after the line has been entered, or a complete program after a program has been entered.

The main problem with syntax checking is one of recovery. It is not very difficult to devise a scheme to check the syntax of (say) an arithmetic expression. The real problem lies in being able to continue with the same line after rejecting an error (i.e., for the program to recover to the state in which it was before the invalid character).

Another aspect of the recovery problem is the deletion of characters already entered. Recovery after an invalid character has been entered means regaining just one state; deleting several characters means regaining any one of several states. If twenty characters had been entered in a line, deletion could call for recovery of any one of these states. Suppose the state of the machine is Sn while waiting for the nth character of a line. The problem is to recover to state S1, S2 etc.

A SOLUTION TO THE RECOVERY PROBLEM

The first attempt, in ACTIV-8, to program such a behaviour avoided the problem of recovery to a large extent. At each stage precautions were taken to recover the state Sn if the nth character proved invalid. The simplest way of doing this was to place restrictions on the syntax. For example, all variables were two characters in length; a letter followed by a number. If the second character entered was not a number the program simply looped until it received one.

With hindsight the solution to the recovery problem in the XB system seems straightforward, but many schemes were devised and discarded in the process. The evolution of the solution lay only partly in meeting the requirements already outlined. A major part of the evolution of the recovery techniques for the XB system lay in adapting the solution to meet new ideas which looked as though they should be able to be incorporated easily. If they proved difficult, the recovery technique was modified until it met the new requirements while still satisfying the old. During this evolution the recovery technique grew both more powerful and simpler; always a good sign! To aid the explanation of the recovery mechanism, etc., reference will be made to the source listing of XB (Appendix G). Such references will be to the decimal serial numbers in the left hand column. Thus a reference to the subroutine to accept arithmetic

expressions will be to 1116-1349.

The first decision was to have a recovery mechanism which had the minimum impact upon the program checking the syntax of the incoming characters. The simplest way of doing this was to have a single location in the machine to which a jump could be made (1434). Because this jump could be made from anywhere in the syntax, the only common point from which recovery could be made was the start of the line. Obviously this meant storing all the input characters in a buffer (INPUT) and having a knowledge of the number of characters which had been accepted (a pointer, INPA1, pointing to the next free position in INPUT). During recovery from the start a pointer (L) was needed to indicate which character should be taken next. Once the pointer L had caught up to the pointer INPA1 recovery was complete and the next character was needed from the user via the keyboard. Hence to recover after an invalid character, INPA1 was decreased by one, L reset to point to the start of INPUT and the program worked its way through all the characters at internal speed. Similarly, to backspace one character, INPA1 was decreased by two.

Another problem was the typing of characters. In the normal case, valid characters had to be typed back to the teleprinter once they were accepted, but during recovery the program had to treat them as internal data and certainly not to be typed once more. The backspace using the backslash (\)

required the line to continue, while the multiple backspace (<nn) called for a new line and the reduced line to be retyped. The control of the typing of characters was handled by another pointer (INPB1) to the INPUT buffer. If no extra typing was needed INPB1 was altered to match INPA1. If a new line was needed, the new line was taken and INPB1 reset to the start of INPUT. In this way the line was retyped during recovery.

Some output from the system may be suppressed (Topic 7.2, page A-47, Appendix A) by the user giving an unsolicited character from the keyboard. Output which may be suppressed is indicated by a variable, OPTION, set to non-zero (5443,5467). The output is by-passed in the subroutine which prints individual characters if OPTION is non-zero and the keyboard flag is set (4040-4045). This approach is used to ensure that the program follows the same path whether the printing is active or not. The alternative of breaking into the program stream was not seriously considered as data may be modified while the output is being prepared.

ACTIV-8 simply did not echo the input character to indicate an error. On a teletype this is quite satisfactory. There is an audible difference between a key pressed followed quickly by a character being typed, and a key being pressed followed by silence. However, on a video terminal it proved to be most unsatisfactory. There is no sound involved in the character appearing on the screen and one loses the

sense that there is something very wrong. To overcome this the bell character (control-G) was introduced. If the character was not valid, the bell character was echoed instead. This restored the impression of an error, even on the video. At first the bell was initiated by the error routine directly and was satisfactory for syntax errors, but characters were rejected for other reasons (see the EDIT facility, Topic 4.7, page A-16, Appendix A) and it seemed a pity not to use the same technique for all invalid characters, irrespective of the cause of their rejection. This was eventually solved by sounding the bell if two successive calls were made for an input character without a call for a character to be typed; if the first character was not echoed, something was wrong.

This emphasised the problem of speed of recovery. When one is simply not echoing, the program gives no signal that it has completed recovery. When the error routine signals the bell, it does so when the error has been detected and before recovery. When the bell is sounded by the second call for an input character, it does so after recovery is complete. So that the user does not lose rapport with the system, the time difference between echoing a valid character and sounding the bell must be minimal. This is the main reason for the programmed check (e.g. 1116-1349, arithmetic expressions) for syntax errors rather than the more compact but slower methods available with a table-

driven scheme. However, a compromise is used in the acceptance of keywords (e.g. 4723-4729).

In essence, the technique is very simple. Suppose the string

1.34E+

has been entered. The only character valid as the next character is a number (1204, and 1976-1983 via 882 and 75-84).

If the next character is a number, it is stored in INPUT (1201 via 891 and 75-84 to 3741-3745), typed (3746-3751) and another character solicited from the keyboard (3753-4036). If it is not, an exit is made (1207 via 960 and 85-92 to 1434-1438) to the error routine. The pointer INPA1 is decreased by one, and a jump made to one of several restart points (say 5049, see 5047). From here the program is able to progress from state S1 at the start of the line to Sn (say) where it was just before the invalid character. Note that (5045, 5046) INPA1 and INPB1 are not reset, thus forcing the program to use the internal characters and not retype them. The bell character is sounded (3841-3846) since no character has been typed (4047) between input characters.

The next point of interest is the expansion of a character in some situations (Topic 7.1, page A-46, Appendix A). An example of this is the compulsory space at the end of the keyword ~ TRUE ~. If a non-space is entered instead of the terminating space, it is saved in INPUT (1531-1539) so

that it is treated as the next character and a space substituted. The value of HBECHO (which controls the bell resulting from two successive calls for an input character) is increased so that the bell sounds unless there are two characters echoed between calls for an input character (1538). This is to allow for the non-space being an invalid character. Similarly the expansion of a special character which has been associated with a compound statement via an ASYNCH statement (4361-4386, Topic 5.18, page A-39, Appendix A) is performed by expanding the string into INPUT, increasing INPA1 to include the new characters, but leaving the pointers L and INPB1 so that the extra characters are both picked up and typed.

An expansion of particular interest is the null statement number (5321-5338). Here a character is used to indicate a null statement number of three spaces, and the most obvious choice is the space character itself. The difficulty arises when a recovery is made. If the space character causes an expansion of one space into three, care must be taken that the system does not end up with five spaces after recovery. This would most likely cause a syntax error, another recovery and a tight loop. This was solved by distinguishing between keyboard characters and recovery characters (5330). A new problem arises where a line containing a null statement number is punched onto paper tape via the SUSPEND statement. One must punch all three

spaces on the paper tape or the typing of the line would not match the line printed via a LIST. However, the first space must not cause an expansion either, else one would end up with five spaces instead of three. This was solved by inserting a special non-printing character (control-7, 5321) in front of the three spaces to suppress the expansion.

Because the syntax of incoming characters is checked, there is an opportunity to store additional information to simplify and speed the process of evaluation. An output buffer (OUTPUT with a pointer SL) receives a copy of the input characters as well as special characters to act as sign-posts for the evaluation. For example, a bracket for subscripts (1226-1231), a bracket for grouping (1131-1134) and a bracket enclosing an argument for a function (1320-1335) are all stored with a different special character imbedded in front of them. Similarly a unary plus (1121-1125), an exponent plus (1191-1195) and an operator plus (1171-1175) are all stored with different indicators. During evaluation the special characters are sought rather than the original characters. Since this "seeded" string must be used to recover the original string at some future time (for program listings, etc), the seed characters must be easily distinguishable. Since the PDP-11 has characters of eight bits and ASCII is a seven bit code, the sign bit of each character was used.

In a sophisticated syntax scheme it was sometimes

necessary to follow one of two possible paths. For example

"IF B AND"

looks like a continuing logical expression such as

"IF B AND B2 THEN TYPE A3"

but it could turn out to be

"IF B AND3=14"

i.e. a variable AND3 rather than a logical operator AND.

To cope with this problem it is necessary to identify logical ends to an expression. After a variable or literal, one may have a logical end of expression or an operator. If the characters turn out not to be a continuation of the expression, a "mini recovery" must be made to the last logical end of expression and an exit taken. Note that a full recovery is neither necessary or possible. It proved to be sufficient to save and restore the two pointers L and SL. In the example above, see 1709-1718, 1735-1737, 1984-1990. The input characters must be recovered from INPUT and new seeds stored into OUTPUT.

It was important that the syntax check graduated from particular to general, otherwise the general syntax could accept a stream and never pass it on to the particular. Also care must be taken that another routine is available to carry on the stream. If not, a later routine may reject a character already echoed to the teleprinter; resulting in a recovery and a tight loop. For example the characters "ACCEPT" could turn into an arithmetic or logical expression

up until acceptance of the fifth character (variables can be four characters long). If a deviation is made from the keyword in the first five characters, a mini recovery is made from the start of the line and the string treated as an expression (4577-4578, 4723-4729, 4970-4972). In the case of an arithmetic or string expression no operator is more than one character long, and it is not necessary to ever perform a mini recovery as for a logical operator such as ' AND '. However care must be taken not to reject a character if a logical end of expression has just occured, but rather pass the character back to the more specialized syntax routine which may be calling the expression. For example 64C would have the first two characters accepted as arithmetic expression but the last (the C) passed over to the more specialized syntax routine for judgement (1178-1186, 1288-1293). If the calling routine were the arithmetic routine looking for an operand for a function, the C would be rejected; any extra character must be a closing bracket (1341-1342). But if it is a TYPE statement looking for an operand it would be accepted (4680-4683). Note that even the comma separating the operands in a TYPE statement is inspected by an expression routine, found not to conform to the syntax for that expression and passed back to the TYPE statement.

In many cases a general expression is required (operand for TYPE, expression statement) as opposed to a particular

type of expression (subscript value, input during execution of an ACCEPT statement). A special routine (1815-1961) is needed to determine the type of expression, to accept the expression from the keyboard and to inform the calling syntax routine. The only problem here is that either the arithmetic or string expressions may become logical expressions by use of a relational operator. For example

```
14*(A1+3)
```

is an arithmetic expression, but

```
14*(A1+3) EQ 6
```

is a logical expression. The mini recovery technique is well suited to this task. If the initial character does not indicate a logical or string expression, an arithmetic expression is taken from the keyboard (1917). If the characters after the arithmetic expression (1919-1948) are a relational operator, a mini recovery is made to the start of the arithmetic expression and a logical expression is sought (1949-1961).

SET OF ALLOWABLE NEXT CHARACTERS

All the techniques described above were well established when the concept of the set of allowable next characters was introduced (see Topic 6, page A-44, Appendix A). This was born largely out of a concern to check all possible characters coming in from the keyboard in any situation. By working systematically through the keyboard

one could classify characters into not acceptable if they signaled an error or acceptable if they were echoed. In order to continue with the next test character, the particular situation must be regained. If the trial character was rejected, recovery was automatic. If the character was accepted, recovery had to be forced by means of a backspace.

In some cases this was meaningless. Suppose the trial character was a colon (treated as a carriage return) which terminated a line which in turn deleted a stored statement. Here it is not possible to reverse this action by a backspace! This was solved by restricting all the syntax routines to merely gather information. If the character was being treated as a test character, this information could be ignored (5369-5375), or if it was being treated in the normal way, it could be acted upon (5057-5065). With this modification, the automation of this "thrashing" of the system was practical. The sixty four ASCII characters (excluding control characters, lower case characters and "<", ">" and "\") were tested. Each character in turn was placed in the "next character" position and followed by several special characters (5508-5514) in case of expansion. The number of internal characters was set to a very large number (5515), typing was suppressed (5516) and a recovery initiated. A jump to the error routine will be made, either because the test character was invalid or because the

special character was detected (3764-3765). A simple test may be made (5523-5525) to determine this.

A string of allowable characters is built up internally (5482-5484, 5499-5502). If this string is empty (i.e. no characters were valid) something is wrong with the X8 system. If more than one character is valid, a fresh line is taken, the set of characters typed, and a recovery made with the original line retyped.

If only one character is valid, it is echoed in the users line, and a fresh scan made for the set of allowable characters in the next position. If this new scan results in only one character, this character is also typed on the users line and the process repeated. If more than one valid character is found, control is returned to the user to continue keying in characters. Note that when the system is following a keyword (the only real situation repeatedly returning one valid character) sixty four test characters (and 64 recoveries) are made for each character typed for the user.

A simple extension of this technique is used to print out a two dimensional array of valid characters for the next two positions.

Lastly, an option exists so that the system will print the set of allowable next characters whenever the user makes an error. This prompts the user with the set of characters from which he must choose. If he was typing a keyword, where

there is only one valid next character, the system will complete the keyword. That is, if the user were typing an UNLESS keyword and had typed the U, then keyed an M instead of an N, the system would type an N instead of the M and continue with L, E, S, S, and the closing space.

An important point is that this facility (a syntax exorciser) is completely independant of the syntax routines. No changes are needed to the exorciser if the syntax is changed.

OTHER ERRORS

Other errors may be checked in the same fashion as the syntax errors. The single subscript of an array must be in the range 0-65535. Hence in the string

A(70000)

the subscript is clearly out of range. Other examples are one of two subscripts (0-255), string subscripts (1-31 at worst), references to sequence or statement numbers (0-999; GO TO, CALL, etc) and the argument of the SQR function (not negative).

In the example above, the closing bracket should be rejected. It is tempting to reject the last zero which changes the subscript from 7000 to 70000, but this would not allow the valid expression

A(70000-X)

It is only when the user indicates the subscript is

completed that any decision can be made about rejecting the terminating character.

Clearly, it is necessary to evaluate the arithmetic expression. This is not possible if reference is made to a variable. There are two cases:-

a) The statement is to be stored. The variable may assume any value when the program is run.

b) The statement is an immediate statement. At first one is tempted to use the current value of variables, but there are two difficulties:-

1) X=-1:
   A(X)+X=1:

2) X=-1:
   X=1; A(X):

In the first case, X will have a value of 1 during the "real" evaluation. Any decision based on the -1 value ignores the possibility of X being redefined before the subscript is evaluated. The second case is similiar.

The checking outlined above has been implemented in the XB system as an option (See Topic 4.13, page A-20, Appendix A). In the case of

A(70000

a pointer to the left hand end of the subscript is saved. When an attempt is made to end the subscript, the expression is copied into an alternate area. The subroutine to

reference the symbol table is conditioned so that reference to a variable causes the evaluation to be aborted. After verifying the syntax of the terminating character, an attempt is made to evaluate the subscript. If the attempt is aborted, the incoming character must be accepted. If the evaluation can be completed, a check on the range of the subscript may be made and if it is outside the range, the character may be rejected.

The time taken to evaluate the arithmetic expression can be a problem. In practice a subscript like

A(1+2+3+4+5-4*4)

can be evaluated and the closing bracket rejected without noticeable delay. However a subscript like

A(SQR(25)-SQR(24))

takes time to evaluate (particularly with decimal floating point software!) and a considerable delay occurs before the closing bracket is accepted. The problem would be reduced if hardware floating point were available. If a real time clock were available, the attempt to evaluate the expression could be aborted (similiar to the symbol table reference) after (say) 40 ms.

If the string

A(SQR(25)-SQR(24))++

were entered, the subscript would be evaluated once to determine whether the closing bracket should be accepted and again during recovery after rejecting the second plus

character. This second evaluation is unnecessary and, once the problem is recognised, can be avoided. However, care has to be taken that the process of determining the set of allowable next characters is not upset (the two cases are very similiar).

In the case of an immediate GO TO with an argument which can be evaluated, a check is made that the sequence or statement number is defined. In the case of a statement number, a check is made that it is unique.

The same idea may be used to monitor the FOR statement. In the statement

```
FOR X=1,3,-.5 A(X)=0:
```

the space after the -.5 could be rejected. Here all three parameters of the FOR loop may be evaluated and the endless loop recognized.

This exercise of rejecting out-of-range subscripts etc, was at first thought to be too difficult to implement. However refinements of any technique depends on the changes needed to incorporate more facilities into a method. (A question of difficult problems vs. inadequate solutions.)

A check on the value for a SQR function may be made with the double set of allowable next characters feature. For

```
SQR(2E4-1E+>
$0123456789$
SQR(2E4-1E+>
```

```
0 )*+-/0123456789^
1 )*+-/0123456789^
2 )*+-/0123456789^
3 )*+-/0123456789^
4 )*+-/0123456789^
5 *+-/0123456789^
6 *+-/0123456789^
7 *+-/0123456789^
8 *+-/0123456789^
9 *+-/0123456789^
SQR(2E4-1E+2):
141.068
>
```

the only numbers which may follow the exponent plus character and which may themselves be followed by a closing bracket are 0 through 4. The numbers 5 through 9 yield negative values for the argument and hence the closing bracket is not allowed.

A forerunner to rejecting nonexistant sequence numbers was the implementation of an immediate statement to delete sequence numbers and change statement numbers (Topics 5.4 and 5.5, pages A-23 to A-26, Appendix A). A rigid syntax structure dictated that the sequence number should be in cols 1, 2 and 3. When the carriage return is given, a check is made that a statement has been stored with that sequence number.

Another possibility (not implemented at the time of writing) is to check on the uniqueness of the statement number of a statement being entered for storage. This check could be made when the second character of the statement number is entered. One complication is that the current statement may be replacing one with the same statement number. If the uniqueness check were to fail, this possibility would have to be explored (by comparing sequence numbers as well). Another consideration is that the user may be well aware that he is duplicating a statement number and his next move would have been to delete the old statement. This places an unnatural restriction on his order of making this change.

Strings longer than 31 characters could be detected at input. That is, in the string

```
"ABCDEFGHIJKLMNOPQRSTUVWXYZ123456
```

the 6 is an error because it is the character which makes the string exceed the maximum length. In the situation above the only character which is valid is a closing quote. Similiarly the string

```
"ABCDEFGHIJKLMNOPQRSTUVWXYZ"&"123456
```

is too long because of the 6. In this case the system would have to recognise that two string literals, with known lengths, were being concatenated. The string

```
"ABCDEFGHIJKLMNOPQRSTUVWXYZ"&H18"123456
```

is also too long because of the 6. Here, although the value

of H1 is unknown at the time of receiving the 6, the string is too long even if H1 is a null string (with a length of zero).

The magnitude of numbers may not exceed .999999E+39. If a number

1.E+64

were entered the system could continuously attempt to convert the number into internal form. In this case the 4 could be rejected.

The above procedure is too simple. One must also allow

99999999999999999999999999999999999999E-20

which is a valid number when the exponent is included, but to a simple minded scheme the fortieth digit is an error and would be rejected. It is only when the user indicates that the number is complete that a check could be made. In the example of 1.E+64 the 4 is the last character since the syntax will not allow more than two exponent characters. In

9999999999999999999999999999999999999999*

the * indicates the end of the number and if the maximum magnitude is exceeded, the * could be rejected.

Another problem appeared in the sample program of Topic 5.19, page A-41 of Appendix A. The condition of validity is that

N GE 1 AND LE 9 AND PO(N) EQ MT

should be true. That is, the position must be on the board and not taken. In practice, this does not achieve the

desired effect and the condition must be broken in two to be

```
IF N LT 1 OR  GT 9 GO ERR
IF PO(N) NE MT GO ERR
```

The reason for this is that if N has a value of 21, say, then PO(21) is out of range, an error exit taken, an error message given and the statement aborted. But the expression is identical to

```
FALSE AND B
```

Here, the value will be FALSE irrespective of B being TRUE, FALSE or undefined. The error status is really indicating that PO(N) NE MT is undefined. In this case, it makes no difference to the final result.

Similarly,

```
TRUE OR B
```

will be TRUE. This difficulty could be overcome by allowing a third value for logical (i.e. undefined). If an undefined were generated during an evaluation, an error message should be stored. If the final value was defined, no harm would be done; if undefined, then the first error message should be given. Note that

```
TRUE AND UNDEFINED
```

should merely yield a partial result of UNDEFINED. There is no need to generate an error message, since clearly the error is due to some earlier condition and an error message exists.

This extra value (UNDEFINED) for a logical variable is

similiar to the floating point values of undefined, plus infinity and minus infinity on the CDC 6600 computer.

The identity

FALSE AND B is always FALSE

is used in optimising code in the I.B.M. FORTRAN compiler for the 360 computer. There it is used to avoid calculations rather than avoid error messages. This rationalization has not been implemented at the time of writing.

FACILITIES FROM INTERNAL CHECKING

Such a large and closely-woven system as the XB system on the PDP-11 requires extensive checking to make sure all possibilities are considered and to assist in isolating errors. Many of these checks and facilites have provided features for the user.

The set of allowable next characters started as a method of checking that the syntax analysis was reasonable and there were no tight loops. In the final system this takes on an advisory role; giving the user an indication of the characters which may be used in a particular situation, either on request or when an error is made.

Because the syntax analysis, if working correctly, guarentees the validity of the statements it is tempting to have no checks in the evaluation stage of the interpretor. However any error not detected in the syntax would go undetected and result in a mysterious error far removed from

its cause. For this reason many checks have been included, although far less than would be needed if these checks were the first line of defense against syntax errors. These checks have been included as assembly options (49,4911).

The fine trace feature, giving an operation by operation analysis of the evaluation of an expression (Topic 4.4, page A-12, Appendix A) is another example. This was introduced to check the evaluation of expressions. As the features within expressions became more sophisticated (such as the elided operand in a relational) the fine trace feature followed. When the possibility of invoking the step-by-step analysis when an error occured was realized the feature was retained in the system, rather than discarded like many other implementation aids.

SYMBOL TABLE

A very simple symbol table is used for the XB system. Two PDP-11 words are used to hold variables (4 characters for simple variables, 2 characters and 16 bits for singly subscripted arrays, and 2 characters and two by 8 bits for doubly subscripted arrays) and two words for data (32 bits).

A sequential search is used to reference variables. The table is not sorted, but rather entries added or altered as assignments are made.

A single area is devoted to storing statements and strings and the symbol table. The variable length data

starts at the low end of the area and increases upwards, while the symbol table (with fixed length data) starts at the high end and increases downwards. This allows optimal use of the area.

A device used in looking up variable length information is of interest. Special characters are used to indicate the start of each item. If the total length of variable length data were n, then n searches need to be made if the data is stored as true variable length with dense packing. A worthwhile improvement in speed may be achieved by padding out the variable length data so that the special characters are always a multiple of m characters from the start of the table. This results in a m-fold increase of speed at the cost of (m-1)/2 characters per item. A multiple of 4 is used in the XB system resulting in a four fold increase in speed for a 7.5% increase in space assuming twenty characters per item on average. (Actually a 370% increase in speed because of the increase in size.)

In order to repeat statements which end in error (Topic 4.4, page A-12, Appendix A) it is necessary to reverse the effect of any assignments made during the statement. An attempt was made to do exactly this, but showed all the signs of being unwieldly. A simpler statement of the problem was to reset the symbol table to its state at the start of the statement. A solution to this equivilant problem proved to be both simple and elegant. The symbol table was

partitioned into a permanent symbol table and an incremental symbol table (initially empty). The permanent symbol table remained unchanged and assignments were made into the incremental symbol table. Any reference was tried first in the incremental table and then, if unresolved, in the permanent table. At the successful conclusion of a statement (or statements) the incremental table was consolidated into the permanent table. If the statement ended in error, the incremental table was simply abandoned.

SYNTAX

The syntax of expressions and statements in the XB system is not complex. The main concern has been exploiting the full duplex typewriter characteristics. Sample flowcharts for arithmetic expressions are given in Appendix F.

There are three types of data which must be recognised. It is desirable to be able to determine the type by the first character to avoid complexities in the recovery problem. Literals identify themselves in this way naturally (e.g. quote for string, space for logical and number or decimal point for real). The first character of variables identifies their type. (An excellent precedent exists in FORTRAN for integer variables!) Initially it was planned to commence logical variables with an L, but the function LOG would cause difficulties. So instead the letter B (for

Boolean) was chosen. Similarly string variables were to start with S but because of SQR and SIN became H (for Hollerith). When strings were introduced, the subroutine depth counter, then SRET, was changed to ZRET, but not changed back when string variables started with H rather than S.

One construction of interest is the elided relational. Here

```
A EQ 2 OR A EQ 3
```

is simplified to

```
A EQ 2 OR 3
```

and

```
N LT 1 OR N GT 99
```

to

```
N LT 1 OR  GT 99
```

This simplification is only possible where a full relational has been given. For example

```
A EQ 2 OR 3
```

is allowed, but not

```
2 OR 3 EQ A
```

A knowledge of past relationals is maintained. It is set (1758) when a full relational is completed and reset whenever a logical variable is used or an assignment made. Also the simplification may not carry from real to string. For example

```
A EQ 2 OR H EQ "2" OR A EQ 3
```

may not be simplified to

```
A EQ 2 OR H EQ "2" OR 3
```

even though the syntax analysis and expression evaluation are well defined. These restrictions are quite arbitrary and designed to prevent the user from confusing situations.

There are two possibilities when recognizing the use of this simplification. In the first, where just the left hand operand is elided, a relational operator follows a logical operator (1655-1659, 1687-1693). In the second a relational operator does not follow the left hand arithmetic operand of what at first sight appears to be a complete relational (1667-1678, 1679-1686).

I am indebted to Mr. Peter Moylan of Newcastle University for bringing a similar construction to my notice. The expression

```
A1 GT A2 AND A2 GT A3
```

may be simplified to

```
A1 GT A2 GT A3
```

This simplification is attractive (especially its implementation) but I have resisted the temptation of including it in the present XB version.

Another construction of interest was that of selective assignment. The construction

```
IF B THEN A=X ELSE C=X
```

may be rewritten as

```
IF B THEN A= ELSE C=X
```

Note that

IF B THEN A ELSE C=X

does not mean the same as the previous expressions because in the XB system assignments like C=X may be treated as an operand. This was considered as being of insufficient benefit for the simple, yet tedious problems involved in implementation.

EDIT FACILITY

The edit facility (Topic 4.7, page A-16, Appendix A) enables the user to change characters already entered in the current line or stored previously.

The change sequence involves nominating a substring in the current line. A subroutine is used to count the number of times the substring exists within the current line. If no match is found, then the change is impossible. The last character entered is either the first character of a substring or a subsequent one (to get this far, the substring must previously have existed within the current line). The last character must then have been in error and should be rejected.

If the substring exists within the current line the character may be echoed. This different criterion to echo caused the decision to give the warning bell character after a second call for an input character rather than a syntax error.

If the substring exists once only (i.e. is unique) within the current line there is a unique next character. This character may be nominated by the user via the '>' character. Once the user has given sufficient of the substring for it to be unique, the rest of the string may be easily nominated by this means. Note that the system cannot advance automatically as with the set of allowable next characters facility; the user must have complete control over the length of the substring he is nominating.

It would be possible to check the syntax of the characters making up the second substring. For example, in the following case

2+C*4<E

E>/C/C)

the ')' could be rejected just as it would be in the sequence

2+C)

However to change

2+C*4

into

(2+C)*4

would mean inserting the opening bracket first and then the closing bracket. The approach finally taken was to allow the user full rein to change the current line and to perform a syntax check when the user indicates that all changes have been made. The current line is presented to the syntax as an

internal stream, and a test made to see if an error would be indicated before the end of the string. If all is well the line is retyped for the user while the system makes a complete recovery.

If a syntax error is indicated, a second scan is made exactly like the first, but with the typing indicator (INPB1) set to type the valid characters as they are accepted. A dollar character is given to make any space character at the end of the line visible. The user is returned to the edit mode with the complete line available.

In this way the user is able to make changes in the line and the system verify that the new line still has valid syntax.

INTERACTION WITH PROGRAM

The XB system has been written to provide the maximum interaction of a user with his program. Because of this it is essential that interplay btween interpretation of stored statements and the user should be as flexible as possible.

When interpreting a stored program, a record must be kept to indicate the current statement and the next. In the XB system it is possible to stop a running program, add extra stored statements and continue. Because this may involve moving statements in storage the only practical means of recording a statement is by sequence or statement number.

Where one statement follows sequentially from another, the address of the next statement is held between statements in exactly this form; e.g. the statement after (sav) 120. Suppose at the time this next statement is 130 and at this point the user regains control (Topic 5.17, page A-38, Appendix A) and inserts a statement with sequence number 124. If the sequence number of the next statement is calculated too early an incorrect return would be made to 130 rather than 124.

Similarly when a CALL or GO TO is made, the target must be described exactly as such. Suppose in a stored program a statement with a sequence number of 160 had a statement number of 23 and the program had just interpreted a GO TO 23 when the user regained control. The user must be free to reassign the statement number to another statement; say 220. If a translation from statement number to sequence number had been made too early, a return to 160 would have been made instead of the correct 220.

A breakpoint facility (Topic 5.8, page A-27, Appendix A) has been provided for the user to regain control at a specified point in his program. It is neccessary to allow the breakpoint to act either before or after the statement to cover the case of a GO TO or CALL.

The user may regain control from the automatic mode (Topic 5.17, page A-38, Appendix A) with provision to return control either to the next statement at full speed with ':',

to the next statement for one statement only with ';' or to any point in the program with a GO TO.

The user may break into the running program and execute a single nominated statement (Topic 5.18, page A-39, Appendix A). Provision has been made to allow this nominated statement to be a CALL statement.

A trace feature is available for variable assignments (Topic 5.9, page A-29, Appendix A) and for listing statements as they are interpreted. Another trace feature (not yet implemented at the time of writing) will be an assignment trace for a particular variable covering all statements rather than the present assignment trace which covers all variables within a particular statement.

CALL and RETURN statements

The CALL statement and its companion, the RETURN statement, provide the means of using other statements in a program and then continuing interpretation with the statement after the CALL statement (Topic 5.14, page A-36, Appendix A). Information must be saved during the execution of a CALL statement so that the RETURN statement may send control back to the correct point. Once this information has been stored, the CALL statement may be, and is, treated as a GO TO statement. This information is held in a relative form. Suppose the CALL were made from sequence number 120 when the next sequence number was 130. If the user regained

control between the CALL and the RETURN and introduced a statement with sequence number 124, the RETURN will be made to 124; if the RETURN address were stored as 130 the system would not be able to follow this change in the program by the user.

In the XB system this information is saved in the symbol table. An array, ZR, holds a stack of return locations and a variable, ZRET, counts the depth of the stack. In the case of a stored CALL a single entry is added to ZR, and ZRET increased by one. For an immediate CALL three entries are needed. The first two are the variables holding the sequence numbers of the previous and next statements. The third is an indicator so that the RETURN can interpret the stack values correctly. A CALL may also be made by an ASYNCH statement, (Topic 5.18, page A-39, Appendix A) and three stack entries are also needed.

The use of the symbol table allows recursive CALLs to be made. The values are held in floating point form under specific variable names so that the user may have access to them. For example, in a recursive subroutine a stack of variables may be referenced using the depth pointer as a subscript as in A(ZRET). The user is not permitted to redefine these variables as this could lead to mysterious errors. However, the exact state of the program has to be stored on paper tape via the SUSPEND statement and be used to reconstruct the program (Topic 5.11, page A-32, Appendix

A). This problem was solved by allowing an assignment character after ZR(N) or ZRET only if a non-printing password (control S, control Y, control S) is given (1248-1272). This allows the value of ZRET to be given with the password included and to look just like any other immediate statements to restore the value of a variable.

THE PRESENT SYSTEM AND THE FUTURE

The present system has been written to explore the possibilities of exploiting the full duplex behaviour of a typewriter in a conversational computing situation. The standard mathematical functions such as SIN, COS etc. have not been implemented to leave space for enhancements on the main theme. The PDP-11 used to implement the XB system has no floating point hardware so a software floating point system was written. With a system allowing a user such ease of communication, constant conversion errors became a problem with a binary internal representation so a decimal representation was used instead. However it would seem sensible to use any hardware floating point and to have an equivalent software system so that they may use the same mathematical functions.

The present XB system has been written to handle only one typewriter. Because of the architecture of the PDP-11 the present program can fairly easily be upgraded to handle several terminals.

Other mediums than paper tape are now available such as cassette and disc. It would be useful to be able to store and retrieve programs from these mediums at much higher transfer rates.

CONCLUSIONS

The concept of dynamic syntax checking via a full duplex typewriter has been developed and demonstrated. The priority given to attending characters entered from the keyboard has been reversed so that checking may be performed quickly enough that it is transparent to the user. This means that any expression or statement entered into the system has valid syntax.

A very powerful, but simple and flexible, method has been evolved to handle a complex language structure. The concept of a set of allowable next characters follows simply and easily from this. A broader view of the validity of expressions has been presented.

Flexible means have been demonstrated in the XB system for a user to interact with his program. A method of saving sequencing information which can cope with dynamic program changes has been used. A facility to repeat exactly any statement ending in error has been included.

The two earlier systems, ACTIV-8 and ACL-NOVA, have been well accepted. The ACL-NOVA system is in use in seven research and university centres. The XB system, which is of

wider scope, should enjoy a similiar usage.

ACKNOWLEDGEMENTS

I wish to acknowledge the use of language concepts from BASIC, FORTRAN, ALGOL and even COBOL. The language is a vehicle to demonstrate the concepts and techniques which have been developed. The edit facility owes a great deal to the change command in the EDIT subsystem of FASBAC. In FASBAC this command is not even recognised until end-of-line is given; in the XB system incoming characters are checked character-by-character for validity.

I wish to acknowledge the support and encouragement of many people over the period of the project.

The atmosphere and facilities provided by the Australian Atomic Energy Commission during the development of the first two programs is appreciated and also the encouragement of Dr.'s J.L.Symonds and D.J.Richardson during this period. Dr. P.L.Sanger was responsible for implementing the ACL system on the NOVA computer.

UCCompunet have provided the facilities needed to develop and implement the XB system. I am grateful for the continuing support of the management; in particular, of Mr.'s Alex Smith, Theo Boehme and Fred Johnson. The encouragement of Mr. Ian G. Nichols has been very much appreciated.

The consistent encouragement of Prof. Austin Keane has

been a major factor in the completion and recording of this project.

REFERENCES

1. 1108 Processing Concepts - UCC 1030.40
UCCompunet Ltd., 23 Cleg St., Artarmon 2064

2. FASBAC Users Guide - UCC 320.50
UCCompunet Ltd., 23 Cleg St., Artarmon 2064

Appendix A

A Guide for Users of the XB system

Appendix A

CONTENTS

1. Introduction

The XB system is a conversational computing system for the DEC PDP-11 computer. Just like BASIC, FORTRAN or ALGOL, XB provides a facility to implement an algorithm.

All communication with the XB system is via a teletypewriter or similar device. A central feature of the XB system is a method of accepting only those characters keyed by the user which are syntactically correct and this applies to all dialogue between the user and the XB system. A character which is incorrect is simply not typed back to the teleprinter and an audible signal may be given to indicate the error. The user is free to continue typing after an error has been made; by this means the user is protected against trivial typing errors.

There are two modes of operation within the system. The first is a user mode where the user is able to present statements and have them executed immediately, or to enter statements to be stored to build up a program. The second is an automatic mode where the stored program is being executed. In the user mode, the user is in control in that he supplies commands to the system, the system obeys them and control returns to him. In the automatic mode, the program is in control. Statements are fetched from the stored program, the system obeys them and a new statement is fetched. There are flexible means of switching between the two modes. A program may be started, the user regain

control, perhaps inspect the values of variables, perhaps change the program and then allow the program to continue.

There are three types of data; real, logical and string. One and two dimensional arrays may be made with real and logical variables. String variables may be subscripted to access individual characters within a string.

Within the XB system, expressions may often be used where a constant or variable is required. Subscripts for arrays may be any real expression; input to the ACCEPT statement may be an expression.

The user is able to save the state of a program on paper tape, reload the program at some later time and continue.

2. Conventions for input

Input to the XB system is a series of lines which are terminated by a carriage return. The visible character colon, ':', is typed in response. Lines may also be terminated by the colon character itself.

When the system is expecting the user to provide input, a prompt character (or characters) is typed in the initial columns of the line. For input to the user mode the prompt is the "greater than" character or '>'.

If an error has been made (such as 1.5 instead of 1.6) characters may be deleted. These deletions take the form

\ Delete the last character. (Continue on same

line.)

<: Delete the last character. (A new line is taken.)

<3: Delete the last 3 characters. (A new line is taken.)

<12 Delete the last 12 characters (carriage return is not needed, new line taken).

The following commands may also be given.

<E Enter the EDIT mode (see Topic 4.7) with the present line as data.

<A Abandon the present line and start again.

The system may be initialized at any time by the keyboard character, control-G.

3. Expressions

There are three types of expression in the XB system. These are logical (or boolean), real (or arithmetic) and string. Topics 3.1, 3.2 and 3.3 discuss features common to all these expressions.

3.1 Assignments

An equals character is used as an assignment character in the XB system and in this users guide. A left pointing arrow (ASCII 137) may optionally be substituted at generation of the XB system. An example of a simple assignment for a real variable is

A=2:

The order for performing an assignment is to evaluate the expression on the right of the assignment operator and then define the variable on the left to have this value. In the XB system an assignment is considered to have an overall value and may be used as an operand. For example

```
2+A=4:
```

has a value of 6. In particular, the ALGOL construction

```
A1=A2=A3=0
```

is available.

Similarly multiple assignments like

```
A1=3*A2=1+A3=1:
```

may be made with assignments being performed from right to left. In this example A3 would be assigned a value of 1, A2 a value of 2 and A1 a value of 6.

3.2 Brackets

Brackets may be used to vary the normal order of evaluation. For example 3*2+1 yields a value of 7 whereas 3*(2+1) has a value of 9.

The part of the expression to be evaluated first is bounded by the left-most closing bracket and its matching opening bracket. This selection is continued until all brackets have been eliminated. For example the expression

```
((2+3)*4)+3*(5+6)
```

would be evaluated as

```
((2+3)*4)+3*(5+6)
```

```
((5)*4)+3*(5+6)
(5*4)+3*(5+6)
(20)+3*(5+6)
20+3*(5+6)
20+3*(11)
20+3*11
20+33
53
```

In the expression

```
A(X)=X=X+1:
```

for an initial value of X of 2, A(2) will be set to 3.

3.3 Conditional clauses

A conditional clause of the form

```
IF b.e. THEN expr1 ELSE expr2
```

may be used as an operand. The b.e. is a logical expression, and expr1 and expr2 are both logical, real or string. If b.e. is true the overall value of the clause is expr1, else it is expr2. For example

```
2+IF B1 THEN 3 ELSE 4:
```

is an arithmetic expression,

```
'A'&IF N1 EQ 1 THEN 'CDE' ELSE H3:
```

is a string expression.

There is a heirachy within these 3 levels. Conditional clauses are considered first, then brackets and then assignments.

3.4 Logical expressions

There are two logical constants

" TRUE " and " FALSE ".

The spaces, but not the quotes, are part of the constants.

Logical variable names may be 1, 2, 3 or 4 characters long; the first character must be the letter "B" and succeeding characters may be either letters or numbers.

Logical array names may be one or two characters long; the first character must be the letter "B" and may be followed by a letter or a number. There may be one or two subscripts; each of which may be a real expression. If there is one subscript, it may range in value from zero to 65535. Each of two subscripts may range in value from zero to 255. No declaration is needed for arrays. The single, double and non-subscripted forms of a variable are distinct and may all be used.

Four logical operators may be used;

" NOT ", " AND ", " OR " and " XOR ".

They have their usual meanings and heirarchy.

Examples

```
>BT OR BF= FALSE  AND BT= TRUE :
TRUE
>BF OR (BT AND BF):
FALSE
>BT XOR IF BT THEN BF ELSE BT:
TRUE
```

3.5 Real expressions

All arithmetic in the XB system is performed with decimal floating point numbers and there are no separate integer numbers. The floating point numbers have a precision of six decimal digits with a range from .1E-39 to .999999E+39 inclusive.

Real variables and arrays are the same as logical variables and arrays except the first character must be a letter other than "B" or "H". (The simple variable ZRET and the single dimensioned array ZR are not available for general use.)

There are five arithmetic operators. They are, in order of heirarchy:-

^ power; e.g. A^3

*, / multiply and divide

+, - add and subtract

Unary plus and minus are available where -A is interpreted as 0-A, +A is 0+A and the above heirarchy is obeyed. The construction 2^3^4 is not permitted, nor is any disguise of this construction.

A set of arithmetic functions such as SQR are available. A list of these is given in the Appendix.

Examples

```
>2*3/4+5:
6.5
>SQR(SQR(4))+1:
```

```
2.41422
>2+A=IF B= TRUE  THEN SQR(5) ELSE 99:
4.23607
```

3.6 String expressions

Any sequence of characters enclosed in quotes may be used as a string. The quote character itself may be indicated by a sequence of two quotes. Examples are

```
"CAT"
"THE QUICK BROWN FOX"
"WON""T"
```

All string variables start with the letter "H" and may be followed by a letter or a number. There are no string arrays as such, but any string variable may be subscripted to indicate reference to individual characters within the string.

There is a single string operator "&", which indicates concatenation. Strings may be concatenated to form longer strings.

An individual character within a string may be redefined with an assignment. A string may be extended by defining its next character. If a string has not been defined, the first character may be defined. A character within a string may be eliminated by redefining it as the null character. In this case the string is reduced in length by one with the higher characters being moved down one

position.

Examples are

H3="AB"&"C":

H3(2)="Z":

H3 would now be "AZC".

H3(4)="E":

H3 would now be "AZCE".

H3(3)="":

H3 would now be "AZE".

3.7 Relational expressions

The validity of a relationship between real expressions has a logical value. As such they may be used as an operand in a logical expression. For example 1 EQ 1 is true, whereas 2 GT 3 is false. The standard six relational operators between real expressions are available. These are

" EQ ", " NE ", " LT ", " GT ", " LE " and " GE ".

The spaces, but not the quotes, are part of the operator. The relational operators are considered in the heirarchy to be after arithmetic operators, but before the logical operators.

Similarly the relationships EQ and NE are meaningful between strings and may be used to yield a logical value. For example

1+2 GT 0 AND "A" EQ "B":

would be false.

Where a logical expression has two adjacent relationals with an identical left hand side, the left hand side of the second relational may be elided. For example

A EQ 2 OR A GT 9:

becomes

A EQ 2 OR GT 9:

Similarly, if two adjacent relationals have identical left hand sides and identical relational operators, then both the left hand side and the operator of the second relational may be elided. For example

A EQ 2 OR A EQ 3:

becomes

A EQ 2 OR 3:

The rule for interpreting the simplified expression is that the missing left hand operand and, if necessary, the missing operator, are copied from the nearest preceding full relational.

4. Facilities and statements in the user mode.

4.1 Expression statements

An expression may be used as a statement. During execution the value of the expression is calculated and unless the leftmost part of the expression consists of a variable assignment, the value of the expression is typed. For example, the value of

```
1+2:
'A'&H1='BC':
```

would be typed, while the value of

```
A1=2+3:
BT= TRUE :
```

would not. In FORTRAN the arithmetic expression above is known as an arithmetic statement.

4.2 TYPE statement

The TYPE statement is for displaying information on the teletypewriter. There are 3 classes of operand.

a) The value of an expression (real, logical or string) may be typed.

b) Either an absolute or a relative positioning operand may be given. The absolute positioning operand has the form

arithmetic expression,'C'

and indicates that the next value should be typed at a particular column. If the desired column is greater than the current column, the teletypewriter carriage will be positioned to the column on that line, otherwise a new line will be taken. The relative positioning operand has the form

arithmetic expression,'X'

and indicates that a number of columns should be skipped.

c) A combination is provided to print an arithmetic value so that its decimal point (or right hand side if there is no decimal point) is positioned in a nominated column.

This takes the form

[a.e.1,a.e.2]

where the value of the second arithmetic expression has its value printed with its decimal point in the column nominated by the first arithmetic expression.

These three types of operand may be used in a TYPE statement separated by commas. For example

TYPE "TEMP. IS ",[15,T1],17C,"DEGREES F":

For T1 values of 1.2 and 212 this would give

TEMP. IS 1.2 DEGREES F

TEMP. IS 212 DEGREES F

4.3 Compound statements

More than one statement may be used on a single line by terminating all but the last statement with "; " (semi-colon, space). The last statement must be terminated with ":" (or carriage return) as usual.

Some statements may not be continued in this way and a list of these is given in the Appendix.

Examples

J=J+1; GO TO 4:

X=Y=0; P=Q=1:

4.4 Statements which end in error

Messages are given for all error conditions. The relevant statement may be repeated with a trace by

responding with a carriage return as the next line. The trace consists of a printout after each operation has been performed. For example

```
>SQR(1+2+3+4+5-4*4):
** ERROR ** SQR(-1) RANGE ERROR
>:
SQR(1+2+3+4+5-4*4)
SQR(1+2+3+4+5-16)
SQR(3+3+4+5-16)
SQR(6+4+5-16)
SQR(10+5-16)
SQR(15-16)
SQR(-1)
** ERROR ** SQR(-1) RANGE ERROR
```

If any assignments were made during the statement(s), their effect is reversed.

Example

```
>A=-3:
>A=A+2; SQR(A):   (Value of A changed from -3 to -1)
** ERROR ** SQR(-1) RANGE ERROR
>A:
-3                (Value of A back to -3)
```

4.5 FOR statement

A FOR statement has the form

FOR J=J1,J2,J3 S1; S2:

where J is a real variable and J1, J2 and J3 are real expressions. The latter part of the statement is interpreted for J=J1, J=J1+J3, etc until J is greater than J2. A limited selection of statements may be used in the latter part of the statement. An indication of these is given in the Appendix.

A check is made that the loop is finite. The increment J3 may be elided, in which case it is chosen to be 1 or -1 as J2 is greater or less than J1.

At the normal completion of the statement, the index variable is not defined, but if a GO TO exit is taken out of the loop, the index variable retains its value.

Examples

```
>FOR X=1,3 TYPE [6,X],[12,SQR(X)]:
     1      1
     2      1.41422
     3      1.73205
>FOR X=3,1 TYPE X:
3
2
1
>FOR X=250,300 A(4,X)=0:
** ERROR ** A(4,256) HAS A SUBSCRIPT OUT OF RANGE
```

4.6 IF statement

An IF statement has been implemented with the form

IF b.e. THEN S1; S2 ELSE S3; S4:

If the logical expression b.e. is true the compound statement S1; S2 is interpreted, else the compound statement S3; S4 is interpreted.

The ELSE part of the statement may be elided to give

IF b.e. THEN S1; S2:

In this case the 'THEN ' characters may also be elided to give

IF b.e. S1; S2:

Examples

IF X=X+1 LE 8 THEN GO TO 3 ELSE GO TO 1:

IF A EQ 2 OR 3 TYPE 'ERROR':

An alternate IF statement of the form

IF(b.e.) S1; S2:

is also available.

The FOR statement may be used in either the THEN or the ELSE part of an IF statement, and the IF statement may be part of a FOR statement.

An arithmetic IF statement of the form

IF(r.e.)n1,n2,n3

has been implemented. The n1, n2 and n3 are real expressions representing statement numbers or sequence numbers. If the value of the r.e. is less than zero a GO TO n1 is made. Similarly if the r.e. is zero or greater than zero a

GO TO n2 or n3 is made. Any of n1, n2 or n3 may be elided. The brackets around the r.e. may also be elided.

Examples

```
IF(A) 21,22,23
IF C 3,,5
IF X(I) 3,4,Y(I)
```

4.7 Edit facility

Lines which are being typed in by the user may be edited at any stage by entering "<E". A new line is taken with a prompt of "E>". The following responses are available:-

a) The character "P". The line is typed.

b) A change sequence consisting of a special character, a string of characters in the line to be changed, the same special character to delimit the string, a string of characters to replace the first and the delimiting special character to terminate the line.

Characters in the first string are dynamically checked for validity. If the string being proposed does not exist in the line being edited, the last character entered is not echoed. If the string entered as the first string is not unique in the line being edited, the special character delimiting the first string is not echoed as a warning. At this point further characters may be entered to uniquely identify the change, or the special character may be entered

again; this time it will be echoed and the first occurance of the string will be taken.

Once the string to be changed has been uniquely identified in the line to be edited, the character '>' may be used to nominate the next character. At any stage during the change sequence, the character '<' may be used to abandon the line.

c) A carriage return indicates that editing has finished and an exit should be taken. At this point a check is made to see that the syntax is correct up to the end of the line. If the syntax is not correct, the line is typed up to the point where an error was detected (a '$' is given at the end of the line) and the edit mode is retained. The full line is once again available. If the syntax is correct and the line terminated by a carriage return, the line is interpreted without being typed again. If the syntax is correct and the line incomplete, the line is typed again and the user may continue his input.

d) The character 'A' causes the line to be abandoned just as if the characters '<A' had been used instead of '<E'.

Example

```
>1.2345<E
E>1.2345                (P entered)
E>/23/.32/              (change 2 characters into 3)
E>1..3245               (P - note two decimal points)
```

```
E>:                    (go back)
1.$                    (extent of valid characters)
E>/.//                 (take out the .)
E>1.3245               (P)
E>:                    (go back)
1.3245+4:              (finish line by adding +4)
5.3245                 (result)
```

4.8 SYMBOLS statement

The SYMBOLS statement may be used to print the symbol table.

4.9 CLEAR statement

The CLEAR statement may be used to clear variables from the symbol table. If no operands are used, all variables are cleared from the symbol table. If one or more variables are used as operands only those variables will be cleared from the symbol table.

Example

```
>A=1; C=2; D=3:
>SYMBOLS:
D=3
C=2
A=1
>CLEAR A,D:
>SYMBOLS:
```

```
C=2
>CLEAR:
>SYMBOLS:
>
```

4.10 Undefined variables

A reference to a variable which has not had a value assigned to it causes an error message to be printed. The user is given the option of assigning a value at that point and having the system repeat the statement.

The system will ignore any detectable errors made by the user in assigning the value and repeat the request.

Example

```
>SQR(A1):
A1 IS UNDEFINED, GIVE VALUE NOW
U>99:
9.9499
>SQR(ZZ):
ZZ IS UNDEFINED, GIVE VALUE NOW
U>XY:
XY UNDEFINED, TRY AGAIN FOR ZZ
U>9/0:
** ERROR ** 9/0  OVERFLOW, TRY AGAIN FOR ZZ
```

An exit to the user mode may be made by responding with a null line (i.e. a carriage return).

4.11 Statement suffix

Some statements may have a suffix indicating conditions under which the statement is to be performed. The forms are

S IF b.e.

S UNLESS b.e.

Examples are

A=200 IF C GT 0:

GO TO 4 UNLESS B:

A list of the statements which may have a suffix is given in the Appendix.

4.12 Comments

Comments may be inserted by starting any line with "C", space. All characters are valid; except that ":", "<", ">" and "\" all retain their usual meanings.

4.13 A further validity check

As well as giving a means of indicating keyboard characters which are invalid for syntax reasons, the full duplex typewriter may be used to prevent some errors of numerical magnitude. Subscripts for arrays must not be negative, and not more than 65535 in the case of single subscripts and 255 in the case of double subscripts. For example, in the lines

A1(70000)

A2(300,

A3(4,100+8*25)

the last character indicates that the subscript is complete. An option exists (see Appendix) in the XB system to attempt to evaluate the subscript expression. If any reference to a variable is made it is not possible to complete the evaluation, the attempt is inconclusive and the system has to accept the closing character.

If, however, the evaluation can be made, and the value is outside the legal limits, then the terminating character can be rejected. Note that in the case of

A2(SQR(25)-SQR(24))

a noticeable time may be taken before the closing bracket is accepted.

Similiarly, the closing bracket in

SQR(1+2+3+4+5-4*4)

would be rejected.

Another example of a numerical bounds check is with statement and sequence numbers. For example

GO TO 1+2-5

CALL 600+600

are both invalid. The bounds check will reject the character which signifies the end of the statement or sequence number.

A further check is made with immediate statements to make sure that referenced sequence numbers are defined and that referenced statement numbers have been defined, but not duplicated.

5. Building a stored program

5.1 Sequence and Statement numbers

Every statement stored as part of a program must have a sequence number. The sequence number is a three digit number in the range 100-999 and must be followed by a space in column four. A space in column one has been interpreted as a request for the next sequence number; this will be ten more than the sequence number of the line last stored (note that this means the sequences ' NOT ', ' TRUE ' and ' FALSE ' cannot be used to start an immediate logical expression).

Statements to be stored may have a two digit statement number in the range 00-99. A single digit number may be used such as 9. A statement number should be followed by a space in column seven and the body of the statement in column eight. However, a non-space in column seven causes the compulsory space to be inserted and a space may be used in column five to automatically generate a null statement number.

5.2 References to statements

There are a number of statements which refer to other statements for the purpose of listing, deleting, etc. These statements have a common format for referencing other statements. There are five variations and the LIST statement is used as an example.

1) LIST:

2) LIST N:

3) LIST N1,N2:

4) LIST N1,:

5) LIST ,N2:

The first form refers to the entire stored program.

In the other four forms, the operand may be a real expression which references a statement by statement or sequence number depending on the value of the expression being in the range 00-99 or 100-999.

The second form references a single statement. The third form references a group of statements from N1 through N2. The fourth form has its second operand elided and this is understood to mean from N1 to the end of the program. The last form means from the start of the program to N2.

5.3 LIST statement

A stored program may be typed by the LIST statement. The operands may assume the five forms discussed in Topic 5.2.

5.4 DELETE statement

Statements may be deleted from the stored program. The operands may assume the five forms discussed in Topic 5.2.

A statement may also be deleted by the sequence "sequence number, space, carriage return". If the sequence

number had not previously been defined the carriage return is rejected; a second carriage return causes the line to be abandoned.

Example

```
>110    X=0:
>120    X=X+1:
>130    X=X+2:
>140    X=X+3:
>150    X=X+4:
>160    X=X+5:
>170    GO TO 120:
>LIST:
110   X=0
120   X=X+1
130   X=X+2
140   X=X+3
150   X=X+4
160   X=X+5
170   GO TO 120
>DELETE 120,130:
>LIST:
110   X=0
140   X=X+3
150   X=X+4
160   X=X+5
170   GO TO 120
```

```
>140 :
>LIST:
110   X=0
150   X=X+4
160   X=X+5
170   GO TO 120
```

5.5 Changing statement numbers

A statement number may be changed by giving the three digit sequence number of the statement, a space, the new statement number and a carriage return. The new statement number may be either two digits, a digit or a space. Once again, if no statement has been stored with the nominated sequence number, the carriage return is rejected; a second carriage return causes the line to be abandoned.

Example

```
>110 99 X=1:
>LIST 110:
110 99 X=1
>110 88 :
>LIST 110:
110 88 X=1
>110 3  :
>LIST 110:
110 3  X=1
>110    :
```

```
>LIST 110:
110   X=1
```

5.6 EDIT statement

A stored statement may be edited with the EDIT statement. Its form is

```
EDIT N:
```

where N is a real expression representing either the sequence number or the statement number of the desired statement.

The actions described for the EDIT facility are now available.

5.7 RUN statement

A stored program may be run by the RUN statement. Execution begins with the lowest sequence number. The symbol table is not altered.

Example

```
>110    X=S=0:
>120    S=S+X*X; IF X=X+1 LE 8 GO 120:
>130    TYPE "SUM IS",S; STOP:
>RUN:
SUM IS 204
CONTROL RETURNED TO USER
```

5.8 Pause Before and Pause After facility

A facility is available to return control to the user before a nominated statement is executed. The statement has the general form

PB N:

The five forms discussed in Topic 5.2 are available. When a statement, which has been the subject of a PB statement, is reached in the stored program, control is given back to the user. A new line is taken; the sequence number of the statement just completed, the sequence number of the statement next to be interpreted and a prompt of "PB>" are all typed.

At this point all the facilities of the user mode are available plus the following features related to the suspended program.

a) The "?" character will cause the sequence numbers of the previous and next statements to be typed. If the next statement has been nominated by a GO TO or CALL statement, the second operand has the form "J N" where N is the statement or sequence number used in the last statement.

b) The ":" (carriage return) character results in a return to the suspended program.

c) The ";" character results in a return to the suspended program for one statement only.

Once the PB condition has been satisfied, it is removed.

The PB condition may also be removed by the statement

PB ^ N:

The same five variations are available.

A similar facility for regaining control after a statement has been interpreted is provided by a PA statement.

Example

```
>110    A=1:
>120    C=2:
>130    D=3:
>140    E=5:
>150    F=7:
>160    G=99:
>170    J=0:
>PB 130:
>RUN:
120 130 PB>SYMBOLS:
C=2
A=1
>?
120 130
>;
130 >?
130 140
>:
END OF PROGRAM
```

5.9 Assignment trace

An assignment trace may be set so that a record is typed as assignments are made to the symbol table.

There are two assignment trace conditions. A global assignment trace may be used to give an assignment trace record for all statements, for selected statements or for no statements.

The global trace may be changed from no statements to selected statements or from selected statements to all statements by

TRON:

Similarly, the global trace may be changed from all statements to selected statements or from selected statements to no statements by

TROFF:

Initially the global trace is set to selected statements. No harm will be done by promoting the trace when it is set to all statements or in demoting it when it is set to no statements. In particular the trace may be set to all statements from any state by

TRON; TRON:

or to selected statements by

TRON; TRON; TROFF:

or to no statements by

TROFF; TROFF:

A local assignment trace may be set on individual

statements by

TRON N:

where the operand may assume the latter four forms of Topic 5.2.

Initially the local assignment trace for each stored statement is off. To reset the local assignment the statement

TROFF N:

may be used where the operand may assume the same four forms as for TRON N.

Another way of looking at the behaviour of the assignment trace is to consider the global trace condition as a variable, say GLTR, having values of -1, 0 and 1. The action of TRON is

IF GLTR LT 1 GLTR=GLTR+1:

The action of TROFF is

IF GLTR GT -1 GLTR=GLTR-1:

The initial value of GLTR is 0.

The local trace may be considered a vector, say LT(N), having values of 0 and 1.

The statement TRON N could be considered to have the action of

LT(N)=1:

and TROFF N to be

LT(N)=0:

As each statement, N, is interpreted, a trace is

printed if

GLTR EQ 1 OR GLTR EQ 0 AND LT(N) EQ 1

Example

```
>110    A=1:
>120    C=2:
>130    D=3:
>140    E=4:
>150    F=5:
>160    G=99:
>170    STOP:
>TRON 130,150:
>RUN:
130 D=3
140 E=4
150 F=5
CONTROL RETURNED TO USER
>TRON:
>RUN:
110 A=1
120 C=2
130 D=3
140 E=4
150 F=5
160 G=99
CONTROL RETURNED TO USER
>TROFF; TROFF:
```

```
>RUN:
CONTROL RETURNED TO USER
```

5.10 List trace

A trace may be set so that a statement is listed each time it is interpreted.

This trace is similar, but distinct, to the assignment trace. All statements have the keyword LTRON rather than TRON.

5.11 SUSPEND statement

The state of the program can be saved on paper tape so that a program may be reinstated at some future time.

The full statement is

SUSPEND:

Interpretation awaits a keyboard character. This gives the user an opportunity to switch on a paper tape punch in parallel with the teletypewriter. Output consists of all stored statements, the symbol table, statements to reset the state of PB, PA, TRON and LTRON conditions and a RESET statement. The RESET statement allows the program to be resumed by giving ":" (carriage return).

If the character "P" is used to signal the start of the SUSPEND statement, output is directed to the high speed punch on the PDP-11 console. No output appears on the teleprinter.

Example

```
>110    A=1:
>120    C=2:
>130    D=3:
>140    E=4:
>150    F=5:
>LTRON 120,130:
>TRON 130,140:
>TROFF:
>LTRON:
>PB 120,150:
>160    STOP:
>SUSPEND:
110    A=1
120    C=2
130    D=3
140    E=4
150    F=5
160    STOP
ZRET=0
LTRON
TROFF
PB 120,150
TRON 130,140
LTRON 120,130
RESET HHHH HHHH
```

5.12 GO TO statement

A GO TO statement is provided so that a statement other than the next sequential statement may be selected as the next statement to be interpreted. The form is

GO TO N:

where N is a real expression. If the value of N is in the range 00-99 it is considered to be a statement number; if it is in the range 100-999 it is considered a sequence number and a value outside these ranges is an error.

Note that the general form of a real expression for N includes the FORTRAN assigned GO TO (GO TO I) and the computed GO TO (GO TO A(I)). The statement may be abbreviated to

GO N:

The GO TO statement may be used to commence automatic execution of a program at any statement.

5.13 ACCEPT statement

An ACCEPT statement may be used in the program to solicit the value of variables from the user. The form is

ACCEPT N1,N2:

Two types of operand may be used. A hollerith literal may be used to type information on the teleprinter. In this regard the ACCEPT statement is identical to the TYPE statement. The second form of operand is a variable, indicating that a value should be obtained from the user and

assigned to this variable. All three data types are available and the input is required to be the same type. The input may be an expression rather than a constant.

For each variable operand of an ACCEPT statement a fresh line is taken, a prompt of "A>" is printed and the system waits for an expression to be typed in. At this stage three responses are recognised

a) The ":" (carriage return) returns the user to the user mode with the line containing the ACCEPT statement as the next statement. The effect of any assignments is reversed.

b) The "?" character causes an identifier line to be typed consisting of the sequence number of the ACCEPT statement, the characters "ACCEPT" and the variable name.

c) An expression of the same type as the variable, followed by ":" (carriage return).

Example

```
>110    ACCEPT "ENTER A1,C2,D",A1,C2,D:
>120    TYPE "RESULT IS ",A1+C2/D; STOP:
>RUN:
ENTER A1,C2,D
A>?                          (Ask for more information)
110 ACCEPT A1
A>99/37:                     (Can give an expression)
A>34.5<E                     (Can edit input)
E>34.5
```

```
E>/4/3/
E>:
33.5:
A>?
110 ACCEPT D
A>.75:
RESULT IS  47.3424
CONTROL RETURNED TO USER
```

5.14 CALL and RETURN statements

A CALL statement is provided to enable other statements in the program to be used as a subroutine. The general form is

CALL N:

where N has the same meaning as in the GO TO statement. Control is passed to the new statement.

The RETURN statement is used to return control to the statement after the most recent CALL. There is only one form of the RETURN statement.

RETURN:

An arithmetic variable, ZRET, and an arithmetic array, ZR(N), are associated with the subroutine facility. The CALL statement is similar to

M ZRET=ZRET+1; ZR(ZRET)=M; GO TO N:

where M is the sequence number of the CALL statement and N is the sequence or statement number of the target statement.

The RETURN statement is similar to

Ztemp=ZR(ZRET); ZRET=ZRET-1; GO TO SUC(Ztemp):

where Ztemp is an internal variable and SUC(Ztemp) is the next statement after Ztemp. The variable ZRET and the array ZR may not be altered by the user, but may be accessed. The CALL statement may be used as an immediate statement.

Example

```
>110    X=2; CALL 9:
>120    STOP:
>130 9  TYPE "SQR IS ",SQR(X); RETURN:
>RUN:
SQR IS  1.41422
CONTROL RETURNED TO USER
>X=88; CALL 9:
SQR IS  9.38085
```

5.15 CONTINUE statement

A CONTINUE statement provides a null operation. More than one statement number may be effectively given to a statement by means of the CONTINUE statement.

5.16 STOP statement

The STOP statement is identical to the TYPE statement except that a return is made to the user mode after it has been interpreted. The STOP statement may have no operands.

5.17 Regaining control from automatic mode

The mode of the system may be changed from automatic to user at any time where input is not expected by hitting any keyboard character. The character is not relevant and its only function is to cause the mode change.

The present statement is completed and control given to the user. In the particular case of the FOR statement, the statement is abandoned and the FOR statement becomes the next statement.

A new line is taken on the teleprinter and a prompt consisting of a sequence number and '>' is given. This sequence number is the sequence number of the line just completed. The full facilities of the user mode are now availiable plus the three features described in Topic 5.8 and reproduced here.

a) The '?' will cause the sequence numbers of the previous and next statements to be typed. If the next statement has been nominated by a GO TO or CALL statement, the second operand has the form 'J N' where N is the statement or sequence number used in the last statement.

b) The ':' (carriage return) character results in a return to the suspended program.

c) The ';' character results in a return to the suspended program for one statement only.

Example

```
>110    X=0:
```

```
>120     X=X+1:
>130     GO TO 120:
>RUN:
130 >X:                  (Get attention with space bar)
53                       (Value of X is 53)
>?                       (Position?)
130 J 120                (Just done 130; Jump to 120)
>;                       (Do one more)
120 >X:                  (Just done 120)
54                       (Value of X is 54)
>?                       (Position?)
120 130                  (Done 120; sequential to 130)
>:                       (Go back at full speed)
```

5.18 ASYNCH statement

Often control is regained from the automatic mode with a specific task in mind. Perhaps a variable will be inspected and control returned to the automatic mode. The ASYNCH statement has been provided to simplify this process. Its form is

ASYNCH $ S1; S2:

A special character, in this case "$", is associated with a compound statement S1; S2. During automatic mode the key "$" may be used to

a) Gain control from the automatic mode,

b) Execute the compound statement as an immediate

statement and

c) Return to the automatic mode.

The six characters from ASCII 041 to 046 (exclamation, double quote, number sign, dollar, percent and ampersand) may all be used in this fashion. The association may be cancelled by the statement

ASYNCH $:

Example

```
>110     X=0:
>120     X=X+1:
>130     GO TO 120:
>ASYNCH $ TYPE "VALUE OF X IS ",X:
>RUN:
VALUE OF X IS  117
VALUE OF X IS  202
VALUE OF X IS  265
VALUE OF X IS  316
```

The special character associated with the compound statement may be used to generate the compound statement from the keyboard. This is effective in the user mode in column one, or in column 8 after giving the preamble for storing a statement. A carriage return is needed to complete the line.

5.19 Sample program

A program to play noughts and crosses is shown below. The game is played on a 3 by 3 layout numbered as

```
1 2 3
4 5 6
7 8 9
```

Internally a vector PO holds the state of the game. This is initiallized to all squares empty (110).

A move from the user is made via the ACCEPT statement at 120. Checks for validity of range (123) and elegibility (126) are made. If all is valid, the position is assigned and a typeout of the game made (130, 320-352).

Now a scan is made over the eight possible diagonals (140-192). The number of spaces (S(J)), the location of spaces (Y(J)), the number of X's (MG(J)) and the number of O's (YG(J)) in all diagonals are counted.

A scan is made to see if the user had just made a winning move (200). A scan is made to see if the program is in a winning position (210) and similarly a scan is made to see if the user is in a winning position (220). If none of these situations apply a move is made in the first available free space, mapped into considering the most favourable positions first by the vector A.

If no positions are left, the game is a draw (250).

The program:-

```
>110 1  FOR X=1,9 PO(X)=MT:
>120 2  ACCEPT "YOUR GO",N:
>123    IF N LT 1 OR  GT 9 TYPE "NOT VALID"; GO 2:
>126    IF PO(N) NE MT TYPE "NOT VALID"; GO 2:
```

```
>130     PO(N)=YOU; CALL 9:
>140     J=1:
>150 3   S(J)=YG(J)=MG(J)=0; X=D(J,1):
>160 99  IF PO(X) EQ MT Y(J)=X; S(J)=S(J)+1:
>170     IF PO(X) EQ YOU YG(J)=YG(J)+1:
>180     IF PO(X) EQ ME MG(J)=MG(J)+1:
>190     IF X=X+D(J,3) LE D(J,2) GO 99:
>192     IF J=J+1 LE 8 GO 3:
>200     FOR J=1,8 IF YG(J) EQ 3 GO 4:
>210     FOR J=1,8 IF MG(J) EQ 2 AND S(J) EQ 1 Z=Y(J); GO 5:
>220     FOR J=1,8 IF YG(J) EQ 2 AND S(J) EQ 1 Z=Y(J); GO 11:
>240     FOR X=1,9 IF PO(A(X)) EQ MT Z=A(X); GO 11:
>250     'DRAWN GAME'; GO 1:
>260 11  TYPE 'MY GO IS ',Z; PO(Z)=ME; CALL 9:
>270     GO 2:
>290 4   'CONGRATULATIONS ON YOUR WIN'; GO 1:
>300 5   TYPE 'MY GO IS ',Z; PO(Z)=ME; CALL 9:
>310     'I WIN'; GO 1:
>320 9   H1=HA(PO(1))&'!'&HA(PO(2))&'!'&HA(PO(3)):
>330     H2=HA(PO(4))&'!'&HA(PO(5))&'!'&HA(PO(6)):
>340     H3=HA(PO(7))&'!'&HA(PO(8))&'!'&HA(PO(9)):
>350     TYPE 2C,H1,2C,'-----',2C,H2,2C,'-----',2C,H3:
>352     RETURN:
>YOU=3:
>ME=2:
>MT=1:
>HA=' XO':
>A(8)=6:
>A(2)=1:
>A(6)=2:
>A(3)=3:
>A(7)=4:
>A(1)=5:
>A(5)=7:
>A(9)=8:
>A(4)=9:
>D(1,1)=1:
>D(1,2)=3:
>D(1,3)=1:
>D(2,1)=4:
>D(2,2)=6:
>D(2,3)=1:
>D(3,1)=7:
>D(3,2)=9:
>D(3,3)=1:
>D(4,1)=1:
>D(4,2)=7:
>D(4,3)=3:
>D(5,1)=2:
>D(5,2)=8:
>D(5,3)=3:
>D(6,1)=3:
```

```
>D(6,2)=9:
>D(6,3)=3:
>D(7,1)=1:
>D(7,2)=9:
>D(7,3)=4:
>D(8,1)=3:
>D(8,2)=7:
>D(8,3)=2:
>ZRET=0:
>RESET IHHI HIIH:

Execution:-

>RUN:
 YOUR GO
A>9:
   ! !
  -----
   ! !
  -----
   ! !O
 MY GO IS  5
   ! !
  -----
   !X!
  -----
   ! !O
 YOUR GO
A>1:
 O! !
  -----
   !X!
  -----
   ! !O
 MY GO IS  3
 O! !X
  -----
   !X!
  -----
   ! !O
 YOUR GO
A>7:
 O! !X
  -----
   !X!
  -----
 O! !O
```

```
 MY GO IS  8
 O! !X
 -----
  !X!
 -----
 O!X!O
 YOUR GO
A>4:
 O! !X
 -----
 O!X!
 -----
 O!X!O
CONGRATULATIONS ON YOUR WIN
```

6. Allowable next characters

The set of allowable next characters may be solicited by entering the character '>'. In the general case, where there is more than one character which could be given next, the '>' is echoed. A fresh line is typed consisting of all the characters (except '<', '>' and '\') which are allowable as the next character. A dollar character is given as the first and last characters of the line to allow the space character to be easily identified. The original line is retyped and the user may continue with his input.

Examples

```
1.2>
$ *+-/0123456789:;E^$
1.2+(>
$(+-.0123456789ACDEFGIJKLMNOPQRSTUVWXYZ$
1.2+(9/7>
$)*+-./0123456789E^$
1.2+(9/7)>
```

```
$ *+-/:;^$
1.2+(9/7):
2.48571
```

If there is only one character allowable, this character is typed instead of the ">". Characters will continue to be typed while there is only one character allowable in the next position. Thus one or more characters may be echoed in response to the ">" character. For example, the CONTINUE statement is uniquely identified by the first five characters, CONTI. If ">" is entered after "CONTI" the system will respond with N, U and E.

Similarly, the set of allowable next characters for two positions ahead may be typed by entering the character ">" a second time.

Example

```
>3+4 >
$EGILNU$
3+4 >
E Q
G ET
I F
L ET
N E
U N
3+4 GT 5:
TRUE
```

In the example above the characters allowed as the n+1th character are given down the page as the first character of each line. The characters from column three are the set of characters allowed as the n+2nd character. For example G is allowed as the n+1th character and E or T as the n+2nd character to form GE or GT.

A mode is available where the set of allowable next characters is given whenever the user makes an error. In the general case, the offending character results in the audible bell character, the '>' character, a new line giving the set of allowable next characters and a line recovering the users line so he may continue.

In the particular case where only one character is correct the system will give the audible bell character and then the correct character. If the user were typing the UNLESS keyword and had typed the U and the N, but had made a mistake typing the L, the system would type the L, E, S, S and the space at the end of the keyword. The effect is that the system will "take over" typing these keywords if the user makes a mistake.

7. Appendix

7.1 Places where characters are expanded

a) A space in column one gives a 3 digit sequence number and a space.

b) A space in column 5 after a sequence number and a space gives 3 spaces for a null statement number.

c) A non-space in column 6 after a sequence number and a single digit statement number gives two spaces and the non-space is presented to column 8.

d) A non-space in column 7 after a sequence number and a statement number gives a space and the non-space is presented to column 8.

e) A non-space after ' TRUE' or ' FALSE' generates a space followed by the non-space.

f) A special character associated with a compound statement via an ASYNCH statement may be used to generate the compound statement.

g) A non-space after a ';' separating statements generates a space followed by the non-space.

7.2 Suppressing output

In some circumstances a large amount of output may be given by the system. The user may suppress output while it is being typed by striking any keyboard character for the following situations.

All error messages.

Any fine trace (i.e. as each operation is being performed).

The output from '>'.

LIST, SYMBOLS and SUSPEND output.

LTRON and TRON (in the automatic mode the keyboard

character will also cause the user mode to be entered).

7.3 Functions

The following functions are available:-

Name	Input	Output
SQR	real exp.	square root
ABS	real exp.	absolute value
INT	real exp.	integer part
RND	real exp.	rounded integer part
LNG	string exp.	length of string
HXT	any exp.	output string

The operation, ^, has not yet been implemented fully. For any arguments its result is one.

7.4 Properties of statements

	immediate	stored	; cont	FOR	suffix
ACCEPT	NO	YES	YES	NO	YES
ASYNCH	YES	NO	IROL	NO	NO
C(omment)	YES	YES	IROL	NO	NO
CALL	YES	YES	NO	NO	YES
CLEAR	YES	YES	NO	YES	NO
CONTINUE	YES	YES	YES	NO	NO
DELETE	YES	YES	NO	NO	NO
EDIT	YES	NO	NO	NO	NO
FOR	YES	YES	IROL	NO	NO
GO TO	YES	YES	NO	YES	YES
IF	YES	YES	IROL	YES	NO
LIST	YES	YES	NO	NO	NO
LTRON,TRON	YES	YES	YES	NO	NO
PA,PB	YES	YES	YES	NO	NO
RESET	YES	NO	NO	NO	NO
RETURN	NO	YES	NO	NO	YES
RUN	YES	YES	NO	NO	YES
STOP	YES	YES	NO	NO	YES
SUSPEND	YES	NO	YES	NO	NO
SYMBOLS	YES	YES	YES	NO	NO
TYPE	YES	YES	YES	YES	YES
expressions	YES	YES	YES	YES	YES

Notes

a) ASYNCH is an immediate statement but is stored.

b) RESET is not available to the user.

c) IROL means Includes Rest Of Line.

7.5 Invoking options

Several special modes may be invoked, with control characters. These control characters may be used at any time the system is expecting input. They are:-

Control-F Prompter on - Topic 4.13

Control-Q Prompter off

Control-C Numeric check on - Topic 6

Control-X Numeric check off

Control-U Fine trace on

Control-V Fine trace off

The fine trace mode exhibits the state of expressions as each operation is performed. The mode is also entered to repeat calculations which ended in error (Topic 4.4).

All these modes are reset at initialization of the system or may be reset individually.

Appendix B

The Development of the ACL system

The Development of the ACL Language and its Implementation ACL-NOVA

By N. W. Bennett† and P. L. Sanger*

The development of a one-terminal system ACTIV-8 for the PDP-8 computer is first described. The concepts of the dynamic syntax checking of keyboard input from the terminal and the interruption of stored program execution via the communicate statement are presented.

The development of the new conversational language ACL and its implementation as a multi-user conversational interpreter ACL-NOVA is then described. The ACL-NOVA system runs as a stand-alone system and provides a powerful conversational computing facility for many users in a time-sharing environment.

1. INTRODUCTION

The first conversational computing facility used at the AAEC Research Establishment was a one-terminal interpretive system called ACTIV-8 (Bennett 1968) developed for a PDP-8 computer. The system was designed to give individual scientists and engineers an easy, direct way of solving small numerical problems and at the same time to provide results more quickly than by use of a large batch processing computer.

ACTIV-8 was implemented on a 4K PDP-8 computer, and proved very popular with those users for whom it was designed. To expand the facilities provided by the ACTIV-8 system a multi-user conversational interpreter was written for a NOVA computer. This interpreter was based on a new conversational language, ACL, developed from the experience gained using ACTIV-8. The implementation of the language is referred to as ACL-NOVA and was carried out on a 12K NOVA computer with five teletypewriter terminals (Sanger 1971).

The development and implementation of the ACTIV-8 language is discussed below, leading to a discussion of the ACL language and its implementation. These discussions are limited to the most important features of ACTIV-8 and ACL. For a more detailed description of the systems the reader is referred to the reports (Bennett 1968 and Sanger 1971).

2. THE ACTIV-8 LANGUAGE

2.1 General Discussion

The design of the ACTIV-8 language was influenced by the fact that the people who would be using the system were already familiar with FORTRAN. The initial ACTIV-8 language was fairly simple. The main statements were an arithmetic statement, a GO TO statement, an arithmetic IF statement, a TYPE statement and an ACCEPT statement. Variable names were restricted to a letter followed by a number, and the only mode of arithmetic was real.

Statements were entered from the teletypewriter terminal in the form

statement number — space — statement

and were used to build up a stored program. Statement numbers were optional and, if present, consisted of two digit numbers in the range 00-99. Statements were stored strictly in the order in which they were entered, and the stored program was executed on receipt of an END statement.

2.2 Dynamic Syntax Checking

The teletypewriter attached to the PDP-8 computer is operated in full-duplex mode. This means that characters may be transmitted between the teletypewriter and the computer in both directions simultaneously. In this case, it also means that a character pressed on the keyboard is transmitted to the computer but is not automatically printed on the teleprinter. In normal practice, a character entered from the keyboard is read by the computer and transmitted back, or "echoed", immediately to the teleprinter, which thus appears to operate like a normal office typewriter. By taking advantage of full-duplex mode operation, the computer can check if the character entered is valid in syntax and, if not, the character is simply not echoed back to the teletypewriter.

The dynamic syntax checking of input was an important part of the ACTIV-8 system and later played an important part in the design and implementation of the ACL language.

To see how this syntax checking was carried out, consider the case of an arithmetic expression. In its simplest terms, an arithmetic expression must consist of operands (i.e. variables and numbers) separated by operators (e.g. * and +). An expression must start with an operand (or unary + or —) and it must finish with an operand. The (simplified) structure of an arithmetic expression is illustrated by the directed-graph shown in Figure 1, and the syntax checking ensured that this pattern was obeyed. As shown in Figure 1, a "valid" Carriage Return was also required to complete a statement. Expressions containing brackets can be checked in the same way. An opening bracket may only appear at the start of an expression or after an operator, while closing brackets may only appear after an operand. An extra condition for brackets is that there may never be more closing brackets than

*Australian Atomic Energy Commission Research Establishment, Lucas Heights, N.S.W. 2232. †Present address, UCCompunet Ltd., 23 Cleg Street, Artarmon, N.S.W. 2064.

opening brackets, and before a carriage return is accepted there must be an equal number of opening and closing brackets.

Reference to the standard mathematical functions took the form of a three letter function name followed by an arithmetic expression enclosed in brackets (e.g. LOG(2)). The first indication of a function name was the combination of a letter followed by another letter instead of the number required for a variable name. An attempt was then made to verify these first two letters in a table of function names. If the second letter was not verified in this way, it was rejected. The character was not echoed to the teleprinter and the program looped to request a new character in its place. If the character was verified, the character was printed and that table entry used to check the rest of the function name.

All the statements in the ACTIV-8 language were checked in this way. For example, if the first character entered is the letter G, then at this stage the statement may be either a GO TO statement or an arithmetic statement. If the second character is the letter O, the user is committed to a GO TO statement. The only sequence of characters allowed from this point is space, T, O, space, number, number and carriage return. If the second character were a number, a commitment to an arithmetic statement would be made. Variable names were restricted to a letter followed by a number to settle this ambiguity as to statement type as soon as possible. It must be emphasised that while this scheme can detect incorrect syntax, it cannot detect the entry of incorrect information. For example, 1.6 has the same syntax as 1.5, and there is no way for the system to detect which number the user really wanted to enter.

When a statement was completed by a "valid" carriage return, it was stored for later execution, except in the case of an END statement which caused the stored program to be executed.

Dynamic syntax checking offered both advantages and disadvantages for the ACTIV-8 system. Since all statements were checked on input there could be no syntax errors at execution time. This saved program space in the system and saved time in processing each statement. However, the sophistication of checking the syntax of incoming characters was, at that time, thought to preclude the possibility of backspacing over errors, particularly on a small machine. Once an error had been made, the entire line had to be abandoned.

2.3 Interpreter versus Compiler

Two broad methods of processing the language are available. A compiler can scan the statements and produce an object program which must be stored and can later be given control. An interpreter also scans the statements, but instead of generating actions to be stored and executed later, those actions are carried out as each statement is considered. Thus the interpretive approach does not require as much space as a compiler, but to execute a program each statement must be decoded from its source form each time control passes to it. The interpretive approach was chosen for the implementation of ACTIV-8 because of the limited space available.

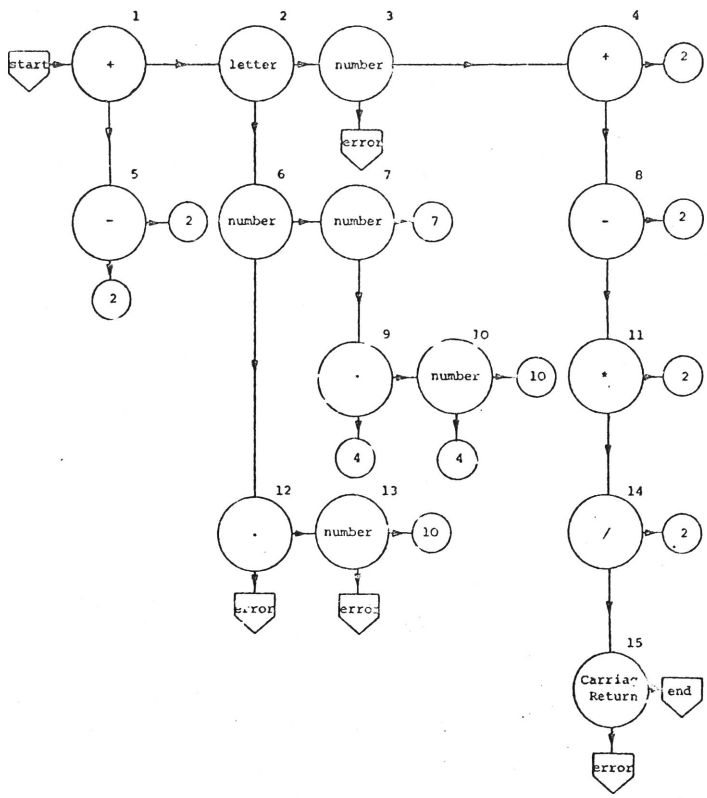


Figure 1: In this diagram incoming characters are checked against individual characters or groups of characters at nodes. If the character is accepted at that node, the character is typed back to the user, another character is accepted and control moves horizontally to the right. If the character is not accepted by that node, control passes vertically downwards to a new node, or to the error exit.
To follow the diagram consider the expression A1*3. Syntax checking begins by comparing the first character, A, with unary plus (node 1). Since these do not match we move vertically down to node 5 and compare A with unary minus. Again there is no match and we see if A is a letter (node 2). This time there is a match, so we move horizontally and compare the next character in the expression, 1, with a number (node 3). A match is obtained and we now compare the next character, *, with + (node 4). There is no match and * is next compared with — (node 8). There is a match when * is compared in node 11 and we now compare the next character, 3, with a letter (node 2). There is no match and we move vertically down to compare 3 with a number (node 6). These match and we finally go through nodes 7, 9, 4, 8, 11, 14, and 15 to match the carriage return at the end of the expression.

An interpreter has a number of advantages over a compiler. For example, when a GO TO statement refers to a statement number that is not present in the source program, the process of compilation cannot be completed in the normal way. In an interpreter execution terminates only if an attempt is made to pass control to a statement with that statement number. The case where two source statements have the same statement number is very similar.

Another feature of an interpreter is that there is always a convenient link between the name of the variable and its value; that is, via the symbol table. With a compilation system, all variable names are translated into machine locations during compilation and the symbol table is not usually accessible during execution. The ACTIV-8 system takes advantage of this feature

in the ACCEPT statement. The ACCEPT statement allows the user to input a value during execution which will be associated with a variable. Because the symbol table is available, ACTIV-8 allows the user to enter an arithmetic expression rather than just a numeric value. This expression is evaluated and its value is assigned to the variable. Note that this arithmetic expression is entered by the user during stored program execution. This proved to be very useful. For example, when the statement ACCEPT A1 is executed, the variable A1 could be increased by one by entering A1+1, A1 could be increased by twenty percent by entering A1*1.2, A1 could be given the same value as another variable by entering this variable name e.g. D4, and if A1 was expected to have a value in centimetres then 7 inches would be entered as 2.54*7.

With a compiler it is not normally possible during execution to check whether the program is using a variable which has been set to a particular value. In an interpreter, values can only be obtained by using a variable name to access the symbol table and this allows undefined variables to be detected.

2.4 The Communicate Statement

The generality of the possible responses to the ACCEPT statement led to the design of the communicate statement. This statement consisted of the question mark character stored as part of the program and its action was determined by the response given by the user at execution time. Some of the possible responses were

(i) An arithmetic statement which is interpreted and control returns to the communicate state. This looping allows the user to give several responses.

(ii) The left hand side of an arithmetic statement, i.e. a variable name followed by an equals sign and a carriage return. The value of the variable is printed on the same line and control returns to the communicate state.

(iii) Two numbers indicating a statement number. Control passes to the statement with that statement number.

(iv) A null response (i.e. a carriage return immediately) indicating the end of the dialogue. Control passes to the statement stored after the communicate statement.

The above features (and others), proved to be very useful for debugging programs, provided the user stored a communicate statement as part of the program. The user can easily determine the value of variables and influence the path of control.

The communicate statement also allows the system to act like a desk calculator, since the program

```
?
END
```

lets the user enter a series of arithmetic statements which can refer to previous results. For example, a user could give the responses

```
A1=3
B1=4
C1=5
S1=(A1+B1+C1)/2
A2=SQR(S1*(S1-A1)*(S1-B1)*(S1-C1))
A2=
```

where the last line causes the value of the variable A2 to be printed.

To allow for the case where a communicate statement may not have been stored as part of a program, a facility was added so that the communicate state could be entered if the user pressed the question mark character during program execution. Processing of the current stored statement is completed and control is given to the communicate state, with the position in the program where the interruption took place being printed. The user can now interact with his program using the responses described above. This feature is also convenient for another user who wants to do a quick "desk" calculation and return control to the stored program (provided, of course, that variables in the suspended program have not been altered unintentionally).

2.5 Implementation

The ACTIV-8 language was implemented on a 4K PDP-8 computer supporting an Anelex line printer, a Burroughs card reader and one teletypewriter terminal. This computer was already being used to provide a card listing facility.

The PDP-8 computer has a 12-bit word which results in a fairly limited instruction set and an addressing scheme which can directly refer to only 256 words from any particular location. The combination of word length, restricted instruction set, fixed page addressing scheme and limited storage available in the PDP-8 computer influenced the design of the ACTIV-8 system. To simplify the task of developing and testing the ACTIV-8 system, a program was written for the IBM 360/50 computer to accept PDP-8 source language from cards, assemble the PDP-8 program into a pseudo-core storage and simulate the actions of executing the assembled program on the PDP-8 computer. This type of program is known as an assembler/simulator.

3. THE ACL LANGUAGE

3.1 General discussion

To expand the facilities provided by the ACTIV-8 system a multi-user conversational interpreter was written for a NOVA computer. This interpreter was based on a more sophisticated language, ACL, developed as a result of the experience gained with ACTIV-8 and a clearer understanding of the needs of this class of user.

3.2 Development of the Language Structure

The most useful development had its origins in the communicate statement. Here the user was able to perform arithmetic statements and GO TO statements in an immediate mode. The ACL language recognised that, in theory, there need be no barrier to any statement being used in this way; and it would be very convenient to have this mode available on first entering the system. Hence there would be two modes

1) The system would accept statements put in from the teletypewriter; execute some statements in an immediate mode and store others.

2) The system would execute previously stored statements in an automatic mode.

To allow statements to be referenced for storage, deletion and editing, a sequence number was introduced. This appeared at the start of statements which were meant to be stored, distinguishing them from statements meant for immediate execution. A requirement on the exact layout of the statements was that the dynamic syntax checking which had proved useful in ACTIV-8 should be easily implemented. In this instance it led to a space character separating the sequence number and the actual statement. The two types of statement looked like this

sequence number — space — statement

statement

For the syntax checking, the sequence number was indistinguishable from a number as part of an arithmetic expression (this possibility is raised later, Section 3.5). The space, which was not allowed in arithmetic expressions, served to give an indication that a sequence number was meant.

Because of the FORTRAN background of most users and because statement numbers were used in ACTIV-8, it was felt that the new language should have statement numbers. Sequence numbers can also serve as targets for transfers of control, but in some cases statement numbers are more convenient. For example, in debugging a program one may wish to change the statement number of a statement. If only the sequence number was available a change might also mean a change in the order in which statements were executed sequentially. As both sequence and statement numbers can be used for transfer of control and they were both integer numbers, an ambiguity arose. This was resolved by having statement numbers from zero to 99 and sequence numbers from 100 to 999.

Again, a space was introduced between the statement number and the statement, this time for purely aesthetic reasons. The two types of statement now had this format

sequence number — space — statement number — space — statement

statement

For those stored statements which did not include a statement number, two spaces could have been used. To simplify the process the user could enter a single space and the system responded with 3 spaces: two for the statement number and one for the compulsory space. Also the user could ask for a new sequence number by giving a space at the start of the line. This sequence number is 10 greater than the last sequence number specified. Thus for a statement to be stored, a new sequence number and a null statement number may be nominated by two spaces.

3.3 Statement Modification

Provision was made for deleting stored statements by entering

sequence number — space — carriage return

Statement numbers could be altered by entering

sequence number — space — new statement number — space — carriage return

A facility was introduced to allow the users to backspace n characters by typing <n, to be able to reconstruct the current line in an EDIT mode (see below) by typing <<, and to be able to cancel the current line by typing <<<.

A statement was provided to allow users to modify stored statements in what is called edit mode. The statement was

EDIT n

where n is an arithmetic statement or expression (see Section 3.5) indicating the sequence number of the statement to be changed. This statement is first displayed on the teletype. The user is now able to compose a new line consisting of characters taken sequentially from the old line or characters inserted from the keyboard. The ability to skip characters from the old line was provided. A complete syntax analysis is carried out on the new line as if it were being entered from the keyboard.

3.4 Variables

Variable names were expanded from a set format of a letter followed by a number in ACTIV-8 to the general form of a letter which may be followed by up to three letters or numbers. Arrays were introduced with array names which were variable names one or two characters long. The arrays could have either one or two subscripts and each subscript was allowed the generality of an arithmetic statement or expression. Function names remained as three characters, such as LOG.

3.5 Arithmetic Statements and Expressions

The basic operands in an expression are variables and numbers. The arithmetic operators are +, —, *, / and ↑ (power) with the usual mathematical hierarchy. Assignments are indicated by ← instead of the equals sign used in FORTRAN and ACTIV-8.

Immediate statements were designed to include the capability of evaluating arithmetic expressions and printing the results. Thus if one entered 2+3, the system would respond with the value 5.

If one wished to assign a value of LOG(2) to a variable, one could use A1←LOG(2). Statements should behave exactly the same way whether they were executed as part of a stored program or in immediate mode. Since a stored program would normally contain arithmetic assignment statements, it is important that these should not cause printing. Thus it was decided that the statement A1←LOG(2) should not cause printing. To inspect the value of A1, one could simply type in A1.

The first rule that determined whether the value of an expression should be printed was the absence of an assignment. A development of this led to the form (A1←6) which would cause both the assignment and its value to be printed. It was convenient to consider the form (A1←6) as having a value like any other operand and to include it as such in expressions; e.g. A2←14*(A1←6). This device of enclosing a complete arithmetic expression in brackets resulted in the first rule being modified.

If an expression began with a variable followed by an assignment arrow, then the expression was called an "arithmetic statement" and its value was not printed; if it commenced in any other way, it was called an

"arithmetic expression" and its value was printed. Thus the values of 2+3 and 4+(A1←6) should be printed, while the values of A1←6 and A23←24+(C3←A1) should not.

Finally, the extra bracket pair around assignments used as operands was dropped on the understanding that assignment should proceed from right to left, with the value of the expression on the right of the assignment arrow being assigned to the variable on the left. Thus the expression A23←24+(C3←A1) becomes A23←24+C3←A1. This allows a very general structure for multiple assignments and includes the type of multiple assignments that occur in ALGOL e.g. A←B←C←2 is allowed.

3.6 ACCEPT and TYPE Statements

The ACCEPT and TYPE statements were included in ACL. While the ACCEPT statement was unchanged, the generality of the TYPE statement was increased in a number of ways. Firstly, literals were allowed so that descriptive information as well as numeric values could be printed. Secondly, a variety of separators provided line control. Thus, if a comma were used as a separator, the next operand would be given on the same line. If a semi-colon were used, the next operand would be given on a new line. The form <arith expression> was introduced as an operand. This form was used to position the carriage to an absolute location. Lastly, a function called DPT was provided to allow numbers on different lines to be typed so that their decimal points were aligned vertically. The value of the DPT function is the position of the decimal point relative to the start of the number. If the number A3 has a value of 89.5 then DPT(A3) would have a value of 3. To ensure that the decimal point of a number is printed in column 10 the statement

TYPE <10-DPT(A3)>,A3

may be used.

3.7 Control Statements

The GO TO statement was expanded to allow an expression instead of an absolute statement number. If the value of the expression was in the range zero to 99, the reference is to a statement number. If the value was in the range 100-999, a sequence number was required. This form encompassed the FORTRAN ASSIGN statement (GO TO I) and also the computed GO TO (GO TO A3(I)).

A CALL statement and its associated RETURN statement were introduced to allow the user to reference other statements within the program as subroutines.

A logical IF statement was provided to give decision making capabilities. The format was

IF(expression.relational operator.expression)
  statement

where the relational operator could be any of the six common FORTRAN relational operators. The generality of the expressions allowed the IF statement to be used as a loop control e.g.

IF(I←I+1..LE.100) GO TO 23

3.8 Supervisory Statements

A number of statements were introduced for auxiliary purposes. The EDIT statement falls into this category. Because of their usage, these statements are only available as immediate statements. A list of all the statements with an indication of these restrictions is given in Table 1.

TABLE 1. List of ACL statements

IMMEDIATE STATEMENTS

Comments

Arithmetic statements

Arithmetic expressions

RUN

GOBTOS {arith stmt or exprn}

LIST [:] [% arith stmt or exprn[,arith stmt or exprn]]

SYMBOLS [:]

CLEAR [% variable[, variable] ···]

SPACE

FTRON

FTROFF

TRON[% arith stmt or exprn]

TROFF [% arith stmt or exprn]

TYPE [% {[:[:]···] operand [{[:[:]···] operand ,}···] :[:]···}···]

where the operands take the form, [<arith stmt or exprn>,] {'characters' ["]arith stmt or exprn}

PR % {arith stmt or exprn}

PA % {arith stmt or exprn}

STOP

END

SUSPEND [:]

EDIT % {arith stmt or exprn}

System Messages

STORED STATEMENTS

Comments

Arithmetic statements

Arithmetic expressions

GOBTOS {arith stmt or exprn}

TYPE [% {[:[:]···] operand [{[:[:]···] operand ,}···] :[:]···}···]

where the operands take the form, [<arith stmt or exprn>,] {'characters' ["]arith stmt or exprn}

ACCEPT % {variable[,variable]···}

CALL % {arith stmt or exprn}

RETURN

CONTINUE

STOP

IF ({arith stmt or exprn} {.EQ. .NE. .GT. .GE. .LT. .LE.} {arith stmt or exprn}) % {arith stmt or exprn / TYPE stmt / ACCEPT stmt / GO TO stmt / CALL stmt / RETURN stmt / CONTINUE stmt / STOP stmt}

PAUSE

END

A RUN statement is used to begin execution of a stored program starting at the statement with the lowest sequence number. A LIST statement is provided to give a listing of the stored statements. A companion SUSPEND statement is designed to record the state of the program as a series of immediate arithmetic statements followed by an immediate GO TO statement. These two statements may be used to prepare a paper tape so that execution may be resumed at some later time.

Trace statements allow assignments that occur in individual stored statements to be printed, or allow every stored statement that is executed to be listed.

Two conditional PAUSE statements are available. A pause before (PB) and a pause after (PA) statement will cause control to be returned to the user just before or just after a specified statement is executed. These two statements provide a valuable checkpoint facility and, together with the trace statements, form a powerful aid to program debugging.

4. DEVELOPMENT OF THE ACL-NOVA SYSTEM

4.1 General Discussion

The ACL language was implemented as a multi-user conversational interpreter on a 12K NOVA computer supporting five teletypewriter terminals.

The NOVA computer has a 16-bit word length, a powerful instruction set and four registers, two of which can be used as index registers. The addressing scheme on the NOVA allows direct reference to page zero (the first 256 locations), 256 locations centred about the program counter and 256 locations centred about each of the index registers. Each instruction occupies one NOVA word. An assembler/simulator for the NOVA computer written to run on the IBM 360/50 computer (Sanger 1970) was also used to develop and test the ACL-NOVA system.

The resident part of the ACL-NOVA system uses the first 6K of the NOVA computer. Parameters such as the number of simultaneous users to be supported by the system, the size of each increment of work area to be allocated to a user and the total amount of work space available to the system are not assembled into the program but must be specified when the system is first loaded into the computer. Work space may be reserved for terminals or left "floating", and when a user signs on at a terminal either the space that was reserved for that terminal or one increment of work space is initialised. Dynamic allocation of work space allows the "floating" work space to be allocated to users who need more space, and this is returned to the system when the user completes work at his terminal.

4.2 Arithmetic

All arithmetic is performed on 32 bit floating point numbers stored as two NOVA words. These numbers have a sign bit, a 7 bit characteristic and a 24 bit fraction. This form corresponds to the short floating point number on an IBM 360 computer and can represent numbers approximately in the range 5.4E-79 <= number <= 7.2E+75. This choice simplified the development of the floating point software because the results obtained from the NOVA simulations could easily be compared with the corresponding IBM 360 hardware floating point instructions. The only difference between the two was that the NOVA routines were set up to round the result of each arithmetic operation to produce more uniform error distributions. The times taken to perform the arithmetic operations were: addition and subtraction, 0.55 msec; multiplication, 1.72 msec; division, 2.96 msec.

4.3. The ACL-NOVA Program

The ACL-NOVA program has two main parts. One part, the Interrupt Handler, syntax checks the input from the terminals and stores it essentially as a string of source characters in a buffer area located in the appropriate user work area. This part of the program also handles any output that must be sent to the terminals.

The second part, the Background Program, controls statement execution. Requests for statement execution from each terminal are treated at the same priority level. Each terminal is serviced in a round-robin approach with the computer processing one statement at a time for each user. Thus the Background Program: (i) executes an immediate statement by interpreting the source string stored in the user buffer area and performing the indicated operations, (ii) stores a statement for later execution by moving the character string from the buffer area to another part of the user work area, and (iii) executes a stored program by fetching one statement at a time from the user work area and moving this into the buffer area for processing as in (i).

The ACL-NOVA program spends most of its time in the Background Program executing statements or checking whether there is a statement to be executed. However, the Background Program can be interrupted at any time by input or output operations at a terminal and these interrupts are serviced completely before control is returned to the Background Program.

4.4 Allocation of Space within a Work Area

The first 135 words of each user's work area is used for various pointers and buffers for the ACL-NOVA system. It includes two flag words and nineteen words of pointers used by the system as well as an input buffer, an internal code buffer and an output buffer. The remaining part of the user work area is used for storing statements and for the symbol table. To provide the most efficient use of core storage, statements are stored starting from the end of the buffer areas and continuing towards the end of the work area, while symbol table entries are stored starting at the end of the work area and continuing backwards towards the start of the work area. In this way a program with many statements but few variables, or a program with few statements but many variables can be handled. By setting up the work area in this way, the task of expanding user areas was simplified and this is discussed in Section 4.9.

4.5 Internal Code Buffer

When a statement is being entered at a terminal, it is syntax checked character by character and a copy of the "valid" characters is kept in the input buffer in input or external code form. At the same time, the

syntax checker translates these characters into an internal code to simplify statement processing and stores them in an internal code buffer. When the input statement reaches a point where the statement type can be determined, an appropriate indicator is stored at the front of the internal code (IC) buffer.

4.6 Editing Statements and Error Correction

When a stored program is listed at a terminal, each statement is translated back into external code form and stored in the output buffer from which it is sent one character at a time to the terminal. To allow statements to be modified using the EDIT statement, the required statement is listed at the terminal in the above way. The pointer to the current position in the output is reset to point to the start of the output buffer.

When the space character is entered, the character indicated by the output buffer pointer is stored in the input buffer as virtual input from the terminal and is syntax checked in the normal way. If the character is valid, the IC buffer is updated, the input buffer pointer increased by one and the output buffer pointer is increased by one; otherwise it is rejected. In the same way, new characters can be entered from the terminal during edit mode and these affect only the input buffer, the input buffer pointer and the IC buffer. When each DEL or RUB OUT character is entered, the output buffer pointer is increased by one, thus deleting one character from the statement.

A similar procedure allows input from the terminal to be modified in edit mode. The only difference is that after the carriage return characters are entered, the current contents of the input buffer are copied into the output buffer before sending a carriage return, line feed to the terminal. The original input is then edited as described above.

The correction of input from a terminal by deleting the last n characters may also be carried out simply. If there are m characters in the input buffer (m is always greater than n), then the input buffer pointer is reset to m-n and the contents of the input buffer are syntax checked in a loop that analyses 1 character, then 2 characters, then 3 characters, . . ., then m-n characters so that the contents of the IC buffer are updated correctly. This multiple scan is particularly important in the case of an IF statement, and at the point in the input where the statement type may be altered by the deletion of input characters.

4.7 Symbol Table

Each symbol table entry uses four NOVA words; two words to store the variable name and two words to store its floating point value. This allows a simple variable name consisting of a letter, which may be followed by up to three letters or numbers, stored in internal code form padded with blanks where necessary as shown in Figure 2(a).

Subscripted variable names were chosen to be a letter, which may be followed by a letter or a number, stored in the first word in internal code form padded with a blank if necessary with the appropriate subscript or subscripts stored in the second word. This allows a singly subscripted variable to have a subscript in the range 0 to 65535, and each of the subscripts of a doubly subscripted variable to be in the range 0 to

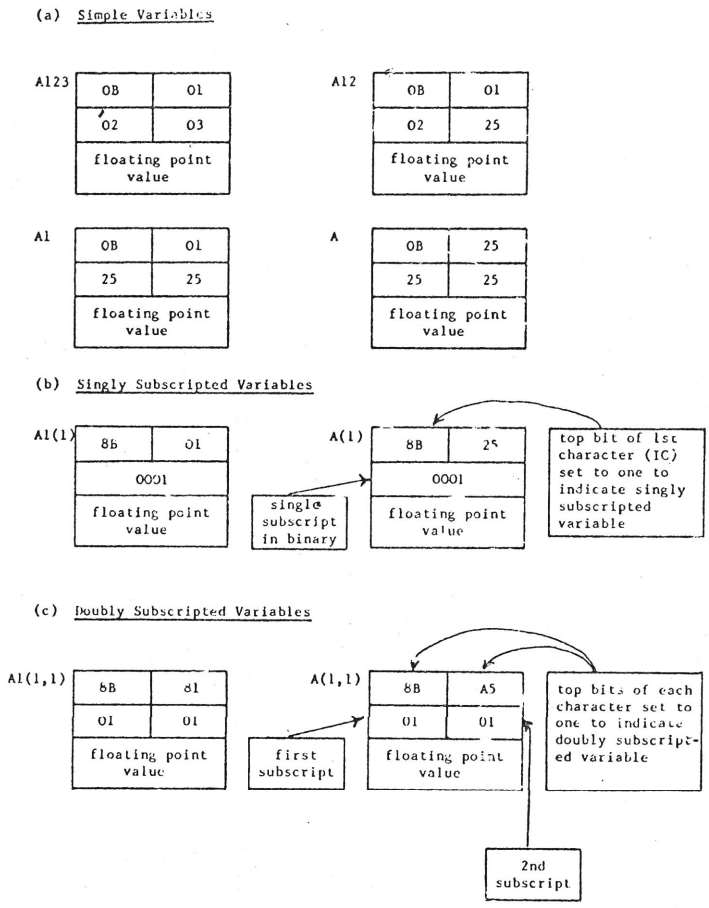


Figure 2: Symbol table entries

255 as shown in Figure 2(b) and 2(c).

The fact that the internal codes for letters and numbers required at most 6 bits was used to distinguish between simple variables, singly subscripted variables and doubly subscripted variables. This means that the top bits of each character in the first word of a symbol table entry can be used as indicators. A singly subscripted variable is set up so that the top bit of the first character in the name is set to one. Doubly subscripted variables are set up with the top bit of each of the characters in the first word of the symbol table entry set to one. This is also shown in Figure 2(b) and 2(c). This scheme has the advantage that there is no conflict when the variables A, A(1), A(1,1) are referenced by a user, as occurs for example in FORTRAN.

The symbol table entries are not ordered and a given entry is located by a sequential search of the symbol table. The search is carried out by adding an entry to the end of the symbol table for the variable required and ensures that a match is obtained from the sequential search. If the match occurs on the last entry of the symbol table, this indicates that the variable is undefined. The choice of a sequential search method simplified the structure of each work area and is adequate for this application where the symbol table is quite small for most users.

4.8 Stored Statement Processing

Execution of a stored program begins with the execution of a RUN statement or an immediate GO TO statement. This causes the sequence number of

the first statement in the stored program or the sequence number of the statement referred to in the GO TO statement to be stored as the sequence number of the next statement to be executed and the stored program to be executed flag is turned on.

When this terminal is next serviced by the round-robin background program this statement is located in the user work area and is copied into the IC buffer for processing. The sequence number of this statement is stored as the sequence number of the statement being executed (this is referenced in error messages, noting pause conditions, for trace output etc.) and the sequence number of the next stored statement (which can easily be located by using the length of the statement being processed) is stored as the sequence number of the next statement to be processed. The stored statement is now executed from the IC buffer. In this process, the IC buffer is partially over-written by replacement operands (see Section 4.10) and this is why the statement cannot be executed directly from the user work area. At this point control is returned to the Background Program.

Sequential statement processing continues in this way until a statement that alters the path of control is executed. The simplest way to do this is to execute a GO TO statement. In this case, once the statement has been processed the sequence number referred to in the GO TO statement is stored as the sequence number of the next statement to be processed.

The CALL statement can also be used to transfer control to another group of stored statements. This statement causes (i) the sequence number of the statement referred to in the CALL statement to be stored as the sequence number of the next statement to be executed, (ii) the sequence number to be used by the RETURN statement to be stored in this CALL statement in the user area, and (iii) the sequence number of this CALL statement to be stored as the sequence number to be used by the RETURN statement. This scheme allows nested CALL statements to be used.

When a RETURN statement is executed, the sequence number to be used by the RETURN statement contains the sequence number of the last CALL statement executed. In this case, the sequence number of the statement following the last CALL statement executed is stored as the sequence number of the next statement to be executed, and the sequence number stored within this CALL statement is stored as the new sequence number to be used by the RETURN statement.

4.9 Expansion of a User's Work Area

When an arithmetic statement or expression that occurs in any statement is executed, an initial scan through the expression counts the number, n, of assignment arrows. During execution of this expression, a maximum of 4n words may be added to the symbol table, and if this space is not available in the user's own area the system allocates the user an extra increment of work space from the "floating" space if this is possible. Similarly if a statement to be stored cannot fit into the user's work area, the system attempts to increase this work area.

When additional space is allocated to a user, that user's symbol table and the other user work areas that are stored above this user area must be copied into higher core locations as shown in Figure 3. The index to the start of the symbol table and the index for the next entry in the symbol table must be updated for this user while the amount of "floating" space available to the system and pointers to the other user work areas must also be updated.

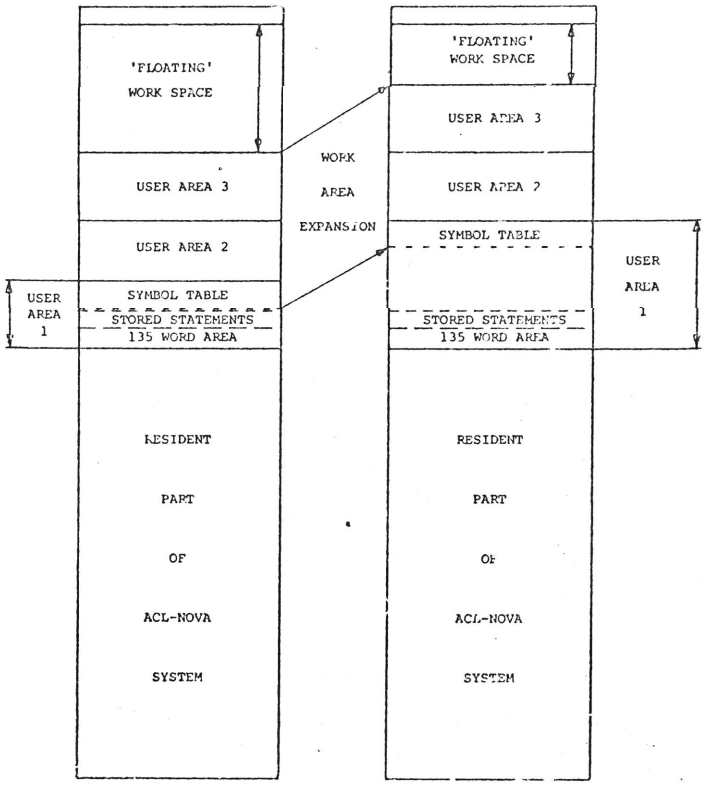


Figure 3: Expansion of user work areas

After the user has completed his calculations by executing an END statement, the extra space that was obtained is returned to the system with the other user work areas stored above this user being shuffled down into lower core locations.

This dynamic allocation of work space gives the system considerable flexibility. The structure of the user work area made this scheme possible, while care was taken to ensure that the pointers in the 135 word area were used as displacements from the start of the user area rather than as absolute addresses.

4.10 Expression Evaluation

If a variable is followed by an assignment arrow, then during statement execution the expression to the right of the assignment arrow is evaluated and its value given to that variable. The variable name and its value in floating point form are stored in the symbol table. When multiple assignments occur in an expression, assuming first of all that the expression is bracket free, then the rightmost assignment arrow is located. The expression to the right of this is evaluated and the resulting value given to the variable. This process is continued until all the assignments have been carried out.

In more complicated expressions that contain a number of levels of brackets plus multiple assignments,

the expression is evaluated by searching first for the rightmost opening bracket. The expression enclosed between the corresponding pair of opening and closing brackets (an expression at zero bracket level) is then evaluated by searching for the rightmost assignment arrow and proceeding as described above. This process is continued until the whole expression is reduced to zero bracket level and evaluated.

During the process of reducing an expression to zero bracket level, subscripted variables are replaced by subscripted variable replacement operands. A four character subscripted variable replacement operand of the form R'pii was chosen (to cope with the simplest form of a subscripted variable, namely A(1)) where the character R' indicates the start of a subscripted variable replacement operand, the character p is a pointer to the position of the next operator or delimiter in the expression, and the two characters ii form a 16-bit index to the appropriate symbol table entry relative to the start of the user work area.

Expressions at zero bracket level (and the bracket pair surrounding the expression, if any) are replaced by simple replacement operands. A three character simple replacement operand of the form Rpi was chosen (to cope with the simplest form of arithmetic expression, namely 3+2) where the character R indicates the start of a simple replacement operand, the character p is a pointer to the position of the next operator or delimiter in the expression, and the character i is an index to the floating point value of the expression. If an expression already contains simple replacement operands, then the simple replacement operand describing the value of the expression is given the same index as the last simple replacement operand in the expression. By using the same simple replacement operand a number of times the number of NOVA words required for saving simple replacement operand values was minimised. The maximum number of simple replacement operands occurs for the expression A*A+B*B+C*C+ . . . up to 71 characters and turns out to be 18. (Because of the mathematical hierarchy the multiplications must be done first.)

However, if statements are being traced by a user, then statement processing is temporarily suspended when a symbol table assignment occurs so that trace output may be printed at the terminal. During this time, statements can be processed for other terminals and consequently the replacement operand index and the values of simple replacement operands must be stored in each user area. The input buffer is not being used at this stage and it is exactly 36 words long (i.e. the space required to store the floating point values of 18 simple replacement operands). Thus the simple replacement operand index is an index to the floating point values relative to the start of the input buffer.

5. CONCLUSIONS

The one-terminal ACTIV-8 system gave individual scientists and engineers an easy, direct way of solving small numerical problems. Statements entered at a terminal were stored in the user work area strictly in the order in which they were typed and the resulting stored program could later be executed. Stored program execution could be interrupted at any time to examine the contents of the symbol table, or to evaluate arithmetic statements, or to examine the value of individual variables or to allow tracing to occur. Stored program execution could then be continued from the same point or some other point in the program.

The ACL-NOVA system allows for stored program execution or for the execution of immediate statements to perform one-time or "desk calculator" calculations, to control the execution of a stored program and to perform various editing and debugging functions. Stored program execution can again be interrupted at any time by pressing the ? character. However this time stored statements can be inserted, modified or deleted as well as any of the immediate statements being executed, before program execution is continued from the same point or some other point in the program.

The EDIT statement is the most important of the error correction facilities provided in the ACL language and it provides a novel way of allowing characters to be copied from, inserted into or deleted from an existing statement. This facility combined with other special features such as ways of suspending stored program execution and powerful tracing statements provide interaction with the users. The flexibility of the IF statement and the generality of the arithmetic statements and expressions, with the freedom of multiple assignments, also adds a great deal of power to the ACL language.

Consideration of the full duplex mode of operation of teletypewriter terminals played an extremely important part in the design of the ACTIV-8 and ACL-NOVA systems. The dynamic syntax checking of keyboard input provides interaction with the computer and protects the user from trivial typing errors. It also has the advantage that statements do not have to be checked for syntax at run time thus improving program execution times.

The implementation of the ACL language as a multi-user conversational interpreter has provided a powerful conversational computing facility for many users in a time-sharing environment. The number of terminals to be supported, the size of work area increments and the available user area are specified when the ACL-NOVA system is first loaded into the computer and work space may be reserved for each terminal or left "floating" to provide the greatest flexibility for the system. The dynamic allocation of work space in the ACL-NOVA system provides for a more efficient use of the available user area than can be obtained from a system having fixed user partititons.

6. ACKNOWLEDGEMENTS

The authors thank Dr. D. J. Richardson and Mr. R. P. Backstrom for valuable discussions throughout this work.

It is interesting to relate the work described in this paper to time, and the above projects were carried out over the following periods: ACTIV-8, January 1968 to July 1968 (NWB); ACL language development, January 1969 to March 1970 (NWB,PLS); ACL-NOVA implementation, May 1969 to May 1971 (PLS, NWB).

REFERENCES

BENNETT, N. W. (1968) ACTIV-8 Users Guide, Unpublished A.A.E.C. Report.

SANGER, P. L. (1971) ACL-NOVA: A Multi-User Conversational Interpreter for the NOVA Computer, AAEC/E221 (revised July 1972).

SANGER, P. L. (1976) NOVASM and NOVASIM — An Assembler and a Simulator for the NOVA and SUPERNOVA Computers Written to Run on an IBM 360 Computer, AAEC/TM566.

Appendix C

The UCC FASTEXT system

Appendix C

This document has been prepared with the use of the UCC FASTEXT system. The original source was entered via a typewriter connected to the UCC FASBAC system as an EDIT file. This enabled the candidate to type the source in free format, proof-read and correct errors.

The EDIT file was then processed by the FASTEXT system to produce a formatted version of the file. Commands may be built into the source file to specify spacing, page size and paragraphs.

The next page is the original input from which the title page and this page were prepared.

```
 ty
.doublespace
.begin
.centre
.space 10
Appendix C
.space 20
The UCC FASTEXT system
.nocentre
.heading 1,2,1
Appendix C
.begin
  This document has been prepared
with the use of the UCC FASTEXT system.
The original source was entered via a typewriter
connected to the UCC FASBAC
system as an EDIT file.
This enabled the candidate to type the source in free
format, proof-read and correct errors.
  The EDIT file was then processed by the FASTEXT
system to produce a formatted version of the file.
Commands may be built into the source file to
specify spacing, page size and paragraphs.
  The next page is the original input from which
the title page and this page were prepared.
>
```

Appendix D

An Assembler/Simulator for the PDP-11

The next eight pages of computer listings are a sample run of the assembler/simulator which was used to develop the XB system on a UNIVAC 1108.

Page 1 shows the assembly of a small program. Lines 17 through 32 are a fixed point divide by ten subroutine for the PDP-11. Lines 11 through 15 are used to set up data, enter the subroutine and display results. The RESET instruction causes the simulator to re-enter the assembler.

There are five columns to the left of the assembler source language on Page 1. The first column is a decimal line number for each source line. The second column is a six-digit octal address of each word of the assembled program and the third column is the octal contents which have been assembled into that word. The fourth and fifth columns are hexidecimal equivilants of the second and third columns.

A cross-reference table showing where symbols have been defined and used is at the bottom of the page. There are several columns to the right of the symbols. The first column is always a one; this indicates the program segment within which the symbol was defined and in the sample this feature is not being used. The second column gives the line where the symbol was defined and succeeding columns give the numbers of the lines where a reference was made to the symbol.

The END statement (line 33) nominates a starting

address for simulation.

The second and third pages show the simulation of the assembled program. Normally there are 9 fields accross each line. These are :-

1) The hexadecimal address of each instruction.

2) The mneumonic of the instruction.

3) The contents of the N, Z, V and C indicators before the instruction.

4) The hexadecimal address (or register name) of the source field.

5) The hexadecimal contents of the source field.

6) The hexadecimal address (or register name) of the destination field.

7) The hexadecimal contents of the destination field before the instruction.

8) The hexadecimal contents of the destination field after the instruction.

9) The contents of the N, Z, V and C indicators after the instruction.

Jumps and successful branches have an indicator printed on the extreme right and a line skipped. In the case of subroutine jumps the contents of the register 6 stack are displayed after the instruction. The first line is a set of hexadecimal addresses from four words below the stack pointer to 12 words above it. The actual value of register 6 is highlighted. The second line is the contents of the

stack. For example the return address of the JSR instruction at HEX 2718 is HEX 271C. The return from subroutine instruction causes the contents of the stack before the instruction to be printed.

The instructions at HEX 271C and 271E display the contents of registers zero and one; the remainder and result of the divide.

Page 4 shows an assembly of two different instructions into the existing assembled program.

Pages 5 and 6 show the simulation of the altered program with the results of 3 and 5 shown.

Page 7 is similar to page 5 but the printout of the divide subroutine has been suppressed (line 1). This results in a much reduced output from the simulation (page 8).

```
    1  023420                2710                    ORG 10000
    2                                                TRON
    3            000000                0000          R0 EQR 0
    4            000001                0001          R1 EQR 1
    5            000002                0002          R2 EQR 2
    6            000003                0003          R3 EQR 3
    7            000004                0004          R4 EQR 4
    8            000005                0005          R5 EQR 5
    9            000006                0006          R6 EQR 6
   10            000007                0007          PC EQR 7
   11  023420    012706     2710       15C6          START MOV =800,R6
       023422    001440     2712       0320
   12  023424    012700     2714       15C0           MOV =21,R0
       023426    000025     2716       0015
   13  023430    004737     2718       09DF           JSR PC,DIV10
       023432    023442     271A       2722
   14  023434    010000     271C       1000           MOV R0,R0
   15  023436    010101     271E       1041           MOV R1,R1
   16  023440    000005     2720       0005           RESET
   17  023442    005001     2722       0A01          DIV10 CLR R1  DIVIDE BY TEN; ANSWER IN R1; REMAINDER IN R0
   18  023444    012702     2724       15C2           MOV =TBL,R2
       023446    023476     2726       273E
   19  023450    000241     2728       00A1          L36 CLC
   20  023452    161200     272A       E280          L15A SUB (R2),R0
   21  023454    103403     272C       8703           BCS L24
   22  023456    066201     272E       6C81           ADD 2(R2),R1
       023460    000002     2730       0002
   23  023462    000773     2732       01FB           BR L15A
   24  023464    062200     2734       6480          L24 ADD (R2)+,R0
   25  023466    021227     2736       2297           CMP (R2),=1
       023470    000001     2738       0001
   26  023472    001366     273A       02F6           BNE L36
   27  023474    000207     273C       0087           RTS PC
   28  023476    023420     273E       2710          TBL DCH 10000
   29  023500    001750     2740       03E8           DCH 1000
   30  023502    000144     2742       0064           DCH 100
   31  023504    000012     2744       000A           DCH 10
   32  023506    000001     2746       0001           DCH 1
   33                                                 END START

DIV10          1   17   13
L15A           1   20   23
L24            1   24   21
L36            1   19   26
PC             1   10   13   27
R0             1    3   12   14   14   20   24
R1             1    4   15   15   17   22
R2             1    5   18   20   22   24   25
R3             1    6
R4             1    7
R5             1    8
R6             1    9   11
START          1   11   33
TBL            1   28   18

REAL TIME CLOCK INTERROGATED AT 11:42:31
```

```
    2710 MOV       0 0 0 0        2712      0320     REG  6      0000                0320          0 0 0 0
    2714 MOV       0 0 0 0        2716      0015     REG  0      0000                0015          0 0 0 0
    2718 JSR       0 0 0 0                                2722   0A01                              0 0 0 0      JSR
0316  0318  031A  031C *031E* 0320  0322  0324  0326  0328  032A  032C  032E  0330  0332  0334
0000  0000  0000  0000  271C  0000  0000  0000  0000  0000  0000  0000  0000  0000  0000

    2722 CLR       0 0 0 0                                REG  1      0000                0000          0 1 0 0
    2724 MOV       0 1 0 0        2726      273E     REG  2      0000                273E          0 0 0 0
0316  0318  031A  031C *031E* 0320  0322  0324  0326  0328  032A  032C  032E  0330  0332  0334
0000  0000  0000  0000  271C  0000  0000  0000  0000  0000  0000  0000  0000  0000  0000
    2728 CLC       0 0 0 0                                                                     0 0 0 0
    272A SUB       0 0 0 0        273E      2710     REG  0      0015                D905          1 0 0 1
    272C BCS       1 0 0 1                                          0003                      1 0 0 1      BR

    2734 ADD       1 0 0 1        273E      2710     REG  0      D905                0015          0 0 0 1
    2736 CMP       0 0 0 1        2740      03E8       2738      0001                              0 0 0 0
    273A BNE       0 0 0 0                                          00F6                      0 0 0 0      BR

0316  0318  031A  031C *031E* 0320  0322  0324  0326  0328  032A  032C  032E  0330  0332  0334
0000  0000  0000  0000  271C  0000  0000  0000  0000  0000  0000  0000  0000  0000  0000
    2728 CLC       0 0 0 0                                                                     0 0 0 0
    272A SUB       0 0 0 0        2740      03E8     REG  0      0015                FC2D          1 0 0 1
    272C BCS       1 0 0 1                                          0003                      1 0 0 1      BR

    2734 ADD       1 0 0 1        2740      03E8     REG  0      FC2D                0015          0 0 0 1
    2736 CMP       0 0 0 1        2742      0064       2738      0001                              0 0 0 0
    273A BNE       0 0 0 0                                          00F6                      0 0 0 0      BR

0316  0318  031A  031C *031E* 0320  0322  0324  0326  0328  032A  032C  032E  0330  0332  0334
0000  0000  0000  0000  271C  0000  0000  0000  0000  0000  0000  0000  0000  0000  0000
    2728 CLC       0 0 0 0                                                                     0 0 0 0
    272A SUB       0 0 0 0        2742      0064     REG  0      0015                FFB1          1 0 0 1
    272C BCS       1 0 0 1                                          0003                      1 0 0 1      BR

    2734 ADD       1 0 0 1        2742      0064     REG  0      FFB1                0015          0 0 0 1
    2736 CMP       0 0 0 1        2744      000A       2738      0001                              0 0 0 0
    273A BNE       0 0 0 0                                          00F6                      0 0 0 0      BR

0316  0318  031A  031C *031E* 0320  0322  0324  0326  0328  032A  032C  032E  0330  0332  0334
0000  0000  0000  0000  271C  0000  0000  0000  0000  0000  0000  0000  0000  0000  0000
    2728 CLC       0 0 0 0                                                                     0 0 0 0
    272A SUB       0 0 0 0        2744      000A     REG  0      0015                000B          0 0 0 0
    272C BCS       0 0 0 0                                          0003                      0 0 0 0
    272E ADD       0 0 0 0        2746      0001     REG  1      0000                0001          0 0 0 0
    2732 BR        0 0 0 0                                          00FB                      0 0 0 0      BR

    272A SUB       0 0 0 0        2744      000A     REG  0      000B                0001          0 0 0 0
    272C BCS       0 0 0 0                                          0003                      0 0 0 0
    272E ADD       0 0 0 0        2746      0001     REG  1      0001                0002          0 0 0 0
    2732 BR        0 0 0 0                                          00FB                      0 0 0 0      BR

    272A SUB       0 0 0 0        2744      000A     REG  0      0001                FFF7          1 0 0 1
    272C BCS       1 0 0 1                                          0003                      1 0 0 1      BR

    2734 ADD       1 0 0 1        2744      000A     REG  0      FFF7                0001          0 0 0 1
    2736 CMP       0 0 0 1        2746      0001       2738      0001                              0 1 0 0
    273A BNE       0 1 0 0                                          00F6                      0 1 0 0
0316  0318  031A  031C *031E* 0320  0322  0324  0326  0328  032A  032C  032E  0330  0332  0334
0000  0000  0000  0000  271C  0000  0000  0000  0000  0000  0000  0000  0000  0000  0000
```

```
    273C RTS        0 1 0 0                         0007                                    0 1 0 0.        RTS

    271C MOV        0 1 0 0       REG  0      0001  REG  0      0001              0001      0 0 0 0
    271E MOV        0 0 0 0       REG  1      0002  REG  1      0002              0002      0 0 0 0
0318  031A  031C  031E *0320* 0322  0324  0326  0328  032A  032C  032E  0330  0332  0334  0336
0000  0000  0000  271C  0000  0000  0000  0000  0000  0000  0000  0000  0000  0000  0000  0000
    2720 RESET      0 0 0 0                                                               0 0 0 0
```

```
      1  023420                2710                    ORG 10000
      2  023420  012706        2710    15C6            MOV =800,R6
         023422  001440        2712    0320
      3  023424  012700        2714    15C0            MOV =53,R0
         023426  000065        2716    0035
      4                                                 END START

R0              1           3
R6              1           2
START           1           4

REAL TIME CLOCK INTERROGATED AT 11:42:33
```

```
    2718 JSR       0 0 0 0                                  2722      0A01                                0 0 0 0          JSR
0316  0318  031A  031C *031E* 0320  0322  0324  0326  0328  032A  032C  032E  0330  0332  0334
0000  0000  0000  0000  271C  0000  0000  0000  0000  0000  0000  0000  0000  0000  0000  0000

    2722 CLR       0 0 0 0                                REG  1      0000                0000          0 1 0 0
    2724 MOV       0 1 0 0        2726      273E          REG  2      0000                273E          0 0 0 0
0316  0318  031A  031C *031E* 0320  0322  0324  0326  0328  032A  032C  032E  0330  0332  0334
0000  0000  0000  0000  271C  0000  0000  0000  0000  0000  0000  0000  0000  0000  0000  0000
    2728 CLC       0 0 0 0                                                                      0 0 0 0
    272A SUB       0 0 0 0        273E      2710          REG  0      0035                D925          1 0 0 1
    272C BCS       1 0 0 1                                                    0003                     1 0 0 1          BR

    2734 ADD       1 0 0 1        273E      2710          REG  0      D925                0035 5        0 0 0 1
    2736 CMP       0 0 0 1        2740      03E8            2738      0001                              0 0 0 0
    273A BNE       0 0 0 0                                                    00F6                     0 0 0 0          BR

0316  0318  031A  031C *031E* 0320  0322  0324  0326  0328  032A  032C  032E  0330  0332  0334
0000  0000  0000  0000  271C  0000  0000  0000  0000  0000  0000  0000  0000  0000  0000  0000
    2728 CLC       0 0 0 0                                                                      0 0 0 0
    272A SUB       0 0 0 0        2740      03E8          REG  0      0035                FC4D          1 0 0 1
    272C BCS       1 0 0 1                                                    0003                     1 0 0 1          BR

    2734 ADD       1 0 0 1        2740      03E8          REG  0      FC4D                0035 5        0 0 0 1
    2736 CMP       0 0 0 1        2742      0064            2738      0001                              0 0 0 0
    273A BNE       0 0 0 0                                                    00F6                     0 0 0 0          BR

0316  0318  031A  031C *031E* 0320  0322  0324  0326  0328  032A  032C  032E  0330  0332  0334
0000  0000  0000  0000  271C  0000  0000  0000  0000  0000  0000  0000  0000  0000  0000  0000
    2728 CLC       0 0 0 0                                                                      0 0 0 0
    272A SUB       0 0 0 0        2742      0064          REG  0      0035                FFD1          1 0 0 1
    272C BCS       1 0 0 1                                                    0003                     1 0 0 1          BR

    2734 ADD       1 0 0 1        2742      0064          REG  0      FFD1                0035 5        0 0 0 1
    2736 CMP       0 0 0 1        2744      000A            2738      0001                              0 0 0 0
    273A BNE       0 0 0 0                                                    00F6                     0 0 0 0          BR

0316  0318  031A  031C *031E* 0320  0322  0324  0326  0328  032A  032C  032E  0330  0332  0334
0000  0000  0000  0000  271C  0000  0000  0000  0000  0000  0000  0000  0000  0000  0000  0000
    2728 CLC       0 0 0 0                                                                      0 0 0 0
    272A SUB       0 0 0 0        2744      000A          REG  0      0035                002B +        0 0 0 0
    272C BCS       0 0 0 0                                                    0003                     0 0 0 0
    272E ADD       0 0 0 0        2746      0001          REG  1      0000                0001          0 0 0 0
    2732 BR        0 0 0 0                                                    00FB                     0 0 0 0          BR

    272A SUB       0 0 0 0        2744      000A          REG  0      002B                0021 !        0 0 0 0
    272C BCS       0 0 0 0                                                    0003                     0 0 0 0
    272E ADD       0 0 0 0        2746      0001          REG  1      0001                0002          0 0 0 0
    2732 BR        0 0 0 0                                                    00FB                     0 0 0 0          BR

    272A SUB       0 0 0 0        2744      000A          REG  0      0021                0017          0 0 0 0
    272C BCS       0 0 0 0                                                    0003                     0 0 0 0
    272E ADD       0 0 0 0        2746      0001          REG  1      0002                0003          0 0 0 0
    2732 BR        0 0 0 0                                                    00FB                     0 0 0 0          BR

    272A SUB       0 0 0 0        2744      000A          REG  0      0017                000D          0 0 0 0
    272C BCS       0 0 0 0                                                    0003                     0 0 0 0
    272E ADD       0 0 0 0        2746      0001          REG  1      0003                0004          0 0 0 0
    2732 BR        0 0 0 0                                                    00FB                     0 0 0 0          BR
```

```
    272A SUB       0 0 0 0       2744      000A    REG 0      000D              0003          0 0 0 0
    272C BCS       0 0 0 0                                            0003                  0 0 0 0
    272E ADD       0 0 0 0       2746      0001    REG 1      0004              0005          0 0 0 0
    2732 BR        0 0 0 0                                            00FB                  0 0 0 0      BR

    272A SUB       0 0 0 0       2744      000A    REG 0      0003              FFF9          1 0 0 1
    272C BCS       1 0 0 1                                            0003                  1 0 0 1      BR

    2734 ADD       1 0 0 1       2744      000A    REG 0      FFF9              0003          0 0 0 1
    2736 CMP       0 0 0 1       2746      0001      2738     0001                            0 1 0 0
    273A BNE       0 1 0 0                                            00F6                  0 1 0 0
0316  0318  031A  031C *031E* 0320  0322  0324  0326  0328  032A  032C  032E  0330  0332  0334
0000  0000  0000  0000  271C  0000  0000  0000  0000  0000  0000  0000  0000  0000  0000  0000
    273C RTS       0 1 0 0                                  0007                            0 1 0 0      RTS

    271C MOV       0 1 0 0     REG 0       0003    REG 0      0003              0003          0 0 0 0
    271E MOV       0 0 0 0     REG 1       0005    REG 1      0005              0005          0 0 0 0
0318  031A  031C  031E *0320* 0322  0324  0326  0328  032A  032C  032E  0330  0332  0334  0336
0000  0000  0000  271C  0000  0000  0000  0000  0000  0000  0000  0000  0000  0000  0000  0000
    2720 RESET     0 0 0 0                                                                 0 0 0 0
```

```
      1                                                    TRKILL DIV10,TBL
      2  023420                2710                        ORG 10000
      3  023420   012706       2710     15C6               MOV =800,R6
         023422   001440       2712     0320
      4  023424   012700       2714     15C0               MOV =803,R0
         023426   001443       2716     0323
      5                                                    END START

DIV10          1          1
R0             1          4
R6             1          3
START          1          5
TBL            1          1

REAL TIME CLOCK INTERROGATED AT 11:42:35
```

```
    2718 JSR        0 0 0 0                              2722       0A01                                  0 0 0 0              JSR
0316  0316  031A   031C *031E* 0320  0322  0324  0326  0328  032A  032C  032E  0330  0332  0334
0000  0000  0000   0000  271C  0000  0000  0000  0000  0000  0000  0000  0000  0000  0000  0000

    271C MOV        0 1 0 0       REG  0       0003     REG  0       0003                 0003                0 0 0 0
    271E MOV        0 0 0 0       REG  1       0050     REG  1       0050                 0050 P              0 0 0 0
0318  031A  031C   031E *0320* 0322  0324  0326  0328  032A  032C  032E  0330  0332  0334  0336
0000  0000  0000.  271C  0000  0000  0000  0000  0000  0000  0000  0000  0000  0000  0000  0000
    2720 RESET      0 0 0 0                                                                  0 0 0 0
```

Appendix E

Use of the XB paper tape system

Appendix E

There are two paper tapes provided with the thesis. They are marked "XB system" and "XB demonstration tape". The XB system is in standard PDP-11 binary loader format and should be loaded normally via the high speed paper tape reader.

The key combination CONTROL-G may be used at any time to initialize the system. A heading describing the version of XB will be typed each time the system is initialized.

The demonstation tape is a collection of about 50 graded examples designed to introduce the major concepts of the system. Each frame may be read by giving a CONTROL-D character. This sets a mode to read from the high speed paper tape reader. The user is free to use the system from the keyboard between frames.

Most of the frames are self-contained. However, an asynchronous interupt from the keyboard can only be made by the user. This is needed to interupt the automatic mode to regain control or to invoke an ASYNCH statement.

Appendix F

Sample flowchart - Arithmetic expressions

Appendix F

The next few pages illustrate the techniques used in accepting or rejecting individual characters to form an arithmetic expression. When a character is accepted, a new character is taken from the keyboard and compared with the characters or groups of characters allowable as the next character. For example, after an exponent E (see flowchart of Number) the next character must be plus, minus or a number. In other situations, such as after a number in the final flowchart, the arithmetic routine cannot judge the validity of characters by itself. If a character is not one of the five operators, an exit is taken and the validity of the character decided elsewhere. The comma in

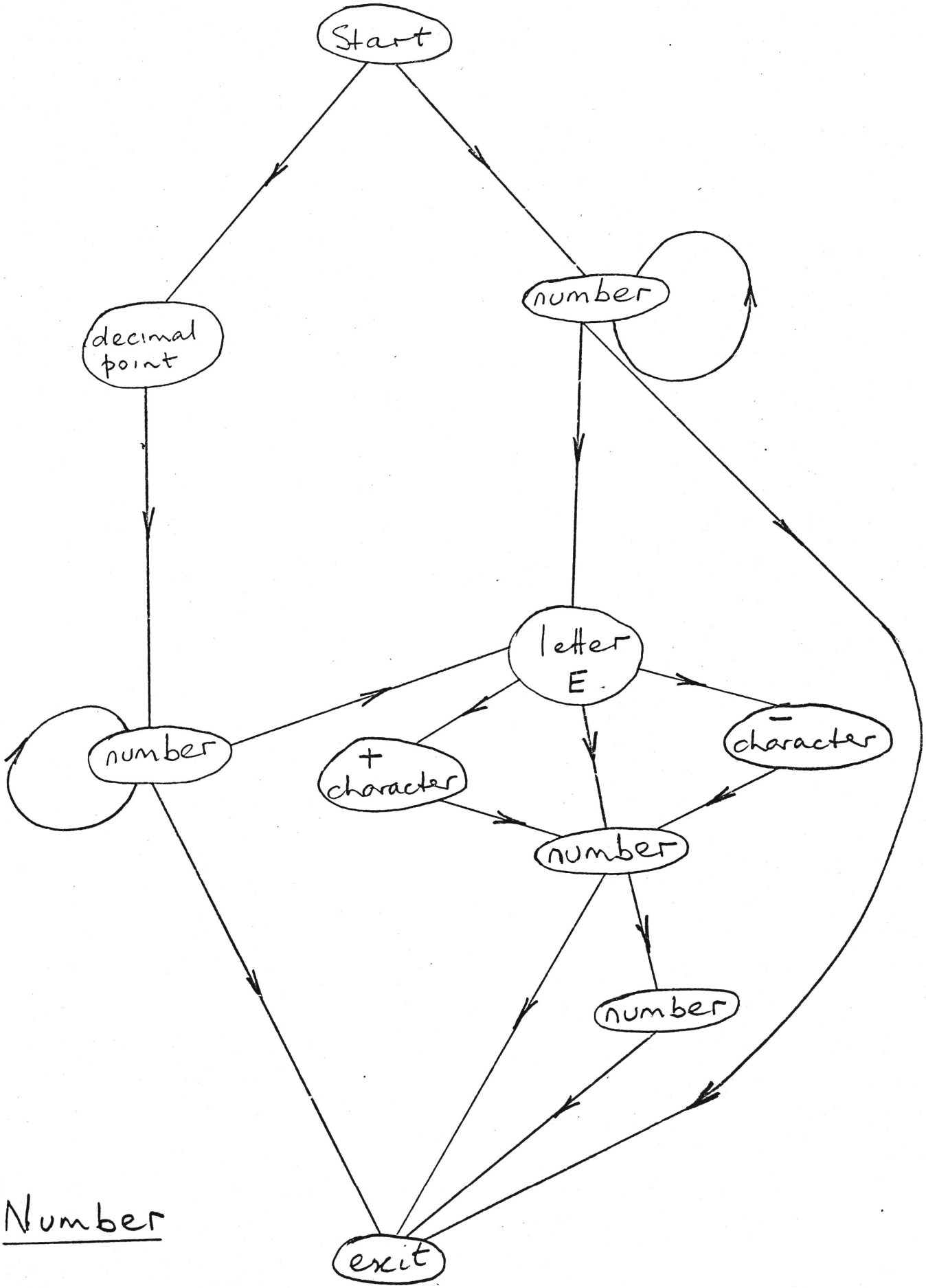
```
TYPE 1+2,
```

would be accepted, but the comma in

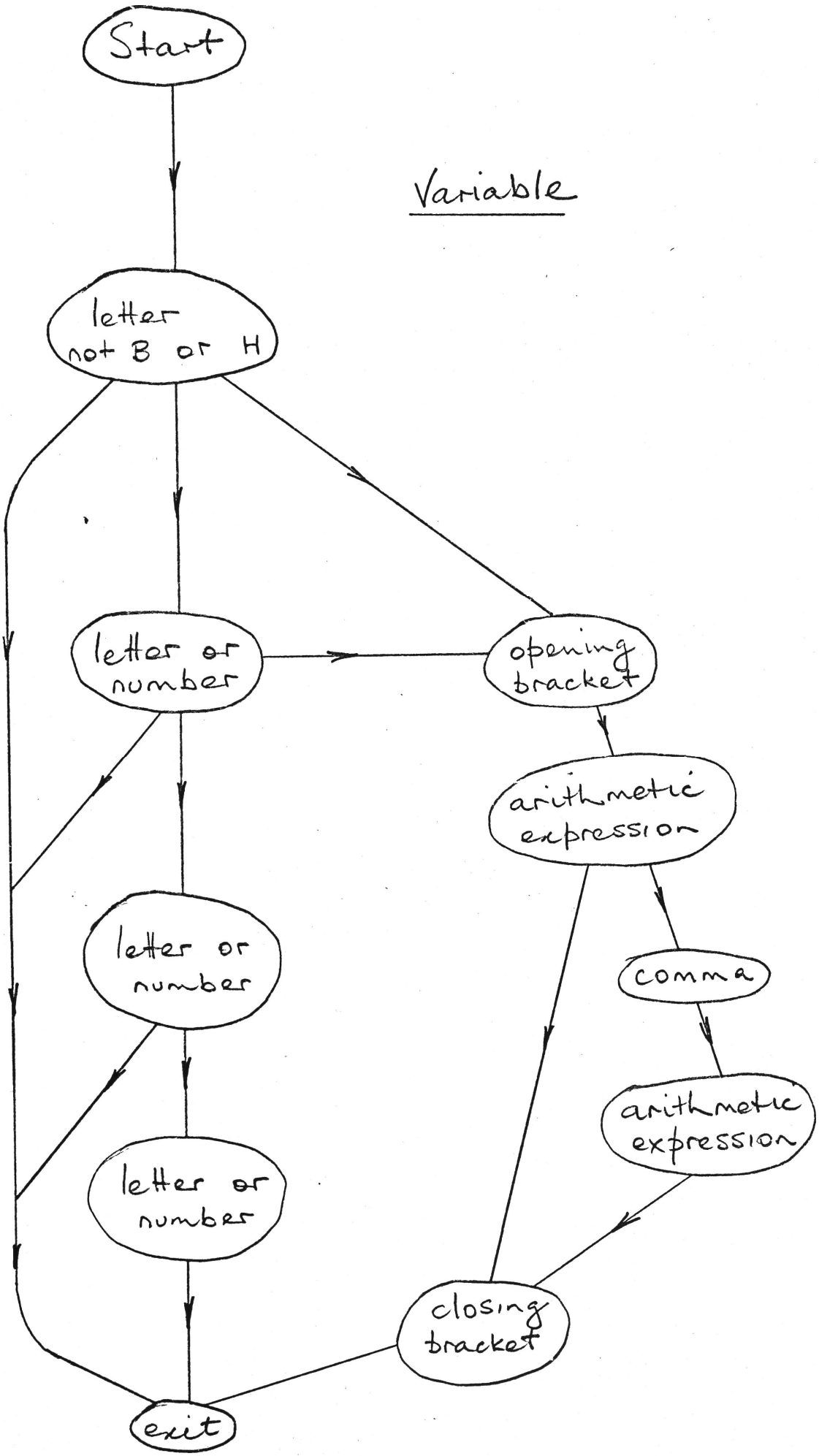
```
A=1+2,
```

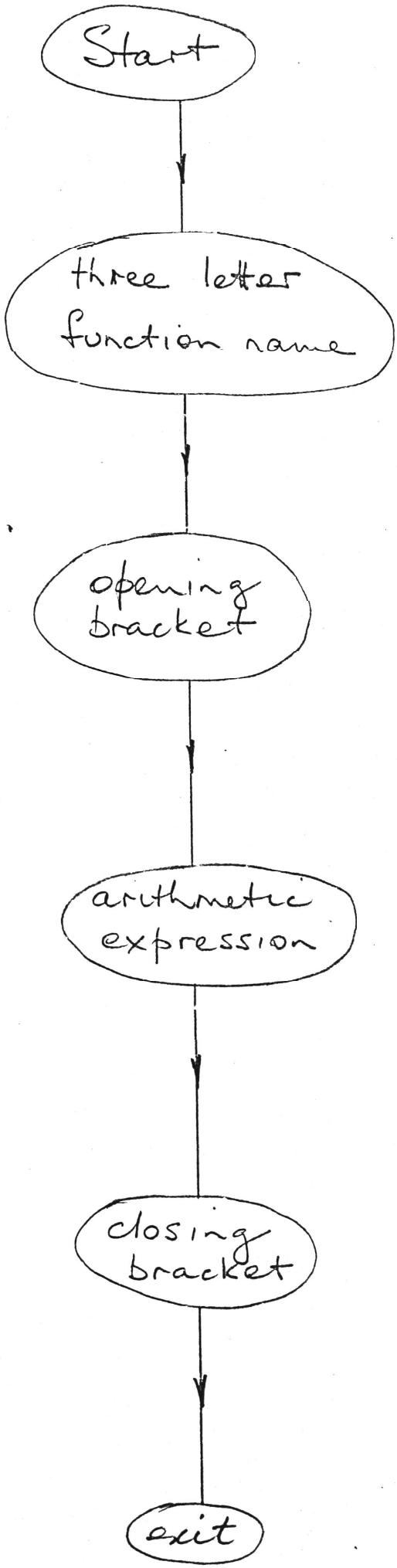
would not be.

As explained in the body of the thesis, the program to check expressions for validity is written in assembler for maximum speed. For arithmetic expressions, see 1116-1349, Appendix G.

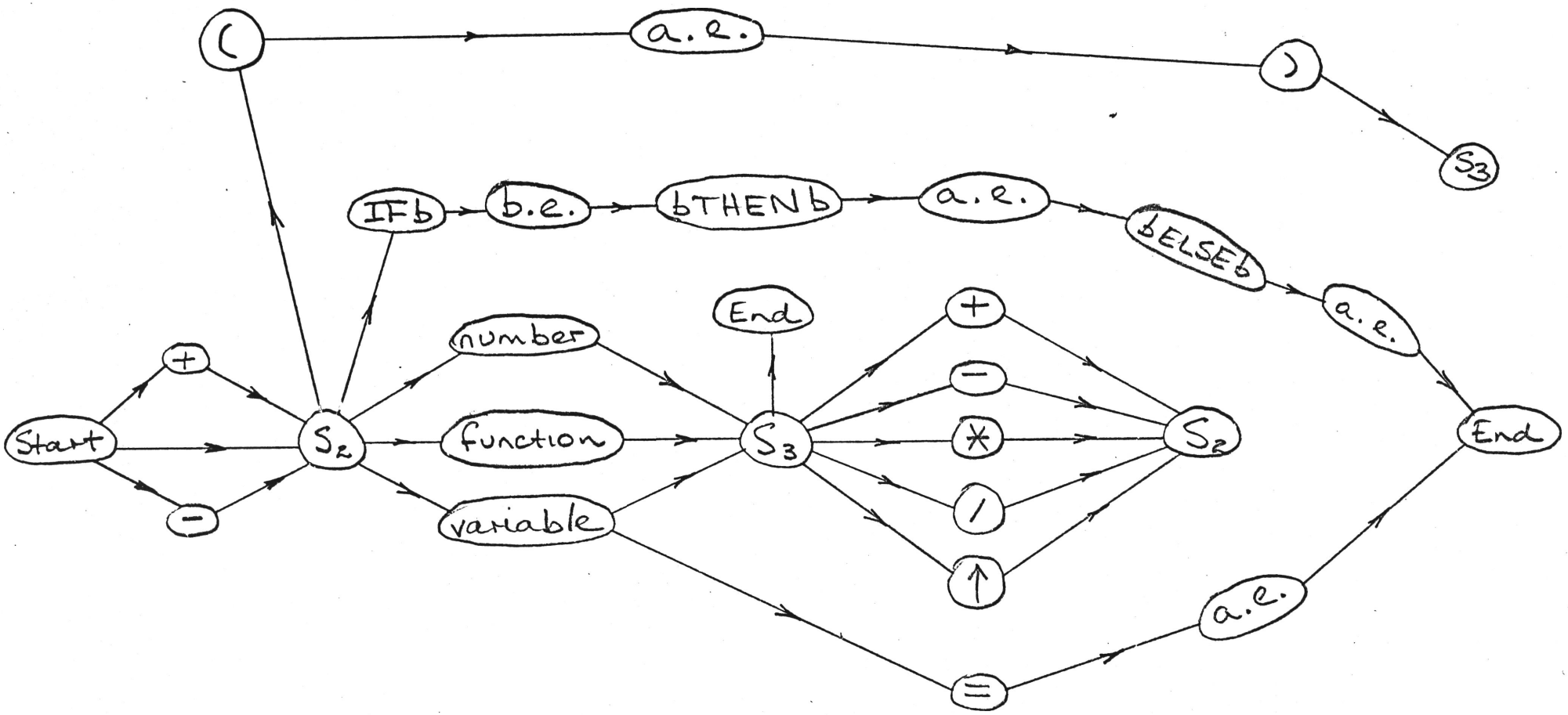

Start
decimal point
number
number
letter E
+ character
− character
number
number
number
exit

Number


Variable
Start
letter not B or H
letter or number
letter or number
letter or number
opening bracket
arithmetic expression
comma
arithmetic expression
closing bracket
exit


Start
three letter function name
opening bracket
arithmetic expression
closing bracket
exit

Function



Arithmetic Expression

Appendix G

Partial Source of the XB system

Appendix G

The next 61 pages are the source statements of the PDP-11 program written as the major part of this project. The source statements given here are not complete; the full program consists of over 8000 statements. That portion given here is the code which implements the algorithms described in the body of the thesis.

A row of asterisks marks the places where part of the program has been left out. Reference is made in the thesis to the program by the decimal line number on the left hand side. The addresses and contents given here are not necessarily the same as in the program supplied on paper tape.

```
 1                                                    TROFF
 2          000000              0000                  K EQR 0
 3          000000              0000                  R0 EQR 0
 4          000001              0001                  R1 EQR 1
 5          000002              0002                  R2 EQR 2
 6          000003              0003                  R3 EQR 3
 7          000004              0004                  R4 EQR 4
 8          000004              0004                  CHKR EQR 4
 9          000004              0004                  STLR EQR 4
10          000005              0005                  R5 EQR 5
11          000006              0006                  R6 EQR 6
12          000007              0007                  PC EQR 7
13          177550              FF68                  PRS EQU O'177550'
14          177552              FF6A                  PRB EQU O'177552'
15          177554              FF6C                  PPS EQU O'177554'
16          177556              FF6E                  PPB EQU O'177556'
17          177560              FF70                  TKS EQU O'177560'
18          177562              FF72                  TKB EQU O'177562'
19          177564              FF74                  TPS EQU O'177564'
20          177566              FF76                  TPB EQU O'177566'
21          177570              FF78                  SWR EQU O'177570'
22          177657              FFAF                  CHSTRI EQU X'FFAF'      INDICATE STRING
23          177724              FFD4                  CHD4EX EQU X'FFD4'       INDICATE OPERATOR
24          175372              FAFA                  FALSE EQU X'FAFA'      REPRESENTATION OF FALSE
25          157336              DEDE                  TRUE EQU X'DEDE'        REPRESENTATION OF TRUE
26          177740              FFE0                  CHSPZR EQU X'FFE0'         SPECIAL ZERO
27          177700              FFC0                  CHC0EX EQU X'FFC0'       INDICATE OPENING BRACKET
28          177772              FFFA                  CHFAEX EQU X'FFFA'      START OF ARITH FIELD
29          177773              FFFB                  CHFBEX EQU X'FFFB'      END OF ARITH FIELD
30          177701              FFC1                  CHC1EX EQU X'FFC1'     INDICATE ARRAY NAME
31          177603              FF83                  CH83EX EQU X'FF83'      INDICATOR AFTER 'IF '
32          177626              FF96                  CH96EX EQU X'FF96'     END OF LOGICAL EXPR AFTER 'IF '
33          177646              FFA6                  CHA6EX EQU X'FFA6'       INDICATOR AT END OF 'TRUE' PART
34          177660              FFB0                  CHB0EX EQU X'FFB0'       INDICATOR AT END OF 'FALSE' PART
35          177703              FFC3                  CHC3EX EQU X'FFC3'      INDICATOR FOR A FUNCTION
36          000367              00F7                  CHF7EX EQU X'00F7'         START OF A STORED LINE
37          177713              FFCB                  CHCBEX EQU X'FFCB'
38          177652              FFAA                  CHAAEX EQU X'FFAA'
39          177732              FFDA                  CHDAEX EQU X'FFDA'
40          177623              FF93                  CH93EX EQU X'FF93'
41          177624              FF94                  CH94EX EQU X'FF94'
42          177625              FF95                  CH95EX EQU X'FF95'
43          177637              FF9F                  CH9FEX EQU X'FF9F'
44          177642              FFA2                  CHA2EX EQU X'FFA2'
45          177730              FFD8                  CHD8EX EQU X'FFD8'
46          177731              FFD9                  CHD9EX EQU X'FFD9'
47          177711              FFC9                  CHC9EX EQU X'FFC9'
48          104210              8888                  STRIND EQU X'8888'   INDICATE STRING IN A+2
49                                                    $4S   INCLUDE CONFIDENCE CODE
50 000000                 0000                         ORG 0
51 000000  002002         0000  0402                   DCH TRAPZZ
52 000002  000000         0002  0000                   DCH 0
53 000004  002002         0004  0402                   DCH TRAPZZ
54 000006  000001         0006  0001                   DCH 1
55 000010  002002         0008  0402                   DCH TRAPZZ
56 000012  000002         000A  0002                   DCH 2
57 000014  015136         000C  1A5E                   DCH STL
58 000016  000000         000E  0000                   DCH 0
59 000020  031310         0010  32C8                   DCH LADDER
60 000022  000000         0012  0000                   DCH 0
```

```
 61  000030              0018                    ORG 0'030'
 62  000030  002004      0018    0404            DCH EMTVEC
 63  000032  000000      001A    0000            DCH 0
 64  000034  002042      001C    0422            DCH TRPVEC
 65  000036  000000      001E    0000            DCH 0
 66  000200              0080                    ORG 128
 67  000200              0080                FFEERR DSB 0          LOOK AT PC:  SHOULD BE LESS THAN OCTAL 400
 68                                              *  AREA FROM 256 TO 1024 FOR R6 STACK
 69                                              * USE THIS FOR PATCH AREA
 70  002000              0400                    ORG 1024
 71  002000              0400                LOCORE DSH 0                    FOR CHECKSUM
 72  002000  002002      0400    0402            DCH TRAPZZ          FOR TOO MANY RTS
 73  002002  104144      0402    8864        TRAPZZ EMT VC0071-EMTTBL  HALT ROUTINE
 74                                              ******************** CHECK ON STACK
 75  002004  010137      0404    105F        EMTVEC MOV R1,SAVR1
     002006  053376      0406    56FE
 76  002010  011601      0408    1381            MOV (R6),R1     RETURN ADDRESS
 77  002012  016616      040A    1D8E            MOV 2(R6),(R6)
     002014  000002      040C    0002
 78  002016  010166      040E    1076            MOV R1,2(R6)
     002020  000002      0410    0002
 79  002022  014101      0412    1841            MOV -(R1),R1  GET INSTRUCTION
 80  002024  042701      0414    45C1            BIC =X'FF00',R1
     002026  177400      0416    FF00
 81  002030  016146      0418    1C66            MOV EMTTBL(R1),-(R6)
     002032  006460      041A    0D30
 82                                              * MOV EMTTBL-X'8800'+1(R1),-(R6)    ******** ALLOW FOR 1'S ARITH
 83  002034  013701      041C    17C1            MOV SAVR1,R1
     002036  053376      041E    56FE
 84  002040  000002      0420    0002            RTI
 85  002042  010137      0422    105F        TRPVEC MOV R1,SAVR1
     002044  053376      0424    56FE
 86  002046  011601      0426    1381            MOV (R6),R1    RETURN ADDRESS
 87  002050  014101      0428    1841            MOV -(R1),R1   GET INSTRUCTION
 88  002052  042701      042A    45C1            BIC =X'FF00',R1
     002054  177400      042C    FF00
 89  002056  016116      042E    1C4E            MOV TRPTBL(R1),(R6)
     002060  006742      0430    0DE2
 90                                              * MOV TRPTBL-X'8900'+1(R1),(R6)  ****** ALLOW FOR 1'S ARITH
 91  002062  013701      0432    17C1            MOV SAVR1,R1
     002064  053376      0434    56FE
 92  002066  000002      0436    0002            RTI
*****************************************************
868  006460              0D30                EMTTBL DSH 0
869  006460  003252      0D30    06AA        VC0001 DCH SW
870  006462  002624      0D32    0594        VC0002 DCH PRNM
871  006464  003110      0D34    0648        VC0003 DCH ADD8
872  006466  003030      0D36    0618        VC0004 DCH TWODAD
873  006470  004326      0D38    08D6        VC0005 DCH PRERR
874  006472  004434      0D3A    091C        VC0006 DCH VALUE
875  006474  030666      0D3C    31B6        VC0007 DCH TYPECH
876  006476  036144      0D3E    3C64        VC0008 DCH SETBLE
877  006500  047612      0D40    4F8A        VC0009 DCH FIXENT
878  006502  004356      0D42    08EE        VC0010 DCH MOVCHR
879  006504  003202      0D44    0682        VC0011 DCH ADD4
880  006506  003310      0D46    06C8        VC0012 DCH MUL11
881  006510  004502      0D48    0942        VC0013 DCH GETN
882  006512  015062      0D4A    1A32        VC0014 DCH NUMB
883  006514  004444      0D4C    0924        VC0015 DCH SHFT
884  006516  031144      0D4E    3264        VC0016 DCH NEWLIN
```

```
865  006520  027016  0D50  2E0E      VC0017 DCH TBLABT
866  006522  047762  0D52  4FF2      VC0018 DCH RETREV
867  006524  036124  0D54  3C54      VC0019 DCH URGE
868  006526  004776  0D56  09FE      VC0020 DCH CONV
869  006530  005374  0D58  0AFC      VC0021 DCH MCONV
890  006532  003356  0D5A  06EE      VC0022 DCH MUL22
891  006534  027134  0D5C  2E5C      VC0024 DCH STK
892  006536  010616  0D5E  118E      VC0025 DCH AREX
893  006540  015056  0D60  1A2E      VC0027 DCH NL
894  006542  004552  0D62  096A      VC0028 DCH MOVEIT
895  006544  027174  0D64  2E7C      VC0030 DCH GNC
896  006546  012470  0D66  1538      VC0031 DCH LOGEX
897  006550  014202  0D68  1882      VC0032 DCH GENEX
898  006552  015106  0D6A  1A46      VC0033 DCH GETK
899  006554  015040  0D6C  1A20      VC0034 DCH LETL
900  006556  002076  0D6E  043E      VC0035 DCH FA
901  006560  005504  0D70  0B44      VC0037 DCH FD
902  006562  005472  0D72  0B3A      VC0040 DCH FM
903  006564  017252  0D74  1EAA      VC0041 DCH SETUPP
904  006566  017270  0D76  1EB8      VC0042 DCH PRESCN
905  006570  017726  0D78  1FD6      VC0043 DCH GHALFS
906  006572  020140  0D7A  2060      VC0045 DCH FILLIN
907  006574  002070  0D7C  0438      VC0046 DCH FS
908  006576  021344  0D7E  22E4      VC0050 DCH LHOP
909  006600  042640  0D80  45A0      VC0055 DCH INTVAR
910  006602  017650  0D82  1FA8      VC0056 DCH PRTVAL
911  006604  046614  0D84  4D8C      VC0057 DCH TBLU
912  006606  025660  0D86  2BB0      VC0060 DCH TBSPPA
913  006610  017322  0D88  1ED2      VC0061 DCH PROUTZ
914  006612  023506  0D8A  2746      VC0062 DCH VGY
915  006614  023052  0D8C  262A      VC0064 DCH FXDS
916  006616  015306  0D8E  1AC6      VC0065 DCH RND
917  006620  024344  0D90  28E4      VC0067 DCH VGZ84
918  006622  025210  0D92  2A88      VC0070 DCH GTEMP
919  006624  035402  0D94  3B02      VC0071 DCH TRAKK
920  006626  025672  0D96  2BBA      VC0072 DCH TBPUT
921  006630  011700  0D98  13C0      VC0073 DCH STREX
922  006632  027556  0D9A  2F6E      VC0074 DCH KEYBCH
923  006634  023366  0D9C  26F6      VC0075 DCH INTINT
924  006636  033400  0D9E  3700      VC0076 DCH REST88
925  006640  033420  0DA0  3710      VC0077 DCH COSUB
926  006642  017372  0DA2  1EFA      VC0088 DCH STOCHR
927  006644  031502  0DA4  3342      VC0101 DCH QSTRNG
928  006646  031554  0DA6  336C      VC0102 DCH VARR
929  006650  033326  0DA8  36D6      VC0103 DCH ELPOSS
930  006652  037040  0DAA  3E20      VC0105 DCH LOOKLN
931  006654  051026  0DAC  5216      VC0106 DCH CONVBD
932  006656  051770  0DAE  53F8      VC0107 DCH FNDZA
933  006660  052072  0DB0  543A      VC0110 DCH LNELIM
934  006662  042032  0DB2  441A      VC0111 DCH INTERP
935  006664  026562  0DB4  2D72      VC0112 DCH TBLCON
936  006666  037510  0DB6  3F48      VC0113 DCH GETST
937  006670  040706  0DB8  41C6      VC0115 DCH EDDY
938  006672  041634  0DBA  439C      VC0116 DCH EDKEY
939  006674  041524  0DBC  4354      VC0117 DCH EDCHK
940  006676  045636  0DBE  4B9E      VC0120 DCH GETARG
941  006700  050604  0DC0  5184      VC0121 DCH GETADR
942  006702  020634  0DC2  219C      VC0122 DCH VGX
943  006704  023162  0DC4  2672      VC0123 DCH INTCON
944  006706  042610  0DC6  4588      VC0124 DCH POSITN
```

```
  945  006710  051072    0DC8    523A        VC0130 DCH SH3500
  946  006712  045660    0DCA    4BB0        VC0131 DCH GET2ND
  947  006714  025352    0DCC    2AEA        VC0134 DCH TBLUPM
  948  006716  046600    0DCE    4D80        VC0135 DCH SRETLK
  949  006720  046214    0DD0    4C8C        VC0137 DCH DBLARG
  950  006722  050424    0DD2    5114        VC0140 DCH DBL2
  951  006724  050252    0DD4    50AA        VC0141 DCH LISTD
  952  006726  052270    0DD6    54B8        VC0142 DCH FND2EN
  953  006730  052006    0DD8    5406        VC0144 DCH FNDZB
  954  006732  052164    0DDA    5474        VC0145 DCH FNDSTN
  955  006734            0DDC                 DSH 3      SPARE
  956  006742            0DE2                TRPTBL DSH 0
  957  006742  004370    0DE2    08F8        VB0002 DCH ST1ERR
  958  006744  002612    0DE4    058A        VB0004 DCH TRSBZ
  959  006746  035640    0DE6    3BA0        VB0005 DCH STRTF
  960  006750  012236    0DE8    149E        VB0011 DCH STSC
  961  006752  013514    0DEA    174C        VB0014 DCH P5
  962  006754  012476    0DEC    153E        VB0016 DCH R21
  963  006756  023522    0DEE    2752        VB0023 DCH VGY0
  964  006760  012242    0DF0    14A2        VB0027 DCH ALT2
  965  006762  010616    0DF2    118E        VB0031 DCH AREX
  966  006764  051174    0DF4    527C        VB0041 DCH MJSCHD
  967  006766  027544    0DF6    2F64        VB0043 DCH RUBB
  968  006770  040716    0DF8    41CE        VB0045 DCH LPEB1
  969  006772  050242    0DFA    50A2        VB0051 DCH INTERB
  970  006774            0DFC                 DSH 3      SPARE
*****************************************
 1116  010616            118E               AREX DSH 0
 1117  010616  162706    118E    E5C6        SUB =2,R6
       010620  000002    1190    0002
 1118  010622  005016    1192    0A0E        CLR (R6)     LAST OPERATOR NOT UP ARROW
 1119  010624  000003    1194    0003        DEBUG  STL SUBR
 1120  010626  177772    1196    FFFA        DCH CHFAEX
 1121  010630  022700    1198    25C0       S1 CMP =E'+',K  IS IT A UNARY PLUS
       010632  000053    119A    002B
 1122  010634  001006    119C    0206        BNE M1
 1123  010636  000003    119E    0003       M2 DEBUG  STL SUBR
 1124  010640  177740    11A0    FFE0        DCH CHSPZR  STORE A SPECIAL ZERO
 1125  010642  000003    11A2    0003        DEBUG  STL SUBR
 1126  010644  177724    11A4    FFD4        DCH CHD4EX  STORE INDICATOR BEFORE OPERATOR
 1127  010646  104054    11A6    882C        EMT VC0024-EMTTBL
 1128  010650  000403    11A8    0103        BR S2
 1129  010652  022700    11AA    25C0       M1 CMP =E'-',K  IS IT A UNARY MINUS
       010654  000055    11AC    002D
 1130  010656  001767    11AE    03F7        BEQ M2  TREAT AS FOR UNARY PLUS
 1131  010660  022700    11B0    25C0       S2 CMP =E'(',K  IS IT AN OPENING BRACKET
       010662  000050    11B2    0028
 1132  010664  001011    11B4    0209        BNE M3
 1133  010666  000003    11B6    0003        DEBUG  STL SUBR    STORE INDICATOR TO THE OPENING BRACKET
 1134  010670  177700    11B8    FFC0        DCH CHC0EX
 1135  010672  104054    11BA    882C        EMT VC0024-EMTTBL
 1136  010674  104056    11BC    882E        EMT VC0025-EMTTBL
 1137  010676  022700    11BE    25C0        CMP =E')',K  MUST END IN A CLOSING BRACKET
       010700  000051    11C0    0029
 1138  010702  001110    11C2    0248        BNE LBJ  IF NOT IS AN ERROR
 1139  010704  104054    11C4    882C        EMT VC0024-EMTTBL
 1140  010706  000436    11C6    011E        BR L315
 1141  010710  022700    11C8    25C0       M3 CMP =E'.',K IS IT A DECIMAL POINT
       010712  000056    11CA    002E
 1142  010714  001002    11CC    0202        BNE M4
```

```
1143  010716  104054    11CE    882C          EMT VC0024-EMTTBL
1144  010720  000417    11D0    010F          BR L34AZX
1145  010722           11D2                  M4 DSH 0        IS IT A NUMBER
1146  010722  104032    11D2    881A          EMT VC0014-EMTTBL
1147  010724  001002    11D4    0202          BNE M5  NO
1148  010726           11D6                  B4 DSH 0
1149  010726  104054    11D6    882C          EMT VC0024-EMTTBL
1150  010730  000502    11D8    0142          BR L43
1151  010732           11DA                  M5 DSH 0        IS IT A LETTER OTHER THAN 'B' OR 'H'
1152  010732  004737    11DA    09DF          JSR PC,LETT
      010734  015024    11DC    1A14
1153  010736  001072    11DE    023A          BNE LBJ  IF NOT IS AN ERROR
1154  010740  005037    11E0    0A1F          CLR VARNM
      010742  053312    11E2    56CA
1155  010744  005037    11E4    0A1F          CLR VARNM+2
      010746  053314    11E6    56CC
1156  010750  110037    11E8    901F          MOVB R0,VARNM     SAVE 1ST CHARACTER OF VARIABLE
      010752  053312    11EA    56CA
1157  010754  104054    11EC    882C          EMT VC0024-EMTTBL
1158  010756  000476    11EE    013E          BR L53AZX
1159  010760           11F0                  L34AZX DSH 0        IS IT A NUMBER
1160  010760  104032    11F0    881A          EMT VC0014-EMTTBL
1161  010762  001060    11F2    0230          BNE LBJ
1162  010764           11F4                  M7 DSH 0
1163  010764  104054    11F4    882C          EMT VC0024-EMTTBL
1164  010766           11F6                  L312 DSH 0        IS IT A NUMBER
1165  010766  104032    11F6    881A          EMT VC0014-EMTTBL
1166  010770  001775    11F8    03FD          BEQ M7
1167  010772  022700    11FA    25C0          L313 CMP =E'E',K  IS IT THE LETTER E
      010774  000105    11FC    0045
1168  010776  001002    11FE    0202          BNE L315
1169  011000  104054    1200    882C          EMT VC0024-EMTTBL
1170  011002  000432    1202    011A          BR L322
1171  011004  022700    1204    25C0          L315 CMP =E'+',K  IS IT A PLUS
      011006  000053    1206    002B
1172  011010  001005    1208    0205          BNE B13
1173  011012  005016    120A    0A0E          M85 CLR (R6)   THIS OPERATOR NOT AN UP ARROW
1174  011014  000003    120C    0003          M85B DEBUG  STL SUBR
1175  011016  177724    120E    FFD4          DCH CHD4EX  STORE INDICATOR BEFORE OPERATOR
1176  011020  104054    1210    882C          EMT VC0024-EMTTBL
1177  011022  000716    1212    01CE          BR S2
1178  011024  022700    1214    25C0          B13 CMP =E'-',K
      011026  000055    1216    002D
1179  011030  001770    1218    03F8          BEQ M85
1180  011032  022700    121A    25C0          CMP =E'*',K  IS IT AN ASTERISK
      011034  000052    121C    002A
1181  011036  001765    121E    03F5          BEQ M85
1182  011040  022700    1220    25C0          CMP =E'/',K  IS IT DIVIDE
      011042  000057    1222    002F
1183  011044  001762    1224    03F2          BEQ M85
1184  011046  023700    1226    27C0          CMP EXPCH,K  IS IT UP ARROW
      011050  052444    1228    5524
1185  011052  001402    122A    0302          BEQ M85A
1186  011054  000137    122C    005F          JMP LEND
      011056  011424    122E    1314
1187  011060  005716    1230    0BCE          M85A TST (R6)      WAS LAST OPERATOR AN UP ARROW
1188  011062  001020    1232    0210          BNE LBJ            YES; IS AN ERROR
1189  011064  005216    1234    0A8E          INC (R6)
1190  011066  000752    1236    01EA          BR M85B
1191  011070  022700    1238    25C0          L322 CMP =E'+',K  IS IT E+
```

```
        011072  000053  123A  002B
1192    011074  001002  123C  0202          BNE B3A
1193    011076          123E                B253 DSH 0
1194    011076  104054  123E  882C          EMT VC0024-EMTTBL
1195    011100  000407  1240  0107          BR L33Y
1196    011102  022700  1242  25C0          B3A CMP =E'-',K  IS IT E-
        011104  000055  1244  002D
1197    011106  001773  1246  03FB          BEQ B253
1198    011110  104032  1248  881A          EMT VC0014-EMTTBL
1199    011112  001004  124A  0204          BNE LBJ
1200    011114          124C                B35 DSH 0
1201    011114  104054  124C  882C          EMT VC0024-EMTTBL
1202    011116  000403  124E  0103          BR L34
1203    011120          1250                L33Y DSH 0          IS IT E^9
1204    011120  104032  1250  881A          EMT VC0014-EMTTBL
1205    011122  001774  1252  03FC          BEQ B35
1206    011124          1254                LBJ DSH 0
1207    011124  104406  1254  8906          TRAP VB0011-TRPTBL
1208    011126          1256                L34 DSH 0          IS IT E99 OR E+99
1209    011126  104032  1256  881A          EMT VC0014-EMTTBL
1210    011130  001325  1258  02D5          BNE L315  MUST BE AN OPERATOR OR FINISH
1211    011132  104054  125A  882C          EMT VC0024-EMTTBL
1212    011134  000723  125C  01D3          BR L315
1213    011136          125E                L43 DSH 0          IS IT A NUMBER
1214    011136  104032  125E  881A          EMT VC0014-EMTTBL
1215    011140  001672  1260  03BA          BEQ B4
1216    011142  022700  1262  25C0          CMP =E'.',K  IS IT A DECIMAL POINT
        011144  000056  1264  002E
1217    011146  001311  1266  02C9          BNE L313
1218    011150  104054  1268  882C          EMT VC0024-EMTTBL
1219    011152  000705  126A  01C5          BR L312
1220    011154          126C                L53AZX DSH 0       IS IT A NUMBER OR A LETTER
1221    011154  104060  126C  8830          EMT VC0027-EMTTBL
1222    011156  001004  126E  0204          BNE L52Z
1223    011160  110037  1270  901F          MOVB R0,VARNM+1    SAVE 2ND CHAR OF VARIABLE
        011162  053313  1272  56CB
1224    011164  104054  1274  882C          EMT VC0024-EMTTBL
1225    011166  000504  1276  0144          BR L51
1226    011170  022700  1278  25C0          L52Z CMP =E'(',K  IS IT AN ARRAY NAME (SINGLE CHARACTER)
        011172  000050  127A  0028
1227    011174  001027  127C  0217          BNE L53
1228    011176          127E                B6A DSH 0
1229    011176  005237  127E  0A9F          INC VARNM+2        INDICATE ONE SUBSCRIPT
        011200  053314  1280  56CC
1230    011202  000003  1282  0003          DEBUG  STL SUBR
1231    011204  177701  1284  FFC1          DCH CHC1EX  STORE INDICATOR (ARRAY NAME)
1232    011206  104054  1286  882C          EMT VC0024-EMTTBL
1233    011210  104056  1288  882E          EMT VC0025-EMTTBL
1234    011212  022700  128A  25C0          CMP =E')',K
        011214  000051  128C  0029
1235    011216  001415  128E  030D          BEQ B73AAA  ONE SUBSCRIPT
1236    011220  022700  1290  25C0          CMP =E',',K  COMMA, SECOND SUBSCRIPT
        011222  000054  1292  002C
1237    011224  001337  1294  02DF          BNE LBJ
1238    011226  005237  1296  0A9F          INC VARNM+2        INDICATE SECOND SUBSCRIPT
        011230  053314  1298  56CC
1239    011232  104054  129A  882C          EMT VC0024-EMTTBL
1240    011234  104056  129C  882E          EMT VC0025-EMTTBL
1241    011236  022700  129E  25C0          CMP =E')',K  MUST FINISH WITH A CLOSING BRACKET
        011240  000051  12A0  0029
```

```
1242  011242  001330    12A2    02D8          BNE LBJ
1243  011244  000402    12A4    0102          BR B73AAA
1244  011246  110037    12A6    901F        B73 MOVB R0,VARNM+3     SAVE 4TH CHAR OF VARIABLE
      011250  053315    12A8    56CD
1245  011252            12AA                B73AAA DSH 0              ACCEPT 4TH CHAR OF VARIABLE
1246  011252  104054    12AA    882C          EMT VC0024-EMTTBL
1247  011254            12AC                L53 DSH 0
1248                                          *  PASSWORD WILL BE IN INPUT IN CASE OF RECOVERY
1249  011254  022737    12AC    25DF          CMP =E'ZR',VARNM        'ZR'
      011256  051132    12AE    525A
      011260  053312    12B0    56CA
1250  011262  001035    12B2    021D          BNE L53A
1251  011264  022737    12B4    25DF          CMP =E'ET',VARNM+2       'ET'
      011266  052105    12B6    5445
      011270  053314    12B8    56CC
1252  011272  001404    12BA    0304          BEQ L53B
1253  011274  022737    12BC    25DF          CMP =X'0001',VARNM+2       ONE SUBSCRIPT
      011276  000001    12BE    0001
      011300  053314    12C0    56CC
1254  011302  001025    12C2    0215          BNE L53A
1255  011304            12C4                L53B DSH 0
1256                                          *    IMMEDIATE MODE ONLY
1257  011304  005737    12C4    0BDF          TST MAJMOD
      011306  053340    12C6    56E0
1258  011310  001235    12C8    029D          BNE L315
1259  011312  022700    12CA    25C0          CMP =O'023',K     CONTROL S
      011314  000023    12CC    0013
1260  011316  001232    12CE    029A        L315AA BNE L315
1261  011320  005237    12D0    0A9F          INC INPB1             MOVE PRINTED POINTER
      011322  053406    12D2    5706
1262  011324  104064    12D4    8834          EMT VC0030-EMTTBL
1263  011326  022700    12D6    25C0          CMP =O'031',K       CONTROL Y
      011330  000031    12D8    0019
1264  011332  001371    12DA    02F9          BNE L315AA
1265  011334  005237    12DC    0A9F          INC INPB1         MOVE  PRINTED POINTER
      011336  053406    12DE    5706
1266  011340  104064    12E0    8834          EMT VC0030-EMTTBL
1267  011342  022700    12E2    25C0          CMP =O'023',K         CONTROL S
      011344  000023    12E4    0013
1268  011346  001363    12E6    02F3          BNE L315AA
1269  011350  005237    12E8    0A9F          INC INPB1         MOVE PRINTED POINTER
      011352  053406    12EA    5706
1270  011354  104064    12EC    8834          EMT VC0030-EMTTBL
1271  011356            12EE                L53A DSH 0
1272  011356  023700    12EE    27C0          CMP ASEQLS,K  IS IT ASSIGNMENT ARROW,  ONLY AFTER VARIABLE
      011360  052420    12F0    5510
1273  011362  001355    12F2    02ED          BNE L315AA
1274  011364  005016    12F4    0A0E          CLR (R6)     NOT AN UP ARROW
1275  011366  000003    12F6    0003          DEBUG  STL SUBR
1276  011370  177724    12F8    FFD4          DCH CHD4EX
1277  011372  104054    12FA    882C          EMT VC0024-EMTTBL
1278  011374  000137    12FC    005F          JMP S1
      011376  010630    12FE    1198
1279  011400            1300                L51 DSH 0       IS IT A NUMBER OR A LETTER
1280  011400  104060    1300    8830          EMT VC0027-EMTTBL
1281  011402  001004    1302    0204          BNE B73AZX
1282  011404  110037    1304    901F          MOVB R0,VARNM+2     SAVE 3RD CHARACTER OF VARIABLE
      011406  053314    1306    56CC
1283  011410  104054    1308    882C          EMT VC0024-EMTTBL
1284  011412  000455    130A    012D          BR L59
```

```
1285  011414  022700  130C  25C0        B73AZX CMP =E'(',K  IS IT AN ARRAY NAME (TWO CHARACTERS)
      011416  000050  130E  0028
1286  011420  001006  1310  0206         BNE B71
1287  011422  000665  1312  01B5         BR B6A  YES
1288  011424         1314               LEND DSH 0
1289  011424  062706  1314  65C6         ADD =2,R6
      011426  000002  1316  0002
1290  011430         1318               STRTS DSH 0
1291  011430  000003  1318  0003         DEBUG  STL SUBR
1292  011432  177773  131A  FFFB         DCH CHFBEX
1293  011434  000207  131C  0087         RTS PC
1294  011436  022700  131E  25C0        B71 CMP =E' ',K  IS THIS A SPACE
      011440  000040  1320  0020
1295  011442  001304  1322  02C4         BNE L53
1296  011444  022737  1324  25DF         CMP =E'I',KCH2  WAS SECOND LAST CHARACTER AN I
      011446  000111  1326  0049
      011450  053320  1328  56D0
1297  011452  001364  132A  02F4         BNE LEND
1298  011454  022737  132C  25DF         CMP =E'F',KCH1  WAS LAST CHARACTER AN F
      011456  000106  132E  0046
      011460  053322  1330  56D2
1299  011462  001360  1332  02F0         BNE LEND
1300  011464  104054  1334  882C         EMT VC0024-EMTTBL
1301  011466  000003  1336  0003         DEBUG  STL SUBR
1302  011470  177603  1338  FF83         DCH CH83EX  STORE INDICATOR FOR IF
1303  011472  104066  133A  8836         EMT VC0031-EMTTBL
1304  011474  000003  133C  0003         DEBUG  STL SUBR
1305  011476  177626  133E  FF96         DCH CH96EX  STORE INDICATOR
1306  011500  004437  1340  091F         JSR CHKR,CHECK
      011502  015170  1342  1A78
1307  011504  000040  1344  0020         DCC E' THEN :'  NEXT CHARACTERS TO BE ' THEN '
      011505  000124  1345  0054
      011506  000110  1346  0048
      011507  000105  1347  0045
      011510  000116  1348  004E
      011511  000040  1349  0020
      011512  000072  134A  003A
1308  011514  104056  134C  882E         EMT VC0025-EMTTBL
1309  011516  000003  134E  0003         DEBUG  STL SUBR
1310  011520  177646  1350  FFA6         DCH CHA6EX  STORE INDICATOR
1311  011522  004437  1352  091F         JSR CHKR,CHECK
      011524  015170  1354  1A78
1312  011526  000040  1356  0020         DCC E' ELSE :'  NEXT CHARACTERS TO BE ' ELSE '
      011527  000105  1357  0045
      011530  000114  1358  004C
      011531  000123  1359  0053
      011532  000105  135A  0045
      011533  000040  135B  0020
      011534  000072  135C  003A
1313  011536  104056  135E  882E         EMT VC0025-EMTTBL
1314  011540  000003  1360  0003         DEBUG  STL SUBR
1315  011542  177660  1362  FFB0         DCH CHB0EX  STORE INDICATOR TO END OF SECOND PART
1316  011544  000727  1364  01D7         BR LEND  MUST BE END OR ELSE A.E. WOULD STILL BE GOING
1317  011546         1366               L59 DSH 0
1318  011546  104060  1366  8830         EMT VC0027-EMTTBL
1319  011550  001636  1368  039E         BEQ B73  FOURTH CHARACTER IS NUMBER OR LETTER
1320  011552  022700  136A  25C0         CMP =E'(',K
      011554  000050  136C  0028
1321  011556  001236  136E  029E         BNE L53  NOT A FUNCTION
1322  011560  012737  1370  15DF         MOV =L315,SUBRET
```

```
        011562  011004    1372    1204
        011564  053016    1374    560E
1323    011566            1376            SUBRAB DSH 0
1324    011566  012703    1376    15C3     MOV =NMS,R3
        011570  012246    1378    14A6
1325    011572  121327    137A    A2D7    LOOPC CMPB (R3),=E':'  FUNCTION; SEE IF NAME IS ON LIST
        011574  000072    137C    003A
1326    011576  001436    137E    031E     BEQ LBSTSC
1327    011600  122337    1380    A4DF     CMPB (R3)+,KCH3
        011602  053316    1382    56CE
1328    011604  001026    1384    0216     BNE AD2
1329    011606  122337    1386    A4DF     CMPB (R3)+,KCH2
        011610  053320    1388    56D0
1330    011612  001024    138A    0214     BNE AD1
1331    011614  122337    138C    A4DF     CMPB (R3)+,KCH1
        011616  053322    138E    56D2
1332    011620  001022    1390    0212     BNE AD0
1333    011622  112337    1392    94DF     MOVB (R3)+,SUBYT
        011624  055020    1394    5A10
1334    011626  000003    1396    0003     DEBUG  STL SUBR
1335    011630  177703    1398    FFC3     DCH CHC3EX  INDICATES FUNCTION
1336    011632  104054    139A    882C     EMT VC0024-EMTTBL
1337    011634  113703    139C    97C3     MOVB SUBYT,R3
        011636  055020    139E    5A10
1338    011640  016301    13A0    1CC1     MOV STBL(R3),R1
        011642  012314    13A2    14CC
1339    011644  004711    13A4    09C9     JSR PC,(R1)
1340    011646            13A6            LNGB DSH 0
1341    011646  022700    13A6    25C0     CMP =E')',K  MUST CLOSE WITH A BRACKET
        011650  000051    13A8    0029
1342    011652  001171    13AA    0279     BNE STSC
1343    011654  104054    13AC    882C     EMT VC0024-EMTTBL
1344    011656  013707    13AE    17C7     MOV SUBRET,PC
        011660  053016    13B0    560E
1345    011662  005203    13B2    0A83    AD2 INC R3
1346    011664  005203    13B4    0A83    AD1 INC R3
1347    011666  062703    13B6    65C3    AD0 ADD =3,R3
        011670  000003    13B8    0003
1348    011672  000737    13BA    01DF     BR LOOPC
1349    011674  000137    13BC    005F    LBSTSC JMP STSC
        011676  012236    13BE    149E
1350    011700            13C0            STREX DSH 0
1351    011700  000003    13C0    0003     DEBUG
1352    011702  177772    13C2    FFFA     DCH CHFAEX
1353    011704            13C4            STREX1 DSH 0
1354    011704  022700    13C4    25C0     CMP =E'I',K
        011706  000111    13C6    0049
1355    011710  001446    13C8    0326     BEQ STRIF1
1356    011712  022700    13CA    25C0     CMP =E'H',K
        011714  000110    13CC    0048
1357    011716  001415    13CE    030D     BEQ STRA1
1358    011720  022700    13D0    25C0     CMP =O'047',K  QUOTE
        011722  000047    13D2    0027
1359    011724  001130    13D4    0258     BNE STRA5
1360    011726  104164    13D6    8874     EMT VC0101-EMTTBL   PICK UP STRING LITERAL
1361    011730  000003    13D8    0003     DEBUG
1362    011732  177637    13DA    FF9F     DCH CH9FEX
1363    011734  022700    13DC    25C0    STRA3 CMP =E'&',K   AMPERSAND
        011736  000046    13DE    0026
1364    011740  001233    13E0    029B     BNE STRTS
```

```
1365  011742  000003    13E2    0003          DEBUG
1366  011744  177724    13E4    FFD4          DCH CHD4EX
1367  011746  104054    13E6    882C          EMT VC0024-EMTTBL
1368  011750  000755    13E8    01ED          BR STREX1
1369  011752  104054    13EA    882C        STRA1 EMT VC0024-EMTTBL    STORE FIRST CHAR
1370  011754  104060    13EC    8830          EMT VC0027-EMTTBL        IS IT LETTER OR NUMBER
1371  011756  001001    13EE    0201          BNE STRA2
1372  011760  104054    13F0    882C          EMT VC0024-EMTTBL
1373  011762  022700    13F2    25C0        STRA2 CMP =E'(',K
      011764  000050    13F4    0028
1374  011766  001054    13F6    022C          BNE STRA6
1375  011770  000003    13F8    0003          DEBUG
1376  011772  177701    13FA    FFC1          DCH CHC1EX
1377  011774  104054    13FC    882C          EMT VC0024-EMTTBL
1378  011776  104056    13FE    882E          EMT VC0025-EMTTBL        AREX
1379  012000  022700    1400    25C0          CMP =E')',K
      012002  000051    1402    0029
1380  012004  001114    1404    024C          BNE STSC
1381  012006  104054    1406    882C          EMT VC0024-EMTTBL
1382  012010  023700    1408    27C0        STRA4 CMP ASEQLS,K       IS IT ASSIGN
      012012  052420    140A    5510
1383  012014  001347    140C    02E7          BNE STRA3
1384  012016  000003    140E    0003          DEBUG
1385  012020  177724    1410    FFD4          DCH CHD4EX
1386  012022  104054    1412    882C          EMT VC0024-EMTTBL
1387  012024  000727    1414    01D7          BR STREX1
1388  012026  004437    1416    091F        STRIF1 JSR CHKR,CHECK
      012030  015170    1418    1A78
1389  012032  000111    141A    0049          DCC E'IF :'
      012033  000106    141B    0046
      012034  000040    141C    0020
      012035  000072    141D    003A
1390  012036  000003    141E    0003          DEBUG
1391  012040  177603    1420    FF83          DCH CH83EX
1392  012042  104066    1422    8836          EMT VC0031-EMTTBL        LOGICAL EXPRESSION
1393  012044  000003    1424    0003          DEBUG
1394  012046  177626    1426    FF96          DCH CH96EX
1395  012050  004437    1428    091F          JSR CHKR,CHECK
      012052  015170    142A    1A78
1396  012054  000040    142C    0020          DCC E' THEN :'
      012055  000124    142D    0054
      012056  000110    142E    0048
      012057  000105    142F    0045
      012060  000116    1430    004E
      012061  000040    1431    0020
      012062  000072    1432    003A
1397  012064  104150    1434    8868          EMT VC0073-EMTTBL
1398  012066  000003    1436    0003          DEBUG
1399  012070  177646    1438    FFA6          DCH CHA6EX
1400  012072  004437    143A    091F          JSR CHKR,CHECK
      012074  015170    143C    1A78
1401  012076  000040    143E    0020          DCC E' ELSE :'
      012077  000105    143F    0045
      012100  000114    1440    004C
      012101  000123    1441    0053
      012102  000105    1442    0045
      012103  000040    1443    0020
      012104  000072    1444    003A
1402  012106  104150    1446    8868          EMT VC0073-EMTTBL
1403  012110  000003    1448    0003          DEBUG
```

```
1404  012112  177660    144A    FFB0          DCH CHB0EX
1405  012114  000137    144C    005F          JMP STRTS
      012116  011430    144E    1318
1406  012120  012703    1450    15C3        STRA6 MOV =NMS,R3
      012122  012246    1452    14A6
1407  012124  121327    1454    A2D7        STRA6N CMPB (R3),=E':'
      012126  000072    1456    003A
1408  012130  001727    1458    03D7          BEQ STRA4
1409  012132  122337    145A    A4DF          CMPB (R3)+,KCH2
      012134  053320    145C    56D0
1410  012136  001016    145E    020E          BNE STRA62
1411  012140  122337    1460    A4DF          CMPB (R3)+,KCH1
      012142  053322    1462    56D2
1412  012144  001014    1464    020C          BNE STRA61
1413  012146  122300    1466    A4C0          CMPB (R3)+,K
1414  012150  001013    1468    020B          BNE STRA60
1415  012152  104054    146A    882C          EMT VC0024-EMTTBL
1416  012154  022700    146C    25C0          CMP =E'(',K
      012156  000050    146E    0028
1417  012160  001026    1470    0216          BNE STSC
1418  012162  012737    1472    15DF          MOV =STRA3,SUBRET
      012164  011734    1474    13DC
      012166  053016    1476    560E
1419  012170  000137    1478    005F          JMP SUBRAB
      012172  011566    147A    1376
1420  012174  005203    147C    0A83        STRA62 INC R3
1421  012176  005203    147E    0A83        STRA61 INC R3
1422  012200  062703    1480    65C3        STRA60 ADD =3,R3
      012202  000003    1482    0003
1423  012204  000747    1484    01E7          BR STRA6N
1424  012206  022700    1486    25C0        STRA5 CMP =E'(',K
      012210  000050    1488    0028
1425  012212  001011    148A    0209          BNE STSC
1426  012214  000003    148C    0003          DEBUG
1427  012216  177700    148E    FFC0          DCH CHC0EX
1428  012220  104054    1490    882C          EMT VC0024-EMTTBL
1429  012222  104150    1492    8868          EMT VC0073-EMTTBL
1430  012224  022700    1494    25C0          CMP =E')',K
      012226  000051    1496    0029
1431  012230  001002    1498    0202          BNE STSC
1432  012232  104054    149A    882C          EMT VC0024-EMTTBL
1433  012234  000637    149C    019F          BR STRA3
1434  012236            149E                STSC DSH 0
1435  012236  005337    149E    0ADF          DEC INPA1       THAT CHARACTER DID NOT HAPPEN
      012240  053404    14A0    5704
1436                              *ALT2 MOV ALT2A,R1
1437                              * JMP (R1)
1438  012242  013707    14A2    17C7        ALT2 MOV ALT2A,PC
      012244  053306    14A4    56C6
1439                              * USE 4TH CHAR TO INDICATE EXPRESSION INSIDE
1440  012246  000123    14A6    0053        NMS DCC E'SQR'
      012247  000121    14A7    0051
      012250  000122    14A8    0052
1441  012251  000000    14A9    0000          DCB 0 AREX
1442  012252  015370    14AA    1AF8          DCH FSQR
1443  012254  000101    14AC    0041          DCC E'ABS'
      012255  000102    14AD    0042
      012256  000123    14AE    0053
1444  012257  000000    14AF    0000          DCB 0 AREX
1445  012260  035142    14B0    3A62          DCH FABS
```

```
1446  012262  000111   14B2   0049          DCC E'INT'
      012263  000116   14B3   004E
      012264  000124   14B4   0054
1447  012265  000000   14B5   0000          DCB 0 AREX
1448  012266  015602   14B6   1B82          DCH FINT
1449  012270  000122   14B8   0052          DCC E'RND'
      012271  000116   14B9   004E
      012272  000104   14BA   0044
1450  012273  000000   14BB   0000          DCB 0 AREX
1451  012274  015306   14BC   1AC6          DCH RND
1452  012276  000114   14BE   004C          DCC E'LNG'
      012277  000116   14BF   004E
      012300  000107   14C0   0047
1453  012301  000002   14C1   0002          DCB 2 STRING
1454  012302  012366   14C2   14F6          DCH LNGTH
1455  012304  000110   14C4   0048          DCC E'HXT'
      012305  000130   14C5   0058
      012306  000124   14C6   0054
1456  012307  000006   14C7   0006          DCB 6 LOGEX
1457  012310  012324   14C8   14D4          DCH SXTSXT
1458  012312  000072   14CA   003A          DCC E': '
      012313  000040   14CB   0020
1459  012314  010616   14CC   118E        STBL DCH AREX
1460  012316  011700   14CE   13C0          DCH STREX
1461  012320  012470   14D0   1538          DCH LOGEX
1462  012322  014202   14D2   1882          DCH GENEX
**********************************************
1500  012470          1538                LOGEX DSH 0
1501  012470  162706   1538   E5C6          SUB =2,R6
      012472  000002   153A   0002
1502  012474  005016   153C   0A0E          CLR (R6)        NO FULL RELATIONAL FOUND
1503  012476  022700   153E   25C0         R21 CMP =E' ',K  IS IT A SPACE
      012500  000040   1540   0020
1504  012502  001107   1542   0247          BNE R12
1505  012504  013737   1544   17DF          MOV L,LOGZ          SAVE STATE
      012506  053324   1546   56D4
      012510  053342   1548   56E2
1506  012512  013737   154A   17DF          MOV SL,LOG2
      012514  053326   154C   56D6
      012516  053344   154E   56E4
1507  012520  104054   1550   882C          EMT VC0024-EMTTBL
1508  012522  022700   1552   25C0          CMP =E'N',K       IS IT ' N'
      012524  000116   1554   004E
1509  012526  001031   1556   0219          BNE P1A
1510  012530  104054   1558   882C          EMT VC0024-EMTTBL
1511  012532  022700   155A   25C0          CMP =E'O',K  IS IT ' NO'
      012534  000117   155C   004F
1512  012536  001162   155E   0272          BNE R21REC
1513  012540  104054   1560   882C          EMT VC0024-EMTTBL
1514  012542  022700   1562   25C0          CMP =E'T',K       IS IT ' NOT'
      012544  000124   1564   0054
1515  012546  001052   1566   022A          BNE STSKA
1516  012550  104054   1568   882C          EMT VC0024-EMTTBL
1517  012552  000003   156A   0003          DEBUG  STL SUBR
1518  012554  177724   156C   FFD4          DCH CHD4EX  STORE INDICATOR
1519  012556  022700   156E   25C0          CMP =E' ',K
      012560  000040   1570   0020
1520  012562  001044   1572   0224          BNE STSKA  MUST BE ' NOT '
1521  012564  104054   1574   882C          EMT VC0024-EMTTBL
1522  012566  022700   1576   25C0          CMP =E' ',K   IS IT A SPACE
```

```
      012570  000040    1578    0020
1523  012572  001053    157A    022B          BNE R12
1524  012574  013737    157C    17DF          MOV L,LOGZ
      012576  053324    157E    56D4
      012600  053342    1580    56E2
1525  012602  013737    1582    17DF          MOV SL,LOG2
      012604  053326    1584    56D6
      012606  053344    1586    56E4
1526  012610  104054    1588    882C          EMT VC0024-EMTTBL
1527  012612  022700    158A    25C0        P1A CMP =E'T',K  IS IT ' T'
      012614  000124    158C    0054
1528  012616  001030    158E    0218          BNE A23
1529  012620  004437    1590    091F          JSR CHKR,CHECK
      012622  015170    1592    1A78
1530  012624  000124    1594    0054          DCC E'TRUE:'  IS IT ' TRUE '
      012625  000122    1595    0052
      012626  000125    1596    0055
      012627  000105    1597    0045
      012630  000072    1598    003A
1531  012632  022700    159A    25C0        KWH2 CMP =E' ',K
      012634  000040    159C    0020
1532  012636  001414    159E    030C          BEQ KWH1A
1533  012640  013704    15A0    17C4          MOV INPA1,R4
      012642  053404    15A2    5704
1534  012644  005304    15A4    0AC4          DEC R4
1535  012646  112724    15A6    95D4          MOVB =E' ',(R4)+
      012650  000040    15A8    0020
1536  012652  110024    15AA    9014          MOVB K,(R4)+
1537  012654  010437    15AC    111F          MOV R4,INPA1
      012656  053404    15AE    5704
1538  012660  005237    15B0    0A9F          INC HBECHO
      012662  053014    15B2    560C
1539  012664  012700    15B4    15C0          MOV =E' ',K
      012666  000040    15B6    0020
1540  012670  104054    15B8    882C        KWH1A EMT VC0024-EMTTBL
1541  012672  104410    15BA    8908          TRAP VB0014-TRPTBL
1542  012674  000137    15BC    005F        STSKA JMP STSC
      012676  012236    15BE    149E
1543  012700  022700    15C0    25C0        A23 CMP =E'F',K
      012702  000106    15C2    0046
1544  012704  001077    15C4    023F          BNE R21REC
1545  012706  004437    15C6    091F          JSR CHKR,CHECK
      012710  015170    15C8    1A78
1546  012712  000106    15CA    0046          DCC E'FALSE:'   IS IT ' FALSE '
      012713  000101    15CB    0041
      012714  000114    15CC    004C
      012715  000123    15CD    0053
      012716  000105    15CE    0045
      012717  000072    15CF    003A
1547  012720  000744    15D0    01E4          BR KWH2
1548  012722  022700    15D2    25C0        R12 CMP =E'(',K  IS IT AN OPENING BRACKET
      012724  000050    15D4    0028
1549  012726  001024    15D6    0214          BNE A13
1550  012730  000003    15D8    0003          DEBUG  STL SUBR
1551  012732  177700    15DA    FFC0          DCH CHC0EX
1552  012734  104054    15DC    882C          EMT VC0024-EMTTBL
1553  012736  104070    15DE    8838          EMT VC0032-EMTTBL
1554  012740  010137    15E0    105F          MOV R1,LOGZ  STORE RESULT IN LOGZ; 0 ARITH, 1 LOGICAL, -1 STRING
      012742  053342    15E2    56E2
1555  012744  022700    15E4    25C0          CMP =E')',K
```

```
      012746  000051    15E6    0029
1556  012750  001351    15E8    02E9          BNE STSKA  MUST END IN CLOSING BRACKET
1557  012752  104054    15EA    882C          EMT VC0024-EMTTBL
1558  012754  005737    15EC    0BDF          TST LOGZ TEST TYPE OF EXPRESSION JUST PICKED UP
      012756  053342    15EE    56E2
1559  012760  001403    15F0    0303          BEQ PPP61             ARITH
1560  012762  100004    15F2    8004          BPL P5TRA             LOGICAL
1561  012764  000137    15F4    005F          JMP P77AA              STRING
      012766  013402    15F6    1702
1562  012770           15F8                  PPP61 DSH 0
1563  012770  000137    15F8    005F          JMP P61
      012772  013434    15FA    171C
1564  012774           15FC                  P5TRA DSH 0
1565  012774  005016    15FC    0A0E          CLR (R6)      NOT FULL RELATIONAL ANY MORE; LOGICAL EXPR
1566  012776  104410    15FE    8908          TRAP VB0014-TRPTBL
1567  013000  022700    1600    25C0         A13 CMP =E'B',K  IS IT A 'B'; LOGICAL VARIABLES START WITH B
      013002  000102    1602    0042
1568  013004  001440    1604    0320          BEQ P3X
1569  013006  022700    1606    25C0          CMP =E'I',K  IS IT AN 'I' POSSIBLY IF
      013010  000111    1608    0049
1570  013012  001465    160A    0335          BEQ P4
1571  013014  000555    160C    016D          BR P6  MUST BE START OF ARITH OR STRING EXPRESS.
1572  013016  022700    160E    25C0         A5 CMP =E'(',K  IS SECOND CHARACTER AN OPENING BRACKET
      013020  000050    1610    0028
1573  013022  001020    1612    0210          BNE P34
1574  013024  000003    1614    0003         A63 DEBUG  STL SUBR
1575  013026  177701    1616    FFC1          DCH CHC1EX  STORE INDICATOR  (ARRAY NAME)
1576  013030  104054    1618    882C          EMT VC0024-EMTTBL
1577  013032  104056    161A    882E          EMT VC0025-EMTTBL
1578  013034  022700    161C    25C0          CMP =E')',K
      013036  000051    161E    0029
1579  013040  001410    1620    0308          BEQ P322  JUST ONE SUBSCRIPT
1580  013042  022700    1622    25C0          CMP =E',',K   IF COMMA A SECOND SUBSCRIPT IS THERE
      013044  000054    1624    002C
1581  013046  001312    1626    02CA          BNE STSKA
1582  013050  104054    1628    882C          EMT VC0024-EMTTBL
1583  013052  104056    162A    882E          EMT VC0025-EMTTBL
1584  013054  022700    162C    25C0          CMP =E')',K  MUST END IN CLOSING BRACKET
      013056  000051    162E    0029
1585  013060  001305    1630    02C5          BNE STSKA
1586  013062           1632                  P322 DSH 0
1587  013062  104054    1632    882C          EMT VC0024-EMTTBL
1588  013064  023700    1634    27C0         P34 CMP ASEQLS,K  IS IT ASSIGNMENT ARROW
      013066  052420    1636    5510
1589  013070  001341    1638    02E1          BNE P5TRA    PICK UP OPERATOR
1590  013072  005016    163A    0A0E          CLR (R6) NOT FULL RELATIONAL ANY MORE; ASSIGNMENT
1591  013074  000003    163C    0003          DEBUG  STL SUBR
1592  013076  177724    163E    FFD4          DCH CHD4EX
1593  013100  104054    1640    882C          EMT VC0024-EMTTBL
1594  013102  104412    1642    890A          TRAP VB0016-TRPTBL
1595  013104  000512    1644    014A         R21REC BR Z2SS
1596  013106           1646                  P3X DSH 0       ACCEPT FIRST CHARACTER OF VARIABLE NAME
1597  013106  005016    1646    0A0E          CLR (R6)  NOT FULL RELATIONAL ANY MORE; BOOLEAN VARIABLE
1598  013110  104054    1648    882C          EMT VC0024-EMTTBL
1599  013112  104060    164A    8830          EMT VC0027-EMTTBL
1600  013114  001340    164C    02E0          BNE A5
1601  013116  104054    164E    882C          EMT VC0024-EMTTBL
1602  013120           1650                  P32 DSH 0       IS THIRD CHARACTER A NUMBER OR A LETTER
1603  013120  104060    1650    8830          EMT VC0027-EMTTBL
1604  013122  001404    1652    0304          BEQ P32A
```

```
1605  013124  022700    1654   25C0          CMP =E'(',K  IS THIRD CHARACTER AN OPENING BRACKET
      013126  000050    1656   0028
1606  013130  001355    1658   02ED          BNE P34
1607  013132  000734    165A   01DC          BR A63
1608  013134            165C                 P32A DSH 0
1609  013134  104054    165C   882C          EMT VC0024-EMTTBL
1610  013136  022700    165E   25C0          CMP =E'(',K
      013140  000050    1660   0028
1611  013142  001005    1662   0205          BNE P32B
1612  013144  012737    1664   15DF          MOV =P5TRA,SUBRET
      013146  012774    1666   15FC
      013150  053016    1668   560E
1613  013152  000137    166A   005F          JMP SUBRAB
      013154  011566    166C   1376
1614  013156            166E                 P32B DSH 0
1615  013156  104060    166E   8830          EMT VC0027-EMTTBL
1616  013160  001341    1670   02E1          BNE P34
1617  013162  104054    1672   882C          EMT VC0024-EMTTBL
1618  013164  000737    1674   01DF          BR P34
1619  013166  013737    1676   17DF          P4 MOV L,LOGZ  SAVE STATE OF POINTERS
      013170  053324    1678   56D4
      013172  053342    167A   56E2
1620  013174  013737    167C   17DF          MOV SL,LOG2
      013176  053326    167E   56D6
      013200  053344    1680   56E4
1621  013202  011637    1682   139F          MOV (R6),LOG4     SAVE STATE OF FULL RELATIONAL
      013204  053346    1684   56E6
1622  013206  104054    1686   882C          EMT VC0024-EMTTBL
1623  013210  022700    1688   25C0          CMP =E'F',K  IS NEXT CHARACTER AN 'F'
      013212  000106    168A   0046
1624  013214  001044    168C   0224          BNE Z2
1625  013216  104054    168E   882C          EMT VC0024-EMTTBL
1626  013220  022700    1690   25C0          CMP =E' ',K  IS NEXT CHARACTER A SPACE
      013222  000040    1692   0020
1627  013224  001040    1694   0220          BNE Z2
1628  013226  104054    1696   882C          EMT VC0024-EMTTBL
1629  013230  000003    1698   0003          DEBUG  STL SUBR
1630  013232  177603    169A   FF83          DCH CH83EX  STORE INDICATOR AFTER 'IF '
1631  013234  104066    169C   8836          EMT VC0031-EMTTBL
1632  013236  000003    169E   0003          DEBUG  STL SUBR
1633  013240  177626    16A0   FF96          DCH CH96EX  STORE INDICATOR
1634  013242  004437    16A2   091F          JSR CHKR,CHECK
      013244  015170    16A4   1A78
1635  013246  000040    16A6   0020          DCC E' THEN :'  NEXT CHARACTERS MUST BE ' THEN '
      013247  000124    16A7   0054
      013250  000110    16A8   0048
      013251  000105    16A9   0045
      013252  000116    16AA   004E
      013253  000040    16AB   0020
      013254  000072    16AC   003A
1636  013256  104070    16AE   8838          EMT VC0032-EMTTBL
1637  013260  010137    16B0   105F          MOV R1,LOGZ  SAVE TYPE OF EXPRESSION
      013262  053342    16B2   56E2
1638  013264  000003    16B4   0003          DEBUG  STL SUBR
1639  013266  177646    16B6   FFA6          DCH CHA6EX  STORE INDICATOR
1640  013270  004437    16B8   091F          JSR CHKR,CHECK
      013272  015170    16BA   1A78
1641  013274  000040    16BC   0020          DCC E' ELSE :'  NEXT CHARACTERS MUST BE ' ELSE '
      013275  000105    16BD   0045
      013276  000114    16BE   004C
```

```
                013277  000123  16BF  0053
                013300  000105  16C0  0045
                013301  000040  16C1  0020
                013302  000072  16C2  003A
1642            013304  005737  16C4  0BDF          TST LOGZ  TEST TYPE
                013306  053342  16C6  56E2
1643            013310  001473  16C8  033B          BEQ P6X                              ARITH
1644            013312  100001  16CA  8001          BPL Z2AA                            LOGICAL
1645            013314  000467  16CC  0137          BR P99VV              STRING
1646            013316          16CE                Z2AA DSH 0
1647            013316  104066  16CE  8836          EMT VC0031-EMTTBL
1648            013320  000003  16D0  0003          DEBUG  STL SUBR
1649            013322  177660  16D2  FFB0          DCH CHB0EX  INDICATE END OF SECOND PART
1650            013324  000547  16D4  0167          BR R32  MUST BE END OF LOGICAL EXPRESSION
1651            013326  013716  16D6  17CE          Z2 MOV LOG4,(R6)
                013330  053346  16D8  56E6
1652            013332  013737  16DA  17DF          Z2SS MOV LOGZ,L  NOT 'IF '; RESTORE POINTERS
                013334  053342  16DC  56E2
                013336  053324  16DE  56D4
1653            013340  013737  16E0  17DF          MOV LOG2,SL
                013342  053344  16E2  56E4
                013344  053326  16E4  56D6
1654            013346  104072  16E6  883A          EMT VC0033-EMTTBL
1655            013350  005716  16E8  08CE          P6 TST (R6)
1656            013352  001404  16EA  0304          BEQ P998B  NO FULL RELATIONAL
1657            013354  100436  16EC  811E          BMI P99CC  PREVIOUS STRING RELATIONAL
1658            013356  004737  16EE  09DF          JSR PC,P61REL       SEE IF ONE OF 6
                013360  013652  16F0  17AA
1659            013362  001442  16F2  0322          BEQ P99GG
1660            013364  022700  16F4  25C0          P99BB CMP =E'H',K
                013366  000110  16F6  0048
1661            013370  001403  16F8  0303          BEQ SZA
1662            013372  022700  16FA  25C0          CMP =O'047',K
                013374  000047  16FC  0027
1663            013376  001015  16FE  020D          BNE P99DD
1664            013400          1700                SZA DSH 0
1665                            1700                *              STRING
1666            013400  104150  1700  8868          EMT VC0073-EMTTBL
1667            013402  004737  1702  09DF          P77AA JSR PC,P61SRL    SEE IF EQ OR NE
                013404  014044  1704  1824
1668            013406  001407  1706  0307          BEQ P99EE      YES
1669            013410  005716  1708  08CE          TST (R6)
1670            013412  100036  170A  801E          BPL STSK2
1671                            170C                *  CASE OF ELIDED LHOP AND REL
1672            013414  000003  170C  0003          P9999 DEBUG
1673            013416  177724  170E  FFD4          DCH CHD4EX
1674            013420  000003  1710  0003          DEBUG
1675            013422  177711  1712  FFC9          DCH CHC9EX
1676            013424  000433  1714  011B          BR P5
1677            013426  104150  1716  8868          P99EE EMT VC0073-EMTTBL
1678            013430  000431  1718  0119          BR P5
1679            013432  104056  171A  882E          P99DD EMT VC0025-EMTTBL
1680            013434  004737  171C  09DF          P61 JSR PC,P61REL
                013436  013652  171E  17AA
1681            013440  001424  1720  0314          BEQ P99FF
1682            013442  005716  1722  08CE          TST (R6)
1683            013444  100421  1724  8111          BMI STSK2
1684            013446  001420  1726  0310          BEQ STSK2
1685                            1728                *  CASE OF ELIDED LHOP AND REL
1686            013450  000761  1728  01F1          BR P9999
```

```
1687  013452  004737  172A  09DF        P99CC JSR PC,P61SRL
      013454  014044  172C  1824
1688  013456  001342  172E  02E2         BNE P998B
1689                                    *  CASE OF ELIDED LHOP FOR STRING
1690  013460  104150  1730  8868         EMT VC0073-EMTTBL
1691  013462  000003  1732  0003        P99JJ DEBUG
1692  013464  177713  1734  FFCB         DCH CHCBEX
1693  013466  000412  1736  010A         BR P5
1694  013470  104056  1738  882E        P99GG EMT VC0025-EMTTBL
1695  013472  000773  173A  01FB         BR P99JJ
1696  013474  104150  173C  8868        P99VV EMT VC0073-EMTTBL
1697  013476  000401  173E  0101         BR P99WW
1698  013500         1740               P6X DSH 0         PICK UP ARITHMETIC OPERATOR
1699  013500  104056  1740  882E         EMT VC0025-EMTTBL
1700  013502         1742               P99WW DSH 0
1701  013502  000003  1742  0003         DEBUG  STL SUBR
1702  013504  177660  1744  FFB0         DCH CHB0EX  INDICATE END OF SECOND PART
1703  013506  000752  1746  01EA         BR P61
1704  013510         1748               STSK2 DSH 0
1705  013510  104406  1748  8906         TRAP VB0011-TRPTBL
1706  013512  104056  174A  882E        P99FF EMT VC0025-EMTTBL
1707  013514  022700  174C  25C0        P5 CMP =E' ',K
      013516  000040  174E  0020
1708  013520  001051  1750  0229         BNE R32  NOT START OF LOGICAL OPERATOR; EXIT
1709  013522  013737  1752  17DF         MOV L,LOGZ  SAVE STATE OF POINTERS
      013524  053324  1754  56D4
      013526  053342  1756  56E2
1710  013530  013737  1758  17DF         MOV SL,LOG2
      013532  053326  175A  56D6
      013534  053344  175C  56E4
1711  013536  104054  175E  882C         EMT VC0024-EMTTBL
1712  013540  005237  1760  0A9F         INC FORPER
      013542  053332  1762  56DA
1713  013544  000004  1764  0004         IOT
1714  013546  000101  1766  0041         DCB E'A',PDB6-LADTAB
      013547  000000  1767  0000
1715  013550  000116  1768  004E         DCB E'N',PDB74-LADTAB
      013551  000002  1769  0002
1716  013552  000104  176A  0044         DCB E'D',PDB74-LADTAB
      013553  000002  176B  0002
1717  013554  000072  176C  003A         DCH E':'
1718  013556  000401  176E  0101         BR B8W1
1719  013560         1770               B8WOW DSH 0
1720  013560  104054  1770  882C         EMT VC0024-EMTTBL
1721  013562  000003  1772  0003        B8W1 DEBUG  STL SUBR
1722  013564  177724  1774  FFD4         DCH CHD4EX  STORE INDICATOR
1723  013566  022700  1776  25C0         CMP =E' ',K  CHECK LAST CHARACTER A SPACE
      013570  000040  1778  0020
1724  013572  001015  177A  020D         BNE B74
1725  013574  104054  177C  882C         EMT VC0024-EMTTBL
1726  013576  104412  177E  890A         TRAP VB0016-TRPTBL
1727  013600  022700  1780  25C0        B6 CMP =E'X',K  ARE TWO CHARACTERS ' X'
      013602  000130  1782  0058
1728  013604  001001  1784  0201         BNE B6EX
1729  013606  104054  1786  882C         EMT VC0024-EMTTBL
1730  013610  022700  1788  25C0        B6EX CMP =E'O',K  ARE CHARACTERS ' XO' OR ' O'
      013612  000117  178A  004F
1731  013614  001004  178C  0204         BNE B74
1732  013616  104054  178E  882C         EMT VC0024-EMTTBL
1733  013620  022700  1790  25C0         CMP =E'R',K  ARE CHARACTERS ' XOR' OR ' OR'
```

```
                013622  000122  1792  0052
1734            013624  001755  1794  03ED          BEQ B8WOW
1735            013626  013737  1796  17DF          B74 MOV LOGZ,L  RESTORE POINTERS
                013630  053342  1798  56E2
                013632  053324  179A  56D4
1736            013634  013737  179C  17DF          MOV LOG2,SL
                013636  053344  179E  56E4
                013640  053326  17A0  56D6
1737            013642  104072  17A2  883A          EMT VC0033-EMTTBL
1738            013644          17A4                R32 DSH 0
1739            013644  062706  17A6  65C6          ADD =2,R6           FULL RELATIONAL SPACE
                013646  000002  17A8  0002
1740            013650  000207  17AA  0087          RTS PC  EXIT
1741            013652  013737  17AC  17DF          P61REL MOV L,LOGZ
                013654  053324  17AE  56D4
                013656  053342  17B0  56E2
1742            013660  013737  17B2  17DF          MOV SL,LOG2
                013662  053326  17B4  56D6
                013664  053344  17B6  56E4
1743            013666  022700  17B8  25C0          CMP =E' ',K
                013670  000040  17BA  0020
1744            013672  001062  17BC  0232          BNE STSTXX
1745            013674  104054  17BE  882C          EMT VC0024-EMTTBL
1746            013676  022700  17C0  25C0          CMP =E'E',K  IS FIRST CHARACTER AN 'E'
                013700  000105  17C2  0045
1747            013702  001020  17C4  0210          BNE B14
1748            013704  104054  17C6  882C          EMT VC0024-EMTTBL
1749            013706  022700  17C8  25C0          CMP =E'Q',K   IS TWO CHARACTERS 'EQ'
                013710  000121  17CA  0051
1750            013712  001043  17CC  0223          BNE STSTST
1751            013714          17CE                P7 DSH 0
1752            013714  104054  17CE  882C          EMT VC0024-EMTTBL
1753            013716  000003  17D0  0003          DEBUG  STL SUBR
1754            013720  177724  17D2  FFD4          DCH CHD4EX  STORE INDICATOR
1755            013722  022700  17D4  25C0          CMP =E' ',K  MUST FINISH WITH SPACE
                013724  000040  17D6  0020
1756            013726  001035  17D8  021D          BNE STSTST
1757            013730  104054  17DA  882C          EMT VC0024-EMTTBL
1758            013732  013766  17DC  17F6          MOV COMPON,2(R6)    TURN ON IF COMPON IS NON-ZERO
                013734  052416  17DE  550E
                013736  000002  17E0  0002
1759            013740  000264  17E2  00B4          SEZ         SET TO EQUAL
1760            013742  000207  17E4  0087          RTS PC          GO BACK
1761            013744  022700  17E6  25C0          B14 CMP =E'N',K  IS FIRST CHARACTER AN 'N'
                013746  000116  17E8  004E
1762            013750  001005  17EA  0205          BNE B2
1763            013752  104054  17EC  882C          EMT VC0024-EMTTBL
1764            013754  022700  17EE  25C0          CMP =E'E',K  IS TWO CHARACTERS NE
                013756  000105  17F0  0045
1765            013760  001020  17F2  0210          BNE STSTST
1766            013762  000754  17F4  01EC          BR P7
1767            013764  022700  17F6  25C0          B2 CMP =E'L',K  IS FIRST CHARACTER AN 'L'
                013766  000114  17F8  004C
1768            013770  001010  17FA  0208          BNE B3B
1769            013772          17FC                B254 DSH 0
1770            013772  104054  17FC  882C          EMT VC0024-EMTTBL
1771            013774  022700  17FE  25C0          CMP =E'E',K IS TWO CHARACTERS 'LE' OR 'GE'
                013776  000105  1800  0045
1772            014000  001745  1802  03E5          BEQ P7
1773            014002  022700  1804  25C0          CMP =E'T',K  IS TWO CHARACTERS 'LT' OR 'GT'
```

```
      014004  000124  1804  0054
1774  014006  001742  1806  03E2          BEQ P7
1775  014010  000404  1808  0104          BR STSTST
1776  014012  022700  180A  25C0        B3B CMP =E'G',K  IS FIRST CHARACTER A 'G'
      014014  000107  180C  0047
1777  014016  001001  180E  0201          BNE STSTST
1778  014020  000764  1810  01F4          BR B254
1779  014022  013737  1812  17DF        STSTST MOV LOGZ,L
      014024  053342  1814  56E2
      014026  053324  1816  56D4
1780  014030  013737  1818  17DF          MOV LOG2,SL
      014032  053344  181A  56E4
      014034  053326  181C  56D6
1781  014036  104072  181E  883A          EMT VC0033-EMTTBL     RECOVER
1782  014040         1820              STSTXX DSH 0
1783  014040  000244  1820  00A4          CLZ                     SET TO NOT EQUAL
1784  014042  000207  1822  0087          RTS PC                GO BACK
1785  014044  013737  1824  17DF        P61SRL MOV L,LOGZ
      014046  053324  1826  56D4
      014050  053342  1828  56E2
1786  014052  013737  182A  17DF          MOV SL,LOG2
      014054  053326  182C  56D6
      014056  053344  182E  56E4
1787  014060  022700  1830  25C0          CMP =E' ',K
      014062  000040  1832  0020
1788  014064  001044  1834  0224          BNE P61SB
1789  014066  104054  1836  882C          EMT VC0024-EMTTBL
1790  014070  022700  1838  25C0          CMP =E'E',K
      014072  000105  183A  0045
1791  014074  001022  183C  0212          BNE P61SA
1792  014076  104054  183E  882C          EMT VC0024-EMTTBL
1793  014100  022700  1840  25C0          CMP =E'Q',K
      014102  000121  1842  0051
1794  014104  001025  1844  0215          BNE P61SC
1795  014106  104054  1846  882C        P61SD EMT VC0024-EMTTBL
1796  014110  000003  1848  0003          DEBUG
1797  014112  177724  184A  FFD4          DCH CHD4EX
1798  014114  022700  184C  25C0          CMP =E' ',K
      014116  000040  184E  0020
1799  014120  001017  1850  020F          BNE P61SC
1800  014122  104054  1852  882C          EMT VC0024-EMTTBL
1801  014124  013766  1854  17F6          MOV COMPCN,2(R6)
      014126  052416  1856  550E
      014130  000002  1858  0002
1802  014132  005466  185A  0B36          NEG 2(R6)
      014134  000002  185C  0002
1803  014136  000264  185E  00B4          SEZ
1804  014140  000207  1860  0087          RTS PC
1805  014142  022700  1862  25C0        P61SA CMP =E'N',K
      014144  000116  1864  004E
1806  014146  001004  1866  0204          BNE P61SC
1807  014150  104054  1868  882C          EMT VC0024-EMTTBL
1808  014152  022700  186A  25C0          CMP =E'E',K
      014154  000105  186C  0045
1809  014156  001753  186E  03EB          BEQ P61SD
1810  014160  013737  1870  17DF        P61SC MOV LOGZ,L
      014162  053342  1872  56E2
      014164  053324  1874  56D4
1811  014166  013737  1876  17DF          MOV LOG2,SL
      014170  053344  1878  56E4
```

```
      014172  053326    187A    56D6
1812  014174  104072    187C    883A          EMT VC0033-EMTTBL
1813  014176  000244    187E    00A4        P61SB CLZ
1814  014200  000207    1880    0087          RTS PC
1815  014202  162706    1882    E5C6        GENEX SUB =4,R6  PICK UP GENERAL EXPRESSION, 2 LOCAL WORDS
      014204  000004    1884    0004
1816  014206  022700    1886    25C0          CMP =E' ',K  IS FIRST CHARACTER A SPACE
      014210  000040    1888    0020
1817  014212  001002    188A    0202          BNE G11BB
1818  014214            188C                BQG1 DSH 0   IF SO GO TO G1 & PICK UP A LOGICAL EXPR.
1819  014214  000137    188C    005F          JMP G1
      014216  015000    188E    1A00
1820  014220            1890                G11BB DSH 0
1821  014220  022700    1890    25C0          CMP =E'B',K  IS FIRST CHAR A 'B'
      014222  000102    1892    0042
1822  014224  001773    1894    03FB          BEQ BQG1 IF SO GO TO G1
1823  014226  013716    1896    17CE          MOV L,(R6)  SAVE STATE OF POINTERS
      014230  053324    1898    56D4
1824  014232  013766    189A    17F6          MOV SL,2(R6)
      014234  053326    189C    56D6
      014236  000002    189E    0002
1825  014240  022700    18A0    25C0          CMP =E'H',K  IS FIRST CHAR AN 'H'
      014242  000110    18A2    0048
1826  014244  001427    18A4    0317          BEQ GENSAA
1827  014246  022700    18A6    25C0          CMP =O'047',K  IS FIRST CHARACTER A QUOTE
      014250  000047    18A8    0027
1828  014252  001424    18AA    0314          BEQ GENSAA
1829  014254  022700    18AC    25C0          CMP =E'(',K  SEE IF OPENING BRACKET
      014256  000050    18AE    0028
1830  014260  001076    18B0    023E          BNE A14
1831  014262  000003    18B2    0003          DEBUG  STL SUBR
1832  014264  177700    18B4    FFC0          DCH CHC0EX  STORE INDICATOR
1833  014266  104054    18B6    882C          EMT VC0024-EMTTBL
1834  014270  104070    18B8    8838          EMT VC0032-EMTTBL
1835  014272  010137    18BA    105F          MOV R1,GENZ  SAVE TYPE OF EXPRESSION, 0 ARITH, 1 LOGICAL
      014274  053356    18BC    56EE
1836  014276  022700    18BE    25C0          CMP =E')',K
      014300  000051    18C0    0029
1837  014302  001007    18C2    0207          BNE STSCXX  MUST BE CLOSING BRACKET
1838  014304  104054    18C4    882C          EMT VC0024-EMTTBL
1839  014306  005737    18C6    0BDF          TST GENZ  TEST TYPE
      014310  053356    18C8    56EE
1840  014312  001543    18CA    0363          BEQ G2  ARITH
1841  014314  100404    18CC    8104          BMI GENSAB     STRING
1842  014316  000137    18CE    005F          JMP A64       LOGICAL
      014320  014764    18D0    19F4
1843  014322            18D2                STSCXX DSH 0
1844  014322  104406    18D2    8906          TRAP VB0011-TRPTBL
1845  014324  104150    18D4    8868        GENSAA EMT VC0073-EMTTBL
1846  014326            18D6                GENSAB DSH 0
1847  014326  022700    18D6    25C0          CMP =E' ',K
      014330  000040    18D8    0020
1848  014332  001046    18DA    0226          BNE STRB9 MUST BE STRING
1849  014334  013737    18DC    17DF          MOV L,GEN6
      014336  053324    18DE    56D4
      014340  053360    18E0    56F0
1850  014342  013737    18E2    17DF          MOV SL,GEN8
      014344  053326    18E4    56D6
      014346  053362    18E6    56F2
1851  014350  104054    18E8    882C          EMT VC0024-EMTTBL
```

```
1852  014352  022700    18EA  25C0          CMP =E'E',K
      014354  000105    18EC  0045
1853  014356  001413    18EE  030B          BEQ STRB2
1854  014360  022700    18F0  25C0          CMP =E'N',K
      014362  000116    18F2  004E
1855  014364  001415    18F4  030D          BEQ STRB3
1856  014366  013737    18F6  17DF        STRB4 MOV GEN6,L
      014370  053360    18F8  56F0
      014372  053324    18FA  56D4
1857  014374  013737    18FC  17DF          MOV GEN8,SL
      014376  053362    18FE  56F2
      014400  053326    1900  56D6
1858  014402  104072    1902  883A          EMT VC0033-EMTTBL
1859  014404  000421    1904  0111          BR STRB9
1860  014406  104054    1906  882C        STRB2 EMT VC0024-EMTTBL
1861  014410  022700    1908  25C0          CMP =E'Q',K   IS IT ' EQ'
      014412  000121    190A  0051
1862  014414  001364    190C  02F4          BNE STRB4
1863  014416  000404    190E  0104          BR STRB5
1864  014420  104054    1910  882C        STRB3 EMT VC0024-EMTTBL
1865  014422  022700    1912  25C0          CMP =E'E',K     IS IT ' NE'
      014424  000105    1914  0045
1866  014426  001357    1916  02EF          BNE STRB4
1867  014430  011637    1918  139F        STRB5 MOV (R6),L    RESTORE TO START OF STRING
      014432  053324    191A  56D4
1868  014434  016637    191C  1D9F          MOV 2(R6),SL
      014436  000002    191E  0002
      014440  053326    1920  56D6
1869  014442  104072    1922  883A          EMT VC0033-EMTTBL
1870  014444  000137    1924  005F          JMP G1        GO BACK AND PICK UP AS LOGICAL
      014446  015000    1926  1A00
1871  014450            1928              STRB9 DSH 0
1872  014450  012701    1928  15C1          MOV =-1,R1
      014452  177777    192A  FFFF
1873  014454  000560    192C  0170          BR G3A
1874  014456  022700    192E  25C0        A14 CMP =E'I',K  IS IT 'I'
      014460  000111    1930  0049
1875  014462  001057    1932  022F          BNE G2  MUST BE ARITH EXP.
1876  014464  104054    1934  882C          EMT VC0024-EMTTBL
1877  014466  022700    1936  25C0          CMP =E'F',K  IS IT 'IF'
      014470  000106    1938  0046
1878  014472  001053    193A  022B          BNE G2
1879  014474  104054    193C  882C          EMT VC0024-EMTTBL
1880  014476  022700    193E  25C0          CMP =E' ',K  IS IT 'IF '
      014500  000040    1940  0020
1881  014502  001047    1942  0227          BNE G2
1882  014504  104054    1944  882C          EMT VC0024-EMTTBL
1883  014506  000003    1946  0003          DEBUG  STL SUBR
1884  014510  177603    1948  FF83          DCH CH83EX  STORE INDICATOR
1885  014512  104066    194A  8836          EMT VC0031-EMTTBL
1886  014514  000003    194C  0003          DEBUG  STL SUBR
1887  014516  177626    194E  FF96          DCH CH96EX  STORE INDICATOR
1888  014520  004437    1950  091F          JSR CHKR,CHECK
      014522  015170    1952  1A78
1889  014524  000040    1954  0020          DCC E' THEN :'  MUST BE ' THEN '
      014525  000124    1955  0054
      014526  000110    1956  0048
      014527  000105    1957  0045
      014530  000116    1958  004E
      014531  000040    1959  0020
```

```
        014532  000072  195A  003A
1890    014534  104070  195C  8838        EMT VC0032-EMTTBL
1891    014536  010137  195E  105F        MOV R1,GENZ  SAVE TYPE OF EXPRESSION
        014540  053356  1960  56EE
1892    014542  000003  1962  0003        DEBUG  STL SUBR
1893    014544  177646  1964  FFA6        DCH CHA6EX  STORE INDICATOR
1894    014546  004437  1966  091F        JSR CHKR,CHECK
        014550  015170  1968  1A78
1895    014552  000040  196A  0020        DCC E' ELSE :'  MUST BE ' ELSE '
        014553  000105  196B  0045
        014554  000114  196C  004C
        014555  000123  196D  0053
        014556  000105  196E  0045
        014557  000040  196F  0020
        014560  000072  1970  003A
1896    014562  005737  1972  0BDF        TST GENZ  TEST TYPE
        014564  053356  1974  56EE
1897    014566  001411  1976  0309        BEQ A2X       ARITH
1898    014570  100004  1978  8004        BPL Z2CC       LOGICAL
1899    014572          197A              DSH 0         STRING
1900    014572  104150  197A  8868        EMT VC0073-EMTTBL
1901    014574  000003  197C  0003        DEBUG
1902    014576  177660  197E  FFB0        DCH CHB0EX
1903    014600  000652  1980  01AA        BR GENSAB   SHOULD TEST FOR ' EQ '; ' NE '.
1904    014602          1982              Z2CC DSH 0
1905    014602  104066  1982  8836        EMT VC0031-EMTTBL
1906    014604  000003  1984  0003        DEBUG  STL SUBR
1907    014606  177660  1986  FFB0        DCH CHB0EX  STORE INDICATOR
1908    014610  000474  1988  013C        BR G1A  SET LOGICAL AND EXIT
1909    014612          198A              A2X DSH 0        PICK UP ARITHMETIC EXPRESSION
1910    014612  104056  198A  882E        EMT VC0025-EMTTBL
1911    014614  000003  198C  0003        DEBUG  STL SUBR
1912    014616  177660  198E  FFB0        DCH CHB0EX  STORE INDICATOR
1913    014620  000407  1990  0107        BR G2NWBY
1914    014622  011637  1992  139F        G2 MOV (R6),L  RESTORE INDICATORS
        014624  053324  1994  56D4
1915    014626  016637  1996  1D9F        MOV 2(R6),SL
        014630  000002  1998  0002
        014632  053326  199A  56D6
1916    014634  104072  199C  883A        EMT VC0033-EMTTBL
1917    014636  104056  199E  882E        EMT VC0025-EMTTBL
1918    014640          19A0              G2NWBY DSH 0
1919    014640  022700  19A0  25C0        CMP =E' ',K  IS IT SPACE
        014642  000040  19A2  0020
1920    014644  001003  19A4  0233        BNE G3  MUST BE ARITH
1921    014646  013737  19A6  17DF        MOV L,GEN6
        014650  053324  19A8  56D4
        014652  053360  19AA  56F0
1922    014654  013737  19AC  17DF        MOV SL,GEN8
        014656  053326  19AE  56D6
        014660  053362  19B0  56F2
1923    014662  104054  19B2  882C        EMT VC0024-EMTTBL
1924    014664  022700  19B4  25C0        CMP =E'L',K  IS IT '.L'
        014666  000114  19B6  004C
1925    014670  001426  19B8  0316        BEQ G21
1926    014672  022700  19BA  25C0        CMP =E'G',K  IS IT ' G'
        014674  000107  19BC  0047
1927    014676  001423  19BE  0313        BEQ G21
1928    014700  022700  19C0  25C0        CMP =E'E',K  IS IT ' E'
        014702  000105  19C2  0045
```

```
1929  014704  001413    19C4  030B          BEQ A33
1930  014706  022700    19C6  25C0          CMP =E'N',K  IS IT ' N'
      014710  000116    19C8  004E
1931  014712  001436    19CA  031E          BEQ A4
1932  014714            19CC                A49 DSH 0
1933  014714  013737    19CC  17DF          MOV GEN6,L
      014716  053360    19CE  56F0
      014720  053324    19D0  56D4
1934  014722  013737    19D2  17DF          MOV GEN8,SL
      014724  053362    19D4  56F2
      014726  053326    19D6  56D6
1935  014730  104072    19D8  883A          EMT VC0033-EMTTBL
1936  014732  000430    19DA  0118          BR G3  SET ARITH AND EXIT
1937  014734            19DC                A33 DSH 0
1938  014734  104054    19DC  882C          EMT VC0024-EMTTBL
1939  014736  022700    19DE  25C0          CMP =E'Q',K  IS IT ' EQ'
      014740  000121    19E0  0051
1940  014742  001364    19E2  02F4          BNE A49
1941  014744  000407    19E4  0107          BR A64
1942  014746            19E6                G21 DSH 0
1943  014746  104054    19E6  882C          EMT VC0024-EMTTBL
1944  014750  022700    19E8  25C0          CMP =E'T',K  IS IT ' LT' OR ' GT'
      014752  000124    19EA  0054
1945  014754  001403    19EC  0303          BEQ A54
1946  014756            19EE                G21ZAB DSH 0
1947  014756  022700    19EE  25C0          CMP =E'E',K  IS IT ' LE' OR ' GE' OR ' NE'
      014760  000105    19F0  0045
1948  014762  001354    19F2  02EC          BNE A49
1949  014764  011637    19F4  139F          A64 MOV (R6),L  YES RELATIONAL OP: RESTORE TO START OF A.E.
      014766  053324    19F6  56D4
1950  014770  016637    19F8  1D9F          MOV 2(R6),SL
      014772  000002    19FA  0002
      014774  053326    19FC  56D6
1951  014776  104072    19FE  883A          EMT VC0033-EMTTBL
1952  015000            1A00                G1 DSH 0          PICK UP LOGICAL EXPRESSION
1953  015000  104066    1A00  8836          EMT VC0031-EMTTBL
1954  015002  012701    1A02  15C1          G1A MOV =1,R1  SET TO LOGICAL
      015004  000001    1A04  0001
1955  015006  000403    1A06  0103          BR G3A
1956  015010            1A08                A4 DSH 0
1957  015010  104054    1A08  882C          EMT VC0024-EMTTBL
1958  015012  000761    1A0A  01F1          BR G21ZAB
1959  015014  005001    1A0C  0A01          G3 CLR R1  SET TO ZERO TO INDICATE ARITHMETIC EXP
1960  015016  062706    1A0E  65C6          G3A ADD =4,R6  RESTORE STACK OVER 2 LOCAL WORDS
      015020  000004    1A10  0004
1961  015022  000207    1A12  0087          RTS PC
1962  015024  022700    1A14  25C0          LETT CMP =E'B',K  LETTER TEST
      015026  000102    1A16  0042
1963  015030  001424    1A18  0314          BEQ NO
1964  015032  022700    1A1A  25C0          CMP =E'H',K
      015034  000110    1A1C  0048
1965  015036  001421    1A1E  0311          BEQ NO
1966  015040  020027    1A20  2017          LETL CMP K,=E'A'
      015042  000101    1A22  0041
1967  015044  100416    1A24  810E          BMI NO
1968  015046  022700    1A26  25C0          CMP =E'Z',K
      015050  000132    1A28  005A
1969  015052  100413    1A2A  810B          BMI NO
1970  015054  000410    1A2C  0108          BR YES
1971  015056            1A2E                NL DSH 0
```

```
1972  015056  104074    1A2E    883C          EMT VC0034-EMTTBL
1973  015060  001407    1A30    0307          BEQ RETPC
1974                                          * JSUBR PC,NUMB
1975                                          * RTS PC
1976  015062  020027    1A32    2017          NUMB CMP K,=E'0'  NUMBER TEST
      015064  000060    1A34    0030
1977  015066  100405    1A36    8105          BMI NO
1978  015070  022700    1A38    25C0          CMP =E'9',K
      015072  000071    1A3A    0039
1979  015074  100402    1A3C    8102          BMI NO
1980  015076  000264    1A3E    00B4          YES SEZ
1981  015100  000207    1A40    0087          RETPC RTS PC
1982  015102  000244    1A42    00A4          NO CLZ
1983  015104  000207    1A44    0087          RTS PC
1984  015106  013701    1A46    17C1          GETK MOV L,R1  RESTORE BUFFER AND CHARACTER IN REG ZERO
      015110  053324    1A48    56D4
1985  015112  162701    1A4A    E5C1          SUB =4,R1
      015114  000004    1A4C    0004
1986  015116  112137    1A4E    945F          MOVB (R1)+,KCH3
      015120  053316    1A50    56CE
1987  015122  112137    1A52    945F          MOVB (R1)+,KCH2
      015124  053320    1A54    56D0
1988  015126  112137    1A56    945F          MOVB (R1)+,KCH1
      015130  053322    1A58    56D2
1989  015132  112100    1A5A    9440          MOVB (R1)+,K
1990  015134  000207    1A5C    0087          RTS PC
1991  015136  010137    1A5E    105F          STL MOV R1,SAVR1
      015140  053376    1A60    56FE
1992  015142  013702    1A62    17C2          MOV SL,R2
      015144  053326    1A64    56D6
1993  015146  011601    1A66    1381          MOV (R6),R1
1994  015150  112122    1A68    9452          MOVB (R1)+,(R2)+
1995  015152  010237    1A6A    109F          MOV R2,SL
      015154  053326    1A6C    56D6
1996  015156  005201    1A6E    0A81          INC R1
1997  015160  010116    1A70    104E          MOV R1,(R6)
1998  015162  013701    1A72    17C1          MOV SAVR1,R1
      015164  053376    1A74    56FE
1999  015166  000002    1A76    0002          RTI
2000  015170  112401    1A78    9501          CHECK MOVB (CHKR)+,R1  CHECK A SEQUENCE OF CHARACTERS
2001  015172  122701    1A7A    A5C1          CMPB =E':',R1
      015174  000072    1A7C    003A
2002  015176  001404    1A7E    0304          BEQ CHECK1
2003  015200  020001    1A80    2001          CMP K,R1
2004  015202  001006    1A82    0206          BNE CHECK2
2005  015204  104054    1A84    882C          EMT VC0024-EMTTBL
2006  015206  000770    1A86    01F8          BR CHECK
2007  015210            1A88                  CHECK1 DSH 0
2008  015210  005204    1A88    0A84          INC CHKR
2009  015212  042704    1A8A    45C4          BIC =X'0001',CHKR
      015214  000001    1A8C    0001
2010  015216  000204    1A8E    0084          CHECK3 RTS CHKR
2011  015220            1A90                  CHECK2 DSH 0
2012  015220  104406    1A90    8906          TRAP VB0011-TRPTBL
**********************************************
3740  027130  004737    2E58    09DF          STKERR JSR PC,TRAKK
      027132  035402    2E5A    3B02
3741  027134  013701    2E5C    17C1          STK MOV SL,R1   LOAD POINTER TO VECTOR
      027136  053326    2E5E    56D6
3742  027140  110021    2E60    9011           MOVB R0,(R1)+     STORE CHARACTER
```

```
3743  027142  010137    2E62    105F          MOV R1,SL         STORE POINTER
      027144  053326    2E64    56D6
3744  027146  013701    2E66    17C1          MOV L,R1
      027150  053324    2E68    56D4
3745  027152  005301    2E6A    0AC1          DEC R1
3746  027154  020137    2E6C    205F          CMP R1,INPB1
      027156  053406    2E6E    5706
3747  027160  101363    2E70    82F3          BHI STKERR
3748  027162  001004    2E72    0204          BNE STK1
3749  027164  112100    2E74    9440          MOVB (R1)+,R0
3750  027166  010137    2E76    105F          MOV R1,INPB1
      027170  053406    2E78    5706
3751  027172  104014    2E7A    880C          EMT VC0007-EMTTBL
3752  027174            2E7C                 STK1 DSH 0
3753  027174  013737    2E7C    17DF         GNC MOV KCH2,KCH3  GET NEXT CHARACTER
      027176  053320    2E7E    56D0
      027200  053316    2E80    56CE
3754  027202  013737    2E82    17DF          MOV KCH1,KCH2
      027204  053322    2E84    56D2
      027206  053320    2E86    56D0
3755  027210  010037    2E88    101F          MOV K,KCH1
      027212  053322    2E8A    56D2
3756  027214  013701    2E8C    17C1          MOV L,R1 POINTER TO LAST + 1 CHARACTER TAKEN FROM STRING
      027216  053324    2E8E    56D4
3757  027220  013702    2E90    17C2          MOV INPA1,R2 POINTS TO NEXT CHARACTER TO BE ENTERED INTO STRING
      027222  053404    2E92    5704
3758  027224  020102    2E94    2042          CMP R1,R2
3759  027226  001412    2E96    030A          BEQ GNC1  IF EQ MEANS THAT THE CHARACTERS HAVE ALL BEEN GIVEN TO THE PROG
3760                               *  WILL HAVE TO GET ONE FROM THE KEYBOARD
3761  027230  112100    2E98    9440          MOVB (R1)+,K    GET NEXT CHAR. FROM STRING; INC THE POINTER
3762  027232  005037    2E9A    0A1F          CLR STRCHR
      027234  055002    2E9C    5A02
3763  027236  010137    2E9E    105F          MOV R1,L  RESTORE THE POINTER
      027240  053324    2EA0    56D4
3764  027242  022700    2EA2    25C0          CMP =X'FFFF',R0
      027244  177777    2EA4    FFFF
3765  027246  001401    2EA6    0301          BEQ GNCA         EDIT OR EXORCISER
3766  027250  000207    2EA8    0087          RTS PC
3767  027252            2EAA                 GNCA DSH 0
3768  027252  104406    2EAA    8906          TRAP VB0011-TRPTBL
3769  027254            2EAC                 GNC1 DSH 0          GET CHARACTER FROM KEYBOARD; MUST NOT CLOBBER R1
3770  027254  104152    2EAC    886A          EMT VC0074-EMTTBL
3771  027256  012737    2EAE    15DF          MOV =63,STRCHR
      027260  000077    2EB0    003F
      027262  055002    2EB2    5A02
3772  027264  010237    2EB4    109F          MOV R2,INPZZ
      027266  055332    2EB6    5ADA
3773  027270  113737    2EB8    97DF          MOVB LASTCH,LASTCH+1
      027272  054644    2EBA    59A4
      027274  054645    2EBC    59A5
3774  027276  110037    2EBE    901F          MOVB R0,LASTCH
      027300  054644    2EC0    59A4
3775  027302  023727    2EC2    27D7          CMP LASTCH,=E'>>'
      027304  054644    2EC4    59A4
      027306  037076    2EC6    3E3E
3776  027310  001414    2EC8    030C          BEQ F2222
3777  027312  022700    2ECA    25C0          CMP =E'<',R0          <
      027314  000074    2ECC    003C
3778  027316  001415    2ECE    030D          BEQ RUBOUT
3779  027 0   022700    2ED0    25C0          CMP =E'>',R0
```

```
      027322  000076    2ED2    003E
3780  027324  001410    2ED4    0308          BEQ GNCB
3781  027326  110022    2ED6    9012          MOVB K,(R2)+ STORE BYTE IN STRING
3782  027330  010237    2ED8    109F          MOV R2,INPA1 STORE POINTER TO STRING
      027332  053404    2EDA    5704
3783  027334  010237    2EDC    109F          MOV R2,L
      027336  053324    2EDE    56D4
3784  027340  000207    2EE0    0087          RTS PC
3785  027342  000137    2EE2    005F         F2222 JMP F22A
      027344  040444    2EE4    4124
3786  027346            2EE6                 GNCB DSH 0
3787  027346  000137    2EE6    005F          JMP FERRET
      027350  037770    2EE8    3FF8
3788  027352            2EEA                 RUBOUT DSH 0
3789  027352  104014    2EEA    880C          EMT VC0007-EMTTBL
3790  027354  012737    2EEC    15DF          MOV =1,RUBAAA
      027356  000001    2EEE    0001
      027360  055334    2EF0    5ADC
3791  027362            2EF2                 RUBA2 DSH 0
3792  027362  104152    2EF2    886A          EMT VC0074-EMTTBL
3793  027364  022700    2EF4    25C0          CMP =E':',R0
      027366  000072    2EF6    003A
3794  027370  001442    2EF8    0322          BEQ RUBA1
3795  027372  022700    2EFA    25C0          CMP =E'E',R0
      027374  000105    2EFC    0045
3796  027376  001456    2EFE    032E          BEQ RUBE
3797  027400  022700    2F00    25C0          CMP =E'A',R0
      027402  000101    2F02    0041
3798  027404  001450    2F04    0328          BEQ RUBA
3799  027406  104052    2F06    881A          EMT VC0014-EMTTBL
3800  027410  001364    2F08    02F4          BNE RUBA2
3801  027412  104014    2F0A    880C          EMT VC0007-EMTTBL
3802  027414  162700    2F0C    E5C0          SUB =E'0',R0
      027416  000060    2F0E    0030
3803  027420  010037    2F10    101F          MOV R0,RUBAAA
      027422  055334    2F12    5ADC
3804  027424            2F14                 RUBA3 DSH 0
3805  027424  104152    2F14    886A          EMT VC0074-EMTTBL
3806  027426  022700    2F16    25C0          CMP =E':',R0
      027430  000072    2F18    003A
3807  027432  001421    2F1A    0311          BEQ RUBA1
3808  027434  104052    2F1C    881A          EMT VC0014-EMTTBL
3809  027436  001372    2F1E    02FA          BNE RUBA3
3810  027440  104014    2F20    880C          EMT VC0007-EMTTBL
3811  027442  013702    2F22    17C2          MOV RUBAAA,R2
      027444  055334    2F24    5ADC
3812  027446  006302    2F26    0CC2          ASL R2
3813  027450  010203    2F28    1083          MOV R2,R3
3814  027452  006303    2F2A    0CC3          ASL R3
3815  027454  006303    2F2C    0CC3          ASL R3
3816  027456  060203    2F2E    6083          ADD R2,R3
3817  027460  060003    2F30    6003          ADD R0,R3
3818  027462  162703    2F32    E5C3          SUB =E'0',R3
      027464  000060    2F34    0030
3819  027466  010337    2F36    10DF          MOV R3,RUBAAA
      027470  055334    2F38    5ADC
3820  027472  012700    2F3A    15C0          MOV =E':',R0
      027474  000072    2F3C    003A
3821  027476            2F3E                 RUBA1 DSH 0
3822  027476  104014    2F3E    880C          EMT VC0007-EMTTBL
```

```
3823  027500  013700  2F40  17C0        MOV INPZZ,R0
      027502  055332  2F42  5ADA
3824  027504  163700  2F44  E7C0        SUB RUBAAA,R0
      027506  055334  2F46  5ADC
3825  027510  010037  2F48  101F        MOV R0,INPA1
      027512  053404  2F4A  5704
3826  027514  020027  2F4C  2017        CMP R0,=INPUT+1
      027516  054515  2F4E  594D
3827  027520  100011  2F50  8009        BPL RUBB
3828  027522  013707  2F52  17C7       ALT3 MOV ALT3A,PC
      027524  055330  2F54  5AD8
3829  027526  104014  2F56  880C       RUBA EMT VC0007-EMTTBL
3830  027530  013707  2F58  17C7        MOV ALT3A,PC
      027532  055330  2F5A  5AD8
3831  027534          2F5C              RUBE DSH 0
3832  027534  000137  2F5C  005F        JMP EDYN
      027536  037532  2F5E  3F5A
3833  027540          2F60              RUBAY2 DSH 0
3834  027540  104014  2F60  880C        EMT VC0007-EMTTBL
3835  027542  000767  2F62  01F7        BR ALT3
3836  027544  012737  2F64  15DF       RUBB MOV =INPUT,INPB1          PRINT ALL OVER
      027546  054514  2F66  594C
      027550  053406  2F68  5706
3837  027552  104036  2F6A  881E        EMT VC0016-EMTTBL
3838  027554  104416  2F6C  890E        TRAP VB0027-TRPTBL
3839  027556          2F6E              KEYBCH DSH 0
3840  027556          2F6E              WALK2 DSH 0
3841  027556  005737  2F6E  0BDF        TST HBECHO
      027560  053014  2F70  560C
3842  027562  001410  2F72  0308        BEQ WALK2A        A TYPE HAS OCCURRED SINCE LAST CALL
3843  027564  005737  2F74  0BDF        TST DINGER
      027566  052414  2F76  550C
3844  027570  001405  2F78  0305        BEQ WALK2A
3845  027572  012700  2F7A  15C0        MOV =X'0007',R0
      027574  000007  2F7C  0007
3846  027576  104014  2F7E  880C        EMT VC0007-EMTTBL
3847  027600  005037  2F80  0A1F        CLR MODE2A  DONT ADD INSULT TO INJURY
      027602  052514  2F82  554C
3848  027604          2F84              WALK2A DSH 0
3849  027604  005737  2F84  0BDF        TST MODE2A
      027606  052514  2F86  554C
3850  027610  001004  2F88  0204        BNE MODLA1
3851  027612  013737  2F8A  17DF        MOV MODE2B,MODE2A
      027614  052426  2F8C  5516
      027616  052514  2F8E  554C
3852  027620  000406  2F90  0106        BR MODLA2
3853  027622  005337  2F92  0ADF       MODLA1 DEC MODE2A
      027624  052514  2F94  554C
3854  027626  012700  2F96  15C0        MOV =E'>',K
      027630  000076  2F98  003E
3855  027632  000137  2F9A  005F        JMP KEYCOM
      027634  030342  2F9C  30E2
3856  027636          2F9E              MODLA2 DSH 0
3857  027636  012737  2F9E  15DF        MOV =1,HBECHO
      027640  000001  2FA0  0001
      027642  053014  2FA2  560C
3858  027644          2FA4              PREV10 DSH 0
3859  027644  005737  2FA4  0BDF        TST SIMULA
      027646  052376  2FA6  54FE
3860  027650  001574  2FA8  037C        BEQ SIMINE
```

```
3861  027652  005737  2FAA  0BDF          TST TKS
      027654  177560  2FAC  FF70
3862  027656  100423  2FAE  8113          BMI WALKZZ
3863  027660  005737  2FB0  0BDF          TST PARIND
      027662  052402  2FB2  5502
3864  027664  001413  2FB4  030B          BEQ PARINQ
3865  027666  005237  2FB6  0A9F          INC PRS
      027670  177550  2FB8  FF68
3866  027672  105737  2FBA  8BDF         PARINC TSTB TKS
      027674  177560  2FBC  FF70
3867  027676  100413  2FBE  810B          BMI PARINE
3868  027700  105737  2FC0  8BDF          TSTB PRS
      027702  177550  2FC2  FF68
3869  027704  100372  2FC4  80FA          BPL PARINC
3870  027706  113700  2FC6  97C0          MOVB PRB,R0
      027710  177552  2FC8  FF6A
3871  027712  000560  2FCA  0170          BR PARCOM
3872  027714  005237  2FCC  0A9F         PARINQ INC TKS
      027716  177560  2FCE  FF70
3873  027720  105737  2FD0  8BDF         PARINB TSTB TKS
      027722  177560  2FD2  FF70
3874  027724  100375  2FD4  80FD          BPL PARINB
3875  027726          2FD6                WALKZZ DSH 0
3876  027726  113700  2FD6  97C0          PARINE MOVB TKB,R0
**********************************************
3937  030254  042700  30AC  45C0         PARCOM BIC =O'177600',R0
      030256  177600  30AE  FF80
3938  030260  005737  30B0  0BDF          TST ASYCHR
      030262  053174  30B2  567C
3939  030264  001411  30B4  0309          BEQ ASYYY
3940  030266  005037  30B6  0A1F          CLR ASYCHR
      030270  053174  30B8  567C
3941  030272  022700  30BA  25C0          CMP =O'007',R0
      030274  000007  30BC  0007
3942  030276  001002  30BE  0202          BNE MODLE2
3943  030300  000137  30C0  005F          JMP SSTRT
      030302  035964  30C2  3B34
3944  030304  000137  30C4  005F         MODLE2 JMP MODLA2
      030306  027636  30C6  2F9E
3945  030310          30C8                ASYYY DSH 0
3946  030310  005700  30C8  08C0          TST R0
3947  030312  001774  30CA  03FC          BEQ MODLE2
3948  030314  022700  30CC  25C0          CMP =O'177',R0
      030316  000177  30CE  007F
3949  030320  001771  30D0  03F9          BEQ MODLE2
3950  030322  022700  30D2  25C0          CMP =O'12',R0
      030324  000012  30D4  000A
3951  030326  001766  30D6  03F6          BEQ MODLE2
3952                                      *
3953                                      *   SEE IF ECHO ONTO HSP
3954                                      *
3955  030330  005737  30D8  0BDF          TST FASTXT
      030332  052404  30DA  5504
3956  030334  001002  30DC  0202          BNE KEYB1
3957  030336  004737  30DE  09DF          JSR PC,HSPSUB
      030340  031174  30E0  327C
3958  030342          30E2                KEYB1 DSH 0
3959  030342          30E2                KEYCOM DSH 0
3960  030342  022700  30E2  25C0          CMP =O'015',R0
      030344  000015  30E4  000D
```

```
3961  030346  001002   30E6   0202          BNE NBS
3962  030350  012700   30E8   15C0          MOV =E':',R0
      030352  000072   30EA   003A
3963  030354           30EC                 NBS DSH 0
3964  030354  032700   30EC   35C0          BIT =O'140',R0    SEE IF CONTROL CHARACTER
      030356  000140   30EE   0060
3965  030360  001004   30F0   0204          BNE NBS123
3966  030362  005037   30F2   0A1F          CLR HBECHO    NEVER DING A CONTROL CHARACTER
      030364  053014   30F4   560C
3967  030366  005037   30F6   0A1F          CLR MODE2A
      030370  052514   30F8   554C
3968  030372           30FA                 NBS123 DSH 0
3969  030372  022700   30FA   25C0          CMP =O'003',R0     CONTROL C; TURN ON TABLE TRACE
      030374  000003   30FC   0003
3970  030376  001005   30FE   0205          BNE TBLRS1
3971  030400  012737   3100   15DF          MOV =63,TBLTRG
      030402  000077   3102   003F
      030404  052410   3104   5508
3972  030406  000137   3106   005F          TBLRS3 JMP KEYBCH
      030410  027556   3108   2F6E
3973  030412  022700   310A   25C0          TBLRS1 CMP =O'030',R0    CONTROL X; TURN OFF TABLE TRACE
      030414  000030   310C   0018
3974  030416  001003   310E   0203          BNE TBLRS2
3975  030420  005037   3110   0A1F          CLR TBLTRG
      030422  052410   3112   5508
3976  030424  000770   3114   01F8          BR TBLRS3
3977  030426           3116                 TBLRS2 DSH 0
3978  030426  022700   3116   25C0          CMP =O'004',R0        CONTROL D; TURN ON PAPER TAPE READER
      030430  000004   3118   0004
3979  030432  001004   311A   0204          BNE COMD
3980  030434  012737   311C   15DF          MOV =63,PARIND
      030436  000077   311E   003F
      030440  052402   3120   5502
3981  030442  000761   3122   01F1          BR TBLRS3
3982  030444           3124                 COMD DSH 0
3983  030444  022700   3124   25C0          CMP =O'005',R0    CONTROL E; TURN OFF PAPER TAPE READER
      030446  000005   3126   0005
3984  030450  001010   3128   0208          BNE COME
3985  030452  005037   312A   0A1F          CLR PARIND
      030454  052402   312C   5502
3986  030456  005737   312E   0BDF          TST SIMULA
      030460  052376   3130   54FE
3987  030462  001751   3132   03E9          BEQ TBLRS3
3988  030464  005037   3134   0A1F          CLR PRS
      030466  177550   3136   FF68
3989  030470  000746   3138   01E6          BR TBLRS3
3990  030472           313A                 COME DSH 0
3991  030472  022700   313A   25C0          CMP =O'010',R0    CONTROL H; TURN ON PUNCH; ECHO INPUT
      030474  000010   313C   0008
3992  030476  001004   313E   0204          BNE COMH
3993  030500  012737   3140   15DF          MOV =63,HSPDAT
      030502  000077   3142   003F
      030504  052400   3144   5500
3994  030506  000737   3146   01DF          BR TBLRS3
3995  030510           3148                 COMH DSH 0
3996  030510  022700   3148   25C0          CMP =O'011',R0     CONTOL I; TURN OFF PUNCH; ECHO INPUT
      030512  000011   314A   0009
3997  030514  001003   314C   0203          BNE COMI
3998  030516  005037   314E   0A1F          CLR HSPDAT
      030520  052400   3150   5500
```

```
3999  030522  000731    3152    01D9          KEYBCC BR TBLRS3
4000  030524            3154                  COMI DSH 0
4001  030524  022700    3154    25C0           CMP =0'017',R0      CONTROL O; TURN ON PUNCH; ECHO OUTPUT
      030526  000017    3156    000F
4002  030530  001004    3158    0204           BNE COMO
4003  030532  012737    315A    15DF           MOV =63,FASTXT
      030534  000077    315C    003F
      030536  052404    315E    5504
4004  030540  000770    3160    01F8           BR KEYBCC
4005  030542            3162                  COMO DSH 0
4006  030542  022700    3162    25C0           CMP =0'020',R0    CONTROL P; TURN OFF PUNCH; ECHO OUTPUT
      030544  000020    3164    0010
4007  030546  001003    3166    0203           BNE COMP
4008  030550  005037    3168    0A1F           CLR FASTXT
      030552  052404    316A    5504
4009  030554  000762    316C    01F2           BR KEYBCC
4010  030556            316E                  COMP DSH 0
4011  030556  022700    316E    25C0           CMP =0'025',R0  CONTROL U; TURN ON PROUT8
      030560  000025    3170    0015
4012  030562  001004    3172    0204           BNE COMU
4013  030564  012737    3174    15DF           MOV =63,PROUT8
      030566  000077    3176    003F
      030570  052406    3178    5506
4014  030572  000753    317A    01EB           BR KEYBCC
4015  030574            317C                  COMU DSH 0
4016  030574  022700    317C    25C0           CMP =0'026',R0  CONTROL V; TURN OFF PROUT8
      030576  000026    317E    0016
4017  030600  001003    3180    0203           BNE COMV
4018  030602  005037    3182    0A1F           CLR PROUT8
      030604  052406    3184    5506
4019  030606  000745    3186    01E5           BR KEYBCC
4020  030610            3188                  COMV DSH 0
4021  030610  022700    3188    25C0           CMP =0'001',K
      030612  000001    318A    0001
4022  030614  001007    318C    0207           BNE MODLB
4023  030616  162700    318E    E5C0          MODLA SUB =0'001',K
      030620  000001    3190    0001
4024  030622  010037    3192    101F           MOV K,MODE2B
      030624  052426    3194    5516
4025  030626  005037    3196    0A1F           CLR MODE2A
      030630  052514    3198    554C
4026  030632  000733    319A    01DB           BR KEYBCC
4027  030634  022700    319C    25C0          MODLB CMP =0'002',K
      030636  000002    319E    0002
4028  030640  001766    31A0    03F6           BEQ MODLA
4029  030642  022700    31A2    25C0           CMP =0'022',R0      CONTROL R
      030644  000022    31A4    0012
4030  030646  001002    31A6    0202           BNE ECH1
4031  030650  000137    31A8    005F           JMP CLRBLL
      030652  052370    31AA    54F8
4032  030654            31AC                  ECH1 DSH 0
4033  030654  022700    31AC    25C0           CMP =0'007',R0      CONTROL G
      030656  000007    31AE    0007
4034                                          * CONTROL G SHOULD WORK UNDER INTERRUPT
4035  030660  001130    31B0    0258           BNE TYPA2
4036  030662  000137    31B2    005F           JMP SSTRT
      030664  035464    31B4    3B34
4037  030666            31B6                  TYPECH DSH 0
4038  030666  005737    31B6    0BDF           TST SIMULA
      030670  052376    31B8    54FE
```

```
4039  030672  001412   31BA  030A        BEQ TYPECZ
4040  030674  005737   31BC  0BDF        TST OPTION
      030676  052502   31BE  5542
4041  030700  001407   31C0  0307        BEQ TYPECZ
4042  030702  105737   31C2  8BDF        TSTB TKS
      030704  177560   31C4  FF70
4043  030706  100004   31C6  8004        BPL TYPECZ
4044  030710  012737   31C8  15DF        MOV =1,ASYCHR
      030712  000001   31CA  0001
      030714  053174   31CC  567C
4045  030716  000511   31CE  0149        BR TYPA2
4046  030720           31D0              TYPECZ DSH 0
4047  030720  006237   31D0  0C9F        ASR HBECHO      HAS BEEN ECHOED
      030722  053014   31D2  560C
4048  030724  022700   31D4  25C0        CMP =0'15',R0
      030726  000015   31D6  000D
4049  030730  001004   31D8  0204        BNE TYPAL1
4050  030732  022737   31DA  25DF        CMP =1,COLKNT
      030734  000001   31DC  0001
      030736  055336   31DE  5ADE
4051  030740  001444   31E0  0324        BEQ TYPAL2        DONT TYPE
4052  030742  022700   31E2  25C0        TYPAL1 CMP =0'12',R0
      030744  000012   31E4  000A
4053  030746  001006   31E6  0206        BNE TYPAL3
4054  030750  005737   31E8  0BDF        TST SKPLF
      030752  055440   31EA  5B20
4055  030754  001403   31EC  0303        BEQ TYPAL3
4056  030756  005037   31EE  0A1F        CLR SKPLF
      030760  055440   31F0  5B20
4057  030762  000433   31F2  011B        BR TYPAL2       DONT TYPE
4058  030764           31F4              TYPAL3 DSH 0
4059  030764  005737   31F4  0BDF        TST SIMULA
      030766  052376   31F6  54FE
4060  030770  001427   31F8  0317        BEQ SIMING
4061  030772  005737   31FA  0BDF        TST PAROP
      030774  052466   31FC  5536
4062  030776  001016   31FE  020E        BNE PAROPA
4063  031000  105737   3200  8BDF        PAROPB TSTB TPS
      031002  177564   3202  FF74
4064  031004  100375   3204  80FD        BPL PAROPB
4065  031006  110037   3206  901F        MOVB R0,TPB
      031010  177566   3208  FF76
4066  031012  005737   320A  0BDF        TST FASTXT
      031014  052404   320C  5504
4067  031016  001413   320E  030B        BEQ PAROPC
4068  031020  022700   3210  25C0        CMP =0'007',R0
      031022  000007   3212  0007
4069  031024  001410   3214  0308        BEQ PAROPC
4070  031026  004737   3216  09DF        JSR PC,HSPAAA
      031030  031262   3218  32B2
4071  031032  000405   321A  0105        BR PAROPC
4072  031034  105737   321C  8BDF        PAROPA TSTB PPS
      031036  177554   321E  FF6C
4073  031040  100375   3220  80FD        BPL PAROPA
4074  031042  110037   3222  901F        MOVB R0,PPB
      031044  177556   3224  FF6E
4075  031046           3226              PAROPC DSH 0
4076  031046  000401   3226  0101        BR TYPAL2
4077  031050           3228              SIMING DSH 0
4078                                      TRON
```

```
4079  031050  010000    3228    1000          MOV R0,R0
4080                                          TROFF
4081  031052            322A                TYPAL2 DSH 0
4082  031052  022700    322A    25C0        TYPCOM CMP =0'15',R0
      031054  000015    322C    000D
4083  031056  001020    322E    0210          BNE TYPA1
4084  031060  022737    3230    25DF          CMP =60,COLKNT
      031062  000074    3232    003C
      031064  055336    3234    5ADE
4085  031066  100002    3236    8002          BPL NEWA1
4086  031070  005000    3238    0A00          CLR R0
4087  031072  104014    323A    880C          EMT VC0007-EMTTBL
4088  031074  022737    323C    25DF        NEWA1 CMP =30,COLKNT
      031076  000036    323E    001E
      031100  055336    3240    5ADE
4089  031102  100002    3242    8002          BPL NEWA2
4090  031104  005000    3244    0A00          CLR R0
4091  031106  104014    3246    880C          EMT VC0007-EMTTBL
4092  031110            3248                NEWA2 DSH 0
4093  031110  012737    3248    15DF          MOV =1,COLKNT
      031112  000001    324A    0001
      031114  055336    324C    5ADE
4094  031116  000207    324E    0087          RTS PC
4095  031120            3250                TYPA1 DSH 0
4096  031120  032700    3250    35C0        TYPX1 BIT =0'140',R0
      031122  000140    3252    0060
4097  031124  001406    3254    0306          BEQ TYPA2
4098  031126  005237    3256    0A9F          INC COLKNT
      031130  055336    3258    5ADE
4099  031132  023727    325A    27D7          CMP COLKNT,=72
      031134  055336    325C    5ADE
      031136  000110    325E    0048
4100  031140  100007    3260    8007          BPL TYPAX
4101  031142  000207    3262    0087        TYPA2 RTS PC
4102  031144  012700    3264    15C0        NEWLIN MOV =0'15',R0
      031146  000015    3266    000D
4103  031150  104014    3268    880C          EMT VC0007-EMTTBL
4104  031152  012700    326A    15C0          MOV =0'12',R0
      031154  000012    326C    000A
4105                                          * JSUBR PC,TYPECH
4106                                          * RTS PC
4107  031156  000643    326E    01A3          BR TYPECH
4108  031160            3270                TYPAX DSH 0
4109  031160  104010    3270    8808          EMT VC0005-EMTTBL
4110  031162  012701    3272    15C1          MOV =TYPMES,R1
      031164  010165    3274    1075
4111  031166  104022    3276    8812          EMT VC0010-EMTTBL
4112  031170  104040    3278    8820          EMT VC0017-EMTTBL      ABORT TABLE
4113  031172  104404    327A    8904          TRAP V80005-TRPTBL
4114  031174  005737    327C    0BDF        HSPSUB TST SIMULA
      031176  052376    327E    54FE
4115  031200  001442    3280    0322          BEQ HSPBBB
4116  031202  022700    3282    25C0          CMP =0'010',R0       CONTROL H
      031204  000010    3284    0008
4117  031206  001437    3286    031F          BEQ HSPBBB
4118  031210  022700    3288    25C0          CMP =0'011',R0      CONTROL I
      031212  000011    328A    0009
4119  031214  001434    328C    031C          BEQ HSPBBB
4120  031216  022700    328E    25C0          CMP =0'017',R0           CONTROL O
      031220  000017    3290    000F
```

```
4121  031222  001431    3292    0319          BEQ HSPBBB
4122  031224  022700    3294    25C0          CMP =O'020',R0        CONTROL P
      031226  000020    3296    0010
4123  031230  001426    3298    0316          BEQ HSPBBB
4124  031232  022700    329A    25C0          CMP =O'004',R0    CONTROL D
      031234  000004    329C    0004
4125  031236  001423    329E    0313          BEQ HSPBBB
4126  031240  022700    32A0    25C0          CMP =O'005',R0    CONTROL E
      031242  000005    32A2    0005
4127  031244  001420    32A4    0310          BEQ HSPBBB
4128  031246  022700    32A6    25C0          CMP =O'003',R0     CONTROL C
      031250  000003    32A8    0003
4129  031252  001415    32AA    030D          BEQ HSPBBB
4130  031254  022700    32AC    25C0          CMP =O'030',R0    CONTROL X
      031256  000030    32AE    0018
4131  031260  001412    32B0    030A          BEQ HSPBBB
4132  031262  005737    32B2    0BDF        HSPAAA TST HSPDAT
      031264  052400    32B4    5500
4133  031266  001407    32B6    0307          BEQ HSPBBB
4134  031270  032737    32B8    35DF          BIT =X'8080',PPS
      031272  100200    32BA    8080
      031274  177554    32BC    FF6C
4135  031276  001771    32BE    03F9          BEQ HSPAAA
4136  031300  100402    32C0    8102          BMI HSPBBB   ERROR
4137  031302  110037    32C2    901F          MOVB R0,PPB       ECHO THE INPUT
      031304  177556    32C4    FF6E
4138  031306            32C6                HSPBBB DSH 0
4139  031306  000207    32C6    0087          RTS PC
4140  031310            32C8                LADDER DSH 0
4141  031310  011604    32C8    1384          MOV (R6),R4
4142  031312  005737    32CA    0BDF          TST FORIND
      031314  053330    32CC    56D8
4143  031316  001405    32CE    0305          BEQ LADLAB
4144  031320  005737    32D0    0BDF          TST FORPER
      031322  053332    32D2    56DA
4145  031324  001414    32D4    030C          BEQ LADOUT
4146  031326  005037    32D6    0A1F          CLR FORPER
      031330  053332    32D8    56DA
4147  031332            32DA                LADLAB DSH 0
4148  031332  112401    32DA    9501          MOVB (R4)+,R1
4149  031334  022701    32DC    25C1          CMP =E':',R1
      031336  000072    32DE    003A
4150  031340  001417    32E0    030F          BEQ LAD1
4151  031342  112437    32E2    951F          MOVB (R4)+,LAD22A
      031344  052604    32E4    5584
4152  031346  020100    32E6    2040          CMP R1,R0
4153  031350  001005    32E8    0205          BNE LAD2
4154  031352  104054    32EA    882C          EMT VC0024-EMTTBL
4155  031354  000766    32EC    01F6          BR LADLAB
4156  031356  005204    32EE    0A84        LADOUT INC R4
4157  031360  112437    32F0    951F          MOVB (R4)+,LAD22A
      031362  052604    32F2    5584
4158  031364  062706    32F4    65C6        LAD2 ADD =4,R6
      031366  000004    32F6    0004
4159  031370  113701    32F8    97C1          MOVB LAD22A,R1
      031372  052604    32FA    5584
4160  031374  016107    32FC    1C47          MOV LADTAB(R1),PC
      031376  031406    32FE    3306
4161  031400  005204    3300    0A84        LAD1 INC R4
4162  031402  010416    3302    110E          MOV R4,(R6)
```

```
4163  031404  000002    3304   0002          RTI
4164  031406            3306                 LADTAB DSH 0
4165  031406  013600    3306   1780          PDB6   DCH B6
4166  031410  013626    3308   1796          PDB74  DCH B74
4167  031412  032520    330A   3550          PDC1A  DCH C1A
4168  031414  033036    330C   361E          PDCZ1  DCH CZ1
4169  031416  033400    330E   3700          PDRST8 DCH REST88
4170  031420  033666    3310   37B6          PDA11  DCH A11
4171  031422  035352    3312   3AEA          PDEXX  DCH EXX
4172  031424  033646    3314   37A6          PDSHGO DCH SHGO
4173  031426  034212    3316   388A          PDB11  DCH B11
4174  031430  035106    3318   3A46          PDJ11  DCH J11
4175  031432  034252    331A   38AA          PDC11  DCH C11
4176  031434  034372    331C   38FA          PDD11  DCH D11
4177  031436  034312    331E   38CA          PDE11  DCH E11
4178  031440  034616    3320   398E          PDG11  DCH G11
4179  031442  034430    3322   3918          PDD51  DCH D51
4180  031444  034560    3324   3970          PDD51A DCH D51AAA
4181  031446  035464    3326   3B34          PDSTRT DCH SSTRT
4182  031450  034652    3328   39AA          PDH11  DCH H11
4183  031452  035170    332A   3A78          PDK11H DCH K11FGH
4184  031454  035076    332C   3A3E          PDH51  DCH H51
4185  031456  035156    332E   3A6E          PDJ51  DCH J51
4186  031460  035224    3330   3A94          PDK11Z DCH K11FGZ
4187  031462  035352    3332   3AEA          PDN11  DCH N11
4188  031464  035340    3334   3AE0          PDJ11A DCH J11A
4189  031466  012256    3336   149E          PDSTSC DCH STSC
4190  031470  034342    3338   38E2          PDSUSZ DCH SUSZ
4191  031472  032636    333A   359E          PDASY1 DCH ASY1
4192  031474  034750    333C   39E8          PDCLR  DCH CLR
4193  031476  035044    333E   3A24          PDCONT DCH CONT
4194  031500  033222    3340   3692          PDFORB DCH FORB
4195                                          * COULD COUNT UP TO 31; THEN WOULD NOT NEED CHECK IN LHOP
4196  031502            3342                 QSTRNG DSH 0
4197  031502  104054    3342   882C           EMT VC0024-EMTTBL
4198  031504  032700    3344   35C0           BIT =O'140',R0     CONTROL CHARACTER
      031506  000140    3346   0060
4199  031510  001403    3348   0303           BEQ QRSTA
4200  031512  022700    334A   25C0           CMP =E':',R0
      031514  000072    334C   003A
4201  031516  001001    334E   0201           BNE QRST
4202  031520  104406    3350   8906          QRSTA TRAP VB0011-TRPTBL
4203  031522            3352                 QRST DSH 0
4204  031522  022700    3352   25C0           CMP =O'47',K               QUOTE
      031524  000047    3354   0027
4205  031526  001365    3356   02F5           BNE QSTRNG
4206  031530  104054    3358   882C           EMT VC0024-EMTTBL
4207  031532  032700    335A   35C0           BIT =O'140',R0   CONTROL CHARACTER
      031534  000140    335C   0060
4208  031536  001770    335E   03F8           BEQ QRSTA
4209  031540  022700    3360   25C0           CMP =O'47',K               QUOTE
      031542  000047    3362   0027
4210  031544  001035    3364   021D           BNE QEND
4211  031546  000003    3366   0003           DEBUG  STL SUBR
4212  031550  177652    3368   FFAA           DCH CHAAEX              INDICATE 2ND QUOTE
4213  031552  000753    336A   01EB           BR QSTRNG
4214  031554  110037    336C   901F          VARR MOVB R0,VARRNM
      031556  053004    336E   5604
4215  031560  105037    3370   8A1F           CLRB VARRNM+1
      031562  053005    3372   5605
```

```
4216  031564  104074    3374    883C          EMT VC0034-EMTTBL
4217  031566  001025    3376    0215          BNE STSK3
4218  031570  104054    3378    882C          EMT VC0024-EMTTBL
4219  031572  022700    337A    25C0          CMP =E'(',K
      031574  000050    337C    0028
4220  031576  001022    337E    0212          BNE A15                 NOT A (
4221  031600  000003    3380    0003         A34 DEBUG  STL SUBR
4222  031602  177701    3382    FFC1          DCH CHC1FX
4223  031604  104054    3384    882C          EMT VC0024-EMTTBL
4224  031606  104056    3386    882E          EMT VC0025-EMTTBL
4225  031610  022700    3388    25C0          CMP =E')',K
      031612  000051    338A    0029
4226  031614  001440    338C    0320          BEQ A24A
4227  031616  022700    338E    25C0          CMP =E',',K
      031620  000054    3390    002C
4228  031622  001007    3392    0207          BNE STSK3
4229  031624  104054    3394    882C          EMT VC0024-EMTTBL
4230  031626  104056    3396    882E          EMT VC0025-EMTTBL
4231  031630  022700    3398    25C0          CMP =E')',K
      031632  000051    339A    0029
4232  031634  001002    339C    0202          BNE STSK3
4233  031636           339E                   A24 DSH 0
4234  031636  104054    339E    882C          EMT VC0024-EMTTBL
4235  031640           33A0                   QEND DSH 0
4236  031640  000207    33A0    0087          A21 RTS PC
4237  031642           33A2                   STSK3 DSH 0
4238  031642  104406    33A2    8906          TRAP VB0011-TRPTBL
4239  031644           33A4                   A15 DSH 0                TEST 2ND
4240  031644  104060    33A4    8830          EMT VC0027-EMTTBL
4241  031646  001374    33A6    02FC          BNE A21
4242  031650  110037    33A8    901F          MOVB R0,VARRNM+1
      031652  053005    33AA    5605
4243  031654  104054    33AC    882C          EMT VC0024-EMTTBL
4244  031656  022700    33AE    25C0          CMP =E'(',K
      031660  000050    33B0    0028
4245  031662  001746    33B2    03E6          BEQ A34
4246  031664  104060    33B4    8830          EMT VC0027-EMTTBL
4247  031666  001364    33B6    02F4          BNE A21
4248  031670  110037    33B8    901F          MOVB R0,VARRNM+2
      031672  053006    33BA    5606
4249  031674  104054    33BC    882C          EMT VC0024-EMTTBL
4250  031676  104060    33BE    8830          EMT VC0027-EMTTBL
4251  031700  001357    33C0    02EF          BNE A21
4252  031702  110037    33C2    901F          MOVB R0,VARRNM+3
      031704  053007    33C4    5607
4253  031706  022727    33C6    25DF          CMP =E'ET',VARRNM+2
      031710  052105    33C8    5445
      031712  053006    33CA    5606
4254  031714  001350    33CC    02E8          BNE A24
4255  031716  022737    33CE    25DF         A24A CMP =E'ZR',VARRNM
      031720  051132    33D0    525A
      031722  053004    33D2    5604
4256  031724  001344    33D4    02E4          BNE A24
4257  031726  000745    33D6    01E5          BR STSK3
*****************************************
4358  032336           34DE                   COSTAT DSH 0   GETS A COMPLETE LINE FROM THE KEYBOARD
4359  032336  013737    34DE    17DF          MOV L,GSAVL     SAVE L
      032340  053324    34E0    56C4
      032342  056354    34E2    5AEC
4360  032344  013737    34E4    17DF          MOV SL,GSAVSL  SAVE SL
```

```
                032346  053326  34C6    56D6
                032350  055356  34C8    5AEE
4361
4362            032352  020027  34CA    2017        ********* IF ! TO &   SEE IF CAN EXPAND INTO INPUT
                                                     CMP K,=0'41'
                032354  000041  34CC    0021
4363            032356  100437  34CE    811F        BMI ZZA001
4364            032360  022700  34D0    25C0        CMP =0'46',K
                032362  000046  34D2    0026
4365            032364  100434  34D4    811C        BMI ZZA001
4366            032366  105760  34D6    88F0        TSTB ASYIND-0'41'(R0)
                032370  052445  34D8    5525
4367            032372  001517  34DA    034F        BEQ ASYNIX
4368            032374  012702  34DC    15C2        MOV =SEQ3,R2
                032376  052570  34DE    5578
4369            032400  112722  3500    95D2        MOVB =E'0',(R2)+
                032402  000060  3502    0030
4370            032404  112722  3504    95D2        MOVB =E'9',(R2)+
                032406  000071  3506    0039
4371            032410  062700  3508    65C0        ADD =0'20',R0
                032412  000020  350A    0010
4372            032414  110012  350C    900A        MOVB R0,(R2)
4373            032416  104250  350E    88A8        EMT VC0144-EMTTBL
4374            032420  013704  3510    17C4        MOV INPA1,R4
                032422  053404  3512    5704
4375            032424  005304  3514    0AC4        DEC R4
4376            032426  062701  3516    65C1        ADD =19,R1
                032430  000023  3518    0013
4377            032432  111100  351A    9240        MOVB (R1),R0
4378            032434  112103  351C    9443        ZZA002 MOVB (R1)+,R3
4379            032436  100776  351E    81FE        BMI ZZA002
4380            032440  022703  3520    25C3        CMP =E':',R3.
                032442  000072  3522    003A
4381            032444  001402  3524    0302        BEQ ZZA003
4382            032446  110324  3526    90D4        MOVB R3,(R4)+
4383            032450  000771  3528    01F9        BR ZZA002
4384            032452          352A                ZZA003 DSH 0
4385            032452  010437  352A    111F        MOV R4,INPA1
                032454  053404  352C    5704
4386            032456          352E                ZZA001 DSH 0
4387            032456  005737  352E    08DF        TST MAJMOD              IS IT TO BE STORED
                032460  053340  3530    56E0
4388            032462  001016  3532    020E        BNE C1A    EDIT CANNOT BE STORED
4389            032464  013702  3534    17C2        MOV SL,R2
                032466  053326  3536    56D6
4390            032470  112722  3538    95D2        MOVB =X'ED',(R2)+
                032472  000355  353A    00ED
4391            032474  010237  353C    109F        MOV R2,SL
                032476  053326  353E    56D6
4392            032500  000004  3540    0004        IOT
4393            032502  000105  3542    0045        DCB E'E',PDC1A-LADTAB
                032503  000004  3543    0004
4394            032504  000104  3544    0044        DCB E'D',PDC1A-LADTAB
                032505  000004  3545    0004
4395            032506  000111  3546    0049        DCB E'I',PDC1A-LADTAB
                032507  000004  3547    0004
4396            032510  000124  3548    0054        DCB E'T',PDC1A-LADTAB
                032511  000004  3549    0004
4397            032512  000040  354A    0020        DCB E' ',PDC1A-LADTAB
                032513  000004  354B    0004
4398            032514  000072  354C    003A        DCB E':'
```

```
4399                                        * JSUBR PC,AREX
4400                                        * RTS PC
4401  032516  104420    354E    8910        TRAP V60031-TRPTBL
4402  032520            3550                C1A DSH 0                    GO BACK
4403  032520  104156    3550    886E        EMT VC0076-EMTTBL
4404                                        * LOOK FOR ASYNCH
4405  032522  005737    3552    0BDF        TST MAJMOD       MUST BE IMMEDIATE BUT WILL BE STORED
      032524  053340    3554    56E0
4406  032526  001043    3556    0223        BNE ASY1
4407  032530  013702    3558    17C2        MOV SL,R2
      032532  053326    355A    56D6
4408  032534  112722    355C    95D2        MOVB =X'EE',(R2)+
      032536  000356    355E    00EE
4409  032540  010237    3560    109F        MOV R2,SL
      032542  053326    3562    56D6
4410  032544  000004    3564    0004        IOT
4411  032546  000101    3566    0041        DCB E'A',PDASY1-LADTAB
      032547  000064    3567    0034
4412  032550  000123    3568    0053        DCB E'S',PDASY1-LADTAB
      032551  000064    3569    0034
4413  032552  000131    356A    0059        DCB E'Y',PDASY1-LADTAB
      032553  000064    356B    0034
4414  032554  000116    356C    004E        DCB E'N',PDASY1-LADTAB
      032555  000064    356D    0034
4415  032556  000103    356E    0043        DCB E'C',PDASY1-LADTAB
      032557  000064    356F    0034
4416  032560  000110    3570    0048        DCB E'H',PDSTSC-LADTAB
      032561  000060    3571    0030
4417  032562  000040    3572    0020        DCB E' ',PDSTSC-LADTAB
      032563  000060    3573    0030
4418  032564  000072    3574    003A        DCH E':'
4419  032566  005237    3576    0A9F        INC ASYDIG       IT IS AN ASYNCH STATEMENT
      032570  053350    3578    5AE8
4420  032572  020027    357A    2017        CMP K,=0'41'
      032574  000041    357C    0021
4421  032576  100415    357E    810D        BMI ASYNIX
4422  032600  022700    3580    25C0        CMP =0'46',K
      032602  000046    3582    0026
4423  032604  100412    3584    810A        BMI ASYNIX
4424  032606  104054    3586    882C        EMT VC0024-EMTTBL
4425  032610  022700    3588    25C0        CMP =E':',K
      032612  000072    358A    003A
4426  032614  001405    358C    0305        BEQ ASYDEL
4427  032616  022700    358E    25C0        CMP =E' ',K
      032620  000040    3590    0020
4428  032622  001003    3592    0203        BNE ASYNIX
4429  032624  104054    3594    882C        EMT VC0024-EMTTBL
4430  032626  000404    3596    0104        BR NEWSUB
4431  032630  000207    3598    0087        ASYDEL RTS PC
4432  032632  000137    359A    005F        ASYNIX JMP STSC          NO GO, POGO
      032634  012236    359C    149E
4433  032636  104156    359E    886E        ASY1 EMT VC0076-EMTTBL       RESTORE POINTERS
4434                                        * JSUBR PC,NEWSUB
4435                                        * RTS PC
4436  032640            35A0                NEWSUB DSH 0
4437  032640  005037    35A0    0A1F        CLR FORIND
      032642  053330    35A2    56D8
4438  032644  005037    35A4    0A1F        CLR FORPER
      032646  053332    35A6    56DA
4439  032650  005037    35A8    0A1F        CLR ELSFOR
```

```
                032652  053334  35AA    56DC
4440            032654  005037  35AC    0A1F    NEW8 CLR INIFCH
                032656  055344  35AE    5AE4
4441            032660  005037  35B0    0A1F     CLR ELSECH
                032662  055342  35B2    5AE2
4442            032664  013737  35B4    17DF    NEW1 MOV L,GSAVL
                032666  053324  35B6    56D4
                032670  055354  35B8    5AEC
4443            032672  013737  35BA    17DF     MOV SL,GSAVSL
                032674  053326  35BC    56D6
                032676  055356  35BE    5AEE
4444            032700  005737  35C0    0BDF     TST INIFCH
                032702  055344  35C2    5AE4
4445            032704  001055  35C4    022D     BNE NEW2
4446            032706  013702  35C6    17C2     MOV SL,R2
                032710  053326  35C8    56D6
4447            032712  112722  35CA    95D2     MOVB =X'EF',(R2)+
                032714  000357  35CC    00EF
4448            032716  010237  35CE    109F     MOV R2,SL
                032720  053326  35D0    56D6
4449            032722  005237  35D2    0A9F     INC FORPER
                032724  053332  35D4    56DA
4450            032726  000004  35D6    0004     IOT
4451            032730  000111  35D8    0049     DCB E'I',PDCZ1-LADTAB
                032731  000006  35D9    0006
4452            032732  000106  35DA    0046     DCB E'F',PDCZ1-LADTAB
                032733  000006  35DB    0006
4453            032734  000040  35DC    0020     DCB E' ',PDCZ1-LADTAB
                032735  000006  35DD    0006
4454            032736  000072  35DE    003A     DCH E':'
4455            032740  005237  35E0    0A9F     INC INIFCH
                032742  055344  35E2    5AE4
4456            032744  104066  35E4    8836     EMT VC0031-EMTTBL
4457            032746  000003  35E6    0003     DEBUG  STL SUBR
4458            032750  177732  35E8    FFDA     DCH CHDAEX
4459                                             *               IF B THEN           ELSE
4460                                             *                   DA            DA    A2
4461                                             *               IF B THEN
4462                                             *                   DA
4463                                             *               IF B
4464                                             *                   DA
4465            032752  022700  35EA    25C0     CMP =E' ',K
                032754  000040  35EC    0020
4466            032756  001026  35EE    0216     BNE STSKXY
4467            032760  104054  35F0    882C     EMT VC0024-EMTTBL
4468            032762  013737  35F2    17DF     MOV L,GSAVL        STORE L AGAIN
                032764  053324  35F4    56D4
                032766  055354  35F6    5AEC
4469            032770  013737  35F8    17DF     MOV SL,GSAVSL  DITTO SL
                032772  053326  35FA    56D6
                032774  055356  35FC    5AEE
4470            032776  005237  35FE    0A9F     INC FORPER
                033000  053332  3600    56DA
4471            033002  000004  3602    0004     IOT
4472            033004  000124  3604    0054     DCB E'T',PDCZ1-LADTAB
                033005  000006  3605    0006
4473            033006  000110  3606    0048     DCB E'H',PDCZ1-LADTAB
                033007  000006  3607    0006
4474            033010  000105  3608    0045     DCB E'E',PDCZ1-LADTAB
                033011  000006  3609    0006
```

```
4475  033012  000116  360A  004E          DCB E'N',PDCZ1-LADTAB
      033013  000006  360B  0006
4476  033014  000040  360C  0020          DCB E' ',PDCZ1-LADTAB
      033015  000006  360D  0006
4477  033016  000072  360E  003A          DCH E':'
4478  033020  005237  3610  0A9F          INC ELSECH
      033022  055342  3612  5AE2
4479  033024  013737  3614  17DF          MOV FORIND,ELSFOR
      033026  053330  3616  56D8
      033030  053334  3618  56DC
4480  033032  000402  361A  0102          BR NEW2
4481  033034  104406  361C  8906        STSKXY TRAP VB0011-TRPTBL
4482  033036  104156  361E  886E        CZ1 EMT VC0076-EMTTBL
4483  033040  013737  3620  17DF        NEW2 MOV L,GSAVL
      033042  053324  3622  56D4
      033044  055354  3624  5AEC
4484  033046  013737  3626  17DF          MOV SL,GSAVSL
      033050  053326  3628  56D6
      033052  055356  362A  5AEE
4485  033054  005737  362C  0BDF          TST FORIND
      033056  053330  362E  56D8
4486  033060  001061  3630  0231          BNE NEW3
4487  033062  013702  3632  17C2          MOV SL,R2
      033064  053326  3634  56D6
4488  033066  112722  3636  9592          MOVB =X'E6',(R2)+
      033070  000346  3638  00E6
4489  033072  010237  363A  109F          MOV R2,SL
      033074  053326  363C  56D6
4490  033076  000004  363E  0004          IOT
4491  033100  000106  3640  0046          DCB E'F',PDFORB-LADTAB
      033101  000072  3641  003A
4492  033102  000117  3642  004F          DCB E'O',PDFORB-LADTAB
      033103  000072  3643  003A
4493  033104  000122  3644  0052          DCB E'R',PDFORB-LADTAB
      033105  000072  3645  003A
4494  033106  000040  3646  0020          DCB E' ',PDFORB-LADTAB
      033107  000072  3647  003A
4495  033110  000072  3648  003A          DCH E':'
4496  033112  005237  364A  0A9F          INC FORIND
      033114  055330  364C  56D8
4497  033116  022700  364E  25C0          CMP =E'H',R0
      033120  000110  3650  0048
4498  033122  001643  3652  03A3          BEQ ASYNIX
4499  033124  022700  3654  25C0          CMP =E'B',R0
      033126  000102  3656  0042
4500  033130  001640  3658  03A0          BEQ ASYNIX
4501  033132  004737  365A  09DF          JSR PC,VARR
      033134  031554  365C  336C
4502  033136  022700  365E  25C0          CMP =E'=',R0
      033140  000075  3660  003D
4503  033142  001233  3662  0298          BNE ASYNIX
4504  033144  104054  3664  882C          EMT VC0024-EMTTBL
4505  033146  104056  3666  882E          EMT VC0025-EMTTBL
4506  033150  000003  3668  0003          DEBUG
4507  033152  177623  366A  FF93          DCH CH93EX        END OF X1 EXPRESSION
4508  033154  022700  366C  25C0          CMP =E',',R0
      033156  000054  366E  002C
4509  033160  001224  3670  0294          BNE ASYNIX
4510  033162  104054  3672  882C          EMT VC0024-EMTTBL
4511  033164  104056  3674  882E          EMT VC0025-EMTTBL
```

```
4512  033166  000003  3676  0003          DEBUG
4513  033170  177624  3678  FF94          DCH CH94EX      END OF X2 EXPRESSION
4514  033172  022700  367A  25C0          CMP =E',',R0
      033174  000054  367C  002C
4515  033176  001004  367E  0204          BNE FORC
4516  033200  104054  3680  882C          EMT VC0024-EMTTBL
4517  033202  104056  3682  882E          EMT VC0025-EMTTBL
4518  033204  000003  3684  0003          DEBUG
4519  033206  177625  3686  FF95          DCH CH95EX
4520  033210  022700  3688  25C0        FORC CMP =E' ',R0
      033212  000040  368A  0020
4521  033214  001206  368C  0266          BNE ASYNIX
4522  033216  104054  368E  882C          EMT VC0024-EMTTBL
4523  033220  000621  3690  0191          BR NEW1
4524  033222  104156  3692  886E        FORB EMT VC0076-EMTTBL
4525  033224  004737  3694  09DF        NEW3 JSR PC,COSUB
      033226  033420  3696  3710
4526  033230  022700  3698  25C0          CMP =E':',R0
      033232  000072  369A  003A
4527  033234  001510  369C  0348          BEQ CEND2
4528  033236  022700  369E  25C0          CMP =E';',R0
      033240  000073  36A0  003B
4529  033242  001007  36A2  0207          BNE D3
4530  033244  104054  36A4  882C          EMT VC0024-EMTTBL
4531  033246  022700  36A6  25C0          CMP =E' ',R0
      033250  000040  36A8  0020
4532  033252  001270  36AA  02B8          BNE STSKXY
4533  033254  104054  36AC  882C          EMT VC0024-EMTTBL
4534  033256  000137  36AE  005F          JMP NEW1
      033260  032664  36B0  35B4
4535  033262  005737  36B2  0BDF        D3 TST ELSECH
      033264  055342  36B4  5AE2
4536  033266  001662  36B6  03B2          BEQ STSKXY
4537  033270  000003  36B8  0003          DEBUG
4538  033272  177732  36BA  FFDA          DCH CHDAEX
4539  033274  004437  36BC  091F          JSR CHKR,CHECK
      033276  015170  36BE  1A78
4540  033300  000040  36C0  0020          DCC E' ELSE :'
      033301  000105  36C1  0045
      033302  000114  36C2  004C
      033303  000123  36C3  0053
      033304  000105  36C4  0045
      033305  000040  36C5  0020
      033306  000072  36C6  003A
4541  033310  000003  36C8  0003          DEBUG
4542  033312  177642  36CA  FFA2          DCH CHA2EX
4543  033314  013737  36CC  17DF          MOV ELSFOR,FORIND
      033316  053334  36CE  56DC
      033320  053330  36D0  56D8
4544  033322  000137  36D2  005F          JMP NEW8
      033324  032654  36D4  35AC
4545  033326  005737  36D6  0BDF        ELPOSS TST ELSECH
      033330  055342  36D8  5AE2
4546  033332  001451  36DA  0329          BEQ CEND2
4547  033334  013737  36DC  17DF          MOV L,GSAVL
      033336  053324  36DE  56D4
      033340  055354  36E0  5AEC
4548  033342  013737  36E2  17DF          MOV SL,GSAVSL
      033344  053326  36E4  56D6
      033346  055356  36E6  5AEE
```

```
4549  033350  005237  36E8  0A9F        INC FORPER
      033352  053332  36EA  56DA
4550  033354  000004  36EC  0004        IOT
4551  033356  000040  36EE  0020        DCB E' ',PDRST8-LADTAB
      033357  000010  36EF  0008
4552  033360  000105  36F0  0045        DCB E'E',PDRST8-LADTAB
      033361  000010  36F1  0008
4553  033362  000114  36F2  004C        DCB E'L',PDRST8-LADTAB
      033363  000010  36F3  0008
4554  033364  000123  36F4  0053        DCB E'S',PDRST8-LADTAB
      033365  000010  36F5  0008
4555  033366  000105  36F6  0045        DCB E'E',PDRST8-LADTAB
      033367  000010  36F7  0008
4556  033370  000040  36F8  0020        DCB E' ',PDRST8-LADTAB
      033371  000010  36F9  0008
4557  033372  000072  36FA  003A        DCH E':'
4558  033374  062706  36FC  65C6        ADD =2,R6
      033376  000002  36FE  0002
4559                                   * JSUBR PC,REST88
4560                                   * RTS PC          BACK TO IF LEVEL
4561  033400  013737  3700  17DF      REST88 MOV GSAVL,L
      033402  055354  3702  5AEC
      033404  053324  3704  56D4
4562  033406  013737  3706  17DF        MOV GSAVSL,SL
      033410  055356  3708  5AEE
      033412  053326  370A  56D6
4563                                   * JSUBR PC,GETK
4564                                   * RTS PC
4565  033414  000137  370C  005F        JMP GETK
      033416  015106  370E  1A46
4566  033420         3710              COSUB DSH 0
4567  033420  000003  3710  0003        DEBUG  STL SUBR
4568  033422  177750  3712  FFD8        DCH CHD8EX
4569  033424  013737  3714  17DF        MOV SL,SLKP1
      033426  053326  3716  56D6
      033430  055360  3718  5AF0
4570  033432  005237  371A  0A9F        INC SL
      033434  053326  371C  56D6
4571  033436  004737  371E  09DF        JSR PC,MOSTAT
      033440  033460  3720  3730
4572  033442  013702  3722  17C2        MOV SLKP1,R2
      033444  055360  3724  5AF0
4573  033446  113712  3726  97CA        MOVB STYPE,(R2)
      033450  055366  3728  5AF6
4574  033452  000003  372A  0003        DEBUG  STL SUBR
4575  033454  177751  372C  FFD9        DCH CHD9EX
4576  033456  000207  372E  0087      CEND2 RTS PC
4577  033460  013737  3730  17DF      MOSTAT MOV L,SAVL
      033462  053324  3732  56D4
      033464  055362  3734  5AF2
4578  033466  013737  3736  17DF        MOV SL,SAVSL
      033470  053326  3738  56D6
      033472  055364  373A  5AF4
4579  033474  005037  373C  0A1F        CLR STOPER
      033476  055346  373E  5AE6
4580                                   * COULD USE 'COM', GET OVER 'C EQ 2' PROBLEM
4581  033500  022700  3740  25C0        CMP =E'C',R0
      033502  000103  3742  0043
4582  033504  001023  3744  0213        BNE C1C
4583  033506  104054  3746  882C        EMT VC0024-EMTTBL
```

```
4584  033510  022700  3748  25C0        CMP =E' ',R0
      033512  000040  374A  0020
4585  033514  001010  374C  0208        BNE C1B
4586  033516          374E              C1A2 DSH 0
4587  033516  104054  374E  882C        EMT VC0024-EMTTBL
4588  033520  022700  3750  25C0        CMP =E':',R0
      033522  000072  3752  003A
4589  033524  001374  3754  02FC        BNE C1A2
4590  033526  012767  3756  15DF        MOV =X'EE',STYPE
      033530  000356  3758  00EE
      033532  055366  375A  5AF6
4591  033534  000207  375C  0087        RTS PC
4592  033536          375E              C1B DSH 0
4593  033536  013767  375E  17DF        MOV SAVL,L
      033540  055362  3760  5AF2
      033542  053324  3762  56D4
4594  033544  013767  3764  17DF        MOV SAVSL,SL
      033546  055364  3766  5AF4
      033550  053326  3768  56D6
4595  033552  104072  376A  883A        EMT VC0033-EMTTBL
4596  033554          376C              C1C DSH 0
4597                                    *  SEE IF GO
4598  033554  005267  376C  0A9F        INC FORPER
      033556  053332  376E  56DA
4599  033560  000004  3770  0004        IOT
4600  033562  000107  3772  0047        DCB E'G',PDA11-LADTAB
      033563  000012  3773  000A
4601  033564  000117  3774  004F        DCB E'O',PDEXX-LADTAB
      033565  000014  3775  000C
4602  033566  000040  3776  0020        DCB E' ',PDEXX-LADTAB
      033567  000014  3777  000C
4603  033570  000072  3778  003A        DCH E':'
4604  033572  012767  377A  15DF        MOV =X'E1',STYPE
      033574  000341  377C  00E1
      033576  055356  377E  5AF6
4605  033600  013767  3780  17DF        MOV L,SAVL
      033602  053324  3782  56D4
      033604  055362  3784  5AF2
4606  033606  013767  3786  17DF        MOV SL,SAVSL
      033610  053326  3788  56D6
      033612  055364  378A  5AF4
4607  033614  005267  378C  0A9F        INC FORPER
      033616  053332  378E  56DA
4608  033620  000004  3790  0004        IOT
4609  033622  000124  3792  0054        DCB E'T',PDSHGO-LADTAB
      033623  000016  3793  000E
4610  033624  000117  3794  004F        DCB E'O',PDSHGO-LADTAB
      033625  000016  3795  000E
4611  033626  000040  3796  0020        DCB E' ',PDSHGO-LADTAB
      033627  000016  3797  000E
4612  033630  000072  3798  003A        DCH E':'
4613  033632          379A              SH2JKL DSH 0
4614  033632  104056  379A  882E        EMT VC0025-EMTTBL
4615  033634  022700  379C  25C0        SH3 CMP =E';',K
      033636  000073  379E  003B
4616  033640  001401  37A0  0301        BEQ STSKNB
4617  033642  000207  37A2  0087        MNDBB6 RTS PC
4618  033644          37A4              STSKNB DSH 0
4619  033644  104406  37A4  8905        TRAP VB0011-TRPTBL
4620  033646  013767  37A6  17DF        SHGO MOV SAVL,L
```

```
      033650  055362  37A8  5AF2
      033652  053324  37AA  56D4
4621  033654  013737  37AC  17DF          MOV SAVSL,SL
      033656  055364  37AE  5AF4
      033660  053326  37B0  56D6
4622  033662  104072  37B2  883A          EMT VC0033-EMTTBL
4623  033664  000762  37B4  01F2          BR SH2JKL
4624  033666  005237  37B6  0A9F         A11 INC FORPER
      033670  053332  37B8  56DA
4625  033672  000004  37BA  0004          IOT
4626  033674  000124  37BC  0054          DCB E'T',PDB11-LADTAB
      033675  000020  37BD  0010
4627  033676  000131  37BE  0059          DCB E'Y',PDJ11-LADTAB
      033677  000022  37BF  0012
4628  033700  000120  37C0  0050          DCB E'P',PDEXX-LADTAB
      033701  000014  37C1  000C
4629  033702  000105  37C2  0045          DCB E'E',PDEXX-LADTAB
      033703  000014  37C3  000C
4630  033704  000040  37C4  0020          DCB E' ',PDEXX-LADTAB
      033705  000014  37C5  000C
4631  033706  000072  37C6  003A          DCH E':'
4632  033710  012737  37C8  15DF          MOV =X'E3',STYPE
      033712  000343  37CA  00E3
      033714  055366  37CC  5AF6
4633  033716         37CE               HBACCH DSH 0
4634  033716  000003  37CE  0003          DEBUG  STL SUBR
4635  033720  000241  37D0  00A1          DCH X'00A1'
4636  033722  022737  37D2  25DF         HB1A CMP =X'E2',STYPE      SEE IF ACCEPT
      033724  000342  37D4  00E2
      033726  055366  37D6  5AF6
4637  033730  001043  37D8  0223          BNE HB2C
4638  033732  022700  37DA  25C0          CMP =O'47',K       SEE IF QUOTE
      033734  000047  37DC  0027
4639  033736  001040  37DE  0220          BNE HB2C
4640  033740  000003  37E0  0003          DEBUG
4641  033742  177772  37E2  FFFA          DCH CHFAEX
4642  033744  104164  37E4  8874          EMT VC0101-EMTTBL
4643  033746  000003  37E6  0003          DEBUG
4644  033750  177637  37E8  FF9F          DCH CH9FEX
4645  033752  000003  37EA  0003          DEBUG
4646  033754  177773  37EC  FFFB          DCH CHFBEX
4647  033756  000003  37EE  0003          DEBUG  STL SUBR
4648  033760  000254  37F0  00AC          DCH X'00AC'
4649  033762         37F2               HB3 DSH 0
4650  033762  022700  37F2  25C0          CMP =E',',K
      033764  000054  37F4  002C
4651  033766  001002  37F6  0202          BNE MNDCC7
4652  033770  104054  37F8  882C          EMT VC0024-EMTTBL
4653  033772  000753  37FA  01EB          BR HB1A
4654  033774  005737  37FC  0BDF        MNDCC7 TST STOPER
      033776  055346  37FE  5AE6
4655  034000  001720  3800  03D0          BEQ MNDBB6
4656  034002  000714  3802  01CC          BR SH3
4657  034004         3804               HBAL DSH 0
4658  034004  104054  3804  882C          EMT VC0024-EMTTBL
4659  034006  104056  3806  882E          EMT VC0025-EMTTBL
4660  034010  000003  3808  0003          DEBUG  STL SUBR
4661  034012  000365  380A  00F5          DCH X'00F5'
4662  034014  022700  380C  25C0          CMP =E',',R0
      034016  000054  380E  002C
```

```
4663  034020  001311  3810  02C9          BNE STSKNB
4664  034022  104054  3812  882C          EMT VC0024-EMTTBL
4665  034024  104056  3814  882E          EMT VC0025-EMTTBL
4666  034026  022700  3816  25C0          CMP =O'135',R0        CLOSING SQUARE BRACKET
      034030  000135  3818  005D
4667  034032  001304  381A  02C4          BNE STSKNB
4668  034034  104054  381C  882C          EMT VC0024-EMTTBL
4669  034036  000751  381E  01E9          BR HB3
4670  034040  022737  3820  25DF        HB2C CMP =X'E2',STYPE        SEE IF ACCEPT
      034042  000342  3822  00E2
      034044  055306  3824  5AF6
4671  034046  001420  3826  0310          BEQ HB2
4672  034050  022700  3828  25C0          CMP =O'133',R0
      034052  000133  382A  005B
4673  034054  001753  382C  03EB          BEQ HBAL                 OPENING SQUARE BRACKET
4674  034056  104070  382E  8838          EMT VC0032-EMTTBL
4675  034060  000003  3830  0003          DEBUG  STL SUBR
4676  034062  000254  3832  00AC          DCH X'00AC'
4677  034064  005701  3834  0BC1          TST R1
4678  034066  001335  3836  02DD          BNE HB3      LOGICAL OR STRING
4679                                     *  ONLY ARITH LEFT
4680  034070  022700  3838  25C0          CMP =E'X',K
      034072  000130  383A  0058
4681  034074  001403  383C  0303          BEQ HB5
4682  034076  022700  383E  25C0          CMP =E'C',K
      034100  000103  3840  0043
4683  034102  001327  3842  02D7          BNE HB3
4684  034104          3844                HB5 DSH 0
4685  034104  104054  3844  882C          EMT VC0024-EMTTBL
4686  034106  000725  3846  01D5          BR HB3
4687  034110  022700  3848  25C0        HB2 CMP =E'B',K
      034112  000102  384A  0042
4688  034114  001006  384C  0206          BNE HB2A
4689  034116  000003  384E  0003          DEBUG  STL SUBR
4690  034120  177711  3850  FFC9          DCH CHC9EX            INDICATE LOGICAL VARIABLE ACCEPT OP.
4691  034122          3852                HB2B DSH 0
4692  034122  104166  3852  8876          EMT VC0102-EMTTBL
4693  034124          3854                HB2D2 DSH 0
4694  034124  000003  3854  0003          DEBUG  STL SUBR
4695  034126  177713  3856  FFCB          DCH CHCBEX
4696  034130  000714  3858  01CC          BR HB3
4697  034132          385A                HB2A DSH 0
4698  034132  022700  385A  25C0          CMP =E'H',K
      034134  000110  385C  0048
4699  034136  001403  385E  0303          BEQ HB2D
4700  034140  000003  3860  0003          DEBUG  STL SUBR
4701  034142  000312  3862  00CA          DCH X'00CA'           INDICATE ARITH VARIABLE
4702  034144  000766  3864  01F6          BR HB23
4703  034146  000003  3866  0003        HB2D DEBUG
4704  034150  000310  3868  00C8          DCH X'00C8'
4705  034152  104054  386A  882C          EMT VC0024-EMTTBL        STORE THE 'H'
4706  034154  104060  386C  8830          EMT VC0027-EMTTBL
4707  034156  001001  386E  0201          BNE HB2D1
4708  034160  104054  3870  882C          EMT VC0024-EMTTBL
4709  034162  022760  3872  25C0        HB2D1 CMP =E'(',K
      034164  000050  3874  0028
4710  034166  001356  3876  02EE          BNE HB2D2
4711  034170  000003  3878  0003          DEBUG
4712  034172  177701  387A  FFC1          DCH CHC1EX
4713  034174  104054  387C  882C          EMT VC0024-EMTTBL
```

```
4714  034176  104056    387E    882E          EMT VC0025-EMTTBL         AREX
4715  034200  022700    3880    25C0          CMP =E')',K
      034202  000051    3882    0029
4716  034204  001217    3884    028F          BNE STSKNB
4717  034206  104054    3886    882C          EMT VC0024-EMTTBL
4718  034210  000745    3888    01E5          BR HB2D2
4719  034212            388A                  B11 DSH 0
4720  034212  005737    388A    0BDF          TST MAJMOD
      034214  053340    388C    56E0
4721  034216  001415    388E    030D          BEQ C11
4722  034220  000004    3890    0004          IOT
4723  034222  000101    3892    0041          DCB E'A',PDC11-LADTAB
      034223  000024    3893    0014
4724  034224  000103    3894    0043          DCB E'C',PDEXX-LADTAB
      034225  000014    3895    000C
4725  034226  000103    3896    0043          DCB E'C',PDEXX-LADTAB
      034227  000014    3897    000C
4726  034230  000105    3898    0045          DCB E'E',PDEXX-LADTAB
      034231  000014    3899    000C
4727  034232  000120    389A    0050          DCB E'P',PDEXX-LADTAB
      034233  000014    389B    000C
4728  034234  000124    389C    0054          DCB E'T',PDSTSC-LADTAB
      034235  000060    389D    0030
4729  034236  000040    389E    0020          DCB E' ',PDSTSC-LADTAB
      034237  000060    389F    0030
4730  034240  000072    38A0    003A          DCH E':'
4731  034242  012737    38A2    15DF          MOV =X'E2',STYPE
      034244  000342    38A4    00E2
      034246  055366    38A6    5AF6
4732  034250            38A8                  PIGGY DSH 0
4733  034250  000622    38A8    0192          BR HBACCH
4734  034252            38AA                  C11 DSH 0
4735  034252  000004    38AA    0004          IOT
4736  034254  000123    38AC    0053          DCB E'S',PDD11-LADTAB          IF NOT 'S' GO TO D11
      034255  000026    38AD    0016
4737  034256  000124    38AE    0054          DCB E'T',PDE11-LADTAB        IF NOT 'ST' GO TO E11
      034257  000030    38AF    0018
4738  034260  000117    38B0    004F          DCB E'O',PDEXX-LADTAB
      034261  000014    38B1    000C
4739  034262  000120    38B2    0050          DCB E'P',PDEXX-LADTAB
      034263  000014    38B3    000C
4740  034264  000072    38B4    003A          DCH E':'
4741  034266  012737    38B6    15DF          MOV =X'E3',STYPE
      034270  000343    38B8    00E3
      034272  055366    38BA    5AF6
4742  034274  022700    38BC    25C0          CMP =E' ',K       IS IT A SPACE
      034276  000040    38BE    0020
4743  034300  001144    38C0    0264          BNE E13        CANNOT BE '; '
4744  034302  104054    38C2    882C          EMT VC0024-EMTTBL
4745  034304  005237    38C4    0A9F          INC STOPER
      034306  055346    38C6    5AE6
4746  034310  000757    38C8    01EF          BR PIGGY
4747  034312  000004    38CA    0004          E11 IOT
4748  034314  000131    38CC    0059          DCB E'Y',PDSUSZ-LADTAB
      034315  000062    38CD    0032
4749  034316  000115    38CE    004D          DCB E'M',PDEXX-LADTAB
      034317  000014    38CF    000C
4750  034320  000102    38D0    0042          DCB E'B',PDEXX-LADTAB
      034321  000014    38D1    000C
4751  034322  000117    38D2    004F          DCB E'O',PDEXX-LADTAB
```

```
                034323  000014    38D3    000C
4752            034324  000114    38D4    004C          DCB E'L',PDSTSC-LADTAB
                034325  000060    38D5    0030
4753            034326  000123    38D6    0053          DCB E'S',PDSTSC-LADTAB
                034327  000060    38D7    0030
4754            034330  000072    38D8    003A          DCH E':'
4755            034332  012737    38DA    15DF         E12 MOV =X'EC',STYPE
                034334  000354    38DC    00EC
                034336  053306    38DE    5AF6
4756            034340  000207    38E0    0087         MNDBA7 RTS PC
4757            034342            38E2                 SUSZ DSH 0
4758            034342  005737    38E2    0BDF          TST MAJMOD
                034344  053340    38E4    56E0
4759            034346  001137    38E6    025F          BNE H11EXX
4760            034350  000004    38E8    0004          IOT
4761            034352  000125    38EA    0055          DCB E'U',PDEXX-LADTAB
                034353  000014    38EB    000C
4762            034354  000123    38EC    0053          DCB E'S',PDEXX-LADTAB
                034355  000014    38ED    000C
4763            034356  000120    38EE    0050          DCB E'P',PDEXX-LADTAB
                034357  000014    38EF    000C
4764            034360  000105    38F0    0045          DCB E'E',PDEXX-LADTAB
                034361  000014    38F1    000C
4765            034362  000116    38F2    004E          DCB E'N',PDSTSC-LADTAB
                034363  000060    38F3    0030
4766            034364  000104    38F4    0044          DCB E'D',PDSTSC-LADTAB
                034365  000060    38F5    0030
4767            034366  000072    38F6    003A          DCH E':'
4768            034370  000760    38F8    01F0          BR E12
4769            034372            38FA                 D11 DSH 0
4770            034372  000004    38FA    0004          IOT
4771            034374  000122    38FC    0052          DCB E'R',PDG11-LADTAB
                034375  000032    38FD    001A
4772            034376  000125    38FE    0055          DCB E'U',PDD51-LADTAB
                034377  000054    38FF    001C
4773            034400  000116    3900    004E          DCB E'N',PDEXX-LADTAB
                034401  000014    3901    000C
4774            034402  000072    3902    003A          DCH E':'
4775            034404  012737    3904    15DF          MOV =X'E6',STYPE
                034406  000346    3906    00E6
                034410  053306    3908    5AF6
4776            034412  022700    390A    25C0          CMP =E':',R0
                034414  000072    390C    003A
4777            034416  001475    390E    033D          BEQ E13
4778            034420  022700    3910    25C0          CMP =E' ',R0
                034422  000040    3912    0020
4779            034424  001472    3914    033A          BEQ E13
4780            034426  000507    3916    0147          BR H11EXX
4781            034430  005737    3918    0BDF         D51 TST MAJMOD
                034432  053340    391A    56E0
4782            034434  001051    391C    0229          BNE D51AAA
4783            034436  000004    391E    0004          IOT
4784            034440  000105    3920    0045          DCB E'E',PDEXX-LADTAB
                034441  000014    3921    000C
4785            034442  000123    3922    0053          DCB E'S',PDEXX-LADTAB
                034443  000014    3923    000C
4786            034444  000105    3924    0045          DCB E'E',PDEXX-LADTAB
                034445  000014    3925    000C
4787            034446  000124    3926    0054          DCB E'T',PDEXX-LADTAB
                034447  000014    3927    000C
```

```
4788  034450  000023   3928   0013          DCB 0'023',PDSTRT-LADTAB
      034451  000040   3929   0020
4789  034452  000031   392A   0019          DCB 0'031',PDSTRT-LADTAB
      034453  000040   392B   0020
4790  034454  000023   392C   0013          DCB 0'023',PDSTRT-LADTAB
      034455  000040   392D   0020
4791  034456  000072   392E   003A          DCH E':'
4792  034460  012737   3930   15DF          MOV =X'E6',STYPE
      034462  000346   3932   00E6
      034464  055366   3934   5AF6
4793  034466  012737   3936   15DF          MOV =2,A
      034470  000002   3938   0002
      034472  052620   393A   5590
4794  034474  012737   393C   15DF          D51BCD MOV =4,SAVE
      034476  000004   393E   0004
      034500  053400   3940   5700
4795  034502  022700   3942   25C0          CMP =E' ',R0
      034504  000040   3944   0020
4796  034506  001022   3946   0212          BNE D51STR
4797  034510  104054   3948   882C          EMT VC0024-EMTTBL
4798  034512  020027   394A   2017          D51ABC CMP R0,=E'H'
      034514  000110   394C   0048
4799  034516  100416   394E   810E          BMI D51STR
4800  034520  022700   3950   25C0          CMP =E'W',R0
      034522  000127   3952   0057
4801  034524  100413   3954   810B          BMI D51STR
4802  034526  104054   3956   882C          EMT VC0024-EMTTBL
4803  034530  005337   3958   0ADF          DEC SAVE
      034532  053400   395A   5700
4804  034534  001366   395C   02F6          BNE D51ABC
4805  034536  005337   395E   0ADF          DEC A
      034540  052620   3960   5590
4806  034542  001354   3962   02EC          BNE D51BCD
4807  034544  022700   3964   25C0          CMP =E':',R0
      034546  000072   3966   003A
4808  034550  001001   3968   0201          BNE D51STR
4809  034552  000207   396A   0087          RTS PC
4810  034554  000137   396C   005F          D51STR JMP SSTRT
      034556  035464   396E   3834
4811  034560  005737   3970   0BDF          D51AAA TST MAJMOD
      034562  053340   3972   56E0
4812  034564  001430   3974   0318          BEQ H11EXX
4813  034566  000004   3976   0004          IOT
4814  034570  000105   3978   0045          DCB E'E',PDEXX-LADTAB
      034571  000014   3979   000C
4815  034572  000124   397A   0054          DCB E'T',PDEXX-LADTAB
      034573  000014   397B   000C
4816  034574  000125   397C   0055          DCB E'U',PDEXX-LADTAB
      034575  000014   397D   000C
4817  034576  000122   397E   0052          DCB E'R',PDEXX-LADTAB
      034577  000014   397F   000C
4818  034600  000116   3980   004E          DCB E'N',PDSTSC-LADTAB
      034601  000060   3981   0030
4819  034602  000072   3982   003A          DCH E':'
4820  034604  012737   3984   15DF          MOV =X'E5',STYPE
      034606  000345   3986   00E5
      034610  055366   3988   5AF6
4821  034612           398A                 E13 DSH 0
4822  034612  000137   398A   005F          JMP SH3
      034614  033634   398C   379C
```

```
4823  034616           398E              G11 DSH 0
4824  034616           398E              G51 DSH 0
4825                                     *  COULD DO AS IMMEDIATE; STORE RETURN ADDRESS AS ZERO
4826  034616  000004   398E   0004        IOT
4827  034620  000103   3990   0043        DCB E'C',PDH11-LADTAB
      034621  000042   3991   0022
4828  034622  000101   3992   0041        DCB E'A',PDCLR-LADTAB
      034623  000066   3993   0036
4829  034624  000114   3994   004C        DCB E'L',PDEXX-LADTAB
      034625  000014   3995   000C
4830  034626  000114   3996   004C        DCB E'L',PDEXX-LADTAB
      034627  000014   3997   000C
4831  034630  000040   3998   0020        DCB E' ',PDEXX-LADTAB
      034631  000014   3999   000C
4832  034632  000072   399A   003A        DCH E':'
4833  034634  012737   399C   15DF        MOV =X'E4',STYPE
      034636  000344   399E   00E4
      034640  053366   39A0   5AF6
4834  034642  000137   39A2   005F        JMP SH2JKL
      034644  033652   39A4   379A
4835  034646           39A6              H11EXX DSH 0
4836  034646  000137   39A6   005F        JMP EXX
      034650  035352   39A8   3AEA
4837  034652  000004   39AA   0004       H11 IOT
4838  034654  000120   39AC   0050        DCB E'P',PDK11H-LADTAB
      034655  000044   39AD   0024
4839  034656  000101   39AE   0041        DCB E'A',PDH51-LADTAB
      034657  000046   39AF   0026
4840  034660  000072   39B0   003A        DCH E':'
4841  034662  012737   39B2   15DF       H13 MOV =X'EA',STYPE
      034664  000352   39B4   00EA
      034666  053366   39B6   5AF6
4842  034670  104170   39B8   8878        EMT VC0103-EMTTBL
4843  034672  022700   39BA   25C0        CMP =E':',R0
      034674  000072   39BC   003A
4844  034676  001526   39BE   0356        BEQ J13
4845  034700  022700   39C0   25C0        CMP =E':',R0
      034702  000073   39C2   003B
4846  034704  001523   39C4   0353        BEQ J13
4847  034706  022700   39C6   25C0        CMP =E' ',R0
      034710  000040   39C8   0020
4848  034712  001013   39CA   020B        BNE HSTSC
4849  034714  104054   39CC   882C        EMT VC0024-EMTTBL
4850  034716  023700   39CE   27C0        CMP PANEG,K     UP ARROW
      034720  052442   39D0   5522
4851  034722  001156   39D2   026E        BNE HH74C
4852  034724  104054   39D4   882C        EMT VC0024-EMTTBL
4853  034726  022700   39D6   25C0        CMP =E' ',K
      034730  000040   39D8   0020
4854  034732  001404   39DA   0304        BEQ HH74B
4855  034734  022700   39DC   25C0        CMP =E':',K
      034736  000072   39DE   003A
4856  034740  001505   39E0   0345        BEQ J13
4857  034742           39E2              HSTSC DSH 0
4858  034742  104406   39E2   8906        TRAP VB0011-TRPTBL
4859  034744           39E4              HH74B DSH 0
4860  034744  104054   39E4   882C        EMT VC0024-EMTTBL
4861  034746  000544   39E6   0164        BR HH74C
4862  034750  005737   39E8   0BDF       CLR TST MAJMOD
      034752  053340   39EA   56E0
```

```
4863  034754  001176  39EC  027E          BNE EXX
4864  034756  005237  39EE  0A9F          INC FORPER
      034760  055332  39F0  56DA
4865  034762  000004  39F2  0004          IOT
4866  034764  000114  39F4  004C          DCB E'L',PDCONT-LADTAB
      034765  000070  39F5  0038
4867  034766  000105  39F6  0045          DCB E'E',PDEXX-LADTAB
      034767  000014  39F7  000C
4868  034770  000101  39F8  0041          DCB E'A',PDEXX-LADTAB
      034771  000014  39F9  000C
4869  034772  000122  39FA  0052          DCB E'R',PDEXX-LADTAB
      034773  000014  39FB  000C
4870  034774  000072  39FC  003A          DCH E':'
4871  034776  012767  39FE  15DF          MOV =X'E7',STYPE
      035000  000347  3A00  00E7
      035002  055366  3A02  5AF6
4872  035004  022700  3A04  25C0          CMP =E':',R0
      035006  000072  3A06  003A
4873  035010  001451  3A08  0331          BEQ CLRTS
4874  035012  022700  3A0A  25C0          CMP =E' ',R0
      035014  000040  3A0C  0020
4875  035016  001351  3A0E  02E9         CLRD BNE HSTSC
4876  035020  104054  3A10  882C          EMT VC0024-EMTTBL
4877  035022  104106  3A12  8876          EMT VC0102-EMTTBL       GET NAME OF VARIABLE
4878  035024  000003  3A14  0003          DEBUG
4879  035026  177713  3A16  FFCB          DCH CHCBEX
4880  035030  022700  3A18  25C0          CMP =E':',R0
      035032  000072  3A1A  003A
4881  035034  001447  3A1C  0327          BEQ CLRTS
4882  035036  022700  3A1E  25C0          CMP =E',',R0
      035040  000054  3A20  002C
4883  035042  000765  3A22  01F5          BR CLRD
4884  035044  000004  3A24  0004         CONT IOT
4885  035046  000117  3A26  004F          DCB E'O',PDEXX-LADTAB
      035047  000014  3A27  000C
4886  035050  000116  3A28  004E          DCB E'N',PDEXX-LADTAB
      035051  000014  3A29  000C
4887  035052  000124  3A2A  0054          DCB E'T',PDEXX-LADTAB
      035053  000014  3A2B  000C
4888  035054  000111  3A2C  0049          DCB E'I',PDEXX-LADTAB
      035055  000014  3A2D  000C
4889  035056  000116  3A2E  004E          DCB E'N',PDEXX-LADTAB
      035057  000014  3A2F  000C
4890  035060  000125  3A30  0055          DCB E'U',PDEXX-LADTAB
      035061  000014  3A31  000C
4891  035062  000105  3A32  0045          DCB E'E',PDEXX-LADTAB
      035063  000014  3A33  000C
4892  035064  000072  3A34  003A          DCH E':'
4893  035066  012767  3A36  15DF          MOV =X'ED',STYPE
      035070  000355  3A38  00ED
      035072  055366  3A3A  5AF6
4894  035074  000207  3A3C  0087          RTS PC
4895  035076  000004  3A3E  0004         H51 IOT
4896  035100  000102  3A40  0042          DCB E'B',PDEXX-LADTAB
      035101  000014  3A41  000C
4897  035102  000072  3A42  003A          DCH E':'
4898  035104  000666  3A44  0186          BR H13
4899  035106  000004  3A46  0004         J11 IOT
4900  035110  000122  3A48  0052          DCB E'R',PDEXX-LADTAB
      035111  000014  3A49  000C
```

```
4901  035112  000117    3A4A    004F          DCB E'O',PDEXX-LADTAB
      035113  000014    3A4B    000C
4902  035114  000116    3A4C    004E          DCB E'N',PDJ51-LADTAB
      035115  000050    3A4D    0028
4903  035116  000072    3A4E    003A          DCH E':'
4904  035120  012737    3A50    150F         J12 MOV =X'EB',STYPE
      035122  000353    3A52    00EB
      035124  055366    3A54    5AF6
4905  035126  104170    3A56    8878          EMT VC0103-EMTTBL
4906  035130  022700    3A58    25C0          CMP =E' ',K
      035132  000040    3A5A    0020
4907  035134  001007    3A5C    0207          BNE J13
4908  035136  104054    3A5E    882C          EMT VC0024-EMTTBL
4909  035140  000447    3A60    0127          BR HH74C
4910  035142           3A62                  FABS DSH 0
4911  035142  004737    3A62    09DF          $4T JSR PC,VERRR1
      035144  004532    3A64    095A
4912  035146  042737    3A66    45DF          BIC =X'8000',A
      035150  100000    3A68    8000
      035152  052620    3A6A    5590
4913  035154           3A6C                  CLRTS DSH 0
4914  035154  000207    3A6C    0087          J13 RTS PC
4915  035156  000004    3A6E    0004          J51 IOT
4916  035160  000106    3A70    0046          DCB E'F',PDEXX-LADTAB
      035161  000014    3A71    000C
4917  035162  000106    3A72    0046          DCB E'F',PDEXX-LADTAB
      035163  000014    3A73    000C
4918  035164  000072    3A74    003A          DCH E':'
4919  035166  000754    3A76    01EC          BR J12
4920  035170           3A78                  K11FGH DSH 0
4921  035170  000004    3A78    0004          IOT
4922  035172  000104    3A7A    0044          DCB E'D',PDK11Z-LADTAB
      035173  000052    3A7B    002A
4923  035174  000105    3A7C    0045          DCB E'E',PDEXX-LADTAB
      035175  000014    3A7D    000C
4924  035176  000114    3A7E    004C          DCB E'L',PDEXX-LADTAB
      035177  000014    3A7F    000C
4925  035200  000105    3A80    0045          DCB E'E',PDEXX-LADTAB
      035201  000014    3A81    000C
4926  035202  000124    3A82    0054          DCB E'T',PDEXX-LADTAB
      035203  000014    3A83    000C
4927  035204  000105    3A84    0045          DCB E'E',PDEXX-LADTAB
      035205  000014    3A85    000C
4928  035206  000072    3A86    003A          DCH E':'
4929  035210  012737    3A88    15DF          MOV =X'E9',STYPE
      035212  000351    3A8A    00E9
      035214  055366    3A8C    5AF6
4930  035216  005237    3A8E    0A9F          INC STOPER
      035220  055346    3A90    5AE6
4931  035222  000411    3A92    0109          BR K11FGY
4932  035224           3A94                  K11FGZ DSH 0
4933  035224  000004    3A94    0004          IOT
4934  035226  000114    3A96    004C          DCB E'L',PDN11-LADTAB
      035227  000054    3A97    002C
4935  035230  000111    3A98    0049          DCB E'I',PDJ11A-LADTAB
      035231  000056    3A99    002E
4936  035232  000123    3A9A    0053          DCB E'S',PDEXX-LADTAB
      035233  000014    3A9B    000C
4937  035234  000124    3A9C    0054          DCB E'T',PDEXX-LADTAB
      035235  000014    3A9D    000C
```

```
4958  035236  000072    3A9E    003A          DCH E':'
4959  035240  012737    3AA0    15DF          MOV =X'E8',STYPE
      035242  000350    3AA2    00E8
      035244  055366    3AA4    5AF6
4960  035246            3AA6                  K11FGY DSH 0
4961  035246  104170    3AA6    8878          EMT VC0103-EMTTBL
4962  035250  022700    3AA8    25C0          CMP =E' ',K
      035252  000040    3AAA    0020
4963  035254  001010    3AAC    0208          BNE K13QWE
4964  035256  104054    3AAE    882C          EMT VC0024-EMTTBL
4965  035260  022700    3AB0    25C0          HH74C CMP =E',',K
      035262  000054    3AB2    002C
4966  035264  001006    3AB4    0206          BNE K12
4967  035266  000003    3AB6    0003          DEBUG  STL SUBR
4968  035270  177763    3AB8    FFF3          DCH X'FFF3'
4969  035272  104054    3ABA    882C          EMT VC0024-EMTTBL
4970  035274  104056    3ABC    882E          EMT VC0025-EMTTBL
4971  035276  000137    3ABE    005F          K13QWE JMP MNDCC7
      035300  033774    3AC0    37FC
4972  035302            3AC2                  K12 DSH 0          GET 1ST OPERAND
4973  035302  104056    3AC2    882E          EMT VC0025-EMTTBL
4974  035304  022700    3AC4    25C0          CMP =E',',K
      035306  000054    3AC6    002C
4975  035310  001372    3AC8    02FA          BNE K13QWE
4976  035312  000003    3ACA    0003          DEBUG  STL SUBR
4977  035314  177763    3ACC    FFF3          DCH X'FFF3'
4978  035316  104054    3ACE    882C          EMT VC0024-EMTTBL
4979  035320  022700    3AD0    25C0          CMP =E' ',R0
      035322  000040    3AD2    0020
4980  035324  001764    3AD4    03F4          BEQ K13QWE
4981  035326  022700    3AD6    25C0          CMP =E':',R0
      035330  000072    3AD8    003A
4982  035332  001761    3ADA    03F1          BEQ K13QWE
4983  035334  104056    3ADC    882E          EMT VC0025-EMTTBL
4984  035336  000757    3ADE    01EF          BR K13QWE
4985  035340  022700    3AE0    25C0          J11A CMP =E'T',K
      035342  000124    3AE2    0054
4986  035344  001002    3AE4    0202          BNE EXX
4987  035346  104054    3AE6    882C          EMT VC0024-EMTTBL
4988  035350  000656    3AE8    01AE          BR J11
4989  035352            3AEA                  N11 DSH 0
4990  035352  013737    3AEA    17DF          EXX MOV SAVL,L
      035354  055362    3AEC    5AF2
      035356  055324    3AEE    56D4
4991  035360  013737    3AF0    17DF          MOV SAVSL,SL
      035362  055364    3AF2    5AF4
      035364  053326    3AF4    56D6
4992  035366  104072    3AF6    883A          EMT VC0033-EMTTBL
4993  035370  012737    3AF8    15DF          MOV =X'E0',STYPE
      035372  000340    3AFA    00E0
      035374  055366    3AFC    5AF6
4994                                          * JSUBR PC,GENEX
4995                                          * RTS PC
4996  035376  000137    3AFE    005F          JMP GENEX
      035400  014202    3B00    1882
4997  035402  010637    3B02    119F          TRAKK MOV R6,A
      035404  052620    3B04    5590
4998  035406  004737    3B06    09DF          JSR PC,NEWLIN
      035410  031144    3B08    3264
4999  035412  004737    3B0A    09DF          TRACE1 JSR PC,NEWLIN
```

```
                035414  031144  3B0C  3264
4980            035416  013704  3B0E  17C4        MOV A,R4
                035420  052620  3B10  5590
4981            035422  004737  3B12  09DF        JSR PC,RESHEX
                035424  022764  3B14  25F4
4982            035426  013701  3B16  17C1        MOV A,R1
                035430  052620  3B18  5590
4983            035432  012104  3B1A  1444        MOV (R1)+,R4
4984            035434  010137  3B1C  105F        MOV R1,A
                035436  052620  3B1E  5590
4985            035440  004737  3B20  09DF        JSR PC,RESHEX
                035442  022764  3B22  25F4
4986            035444  023737  3B24  27DF        CMP A,STKTOP
                035446  052620  3B26  5590
                035450  052430  3B28  5518
4987            035452  100757  3B2A  61EF        BMI TRACE1
4988            035454  000000  3B2C  0000        HALT
4989            035456  005737  3B2E  08DF        TST SWR
                035460  177570  3B30  FF78
4990            035462  001066  3B32  0236        BNE STRTF
4991            035464          3B34            SSTRT DSH 0
4992            035464  005737  3B34  08DF        TST SIMULA
                035466  052376  3B36  54FE
4993            035470  001404  3B38  0304        BEQ SIMDIM
4994            035472  000005  3B3A  0005        RESET
4995            035474  012737  3B3C  15DF        MOV =X'7FF8',SYMHI
                035476  077770  3B3E  7FF8
                035500  052422  3B40  5512
4996            035502          3B42            SIMDIM DSH 0
4997            035502  013706  3B42  17C6        MOV STKTOP,R6
                035504  052430  3B44  5518
4998            035506  010637  3B46  119F        MOV R6,COLKNT
                035510  055336  3B48  5ADE
4999            035512  012700  3B4A  15C0        MOV =O'005',R0
                035514  000005  3B4C  0005
5000            035516  004737  3B4E  09DF        JSR PC,HSPAAA
                035520  031262  3B50  3282
5001            035522  012737  3B52  15DF        MOV =80,SAVE
                035524  000120  3B54  0050
                035526  053400  3B56  5700
5002            035530          3B58            SIMXXX DSH 0
5003            035530  005000  3B58  0A00        CLR R0
5004            035532  004737  3B5A  09DF        JSR PC,HSPSUB
                035534  031174  3B5C  327C
5005            035536  005337  3B5E  0ADF        DEC SAVE
                035540  053400  3B60  5700
5006            035542  001372  3B62  02FA        BNE SIMXXX
5007            035544  012700  3B64  15C0        MOV =O'007',R0
                035546  000007  3B66  0007
5008            035550  004737  3B68  09DF        JSR PC,HSPSUB
                035552  031174  3B6A  327C
5009            035554  012737  3B6C  15DF        MOV =X'0100',STSEQ
                035556  000400  3B6E  0100
                035560  055372  3B70  5AFA
5010            035562  012701  3B72  15C1        MOV =CRUD,R1        SET UP ADDRESS
                035564  052464  3B74  5534
5011            035566  012700  3B76  15C0        MOV =15,R0    NUMBER OF WORDS
                035570  000017  3B78  000F
5012            035572  005021  3B7A  0A11        SSLOOP CLR (R1)+
5013            035574  005300  3B7C  0AC0        DEC R0
```

```
5014  035576  001375    3B7E    02FD          BNE SSLOOP
5015                                          * CLR (R1)+         CRUD
5016                                          * CLR (R1)+         PAROP
5017                                          * CLR (R1)+         TRACE
5018                                          * CLR (R1)+         GLLTR
5019                                          * CLR (R1)+         GLTTR
5020                                          * CLR (R1)+         LSTERR
5021                                          * CLR (R1)+         PREVAD
5022                                          * CLR (R1)+         SPECTF
5023                                          * CLR (R1)+       PREVCH
5024                                          * CLR (R1)+         OPTION
5025                                          * CLR (R1)+       WUNMOR
5026                                          * CLR (R1)+     ASYIND
5027                                          * CLR (R1)+     ASYIND+2
5028                                          * CLR (R1)+     ASYIND+4
5029                                          * CLR (R1)+     MODE2A
5030  035600  104036    3B80    881E          EMT VC0016-EMTTBL
5031  035602  012701    3B82    15C1          MOV =HEADY,R1
      035604  007324    3B84    0ED4
5032  035606  104022    3B86    8812          EMT VC0010-EMTTBL
5033  035610  104016    3B88    880E          EMT VC0008-EMTTBL
5034  035612  013737    3B8A    17DF          MOV STORLO,STPTR
      035614  052424    3B8C    5514
      035616  052546    3B8E    5566
5035  035620  013737    3B90    17DF          MOV STORLO,SYMLO
      035622  052424    3B92    5514
      035624  052462    3B94    5532
5036  035626  005037    3B96    0A1F        DROP CLR RETADR       NO RETURN ADDRESS
      035630  052554    3B98    556C
5037  035632            3B9A                STRTFP DSH 0
5038  035632  012737    3B9A    15DF          MOV =E'1',STATE
      035634  000061    3B9C    0031
      035636  052550    3B9E    5568
5039  035640  005037    3BA0    0A1F        STRTF CLR OPTION
      035642  052502    3BA2    5542
5040  035644  013706    3BA4    17C6          MOV STKTOP,R6
      035646  052430    3BA6    5518
5041  035650  104036    3BA8    881E          EMT VC0016-EMTTBL
5042  035652  012700    3BAA    15C0          MOV =E'>',R0
      035654  000076    3BAC    003E
5043  035656  104014    3BAE    880C          EMT VC0007-EMTTBL
5044  035660  013706    3BB0    17C6        STRTF2 MOV STKTOP,R6
      035662  052430    3BB2    5518
5045  035664  012737    3BB4    15DF          MOV =INPUT,INPA1        SET POINTER TO BUFFER; NO STORED STRING
      035666  054514    3BB6    594C
      035670  053404    3BB8    5704
5046  035672  012737    3BBA    15DF          MOV =INPUT,INPB1    ALL CHARACTERS TO BE TYPED
      035674  054514    3BBC    594C
      035676  053406    3BBE    5706
5047  035700  012737    3BC0    15DF          MOV =RESTRZ,ALT2A       STSC RESTART
      035702  035714    3BC2    3BCC
      035704  053306    3BC4    56C6
5048  035706  012737    3BC6    15DF          MOV =STRTF,ALT3A        ABORT RESTART
      035710  035640    3BC8    3BA0
      035712  055330    3BCA    5AD8
5049  035714  013706    3BCC    17C6        RESTRZ MOV STKTOP,R6
      035716  052430    3BCE    5518
5050  035720  010637    3BD0    119F          MOV R6,TYSMR6           SAVE REG 6
      035722  052532    3BD2    555A
5051  035724  012737    3BD4    15DF          MOV =LOOKLN,TYSAM
```

```
      035726  037040    3BD6    3E20
      035730  053310    3BD8    56C8
5052  035732  104172    3BDA    887A          EMT VC0105-EMTTBL
5053  035734  010037    3BDC    101F          MOV R0,ASYR0
      035736  055352    3BDE    5AEA
5054  035740  104014    3BE0    880C          EMT VC0007-EMTTBL
5055  035742  104036    3BE2    881E          EMT VC0016-EMTTBL
5056  035744  012737    3BE4    15DF          MOV =63,SKPLF   SKIP THE NEXT LINE FEED
      035746  000077    3BE6    003F
      035750  055440    3BE8    5B20
5057  035752  013703    3BEA    17C3        LPEA2 MOV OUT,R3
      035754  055370    3BEC    5AF8
5058  035756  006303    3BEE    0CC3          ASL R3      DOUBLE
5059  035760  016307    3BF0    1CC7          MOV JTBL(R3),PC
      035762  035764    3BF2    3BF4
5060  035764  036176    3BF4    3C7E        JTBL DCH SUBVID       0 - GO BACK TO STORED PROGRAM
5061  035766  036636    3BF6    3D9E          DCH AMCOL2        1 - IMMEDIATE STATEMENT
5062  035770  036610    3BF8    3D88          DCH DELSTM      2 - DELETE STATEMENT
5063  035772  036616    3BFA    3D8E          DCH DELNUM      3 - CHANGE STATEMENT NUMBER
5064  035774  036426    3BFC    3D16          DCH FUULIN        4 - FULL LINE TO BE STORED
5065  035776  036000    3BFE    3C00          DCH COL1NM          5 - REQUEST FOR POSITION
**********************************************
5268  037040            3E20                LOOKLN DSH 0
5269  037040  005037    3E20    0A1F          CLR OUT           CLEAR RESULT WORD
      037042  055370    3E22    5AF8
5270  037044  005037    3E24    0A1F          CLR CRUD
      037046  052464    3E26    5534
5271  037050  005037    3E28    0A1F          CLR ASYDIG
      037052  055350    3E2A    5AE8
5272  037054  012737    3E2C    15DF          MOV =INPUT,L
      037056  054514    3E2E    594C
      037060  053324    3E30    56D4
5273  037062  012737    3E32    15DF          MOV =OUTPUT,SL
      037064  053410    3E34    5708
      037066  053326    3E36    56D6
5274  037070  104064    3E38    8834          EMT VC0030-EMTTBL
5275  037072  022700    3E3A    25C0          CMP =O'077',R0
      037074  000077    3E3C    003F
5276  037076  001017    3E3E    020F          BNE COL00
5277  037100  005737    3E40    0B0F          TST RETADR
      037102  052554    3E42    556C
5278  037104  001425    3E44    0315          BEQ SSTTSC
5279  037106  000555    3E46    016D          BR Z5Z5
5280  037110  005037    3E48    0A1F        AMCOL1 CLR MAJMOD
      037112  053340    3E4A    56E0
5281  037114  012737    3E4C    15DF          MOV =INPUT+1,L
      037116  054515    3E4E    594D
      037120  053324    3E50    56D4
5282  037122  012737    3E52    15DF          MOV =OUTPUT,SL
      037124  053410    3E54    5708
      037126  053326    3E56    56D6
5283  037130  104072    3E58    883A          EMT VC0033-EMTTBL
5284  037132  104206    3E5A    8886          EMT VC0113-EMTTBL
5285  037134  000552    3E5C    016A          BR Z1Z1
5286  037136            3E5E                COL00 DSH 0
5287  037136            3E5E                ASYNXZ DSH 0
5288  037136  022700    3E5E    25C0          CMP =E':',R0
      037140  000072    3E60    003A
5289  037142  001007    3E62    0207          BNE MABEL
5290  037144  005737    3E64    0BDF          TST LSTERR
```

```
      037146  052476  3E66  553E
5291  037150  001146  3E68  0266          BNE Z0Z0
5292  037152         3E6A                MAXINE DSH 0
5293  037152  005737  3E6A  0BDF          TST RETADR
      037154  052554  3E6C  556C
5294  037156  001143  3E6E  0263          BNE Z0Z0
5295  037160         3E70                SSTTSC DSH 0
5296  037160  104406  3E70  8906          TRAP VB0011-TRPTBL
5297  037162  022700  3E72  25C0        MABEL CMP =E';',R0
      037164  000073  3E74  003B
5298  037166  001771  3E76  03F9          BEQ MAXINE
5299  037170  012737  3E78  15DF          MOV =63,MAJMOD     ASSUME WILL BE STORED
      037172  000077  3E7A  003F
      037174  053340  3E7C  56E0
5300  037176  022700  3E7E  25C0        COL1 CMP =E' ',R0
      037200  000040  3E80  0020
5301  037202  001644  3E82  03A4          BEQ GETNEX           GET NEXT SEQ NUMBER AND A SPACE
5302  037204  022700  3E84  25C0          CMP =E'0',R0
      037206  000060  3E86  0030
5303  037210  001737  3E88  03DF          BEQ AMCOL1
5304  037212  104032  3E8A  881A          EMT VC0014-EMTTBL
5305  037214  001335  3E8C  02DD          BNE AMCOL1           NOT A NUMBER; ABORT BACK TO COL 1
5306  037216         3E8E                COL2B4 DSH 0          ACCEPT COL 1          'N'
5307  037216  104054  3E8E  882C          EMT VC0024-EMTTBL
5308  037220         3E90                COL2 DSH 0
5309  037220  104032  3E90  881A          EMT VC0014-EMTTBL
5310  037222  001332  3E92  02DA          BNE AMCOL1
5311  037224  104054  3E94  882C          EMT VC0024-EMTTBL
5312  037226         3E96                COL3 DSH 0
5313  037226  104032  3E96  881A          EMT VC0014-EMTTBL
5314  037230  001327  3E98  02D7          BNE AMCOL1
5315  037232  104054  3E9A  882C          EMT VC0024-EMTTBL
5316  037234  022700  3E9C  25C0        COL4 CMP =E' ',R0
      037236  000040  3E9E  0020
5317  037240  001323  3EA0  02D3          BNE AMCOL1
5318  037242  104054  3EA2  882C          EMT VC0024-EMTTBL
5319  037244  022700  3EA4  25C0        COL5 CMP =E':',R0          'NNN :'
      037246  000072  3EA6  003A
5320  037250  001511  3EA8  0349          BEQ Z2PREV           DELETE STORED STATEMENT
5321  037252  022700  3EAA  25C0          CMP =O'032',R0
      037254  000032  3EAC  001A
5322  037256  001011  3EAE  0209          BNE COL5XY
5323  037260  005037  3EB0  0A1F          CLR HBECHO
      037262  053014  3EB2  560C
5324  037264  005337  3EB4  0ADF          DEC L
      037266  053324  3EB6  56D4
5325  037270  005337  3EB8  0ADF          DEC INPA1
      037272  053404  3EBA  5704
5326  037274  004737  3EBC  09DF          JSR PC,GNC         DONT STORE THE CONTROL Z
      037276  027174  3EBE  2E7C
5327  037300  000421  3EC0  0111          BR COL6B4
5328  037302  022700  3EC2  25C0        COL5XY CMP =E' ',R0
      037304  000040  3EC4  0020
5329  037306  001014  3EC6  020C          BNE COL5N
5330  037310  005737  3EC8  0BDF          TST STRCHR
      037312  055002  3ECA  5A02
5331  037314  001413  3ECC  030B          BEQ COL6B4
5332  037316  013704  3ECE  17C4          MOV INPA1,R4
      037320  053404  3ED0  5704
5333  037322  005304  3ED2  0AC4          DEC R4
```

```
5334  037324  110024    3ED4    9014          MOVB R0,(R4)+
5335  037326  110024    3ED6    9014          MOVB R0,(R4)+
5336  037330  110024    3ED8    9014          MOVB R0,(R4)+
5337  037332  010437    3EDA    111F          MOV R4,INPA1
      037334  053404    3EDC    5704
5338  037336  000402    3EDE    0102          BR COL6B4
5339  037340            3EE0                  COL5N DSH 0
5340  037340  104032    3EE0    881A          EMT VC0014-EMTTBL
5341  037342  001262    3EE2    0282          BNE AMCOL1
5342                                          *     NOW COMMITED TO STORING STATEMENT
5343  037344            3EE4                  COL6B4 DSH 0          ACCEPT COL 5  'NNN N'  OR  'NNN  '
5344  037344  104054    3EE4    882C          EMT VC0024-EMTTBL
5345  037346  022700    3EE6    25C0          COL6 CMP =E' ',R0
      037350  000040    3EE8    0020
5346  037352  001402    3EEA    0302          BEQ COL6A
5347  037354  104032    3EEC    881A          EMT VC0014-EMTTBL
5348  037356  001300    3EEE    02C0          BNE SSTTSC
5349  037360            3EF0                  COL6A DSH 0          ACCEPT COL 6   'NNN NN'  OR  'NNN N '
5350  037360  104054    3EF0    882C          EMT VC0024-EMTTBL
5351                                          *  OR  'NNN   '
5352  037362  022700    3EF2    25C0          COL7 CMP =E':',R0
      037364  000072    3EF4    003A
5353  037366  001440    3EF6    0320          BEQ Z3PREV           CHANGE STATEMENT NUMBER   'NNN NN:'
5354  037370  022700    3EF8    25C0          CMP =E' ',K
      037372  000040    3EFA    0020
5355  037374  001414    3EFC    030C          BEQ COL7A
5356  037376  013704    3EFE    17C4          MOV INPA1,R4
      037400  053404    3F00    5704
5357  037402  005304    3F02    0AC4          DEC R4
5358  037404  112724    3F04    95D4          MOVB =E' ',(R4)+
      037406  000040    3F06    0020
5359  037410  110024    3F08    9014          MOVB K,(R4)+
5360  037412  010437    3F0A    111F          MOV R4,INPA1
      037414  053404    3F0C    5704
5361  037416  005267    3F0E    0A9F          INC HBECHO      MAKE IT TWO
      037420  053014    3F10    560C
5362  037422  012700    3F12    15C0          MOV =E' ',K
      037424  000040    3F14    0020
5363  037426            3F16                  COL7A DSH 0
5364  037426  104054    3F16    882C          EMT VC0024-EMTTBL
5365  037430  022700    3F18    25C0          CMP =E':',R0       'NNN NN :'
      037432  000072    3F1A    003A
5366  037434  001415    3F1C    030D          BEQ Z3PREV
5367  037436  104206    3F1E    8886          EMT VC0113-EMTTBL
5368  037440  000402    3F20    0102          BR Z4Z4
5369  037442  005267    3F22    0A9F          Z5Z5 INC OUT      GIVE LOCATION
      037444  055370    3F24    5AF8
5370  037446  005267    3F26    0A9F          Z4Z4 INC OUT       FULL LINE TO STORE
      037450  055370    3F28    5AF8
5371  037452  005267    3F2A    0A9F          Z3Z3 INC OUT CHANGE STATEMENT NUMBER
      037454  055370    3F2C    5AF8
5372  037456  005267    3F2E    0A9F          Z2Z2 INC OUT   DELETE STORED STATEMENT
      037460  055370    3F30    5AF8
5373  037462  005267    3F32    0A9F          Z1Z1 INC OUT IMMEDIATE LINE
      037464  055370    3F34    5AF8
5374  037466            3F36                  Z0Z0 DSH 0  REPEAT/START STORED LINE
5375  037466  000207    3F36    0087          RTS PC
5376  037470  005267    3F38    0A9F          Z3PREV INC OUT
      037472  055370    3F3A    5AF8
5377  037474            3F3C                  Z2PREV DSH 0
```

```
5378  037474  104176    3F3C    887E          EMT VC0107-EMTTBL
5379  037476  012700    3F3E    15C0          MOV =E':',R0
      037500  000072    3F40    003A
5380  037502  005701    3F42    0BC1          TST R1
5381  037504  001364    3F44    02F4          BNE Z2Z2
5382  037506  000624    3F46    0194          BR SSTTSC
5383  037510            3F48                 GETST DSH 0
5384  037510  004737    3F48    09DF          JSR PC,COSTAT
      037512  032336    3F4A    34DE
5385  037514  022700    3F4C    25C0          CMP =E':',R0
      037516  000072    3F4E    003A
5386  037520  001217    3F50    028F          BNE SSTTSC
5387  037522  013701    3F52    17C1          MOV SL,R1
      037524  053326    3F54    56D6
5388  037526  110011    3F56    9009          MOVB R0,(R1)       STORE THE COLON
5389  037530  000207    3F58    0087          RTS PC
*****************************************
5441  037770  010237    3FF8    109F         FERRET MOV R2,R2FERR
      037772  052536    3FFA    555E
5442  037774  012737    3FFC    15DF          MOV =FRR1,FRETRN
      037776  040010    3FFE    4008
      040000  053012    4000    560A
5443  040002  005237    4002    0A9F          INC OPTION
      040004  052502    4004    5542
5444  040006  000473    4006    0138          BR FERSUB
5445  040010            4008                 FRR1 DSH 0
5446  040010  122737    4008    A5DF          CMPB =X'FF',FERSTK+1
      040012  000377    400A    00FF
      040014  054401    400C    5901
5447  040016  001043    400E    0223          BNE FERRYY       MORE THAN ONE
5448  040020  005737    4010    0BDF          TST MODE2B
      040022  052426    4012    5516
5449  040024  001040    4014    0220          BNE FERRYY                DONT GO AHEAD IF PROMPTING
5450  040026  113700    4016    97C0         FERRJJ MOVB FERSTK,R0
      040030  054400    4018    5900
5451  040032  005037    401A    0A1F          CLR LASTCH
      040034  054644    401C    59A4
5452  040036  022700    401E    25C0          CMP =E':',R0
      040040  000072    4020    003A
5453  040042  001454    4022    032C          BEQ FERRTT
5454  040044  013701    4024    17C1          MOV R2FERR,R1
      040046  052536    4026    555E
5455  040050  110021    4028    9011          MOVB R0,(R1)+
5456  040052  010137    402A    105F          MOV R1,R2FERR
      040054  052536    402C    555E
5457  040056  104014    402E    880C          EMT VC0007-EMTTBL
5458  040060  012737    4030    15DF          MOV =FRR2,FRETRN
      040062  040070    4032    4038
      040064  053012    4034    560A
5459  040066  000443    4036    0123          BR FERSUB
5460  040070            4038                 FRR2 DSH 0
5461  040070  122737    4038    A5DF          CMPB =X'FF',FERSTK+1
      040072  000377    403A    00FF
      040074  054401    403C    5901
5462  040076  001753    403E    03EB          BEQ FERRJJ
5463  040100  013737    4040    17DF          MOV R2FERR,INPA1
      040102  052536    4042    555E
      040104  053404    4044    5704
5464  040106  012737    4046    15DF          MOV =INPUT+1000,INPB1
      040110  056464    4048    5D34
```

```
        040112  055406  404A  5706
5465    040114  005000  404C  0A00          CLR R0              GIVE A NULL
5466    040116  104014  404E  880C          EMT VC0007-EMTTBL         TYPECH
5467    040120  005337  4050  0ADF          DEC OPTION
        040122  052502  4052  5542
5468    040124  104416  4054  890E          TRAP VB0027-TRPTBL   ALT2
5469    040126  112700  4056  95C0         FERRYY MOVB =E'>',R0
        040130  000076  4058  003E
5470    040132  104014  405A  880C          EMT VC0007-EMTTBL
5471    040134  104036  405C  881E          EMT VC0016-EMTTBL
5472    040136  012700  405E  15C0          MOV =E'$',R0
        040140  000044  4060  0024
5473    040142  104014  4062  880C          EMT VC0007-EMTTBL
5474    040144  012701  4064  15C1          MOV =FERSTK,R1
        040146  054400  4066  5900
5475    040150  104022  4068  8812          EMT VC0010-EMTTBL
5476    040152  012700  406A  15C0          MOV =E'$',R0
        040154  000044  406C  0024
5477    040156  104014  406E  880C          EMT VC0007-EMTTBL
5478    040160  013737  4070  17DF          MOV INPZZ,INPA1
        040162  055552  4072  5ADA
        040164  053404  4074  5704
5479    040166  005337  4076  0ADF          DEC OPTION
        040170  052502  4078  5542
5480    040172  104424  407A  8914          TRAP VB0043-TRPTBL
5481    040174  104144  407C  8864         FERRTT EMT VC0071-EMTTBL  HALT ROUTINE
5482    040176  012700  407E  15C0         FERSUB MOV =FERSTK,R0
        040200  054400  4080  5900
5483    040202  010037  4082  101F          MOV R0,FERPRT
        040204  055112  4084  5A4A
5484    040206  012710  4086  15C8          MOV =X'00FF',(R0)
        040210  000377  4088  00FF
5485    040212  012700  408A  15C0          MOV =O'040',R0
        040214  000040  408C  0020
5486    040216  000406  408E  0106          BR FERR2
5487    040220  013700  4090  17C0         FERR3 MOV FERCHR,R0
        040222  052540  4092  5560
5488    040224  005200  4094  0A80          INC R0
5489    040226  022700  4096  25C0          CMP =O'140',R0
        040230  000140  4098  0060
5490    040232  001426  409A  0316          BEQ FERFIN
5491    040234  010037  409C  101F         FERR2 MOV R0,FERCHR
        040236  052540  409E  5560
5492    040240  022700  40A0  25C0          CMP =E'<',R0
        040242  000074  40A2  003C
5493    040244  001765  40A4  03F5          BEQ FERR3
5494    040246  022700  40A6  25C0          CMP =E'>',R0
        040250  000076  40A8  003E
5495    040252  001762  40AA  03F2          BEQ FERR3
5496    040254  012737  40AC  15DF          MOV =FERRZZ,FSRETN
        040256  040264  40AE  4084
        040260  054646  40B0  59A6
5497    040262  000414  40B2  010C          BR FTEST
5498    040264  001355  40B4  02ED         FERRZZ BNE FERR3       NO GOOD
5499    040266  013702  40B6  17C2          MOV FERPRT,R2
        040270  055112  40B8  5A4A
5500    040272  113722  40BA  97D2          MOVB FERCHR,(R2)+
        040274  052540  40BC  5560
5501    040276  112712  40BE  95CA          MOVB =X'FF',(R2)
        040300  000377  40C0  00FF
```

```
5502  040302  010237  40C2  109F          MOV R2,FERPRT
      040304  055112  40C4  5A4A
5503  040306  000744  40C6  01E4          BR FERR3
5504  040310  013707  40C8  17C7          FERFIN MOV FRETRN,PC          RETURN
      040312  053012  40CA  560A
5505  040314  013737  40CC  17DF          FTEST MOV ALT2A,ALT2AF
      040316  053306  40CE  56C6
      040320  052534  40D0  555C
5506  040322  012737  40D2  15DF          MOV =FERR1,ALT2A
      040324  040412  40D4  410A
      040326  053306  40D6  56C6
5507  040330  013702  40D8  17C2          MOV R2FERR,R2
      040332  052536  40DA  555E
5508  040334  110022  40DC  9012          MOVB R0,(R2)+
5509  040336  012701  40DE  15C1          MOV =X'00FF',R1
      040340  000377  40E0  00FF
5510  040342  110122  40E2  9052          MOVB R1,(R2)+
5511  040344  110122  40E4  9052          MOVB R1,(R2)+
5512  040346  110122  40E6  9052          MOVB R1,(R2)+
5513  040350  110122  40E8  9052          MOVB R1,(R2)+
5514  040352  110122  40EA  9052          MOVB R1,(R2)+
5515  040354  012737  40EC  15DF          MOV =INPUT+1000,INPA1
      040356  056464  40EE  5D34
      040360  053404  40F0  5704
5516  040362  012737  40F2  15DF          MOV =INPUT+1000,INPB1
      040364  056464  40F4  5D34
      040366  053406  40F6  5706
5517  040370  013706  40F8  17C6          MOV TYSMR6,R6
      040372  052532  40FA  555A
5518  040374  013701  40FC  17C1          MOV TYSAM,R1
      040376  053310  40FE  56C8
5519  040400  004711  4100  09C9          JSR PC,(R1)
5520  040402  013737  4102  17DF          MOV ALT2AF,ALT2A
      040404  052534  4104  555C
      040406  053306  4106  56C6
5521  040410  000407  4108  0107          BR FERR9
5522  040412  013737  410A  17DF          FERR1 MOV ALT2AF,ALT2A
      040414  052534  410C  555C
      040416  053306  410E  56C6
5523  040420  013701  4110  17C1          MOV L,R1
      040422  053324  4112  5604
5524  040424  114100  4114  9840          MOVB -(R1),R0
5525  040426  100004  4116  8004          BPL FERR3A
5526  040430  013701  4118  17C1          FERR9 MOV FSRETN,R1
      040432  054646  411A  59A6
5527  040434  000264  411C  00B4          SEZ             SET TO EQUAL
5528  040436  000111  411E  0049          JMP (R1)        ********** DONT CONDENSE
5529  040440  013707  4120  17C7          FERR3A MOV FSRETN,PC        CLEAR EQUAL; RETURN
      040442  054646  4122  59A6
5530  040444  010237  4124  109F          F22A MOV R2,F22R2A
      040446  054652  4126  59AA
5531  040450  005237  4128  0A9F          INC OPTION
      040452  052502  412A  5542
5532  040454  012700  412C  15C0          MOV =O'040',R0
      040456  000040  412E  0020
5533  040460  000403  4130  0103          BR F22B
5534  040462  013700  4132  17C0          F22C MOV FERCHR,R0
      040464  052540  4134  5560
5535  040466  005200  4136  0A80          INC R0
5536  040470  010037  4138  101F          F22B MOV R0,FERCHR
```

```
      040472  052540    413A    5560
5537  040474  022700    413C    25C0          CMP =0'140',R0
      040476  000140    413E    0060
5538  040500  001474    4140    033C          BEQ F22FIN
5539  040502  022700    4142    25C0          CMP =E'<',R0
      040504  000074    4144    003C
5540  040506  001765    4146    03F5          BEQ F22C
5541  040510  022700    4148    25C0          CMP =E'>',R0
      040512  000076    414A    003E
5542  040514  001762    414C    03F2          BEQ F22C
5543  040516  013737    414E    17DF          MOV F22R2A,R2FERR
      040520  054652    4150    59AA
      040522  052536    4152    555E
5544  040524  012737    4154    15DF          MOV =F22D,FSRETN
      040526  040534    4156    415C
      040530  054646    4158    59A6
5545  040532  000670    415A    01B8          BR FTEST
5546  040534  001352    415C    02EA        F22D BNE F22C     NO GOOD
5547  040536  104036    415E    881E          EMT VC0016-EMTTBL
5548  040540  013700    4160    17C0          MOV FERCHR,R0
      040542  052540    4162    5560
5549  040544  104014    4164    880C          EMT VC0007-EMTTBL
5550  040546  012700    4166    15C0          MOV =E' ',R0
      040550  000040    4168    0020
5551  040552  104014    416A    880C          EMT VC0007-EMTTBL
5552  040554  122737    416C    A5DF          CMPB =E':',FERCHR
      040556  000072    416E    003A
      040560  052540    4170    5560
5553  040562  001437    4172    031F          BEQ F23FIN
5554  040564  012700    4174    15C0          MOV =0'040',R0
      040566  000040    4176    0020
5555  040570  000403    4178    0103          BR F23B
5556  040572  013700    417A    17C0        F23C MOV FERCHB,R0
      040574  054650    417C    59A8
5557  040576  005200    417E    0A80          INC R0
5558  040600  010037    4180    101F        F23B MOV R0,FERCHB
      040602  054650    4182    59A8
5559  040604  022700    4184    25C0          CMP =0'140',R0
      040606  000140    4186    0060
5560  040610  001724    4188    0304          BEQ F22C
5561  040612  022700    418A    25C0          CMP =E'<',R0
      040614  000074    418C    003C
5562  040616  001765    418E    03F5          BEQ F23C
5563  040620  022700    4190    25C0          CMP =E'>',R0
      040622  000076    4192    003E
5564  040624  001762    4194    03F2          BEQ F23C
5565  040626  013737    4196    17DF          MOV F22R2A,R2FERR
      040630  054652    4198    59AA
      040632  052536    419A    555E
5566  040634  005237    419C    0A9F          INC R2FERR
      040636  052536    419E    555E
5567  040640  012737    41A0    15DF          MOV =F23D,FSRETN
      040642  040650    41A2    41A8
      040644  054646    41A4    59A6
5568  040646  000622    41A6    0192          BR FTEST
5569  040650  001350    41A8    02E8        F23D BNE F23C
5570  040652  013700    41AA    17C0          MOV FERCHB,R0
      040654  054650    41AC    59A8
5571  040656  104014    41AE    880C          EMT VC0007-EMTTBL
5572  040660  000744    41B0    01E4          BR F23C
```

```
5573  040662  012700  4182  15C0      F23FIN MOV =E'$',R0
      040664  000044  4184  0024
5574  040666  104014  4186  880C       EMT VC0007-EMTTBL
5575  040670  000674  4188  01BC       BR F22C
5576  040672  005337  418A  0ADF      F22FIN DEC OPTION
      040674  052502  418C  5542
5577  040676  013737  418E  17DF       MOV INPZZ,INPA1.
      040700  055332  41C0  5ADA
      040702  053404  41C2  5704
5578  040704  104424  41C4  8914       TRAP VB0043-TRPTBL
```

Supplementary

Historical

Perspective

The XB system is an experimental system designed to demonstrate the practicability of detecting syntax errors by echo-checking individual characters as they are presented via a teletypewriter.

The state of the art of computing, upon which the XB system builds, can be broken down into three main areas of interest, each with their own history. These areas are :-

a) The Development of Electronic Computers.

b) The Development of Programming Languages.

c) The Development of On-line Computing.

1. Development of Electronic Computing.

The first electronic computer was a machine called ENIAC (Electronic Numerical Integrator And Calculator) built by Eckert and Mauchly at the University of Pennsylvania in 1946.

This machine was based on eighteen thousand vacuum tubes and programs to be run had to be installed by engineers who changed the wiring among its various components. The technology used in computers has grown from vacuum tubes via the transistor, invented in 1948, to large-scale integrated circuits. Three generations of computers based on these technologies correspond to the years 1946-1959, 1959-1965 and 1965 to present.

Electronic computers and the programs which drive them have each had a large effect on the evolution of the other. Programs are naturally written to most effectively use the hardware which is available and hardware is built with the needs of the programs

in mind.

The PDP-11, apart from being the electronic computer used to implement the XB system, is typical of the state of the present art of electronic computers. It is one of a family of computers, each having a similiar instruction set which enables programs developed on a small, slow member of the family to be run on a larger, faster member. Other examples of families are the IBM 360 series and the CDC Cyber series. The evolution of these families has been brought about by a recognition of the enormous task of rewriting large programs to take advantage of faster cheaper computers, a clearer understanding of instruction sets and the passing of computers from the experimental stage to being a tool.

The internal organization of electronic computers has evolved in the direction of large instruction sets (including floating point instructions), priority interrupt systems, multiple registers and stack techniques (which allow recursive subroutine calls, used heavily by the XB system).

High speed memory systems have developed from electostatic storage systems to solid-state memories with cycle speeds of a few hundred nanoseconds. Early machines were considered adequate with a thousand words of storage while the IBM 360 series has been designed to address four million words of storage.

An excellent historical survey of the development of Electronic Computers has been written by ROSEN-69. See also ALT-72, SPRAGUE-72, WILKES-60 and BELL-71 (Chap. 3 esp.).

2. Development of Programming Languages

At the level where instructions are electronically interpreted in a computer the language used is normally expressed in binary code. In the first computers it was necessary for the programmer to write a program as a set of binary instructions. For example, on the IBM 704 the instruction

011011 000000 000000 000000 000000 110000

would mean: place the contents of storage location 48 in the accumulator.

A first step in relieving the problem of dealing totally in binary was to introduce mnemonic codes for instructions and the instruction above became

CLA 000000 000000 000000 000000 110000

Another development was to condense 3 bits into an octal number

CLA 00 00 00 00 60

or the address field on the right could be expressed in decimal

CLA 00 00 00 00 48

If it was found necessary to later insert a new instruction in a program this could cause references to some instructions to be incorrect. A partial solution was provided by a concept of regional addressing where a program was divided into sections and a modification to one section did not upset the entire program.

The concept of symbolic addresses was developed, so that the sample instruction became

CLA TEMP

where TEMP was a storage location defined elsewhere by the programmer. The translation of the mnemonic and the allocation of storage was made via the machine itself by a program called an assembler. One of the first assemblers was for the IBM 704 and known as SAP (Symbolic Assembly Program).

While such an assembler made life a lot easier for the programmer, he still had to write a sequence such as

```
CLA C
MPY D
ADD B
STO A
```

when he really wanted to form A equal to B+C*D. This led to the development of higher-level languages, the most widely used being FORTRAN (FORmula TRANslation). The advantages of higher-level languages (vs assemblers) has been given by SAMMETT-68 (pp 14-17). These are

1. Ease of learning.
2. Ease of coding and understanding.
3. Ease of debugging.
4. Ease of maintenance and documenting.
5. Ease of conversion.
6. Reduced elapsed time for problem solving.

In general, higher level languages allow the programmer to interface more directly with the problem he is trying to solve.

SAMMETT-68 (pp 17-19) also discusses these disadvantages:-

1. Time required for compiling.

2. Inefficient object code.

3. Difficulties in debugging without learning machine language.

4. Inability of the language to express all needed operations.

Despite these objections, the use of higher-level languages is usually preferred where the problem may be expressed in a higher-level language. Much work has been put into reducing the disadvantages listed above.

2.1 FORTRAN

A landmark quoted by SAMMETT-68 (p 143) is a document marked "PRELIMINARY REPORT, Specifications for the IBM Mathematical FORmula TRANslating system, FORTRAN" dated 10th November, 1954 issued by the programming Research Group, Applied Science Division of IBM for the 704 computer.

The 704 FORTRAN system was issued early in 1957 and a new FORTRAN compiler FORTRAN II was issued in June 1958 (thus causing the earlier FORTRAN to become FORTRAN I). These developments initiated a flood of FORTRAN compilers (709 & 650 in late 1958; 1620 & 7070 in 1960 and 7030 (Stretch) in 1962) by IBM and other companies. By 1963 virtually all manufacturers had FORTRAN compilers. OSWALD-64 cites the existance of 43 FORTRAN compilers.

Incompatibilities crept in because of this flood of FORTRAN

compilers. Methods of implementation differed not only between manufacturers but even within the same manufacturer for different machines. In an effort to standardise and also introduce new features in a controlled fashion, the SHARE FORTRAN Committee in March, 1961 went on record as favouring a new, improved FORTRAN language. In May, 1962 a Committee working for the American Standards Association was formed and a publication "American Standard FORTRAN" was approved dated March 7th, 1966. At the same time a proper subset known as Basic FORTRAN was also defined.

The development of FORTRAN has had a very significant impact on computing. Because it was the first such language and because such enormous effort has been spent on education, compiler development and programs, it is difficult for FORTRAN to take advantage of research into programming languages.

2.2 ALGOL

In the late fifties many groups were working on programming languages. In October 1957 GAMM (German Association for applied Mathematics and Mechanics) and the ACM (the Association for Computing Machinery, the American computer society formed in 1947) joined efforts to produce a language first known as IAL (International Algorithmic Language) and later as ALGOL 58 (ALGOrithmic Language). Despite the interest of the ACM, ALGOL was (and still is) largely a European language. ALGOL 60 followed from an international meeting held in Paris in January, 1960 and an "Algorithms" section appeared in the Communications of the ACM

in February of that year for the dissemination of basic programming algorithms (written in ALGOL).

SAMMETT-68 (p 192) discusses these contributions to the state of the art made by ALGOL:-

1) block structure and defining the scope of variables.

2) a formal language definition.

3) recursive procedures.

4) a significant embedding capability for differing subunits.

5) a general simplicity combined with power for stating computational processes.

6) concepts of separate reference, publication and hardware implementation languages.

7) a requirement for the development of better implementation techniques.

8) the spawning of a significant number of languages as outgrowths (such as NELIAC, MAD and JOVIAL).

However ALGOL was designed as a language for procedures internal to a machine and facilities for input and output of data were simply not defined and were left to individual implementations to define (inevitably, differently).

2.3 COBOL

While these developments were going on in the programming languages meant for tecnical purposes, a parallel development was happening for a business language.

In June 1959 a committee was formed by six manufacturers

(Burroughs, IBM, Honeywell, RCA, Remington Rand and Sylvania Electric Products) and two government agencies (Air Material Command, USAF and David Taylor Model Basin, Department of the Navy) and known as CODASYL (COnference on Data SYstems Languages) Short-range committee. COBOL (COmmon Business Oriented Language) specifications were designed by December of that year.

There have been three main contributions to the state of the art by the COBOL language. (See SAMMETT-68 p 375).

The first is the separation of any COBOL program into four main divisions:-

a) An IDENTIFICATION division, which does not contribute to the final running program but its requirements are an aid to documenting the program.

b) An ENVIRONMENT division which collects the hardware descriptions together. This division associates all input and output files with the hardware implementation. It is not intended that this division be independent of a particular machine.

c) A DATA division describes all the data used in the program. This is intended to be machine independent but in practise there is a conflict between describing items in a machine-independent fashion at the loss of efficiency, or choosing efficiency by describing items in a way which is well suited to the particular machine at the cost of machine dependence.

d) A PROCEDURE division where the actions to be taken on the data are described. The same conflict arises between efficiency

and independence for the PROCEDURE division.

The second contribution is the accent on handling large files and the third is the development of a language which is very natural to read (although not necessarily to write) and allows long mnemonic names.

2.4 PL/I

The language now known as PL/I arose from inadequacies of FORTRAN in two main areas; character handling and interactions with modern operating systems. (See SAMMETT-68 p 540).

A committee was formed in September 1963 by SHARE and IBM to look at ways of extending FORTRAN. As this work progressed, it was felt by this Advanced Language Committee that trying to preserve compatibility with FORTRAN would prevent very worthwhile concepts from ALGOL and COBOL being included. The FORTRAN compatibility was abandoned and after a series of reports and a change of name (NPL for New Programming Language met with objection from the National Physical Laboratory) an official manual for PL/I was issued in early 1965 by IBM and in August 1966 the first compiler for the system 360 was released.

By building on the experience gained from FORTRAN, ALGOL and COBOL and also by abandoning considerations of compatibility with an existing language, PL/I may eventually replace these languages with a single language, suitable over a wide range of applications.

See CURRIE-66 for a comparison between PL/I and ALGOL, COBOL

and FORTRAN.

3. Development of On-line Computing.

All the programming languages described so far were primarily designed to specify a computing procedure in its entirety. A file of input (program and data) was presented to the machine, usually via a card reader, and a file of output was returned, usually via a printer.

In the early sixties programmers began to feel that a typewriter directly connected to a computer and, also, a file system could be a very effective substitute for a card punch and card reader.

LAWRENCE-72 (p 590) summarises some of the advantages of this approach. These are

1) Direct program entry which eliminates the keypunching cycle.

2) Syntax checking on a line-by-line basis.

3) Line by line execution.

4) An editing capability.

5) Data set handling.

6) Direct input/output.

7) Use of an error-detecting fast-compiling system for debugging and later use of an optimizing fast-executing system.

8) Immediate execution.

9) Stop, modify and continue capability.

10) Object deck storage.

See also MEADOW-70 pp 281-293.

3.1 QUICKTRAN

Work on QUICKTRAN was started in 1961 in IBM. The objective was to use standard computing equipment and to write a system which would be largely compatible with an existing language but with powerful debugging and terminal control facilities added. The equipment used were IBM 7040/44 computers and 1050 terminals and the language Basic FORTRAN. A first version of this system was running in mid 1963.

There were two modes; a COMMAND mode (where statements were executed immediately) and a PROGRAM mode where statements were stored to build a program. A number of program control statements were introduced to assist in debugging.

SAMMETT-68 (p 229) summarises QUICKTRAN like this: "QUICKTRAN was significant from several viewpoints. It was the first on-line system using standard equipment; it retained compatibility with an existing language and thus made possible for the user to debug a program on-line and then to use a regular FORTRAN compiler for batch production runs." See also MARTIN-73 pp 55-58.

3.2 JOSS

JOSS (Johnniac Open Shop System) is a system developed at the RAND Corporation in 1963-64 to run on the JOHNNIAC computer. A goal of JOSS (SAMMETT-68 p 218) was to demonstrate "the value of on-line access to a computer via an appropriate language." "It was designed to give the individual scientist or engineer an

easy, direct way of solving his small numerical problems without a large investment in learning to use an operating system, a compiler and debugging tools or in explaining his problems to a professional computer programmer and in checking the latter's results."

JOSS uses a specially-designed typewriter (BAKER-67) with upper- and lower-case letters, the 10 digits and a set of special characters including (for example) a single character for less than or equal to (a combination of < and =).

There are two modes of entry of statements; a DIRECT mode where statements are executed and an INDIRECT mode where statements are stored to be run as a program. The INDIRECT statements have a line label which consists of numbers which can have a decimal point in them to allow for insertion of lines. A step refers to a single line, e.g. 3.2, while a part refers to all lines with the designated number at the left of the decimal point.

Variables consist of a single upper- or lower- case letter which can have two subscripts separated by commas. The value of subscripts must be between 0 and 99.

Only arithmetic variables are permitted, but they can be either integer or floating point. JOSS stores all numbers internally as floating point.

Operations are +, -, . (for multiply), /, and * (for exponentiation). Conditions can be quite complex as in

```
if a<b<c and d=e
```

The assignment statement is of the form

Set var = expression

The transfer of control is to a step as in

To step 1.35

or to a part as in

To part 4.

Do statements may refer to a single step or to a part; e.g.

Do step n for x = a(b)c

or

Do part n for x = a(b)c

where a and c are initial and terminal values and b the increment. A list is also allowed as in

Do step 3.1 for x=1,2,3,100

A full set of functions are available. Both SAMMETT-68 pp 217-226 and MEADOW-70 pp 293-323 have summaries of JOSS.

3.3 BASIC

BASIC (Beginners All-purpose Symbolic Instruction Code) is a system developed at Dartmouth College in 1965 by Kemeny and Kurtz for the GE 225. It was intended for students as a first language and a stepping stone towards FORTRAN or ALGOL.

As in QUICKTRAN and JOSS there are two modes of statement entry; an immediate and a program-storage mode. A statement number (between 1 and 99999) acts to indicate the sequence of the stored program and as a target for both GOTO and GOSUB statements.

Variables in BASIC are a single letter possibly followed by

a single digit. Single letter variables can have one or two subscripts.

Only arithmetic data is available, but more advanced BASIC systems have features such as string variables and matrix manipulation. (See DATA GENERAL CORPORATION-74, HEWLETT-PACKARD-68).

Assignment statements are of the form

```
LET var = expression
```

The transfer of control statement is of the form

```
GOTO 123
```

A subroutine call is of the form

```
GOSUB 345
```

with its companion

```
RETURN
```

A simple IF statement of the form

```
IF A<B THEN 23
```

transfers control if the condition is met.

The loop control is of the form

```
FOR I=1 TO 9
```

and is ended by its companion

```
NEXT I
```

SAMMETT-68 pp 229-232 discusses the original BASIC system.

3.4 AMTRAN

AMTRAN (Automatic Mathematical TRANslation) is a system developed at NASA in Huntsville, Alabama. The basic objective of the work was to provide an on-line system to facilitate the

solution of mathematical problems by non-professional programmers.

The over-all system was inspired by the work of Culler and Fried and was influenced by JOSS. (See SAMMETT-68 pp 258-264).

The main aim (REINFELDS-72 p 293) of AMTRAN is to exploit the general n-dimensional rectangular array as a basic data structure using operators derived from concepts familiar to a numerical analyst. The language is designed to eliminate as much explicit array component loop writing as possible.

In Fortran, Algol or PL/I, the basic data item is a single entity: number or character or whatever. In AMTRAN the basic data item is an array of entities of the same type.

The following examples are taken from REINFELDS-73 pp 9-14. AMTRAN operands are either numbers or strings. For example

```
XXX = 5
W = 'A STRING'
```

Automatic arrays form the backbone of the AMTRAN system. Memory is allocated dynamically and arrays may be created or changed in dimension without prior declarations. Arrays may be created by concatenation,

```
X=1&2&5, TYPE X
1
2
5
```

or by an ARRAY operator

```
XXX = ARRAY(0,10,5), TYPE XXX
0
2
4
6
8
10
```

Subscripting is performed by a SUB operator. For example

```
W='A STRUNG',W SUB 5 = 'I', TYPE W

A STRING
```

A volume of the ACM SIGPLAN Notices was devoted to AMTRAN (REINFELDS-71).

3.5 APL

ROSEN-72 p 593 describes the history of APL. "The APL language was developed over many years by Kenneth Iverson at the IBM research Laboratories - a development which culminated in the publication of his book in 1962. (IVERSON-62). The language and the book achieved considerable notice, mostly in academic circles. It seemed useful as a publication language for describing hardware and software computing algorithms but, at least originally, was not considered important as a practical programming language. In 1966 Iverson and his colleagues at IBM designed and installed an elegant time-sharing system on an IBM 360/50 based on the use of APL as a programming language. APL differs from most so-called higher-level languages in that it attempts to emulate and exploit the conciseness and elegance of

mathematical notation in the expression of algorithms. It provides many operators on scalars and vectors and matrices, and some of the operators do complicated things. The resulting conciseness is attractive to users of slow terminals, since powerful algorithms can be expressed in a few lines. Some people find this unattractive, claiming that the language is cryptic and confusing, but some, especially sophisticated programmers, love it."

MARTIN-73 p 58 also describes this dichotomy. "The computer world became divided on APL into enthusiasts and detractors. Few persons familiar with the topic remained neutral." "Often the enthusiasm will rival that of a religious convert."

SAMMETT-68 pp 247-252 gives a short summary of the APL/360 language. A fuller description is given in IBM-69.

3.6 The development of the syntax-checking systems.

The three syntax-checking systems which are described in the thesis all indicate syntax errors to the user at the time an invalid character is entered. This is in contrast to compilers which were written to receive input from a card reader. These compilers were usually multipass systems and depended on the entire program being available. With on-line computing, compilers evolved which gave error messages on a line-by-line basis (See LAWRENCE-72 p 590).

The syntax checking systems exploit the full duplex behaviour of a teletype so that syntax errors can be detected on a character-by-character basis. This approach means that the user

can correct his error immediately and messages relating to syntax errors are not needed.

The first of these systems (ACTIV-8) was written in early 1968 for a D.E.C. PDP-8 computer and was a first attempt to exploit the full-duplex behaviour of the teletype. The syntax of the language implemented was kept very simple and the problem of recovery to an earlier point in the same line was avoided. ACTIV-8 had only a program entry mode and the only type of data permitted was real.

Variables were a letter followed by a number. This very simple form made a decision between a command and an assignment statement possible at the second character.

Transfer of control was a GO TO such as

```
GO TO 21
```

and conditional transfer of control could be made by an arithmetic IF such as

```
IF(A3-100) 18,21,28
```

Input to ACTIV-8 during execution allowed expressions including references to defined variables. For example, an input value equal to A3 plus 10% was A3*1.1 and 14 inches could be expressed in centimetres as 2.54*14.

The second system (ACL) was written between 1969 and 1971 with Dr. P.L. Sanger for a D.G.C. NOVA computer. This system implemented a more complex language and allowed an immediate execution mode as well as a program entry mode. Like ACTIV-8, the only type of data permitted was real; although ACL did allow IF

statements similar to the Fortran logical IF.

The third system (XB) was written in 1972 and 1973 for a D.E.C. PDP-11 computer. The language implemented is much more complex than the ACTIV-8 or ACL systems. There are three data types allowed (real, boolean and string) and these may all be combined in one expression such as

```
IF B2 AND R1 GT 2 H3='12'&H2
```

Assignments in XB may occurr at several levels.

```
A1=3*A2=1+A3=1
```

is evaluated from right to left (at assignment level). A3 is given a value of 1; A2 a value of 2 and A1 a value of 6.

Transfer of control is via a GO TO statement like

```
GO TO 12
```

or

```
GO TO I
```

where I has previously been defined or

```
GO TO X(I)
```

which gives a computed GO TO capability.

Conditionals are performed by a logical IF statement of the form

```
IF logical THEN statement ELSE statement
```

or

```
IF logical THEN statement
```

or the THEN may be elided to

```
IF logical statement
```

such as

3 S=S+X(I); IF I=I+1 LE 8 GO TO 3

There is a FOR statement such as

FOR X=150,200 A(X)=0

There is a powerful EDIT statement which allows source language previously entered to be changed.

All these statements and commands are entered under the watchdog checking of the syntax. In the XB system it is not possible to make any syntax errors; on the other hand, any expression which is correct in syntax is permitted.

It is necessary to echo correct characters quickly enough to retain the confidence of the user. In order to do this a method similiar to CONWAY-63 was evolved. Timing tests on the XB system show a complete recovery time of about 8 ms for a line of 72 characters; a per character acceptance time of about 125 microseconds using a PDP 11/50.

A formal grammar approach to language analysis is presented by HIGMAN-73, GRIES-71 and FELDMAN-68. The techniques presented in the thesis allow the syntax checking of a language more general than LR(1) (FELDMAN-68 pp 89-90) as shown in the example on p 15 of the thesis. It is necessary only to determine whether an incoming character is acceptable. The language string entered may be ambiguous (as in the example on p 15 of the thesis) but this is not important - echo checking can still proceed.

The Text Editor for the Nova (DATA GENERAL CORPORATION-73 p 1-1) also exploits the full duplex behaviour of a teletype with its rubout character. If the user types

THE ROW

and he really wanted

THE BROW

he can delete three characters by entering three rubouts. These rubouts echo the character being deleted, so the line will look like

THE ROWWOR

and the user can continue thus

THE ROWWORBROWN DERBY

and the information entered into the system will be

THE BROWN DERBY

On the DEC SYSTEM 10, the system maintains the security of user passwords by not echoing the information back to the teletype. (DIGITAL EQUIPMENT CORPORATION - 1972, p 2-129)

4. References.

ALT, F. L., "Archaeology of Computers - Reminiscences, 1945-1947", Communications of the ACM, Vol. 15, No. 7, July 1972, pp 693-694.

BAKER, C. L., "JOSS: Console design", RAND Corporation, RM-5218-PR, 1967.

BELL, C. G. and NEWELL, A., "Computer Structures: Readings and Examples", McGraw-Hill, 1971.

CONWAY, M. E., "Design of a Separable Transition-Diagram Compiler", Communications of the ACM, Vol. 6, No. 7, July 1963, pp 396-408.

CURRIE, R. L., "PL/I Compared with ALGOL, COBOL and FORTRAN", Proceedings 3rd Australian Computer Conference, Canberra, Australia, 1966, pp 377-379.

DATA GENERAL CORPORATION, "Extended BASIC", 1974.

DATA GENERAL CORPORATION, Nova Text Editor, 1973.

DIGITAL EQUIPMENT CORPORATION, DEC SYSTEM 10 - Users Handbook, Pt. II, 1972.

FELDMAN, J. and GRIES, D., "Translator Writing Systems", Communications of the ACM, Vol. 11, No. 2, February 1968, pp 77-113.

GRIES, D., "Compiler Construction for Digital Computers", Wiley, 1971.

HEWLETT-PACKARD, BASIC LANGUAGE, 1968.

HIGMAN, B., "A Comparative Study of Programming Languages", MacDonald, 1973.

IBM, "APL/360 Primer", C20-1702, 1969.

IVERSON, K.E., "A Programming Language", Wiley, 1962.

LAWRENCE, L. G., "Remote Computing for Program Development", Proceedings 5th Australian Computer Conference, Brisbane, Australia, 1972, pp 588-595.

MARTIN, J., "Design of Man-Computer Dialogues", Prentice-Hall, 1973.

MEADOW, C. T., "Man-Machine Comunication", Wiley 1970.

OSWALD, H., "The Various FORTRANS", DATAMATION, Vol. 10, No. 8, August 1964, pp 25-29.

REINFELDS, J. (Ed), "AMTRAN", ACM SIGPLAN Notices, Vol. 6, No. 11, November 1971.

REINFELDS, J., "AMTRAN 71", Proceedings 5th Australian Computer Conference, Brisbane, Australia, 1972, pp 292-297.

REINFELDS, J., "AMTRAN HANDBOOK", Interactive Computer Systems, Inc., 1973.

ROSEN, S., "Electronic Computers: A Historical Survey", ACM Computing Surveys, Vol. 1, No. 1, March 1969 pp 7-36.

ROSEN, S., "Programming Systems and Languages 1965-1975", Communications of the ACM, Vol. 15, No. 7, July 1972, pp 591-600.

SAMMETT, J. E., "Programming Languages: History and Fundamentals", Prentice-Hall, 1968.

SAMMETT, J. E., "Programming Languages: History and Future", Communications of the ACM, Vol. 15, No. 7, July 1972, pp 601-610.

SPRAGUE, R. E., "A Western View of Computer History", Communications of the ACM, Vol. 15, No. 7, July 1972, pp 686-692.

WILKES, M. V., "Computers Then and Now", Journal of the ACM, Vol. 15, No. 1, January 1968, pp 1-7.

Demon — An Automatic System For Solving Ordinary Differential Equations

By N. W. BENNETT,
Applied Mathematics and Computing Section,
A.A.E.C. Research Establishment, Lucas Heights,
Private Mail Bag, Sutherland, N.S.W.

SUMMARY

Demon is a system for rapid solution of problems involving ordinary differential equations. It accepts differential equations in the D-notation and generates a FORTRAN program to solve the problem. A means is provided for solving boundary equation problems. New integration methods may be defined to DEMON together with the problem. These methods may range from Predictor-Corrector to Gaussian quadrature. Non self-starting methods are automatically started. The DEMON program has been written in COBOL and should be suitable for any large machine. Using the A.A.E.C.'s 32K IBM-7040 at Lucas Heights, N.S.W., the time taken from completely defining the problem to receipt of the results may be as low as three minutes.

1. Introduction

This paper describes a system called DEMON[1] which has been written for a digital computer so that a person can solve problems involving ordinary differential equations much faster than by conventional methods.

The system allows the problem to be described in a language very similar to FORTRAN, and then produces a complete FORTRAN program to solve the given problem.

There are two reasons for the speed of the system. Firstly, problems can be specified in a language which has been oriented towards describing differential equations. The specification is usually very short since it only has to describe the problem and not specify how to solve it. Secondly, speed is gained because the problem is handled entirely by the computer from this stage.

DEMON is rather like a library information retrieval system and depends on a large skeleton program. Parts of this are selected and modified to fit the specified problem and are then moulded into a complete FOTRAN program.

2. Typical Problems and Essential Notation

The following specification is an example of a problem which may be posed to the DEMON system.

```
T = 0.0                    1
Y = 0.0                    2
STEP = 0.01                3
ENTER DEMON                4
(T = 1.)                   5
FUNCTION                   6
DY = 1.0 + Y**2            7
END DEMON                  8
CALL EXIT                  9
END                       10
```

The problem described is "Integrate the differential equation (7)

$$\frac{dy}{dt} = 1 + y^2,$$

with initial values t = 0.0 (1) and y = 0.0 (2) using a step size of 0.01 (3). When the point t = 1. (5) is reached, pass control back to the monitor (9)."

The problem has thus been described in 10 statements, or which four are directions to the system.

The eventual solution given to this problem is the numerical values of T and Y, which are printed with identifying format. For example, the first line of the output would be:

T = 0.0 Y = 0.0

The total time taken to solve this problem from the specification was 204 seconds on the IBM-7040 at Lucas Heights. This was composed of:

generation	64 seconds
compilation and loading	66 seconds
execution	4 seconds.

The discrepancy in the total is due to dead time, such as tape rewinding.

The language for describing differential equations, relies heavily on two devices of notation.

The first is the D-notation which is well accepted and may be punched on a card for input to the machine. The following example illustrates the use of the D-notation in DEMON:

a = 1 + y × z'	A = 1. + Y*DZ
y' = z + sin (a)	DY = Z + SIN (A)
z'' = —y + z'	D2Z = —Y + DZ

Conventional FORTRAN may be mixed with the differential equations. Suppose a differential equation changes when T is equal to A. This may be described by:

```
      IF (T —A) 1, 2, 2
1  DY = F (T, Y)
      GO TO 3
2  DY = G (T, Y)
3  CONTINUE.
```

Any number of differential equations of differing order will be accepted by the DEMON system.

The second device of notation is used to refer to points along the integration path. The statement

(T = 1.)

was used in the sample problem to indicate the last point of the integration. More complex points may be described; for example

(X*X + Y*Y = 25.)

indicates the point on the integration path which is five units from the origin.

Points that occur more than once may also be described. The notation:

(Y = 0.0, 3)

describes the point where y is zero for the third time.

The notation for describing points may also be extended for describing boundary equations. The following problem illustrates this.

```
T = 0.0                              1
Y = 0.0                              2
A = 1.                               3
STEP = 0.01                          4
ENTER DEMON                          5
ALTER (A)                            6
Y (T = 1.) = 1.5, .001               7
FUNCTION                             8
DY = 1. + A*Y**2                     9
END DEMON                           10
CALL EXIT                           11
END                                 12
```

This problem is very similar to the first example. The differential equation (9) has a slightly different form. A boundary equation states that A should be altered (6) so that y will be 1.5 when t is equal to one. An initial estimate of A has been given (3).

A variation of Newton's method is used in the generated program to find the required value of A.

If the first integration does not satisfy the boundary equation to within a certain tolerance (as shown on statement 7), another integration is made with a slightly changed value of A. From these two integrations, an extrapolation may be made to a better approximation for A. This cycle is then repeated until the boundary equation is satisfied.

The total time taken to run this problem on the IBM-7040 at Lucas Heights was 229 seconds.

DEMON has been designed to solve any number of boundary equations, but no guarantee can be given that the process will converge for every problem.

3. Systems Language

The DEMON generator, which produces the FORTRAN program, has been written in COBOL and consists of six thousand statements.

There are several reasons for choosing the COBOL language. The most important was its ability to handle characters and strings of characters. The following statements illustrate this:

```
IF CN (CNPT) = ' ( ' ADD 1 TO BC.
MOVE 'GO TO 4108' TO CARD-AREA.
```

With such a large program, it was also important to have it written in a language which is independent of the machine.

FORTRAN was chosen for the generated program because it is the major scientific computing language used at Lucas Heights. The generated program is built up from only the most basic FORTRAN, so that the entire DEMON system is machine independent.

4. Implementation of Demon

Since the program generator is much slower than the FORTRAN compiler, some interesting compromises had to be made. The generated program may contain the following statements:

```
GO TO 4108
4108 CONTINUE.
```

It turns out that it is much faster in total time to leave these redundant statements in the program rather than attempt to remove them.

These statements were created because the generated program must be correct for any combination of its basic building blocks. In a different combination there may be a block of statements between the two shown and statement 4108 may be the start of another block. However, since statement 4108 has been introduced, a redundant statement has to be used.

DEMON has been in use on the 32K IBM-7040 at Lucas Heights since April 1965. Problems using DEMON are run in one job by generating the FORTRAN program on to a scratch tape and then compiling and executing the program from this tape.

5. Other Applications of Demon

Three more features of DEMON have proved to be very useful.

The first is that new integration methods may be defined to DEMON with the problem. The way in which

$$y' = f(y)$$

should be integrated is described; from this the method may be applied to the particular problem.

Both self-starting and non-self-starting methods may be defined. The non-self-starting methods are automatically started by DEMON. The methods may range from simple Runge Kutta to high order Predictor-corrector methods which change their step size. An outline of the definition of integration methods is given in the Appendix.

Sometimes sets of differential equations have a common structure. These may be described to DEMON by using dimensioned differential equations in a DO-loop. For example

```
DO 1 K = 2, 99
1 DX(K) = C*(X(K + 1) —2.*X(K) + X(K —1))/DELX**2
```

may be used to describe the flow of heat in a bar of metal. The equations of objects under mutual gravitational fields may also be cast in this form.

DEMON may also be used to solve problems not really involving differential equations. The best example of this is in non-linear equations.

Consider the equations

$$xy = .25$$
$$(x —1)^2 + y^2 = 1.$$

The problem of solving these equations may be described to DEMON as follows:

```
T = 0.0                                  1
X = 1.0                                  2
Y = 1.0                                  3
STEP = 10.                               4
ENTER DEMON                              5
ALTER (X)                                6
ALTER (Y)                                7
R (T = 0.0) = 0.0, .0001                 8
P (T = 0.0) = 0.0, .0001                 9
RELAX (.7)                               10
FUNCTION                                 11
R = X*Y —.25                             12
P = (X —1.)**2 + Y**2 —1.                13
END DEMON                                14
CALL EXIT                                15
END                                      16
```

Here the equations have been posed as boundary equations for a trivial differential equation. The resulting program gives a Newton's method solution to the equations. The RELAX statement (10) gives the fraction of the estimated correction which should be applied; this is done to aid convergence. This particular problem converged to the required accuracy in 9 iterations and the total machine time was 235 seconds.

6. Conclusion

Programs produced by DEMON are usually slightly slower and larger than an equivalent hand-written program. However, since the DEMON produced program is obtained very simply and invariably compiles correctly, results may be produced very much faster by using DEMON. This has been confirmed by experience in using DEMON at Lucas Heights.

7. Reference

1. Bennett, N. W. (1965). — DEMON — A program generator for problems involving ordinary differential equations. AAEC/E142.

APPENDIX

New integration methods may be described to DEMON. A simplified explanation of the process is given here. The method described treats vectors directly. In practice DO-statements and subscripted variables would be used.

The second order Runge Kutta method illustrates the basic features of the technique:

```
DY2 = FUNCTION (Y2)                      1
LOOP                                     2
ROTATE (Y1, Y2) (DY1, DY2)               3
P = Y1 + STEP*DY1                        4
DP = FUNCTION (P)                        5
Y2 = Y1 + .5*STEP*(DY1 + DP)             6
DY2 = FUNCTION (Y2)                      7
END                                      8
```

This description shows how Y2 and DY2 may be calculated from Y1 and DY1.

In the beginning the initial values are stored in Y2. Statement 1 indicates that the derivative of Y2 should be evaluated and placed in DY2. Statements 5 and 7 have a similar indication. The LOOP statement (2) indicates the entry point for a new integration step. Statement 3 places the information Y2 and DY2 into Y1 and DY1 so that a new integration step can be made. This second order Kunge Kutta is the standard method used in DEMON.

A non-self-starting method may also be used. The method

$$y_2 = 5y_0 —4y_1 + 2hy'_0 + 4hy'_1$$

illustrates this:

```
DY2 = FUNCTION (Y2)                                  1
EXAMINE                                              2
ROTATE (YZ, Y1, Y2) (DYZ, DY1, DY2)                  3
CALCULATE (Y1, DY1, Y2, DY2)                         4
LOOP                                                 5
ROTATE (YZ, Y1, Y2) (DYZ, DY1, DY2)                  6
Y2 = 5.*YZ —4.*Y1 + 2.*STEP*(DYZ + 2.*DY1)           7
DY2 = FUNCTION (Y2)                                  8
END                                                  9
```

The EXAMINE statement (2) indicates that the values in Y2 and DY2 are complete and may be used for output and other purposes. Statement 4 is translated as a call to the self-starting method.

It is also possible to describe Gaussian quadrature methods and methods that change their step size.

ABACUS—A Fast Fortran System For The IBM/360

By N. W. Bennett*

ABACUS is a fast FORTRAN system for the IBM 360 computer. The U.S.A.S.I. standard FORTRAN has been implemented without any extensions, to provide a reference point for teaching and programme interchange. A large proportion of the system is devoted to the detection of errors and the production of diagnostics. A simple monitor ensures rapid transition from one job to the next.

Introduction

The IBM 360/50 computer system is supported by a large operating system, which includes many language compilers. With such an operating system there is an inevitable overhead in recognising a job being presented, invoking the appropriate compiler and running the compiled program. For long-running jobs this overhead is insignificant, but for small jobs overhead can be the major part of the job time. Compilation of a small program may also be a large part of the job time.

For these reasons it is desirable to have an additional compiler which can compile a small program as quickly as possible, execute it and proceed to the next job with a minimum of delay. This report describes such a compiler called ABACUS.

The easiest way to keep overhead down is to restrict a stream of jobs to the one language and to keep the compiler resident in the machine from one job to the next. With this type of compiler a single pass of the source statements is made, and the compiled program is generated into a reserved area of core storage where it can be executed. Programs in compiled form should be neither produced nor accepted because compilation would be slowed down to produce an object deck and input with an object deck would be only marginally faster than with the original source deck.

An alternative of generating a program onto disc and then reading it back into the machine was rejected because of the overhead involved and the possibility of a future expansion of the core storage.

Naturally, if a compiler is written to be as fast as possible the compiled program cannot be as efficient as that produced by a compiler without this bias. Also if the compiler is resident during execution the space available to the object program is less. Many systems written along these lines are now available, but when the ABACUS project was started in August 1966, there was no fast FORTRAN compiler available for the 360 computer.

Choice of FORTRAN

FORTRAN is the language most used by nuclear research establishments throughout the world, for both domestic use and for program interchange. It is used at the A.A.E.C.'s Research Establishment, Lucas Heights and it was obvious that any fast processor for small jobs should use the FORTRAN language.

The United States of America Standards Institute (1966) has approved a standard for FORTRAN which has been set so that a FORTRAN program may be written for one machine and then run on another, if desired. Many computer companies have implemented this standard FORTRAN for their particular machine, but in doing this, a lot of extensions have been added to the standard. Some of these extensions have filled a genuine need in the language, but others have introduced discrepancies which the standard was intended to remove. One of the worst of these is an extension made by two companies which looks the same, but has different meanings. Honeywell Inc. (1966) interpret A**B**C as (A**B)**C, whereas IBM (1966) regard it as A**(B**C).

It was decided to write a fast FORTRAN compiler for the 360 and to implement the complete standard FORTRAN or as large a subset of it as possible. Because of space limitations it was not possible to implement the COMPLEX data type or the EXTERNAL statement. Also the EXTERNAL statement creates complications which the author considered outweighed its usefulness. The PAUSE statement was not implemented because it conflicts with the concept of fast job processing.

Two popular extensions to the standard were implemented, but these may be deleted by use of a compile option as suggested by the American Nuclear Society (Communications of the A.C.M. 1966). These extensions exist in the IBM FORTRAN system, so their inclusion does not preclude testing a job with ABACUS, and then running it for much longer periods using the IBM system. The extensions are

(1) The 'END=' option in the READ statement which gives the address of an end of file routine.
(2) The use of quotes to delimit Hollerith fields within FORMAT statements.

Description of the 360 at Lucas Heights

The IBM 360/50 at Lucas Heights has 256K bytes

* *Australian Atomic Energy Commission.* *Manuscript received September, 1968.*

of core storage. These bytes are eight bit characters. A single word (or one storage unit as defined by the standard) consists of 4 bytes. A card reader, card punch and printer are attached on a multiplexor channel, with 4 discs and 2 tapes connected to two selector channels. The ABACUS system resides on a disc, but once it has been read into core storage, the discs and tapes are ignored.

The ABACUS system is organized along the lines already outlined. It consists of over 150 assembler subroutines which take up 162K bytes. The symbol table uses 12K bytes, the compiled program may use 16K bytes for variables and arrays, 8K bytes are reserved for storage of constants and addresses, the compiled program may be up to 24K bytes long, and the rest of the machine (34K) is used by the IBM operating system.

The desirability of having a single pass over the source language forces a requirement on the order in which specification statements may be presented. This required order is:—

(1) DIMENSION or type statements
(2) COMMON statements
(3) EQUIVALENCE statements.

The DIMENSION and type statements must precede COMMON statements so that addresses in COMMON can be assigned while the COMMON statement is being scanned. Similarly, the DIMENSION and COMMON statements must precede the EQUIVALENCE statements so that a complete description of an array or variable is available for processing the equivalence relationships.

Addressing on the 360

The 360 system has been designed for a maximum core storage of 16M bytes. To reference a position in this maximum size store requires a 24 bit address and the designers considered a 24 bit address in every memory reference instruction to be wasteful of storage space, particularly when an actual store of 256K requires only an 18 bit address. Instead, memory reference instructions contain a reference to one of 16 general purpose registers and also a positive displacement of 12 bits. When referenced in this way, the general purpose register is called a base register. The address to memory is obtained by adding the displacement to the contents of the base register. With this addressing scheme an origin in a base register can only be used to address an expanse of 4096 bytes. To address a location outside this expanse, the origin in the base register must be changed, or another base register containing a suitable origin must be used.

This addressing structure creates problems in the organisation of the compiled program. Firstly, reference must be made to different points in the compiled program; to statement numbers, subroutine entries, etc. These references are made only occasionally and in ABACUS an address is loaded into a register which is used as a base. This base is then used by an instruction with a zero displacement. This approach is also needed because of the single pass structure of the system.

Reference to variables is a more serious problem. Execution would be very slow if a double reference were needed for each variable. Because of the small amount (4K words) of storage available to the compiled program, a much faster and more pleasing approach may be taken.

The base register designation in the instruction is to the left of the displacement. Therefore if an increment were made to the displacement in an instruction, any overflow would cause the numerically next higher register to be indicated. Now this overflow does not matter if the next register contains an address 4096 higher than the first. In effect, by this arrangement the displacement within the instruction can be extended. Six of the 16 registers are used in ABACUS to reference the 4K words of data and 2K words of pointers to statement numbers, subroutine entries, etc.

When extra core storage becomes available for the 360/50 at Lucas Heights, the complete standard FORTRAN will be implemented. The addressing scheme described above is valuable mainly for variables. Currently the 4K words of data contain both variables and arrays. The arrays could be taken out of this region with a small loss in performance due to double referencing those array elements which have absolute subscripts. Array elements whose subscripts have to be calculated are double referenced either way. For variables in COMMON and variables in equivalence with array elements, a double reference addressing scheme would be needed.

The Blank Character in FORTRAN

In most statements in FORTRAN (including standard FORTRAN) the blank character has no meaning and may be used freely to improve the appearance of the program. The author feels that blanks imbedded in words, or words without a blank separating them, do not improve the appearance of a program. A compiler can be more easily written to recognise individual statements in the FORTRAN language if the restrictions suggested above are made. As an example, consider the two statements:—

```
DO 2 J = 1,3
DO 2 J = 1.3
```

For compilers which are tolerant to blanks the first statement is a DO statement and the second is an arithmetic statement. However, considerable analysis of one statement must be made before it is reclassified as a statement of the other type. For ABACUS the first statement is a valid DO statement and the second is an invalid DO statement. This approach has led to some conflicts. People used to writing

```
GOTO28
```

are not pleased at having the statement rejected, but why should all programs be compiled more slowly because a few users have poor punctuation?

Treatment of Errors—Compilation

One of the aims of ABACUS was to produce good error messages. During compilation an extensive check is made of the statements. For all errors de-

tected a full message is produced, with a pointer to a card column where needed. The parts of the compiler used to check for errors and all the possible error messages take up a large amount of storage. Further action depends on the type of error. If the error is in an executable statement, instructions are generated so that execution will be terminated if control reaches that point. If the error is in a specification statement, it will affect the entire program unit. If the error is in the mainline program, execution is deleted; if it is in a sub-program, execution will be terminated on entry. For statement numbers missing or subprograms missing, their address link is filled in so that control will pass to an error routine if they are referenced.

Treatment of Errors—Execution

The operating system must consider the invocation of the batch processor as a single job. If the operating system is allowed to follow its usual course when, say, an overflow occurs, the remaining jobs in the batch may be flushed. Because of this unsympathetic behaviour of the operating system and because so few of its features are used, ABACUS has its own small operating system to handle input-output, timing and interrupts.

There are two modes of operation on the 360. One is a supervisor state where all instructions are legal, and the second is a problem state where instructions such as input-output are illegal. A special instruction, supervisor call, is provided for a program in the problem state to request supervisor services. A special supervisor call, SVC 255, has been implemented at Lucas Heights to allow a program in problem state to enter the supervisor state. This feature is used by ABACUS to give control to its own operating system.

Errors during execution can be divided into errors which any program might make, such as an overflow, and those which violate the rules of FORTRAN, such as the subscript for an array element being out of range. The first type of error causes an interrupt and information needed to diagnose the fault is available to the system.

It is much more difficult to detect the second type of error and gather information about it. The program produced by ABACUS contains checks for this type of error and, if one should occur, provides the information necessary to diagnose it. One example of this is the array subscripts already mentioned. Another is a check that none of the parameters of a DO statement is changed during its range This is achieved by making a copy of the parameters at the start of the loop and comparing them at the end of the loop.

The Disassembler

A disassembler is part of the ABACUS system and this is a program which can provide an assembler language listing from basic machine language. The disassembler was used during development of the system to check the compiled program. This feature which takes up little space, was invaluable to the author when checking the generated program. The disassembler is used in the completed program to give an assembler language listing of the compiled program (if the user desires), and also to give a list of any statement which signals an error during execution. Several tables are available to the disassembler in order to make the listing more readable. The first of these is the symbol table, which is retained throughout the job. Addresses found in instructions can be used to find the relevant variable name in the symbol table; this name may then be included in the listing. The symbol table entry also gives the variable type. This means the current value of the variable may be given in the appropriate format. Many instructions are only generated in unique circumstances. A table of these is available to the disassembler and when they occur a relevant comment can be included in the listing. During compilation a table containing the lengths of individual statements is built; this is used later to separate the compiled program into the original statements for listing.

The Epilogue

At the end of execution an epilogue is given which is a summary of the execution giving the time taken, the number of cards read, the number of lines printed and the number of cards punched. The last card read and the last card punched are reproduced in the epilogue. The reasons for termination are given. This may merely be an indication that control reached a STOP statement or it may be much longer. Suppose the error was an overflow in an arithmetic statement. Using the address of the overflow, the disassembler is able to list the complete arithmetic statement in question. As mentioned before, this list will contain the names of the variables in the statement and their values. Every fault is handled in this way. A message gives the nature of the fault and the list of the relevant statement gives added detailed information. If the fault occurs in a subprogram, the path back to the mainline program is followed, giving a list of each call statement, including the value of all arguments.

A compile option is available to give the program's path prior to termination. The compiled program has an instruction inserted before the normal code for each statement. During execution this instruction calls a trace subroutine which records the latest 1000 return addresses and counts the number of calls. This count and these addresses are given in the epilogue. If the program had been in an endless loop, the actual loop could be determined by an analysis of the statement locations. This could also be done by the system, but lack of space prevents its inclusion in ABACUS. Another option is available to list the value of all variables and arrays after execution. This provides a valuable aid to locating errors.

The Batch Control Card

It is usual for a system to have different options available. Among these may be the option to produce a list of the variables and statement numbers in the program, or to execute not longer than a set time. So that every job need not explicitly state all the options, a standard set of options is normally adopted. If a job does not state a particular option, the relevant

standard state can be taken. This standard set of options is usually built into a system, but with ABACUS a standard state may be set for a batch of jobs. This is done with a control card which precedes the batch and also specifies if an option may be changed by an individual job. There are two classes of options. The first contains qualitative options, such as a list of the variables and statement numbers. The control card states the standard option and also whether it may be changed. If it may not be changed, the user's attempt to do so will be fruitless. The second class contains quantitative options, such as a time limit for execution. As well as the standard length of time, the control card sets a maximum time which the user cannot exceed. Suppose the standard limit is 10 secs. and the maximum limit 60 secs. If the individual job does not set its own limit, 10 secs. will be taken. If the job specifies any time up to 60 secs. this will be allowed, but any attempt to request more than the 60 secs. will be taken as 60 secs. only.

This concept of a maximum limit also applies to all quantitative options such as the number of lines printed and the number of cards punched.

Performance

ABACUS will run each small job in approximately 2 seconds. This compares very favourably with the equivalent IBM time of 61 secs. Naturally, the execution speed of ABACUS is not as good as the longer compiling IBM system; this is also due to the time required for checking errors. For jobs which use many subscripted variables, the comparative speed may be about 35 per cent, but a carefully written program using only simple variables may execute at about 95 per cent. The size of the source program naturally varies with the complexity of the statements. If the statements were all as simple as

$$A = B + C$$

then ABACUS could handle a program with two thousand statements. The realistic limit would be about 500 to 700 statements. It has been very worthwhile to include the check for execution errors with the production of full error messages. Users of the ABACUS system have commented that errors are well diagnosed and that this has enabled them to correct their programs more quickly.

Conclusions

Possibly the most valuable lesson learnt from writing ABACUS is that detecting errors and producing good diagnostics takes up a lot of space. Apart from the fact that parts of standard FORTRAN are not implemented because of space limitations, the design aims of the project have been realised. The time taken to process small jobs has been greatly reduced from that required by the IBM system. The time required to find and correct errors has been reduced by the provision of execution checking and good messages.

References

Communications of the A.C.M. (1966)—Letter to the Editor, Program Exchange in Nuclear Science and Engineering Applications. *Comm. Assoc. Comp. Mach.* Vol. 9, No. 9, p. 698.

Honeywell Inc. (1966)—*Software Manual*, Series 200, Fortran Compiler D, 027, pages 2-2 and 2-3, Honeywell EDP Division, Massachusetts.

I.B.M. Corporation (1966)—IBM System 360 Fortran IV Language, File No. S360-25, Form C28-6515-4, page 19, IBM Corporation, New York.

United States of America Standards Institute (1966)—*U.S.A. Standard Fortran*, X3.9-1966.