AN ABSTRACT OF THE THESIS OF

SUSAN PAULA MARGOLIS for the MASTER OF ARTS
(Name of student) (Degree)

in Mathematics presented on December 17, 1969
(Major) (Date)

Title: A TURING MACHINE PROGRAMMING LANGUAGE

Abstract approved: Redacted for Privacy
(Professor Harry Goheen)

This thesis documents a new language which facilitates the construction of Turing machines. The language translator is written in Compass and has been debugged and is available for use on the CDC 3300.

A Turing Machine Programming Language

by

Susan Paula Margolis

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Arts

June 1970

APPROVED:

Redacted for Privacy

Professor of Mathematics
in charge of major

Redacted for Privacy

Acting Chairman of Department of Mathematics

Redacted for Privacy

Dean of Graduate School

Date thesis is presented December 17, 1969

Typed by Charlene Laski for Susan Paula Margolis

ACKNOWLEDGEMENTS

I would like to thank Professor H.E. Goheen for his help and encouragement in the development of this thesis.

I would like to thank Charlene Laski for typing the thesis.

TABLE OF CONTENTS

LIST OF FIGURES

A TURING MACHINE PROGRAMMING LANGUAGE

I. INTRODUCTION

This thesis develops a symbolic language to facilitate the construction of Turing machines. By now, the importance of Turing machines is well-known. They are recognized as an excellent tool in constructive proof theory, as well as an important source of examples in automata theory. For an introduction to Turing machine theory, see Davis (4), Hermes (5). For important examples of the power of Turing machines see Kleene (6) and Minsky (9).

Existing techniques for constructing Turing machines are inadequate. Construction of symbolic languages for processing automata is an important but neglected area of automatic programming. As Knuth has noted "... most current papers on automata theory have not advanced past the 'machine language' programming stage; authors have been reluctant to simplify their constructions except by using classical mathematical notations which are cumbersome for this purpose.

Symbolic programs are considerably easier to write and to read, and they are less likely to contain careless errors. So it is surprising that these ideas have not already spread from computer programming to automata programming."[1]

The only previous Turing machine simulator language in the literature belongs to M.W. Curtis (3). Curtis's language, written for the IBM 1620, seems to have all the failings noted by Knuth above. It is a language at the machine level and employs mnemonics quite divorced from the structure of Turing machine operations.

The language to be described is a higher level symbolic language, which has been implemented on the CDC 3300. A program processed in this language yields as output a Turing machine in a form usable by the TASP[2] compiler (if desired) as well as a printout of the resulting quintuples.

To test the processor, Watanabe's Universal Turing machine (12) is encoded in this language, and the resulting program may be compared with the encoding of a universal Turing machine by Curtis, who also used Watanabe's version.

The possibility of encoding a Universal Turing machine gives a proof of the universality of the language in encoding algorithms. This fact is also proved directly in Chapter 3.

---

1. Knuth, D.E. and R.H. Bigelow. Programming language for Automata. Journal of the Association of Computing Machines. 14:615. 1967.

2. TASP is a Turing Automation Simulation Program.

II. DESCRIPTION

This section consists of an informal description of the programming language, followed by rigorous definition of allowable syntactical structures using the notation of Backus-Naur (2).

In order to describe the language, we shall study it in connection with an example. Consider the problem of devising an algorithm to transform arithmetic expressions into Lukasiewicz Normal Form (8). A Turing machine which will perform such an algorithm is given below:

	H	D	S	(	X	Y	Z	)	+	-	*	/
1	2RH	1RS	2RH		1RX	1RY	1RZ	1RH	1R+	1R-	1R*	1R/
2	3L)	2RD	!	2R(	11LX	11LY	11LZ	3L)	2R+	2R-	2R*	2R/
3	1RS	3LD		9RD					4RD	5RD	6RD	7RD
4	4RH	4RD	8L+	4R(	4RX	4RY	4RZ	4R)	4R+	4R-	4R*	4R/
5	5RH	5RD	8L-	5R(	5RX	5RY	5RZ	5R)	5R+	5R-	5R*	5R/
6	6RH	6RD	8L*	6R(	6RX	6RY	6RZ	6R)	6R+	6R-	6R*	6R/
7	7RH	7RD	8L/	7R(	7RX	7RY	7RZ	7R)	7R+	7R-	7R*	7R/
8	8LH	2RD		8R(	8RX	8RY	8RZ	8L)	8R+	8L-	8L*	8L/
9		9RD			13RD	14RD	15RD	10RD	9R+	9R-		
10	3L)	2R*		10L(	10LX	10LY	10LZ	3L)	2R+	2R-	2R*	2R/
11	9RH	11LD		9R(					12L+	12L-	9R*	9R/
12	9RH	12LD		9R(					4RD	5RD	6RD	7RD
13	13RH		16LX	13R(	13RX	13RY	13RZ	13R)	13R+	13R-	13R*	13R/
14	14RH		16LY	14R(	14RX	14RY	14RZ	14R)	14R+	14R-	14R*	14R/
15	15RH		16LZ	15R(	15RX	15RY	15RZ	15R)	15R+	15R-	15R*	15R/
16	16LH	10RD		16L(	16RX	16LY	16LZ	16L)	16L+	16L-	16L*	16L/

Figure 1

The above assumes the arithmetic expression is orginally given in the standard form with delimiter "h" at each end. We also follow the convention that the Turing machine is to commence its operation in State 1, looking at the left delimiter "h", and that the expression involves only the three variables x,y, and z; the four operations, viz. multiplication, "*", addition, "+", subtraction, "-", and division, "/", and left parenthesis, "(", and right parenthesis, ")".

One may describe the Turing machine of Figure 1 in the following terms. For purposes of description consider the program divided into a number of blocks; first the heading block followed by a number of state blocks and finally the terminator block. In the heading block, the first card is the title card which is read by the processor and then printed at the top of each page of output with the time and date of the run. The next card is the Turing machine card. This card is necessary since more than one Turing machine may be processed in the same run.

The third card of the deck is the alphabet card. This card designates to the processor the set of characters which the Turing machine will recognize as its alphabet. The choice of characters for the Turing machine alphabet is limited to 47 of the 48 Standard Keypunch Characters ABC ... XYZ 012...9 + - / * . , = ( ) $ the 48th character (space or blank) being reserved as a separator.

Next follows the state blocks. Each of these blocks is equivalent to a Turing machine state as described by its set of quintuples in that state.

The last card of the deck is the termination card END. This card denotes the end of compilation. At this point, if no errors have been detected, the quintuples are printed and the TASP processor is called. If any error is detected an appropriate message is printed along with the line number of the place of error detection and compilation is terminated.

A useful feature of this language is that state blocks correspond in a one-to-one fashion with states in the compiled Turing machine. The faithfulness of the number of states demonstrates the usefulness of this language in researches involving minimal state machines.

In terms of the above discussion the Turing machine of Figure 1 is presented in Figure 2.

Figure 2.

A TRANSFORMER OF ARITHMETIC EXPRESSION TO POLISH STRINGS

```
       ( A LIST IS FORMED TO THE RIGHT OF THE INPUT STRING
       |   WHICH WHEN DONE CONTAINS THE POLISH STRING
HEAD   |
BLOCK <  TURING MACHINE 1
       |
       ( ALPHABET H D S ( X Y Z ) + - * /
```

(continued)

```
         STATE 1  BEGIN AND CLEAN UP AT THE END
         SEARCH RIGHT FOR H AND D AND S AND )
              H GO RIGHT TO STATE 2
STATE         D CHANGE TO S
BLOCK         S CHANGE TO H AND GO RIGHT TO STATE 2
              ) CHANGE TO H
         STATE 2  LOOK FOR THE FIRST X Y Z OR H
         SEARCH RIGHT FOR H AND S AND X AND Y AND Z AND )
              H CHANGE TO ) AND GO LEFT TO STATE 3
              S GO TO STOP
              X GO LEFT TO 11
              Y GO LEFT TO 11
              Z GO LEFT TO 11
              ) GO LEFT TO 3
         STATE 3  LOOK FOR FIRST OPERATOR
         SEARCH RIGHT FOR H ( + - * /
              H CHANGE TO S AND GO R TO 1
              ( CHANGE TO D AND GO R TO 9
              + CHANGE TO D AND GO R TO 4
              - CHANGE TO D AND GO R TO 5
              * CHANGE TO D AND GO R TO 6
              / CHANGE TO D AND GO R TO 7
         STATE 4  PUT THE + ON THE OUTPUT SIDE
         SEARCH RIGHT FOR S
              S CHANGE TO + AND GO LEFT TO 8
```

(continued)

```
STATE 5  PUT THE - ON THE OUTPUT SIDE

SEARCH RIGHT FOR S

     S CHANGE TO - AND GO LEFT TO 8

STATE 6  PUT THE * ON THE OUTPUT SIDE

SEARCH RIGHT FOR S

     S CHANGE TO * AND GO LEFT TO 8

STATE 7  PUT THE / ON THE OUTPUT SIDE

SEARCH RIGHT FOR S

     S CHANGE TO / AND GO LEFT TO 8

STATE 8  GO BACK TO INPUT STRING

SEARCH LEFT FOR D

     D GO RIGHT TO 2

STATE 9  LOOK FOR THE VARIABLES IN INPUT AND
         REMOVE THEM

SEARCH RIGHT FOR X Y Z )

     X CHANGE TO D AND GO RIGHT TO 13

     Y CHANGE TO D AND GO RIGHT TO 14

     Z CHANGE TO D AND GO RIGHT TO 15

     ) CHANGE TO D AND GO RIGHT TO 10

STATE 10  * AND / WRITTEN AFTER PRECEDENCE RULES
          EXECUTED

SEARCH LEFT FOR H ( + - * /

     H CHANGE TO ) AND GO LEFT TO 3

     D CHANGE TO * AND GO RIGHT TO 2

     + GO R TO 2

     - GO R TO 2
```

(continued)

```
     * GO R TO 2

     / GO R TO 2

STATE 11  FIND FIRST OPERATOR AFTER A VARIABLE

SEARCH LEFT FOR H ( + - * /

     H GO R TO 9

     ( GO R TO 9

     + GO L TO 12

     - GO L TO 12

     * GO R TO 9

     / GO R TO 9

STATE 12  PRECEDENCE CHECK ON OPERATORS + AND -
          WRITTEN

SEARCH LEFT FOR H ( + - * /

     H GO R TO 9

     ( GO R TO 9

     + CHANGE TO D AND GO R TO 4

     - CHANGE TO D AND GO R TO 5

     * CHANGE TO D AND GO R TO 6

     / CHANGE TO D AND GO R TO 7

STATE 13  PUT THE X ON THE OUTPUT SIDE

SEARCH R FOR S

     S CHANGE TO X AND GO LEFT TO 16

STATE 14  PUT THE Y ON THE OUTPUT SIDE

SEARCH R FOR S

     S CHANGE TO Y AND GO LEFT TO 16
```

(continued)

```
STATE 15  PUT THE Z ON OUTPUT SIDE
SEARCH R FOR S
     S CHANGE TO Z AND GO LEFT TO 16
STATE 16  GO BACK TO INPUT STRING
SEARCH L FOR D
     D GO R TO 10
END
```

Syntatical Construction

1. Basic Symbols
<external alphabet character> :: = <letter>|<digit>|
<special character>

1.1 <letter> :: = A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|
Q|R|S|T|U|V|W|X|Y|Z

1.2 <digit> :: = 0|1|2|3|4|5|6|7|8|9|

1.3 <special character> :: = +|-|/|*|.|,|=|(|)|$

2. <delimiter> :: = <operator>|<separator>|<bracket>|
<declarator>

2.1 <operator> :: = search|go|change|changing

2.2 <separator> :: = for|to|and|<blank>

2.3 <bracket> :: = state|Turing machine|end

2.4 <declaration> :: = alphabet

3. <expression> :: = <search expression>|<replacement
expression>|<transfer expression>

3.1 <search expression> :: = search<direction>for
<list>

3.1.1 <direction> :: = left|right|R|L|

3.1.2 <list> :: = <external alphabet character>|
<list><blank><external alphabet character>

3.2 <replacement statement> :: = <u>changing</u> <external alphabet character> <u>to</u> <external alphabet character> | <u>changing</u> <u>rest</u> <u>to</u> <external alphabet character>

<replacement statement 2> :: = <external alphabet character> <u>change</u> <u>to</u> <external alphabet character>

3.3 <transfer expression> :: = <u>go</u> <direction> <u>to</u> <label> | go to <label>

3.3.1 <label> :: = <u>state</u> <number>|<number>| <u>stop</u>

<number>* :: = <digit>|<number><digit>

*restricted to three digits by our TASP processor

4. Syntax

4.1 <basic statement> :: = <search expression>|
<search expression><replacement expression 1>

4.2 <statement> :: = <transfer statement>|
<replacement statement>

4.2.1 <transfer statement> :: = <external alphabet character><transfer expression>

4.2.2 <replacement statement> :: = <replacement

expression 2>|<replacement expression 2>
and <transfer expression>|<replacement
expression 2><blank><transfer expression>

4.3 <machine block> :: = <block head><declaration>
<state block>|<machine block><state block>|
<machine block><machine block>

4.3.1 <block head> :: = Turing machine <number>
(comment)

4.3.2 <declaration> :: = alphabet <list>

4.3.3 <state block> :: = <block label><basic
statement><statement>|<state block>
<statement>

4.3.3.1 <block label> :: = state
<number> (comment)

4.4 <program> :: = <machine block> end

III. IMPLEMENTATION OF THE LANGUAGE

Memory Utilization by the Translator.

The function of the translator is to create a disk file of quintuples for the TASP (11) processor from each input source program. Usually, the source input consists of a card deck of program blocks as described in Chapter 1 preceded by control cards in the format for OS3 (10) whose function it is to load the translator into the machine memory. However, by using other control card options, the source may be put in by other input devices.

The translator uses 73 file blocks of core storage, which is then overlayed, by the TASP processor (11), after the compilation of the quintuple list.

Translator-time Error Detection.

The translator prints out diagnostics after each detection of a source language error. Translation continues however, until the terminator card is reached or until 125 errors have been detected, which ever event occurs first. After detection of the first error, the TASP processor call is suppressed.

Translator Outputs.

The translator output is normally given to the standard system output file, the printer. A file for the

TASP processor is also created. The output for both files consists of the quintuples of the Turing machine which has been complied. The programmer may request the output on a different device by using other control card options.

Only columns 1 to 72 of each card are interpreted by the translator. The other eight columns are available to the user as a label field, comment field, or sequence field. The Turing machine card, alphabet card, and state card must begin in column 1. All other cards have a free format. At translation time, each card is counted and assigned a line count starting with "1" for reference in error messages.

The format of the source deck is given in Figure 3 and the error diagnostic list in Figure 4.

Figure 3. Source Deck.

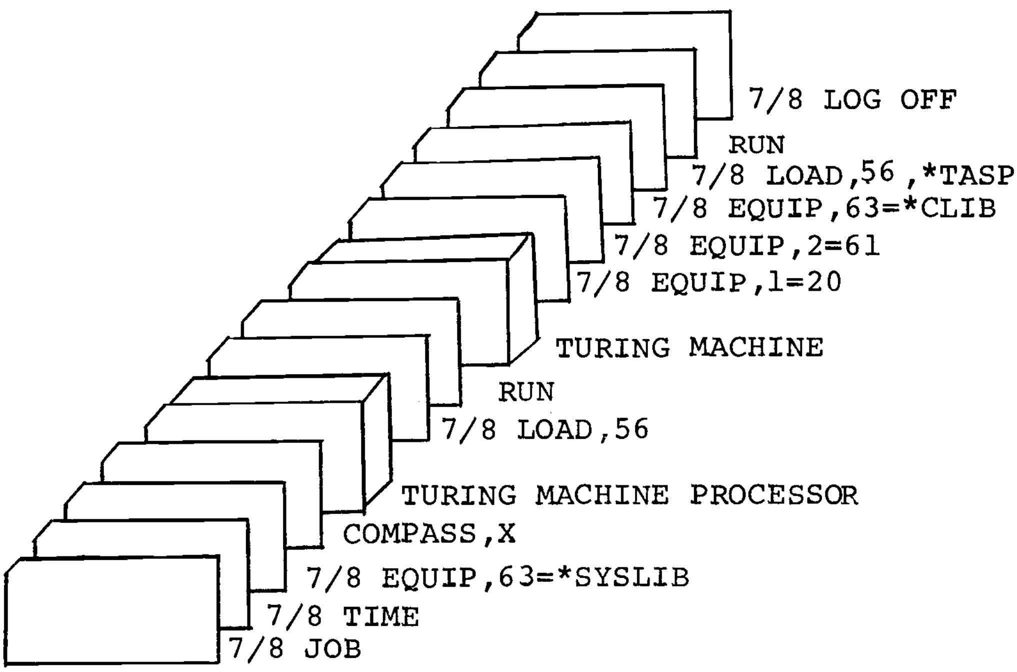


Figure 4. Diagnostic List.

NO TURING CARD

missing a Turing machine card

NO ALPHABET

missing an alphabet card

NO END

missing an end card

TOO MANY ERRORS

more than 125 errors encountered

NO DIRECTION

no direction or an incorrect symbol was found after a search or a go

ILLEGAL STATEMENT AFTER CHANGE

means as stated

SYNTAX ERROR

an illegal statement encountered perhaps a misspelling

UNDEFINED TASK

an illegal expression encountered

NOT IN THE ALPHABET

alphabet character written which is not listed on alphabet card

ERROR IN SYMBOL LIST

a word or two character symbol in symbol list

IV. UNIVERSALITY OF THE TURING MACHINE PROGRAMMING LANGUAGE PROCESSOR LANGUAGE

In this section the main assertion stated previously is proved. We show that any Turing machine quintuple may be encoded in this language. This will demonstrate that algorithms which are capable of being encoded as Turing machines may be written in this language, and thus any algorithm may be encoded in this language.

Lemma 1.

Let S be a set of programs which perform algorithms written in language L_1 and let T be a set of programs which perform algorithms written in language L_2. Let P be a mapping from S to T such that if m is a program of S and the correspondent in T is Pm then

i) The set of inputs to a program m is the same set as the set of inputs to Pm.

ii) If i is an input to m then Pm(i) = m(i). Thus the outputs are the same.

If such a function P does exist then every algorithm encodable in L_1 as an element of S is encodable in L_2 as an element of T.

Proof.

This follows from the definitions. Note that our translator from Turing machine language L_1 to Turing machine L_2 is such a function.

THEOREM 1.

Consider a function $P: S \rightarrow T$ that satisfies the hypotheses of Lemma 1, and suppose there exists a map $Q: T \rightarrow S$ such that PQ is the identity map on S. Then the set of algorithms encodable in L_1 as elements of S equals the set of algorithms encodable in L_2 as elements of T.

<u>Proof</u>.

By Lemma 1 it is sufficient to demonstrate that every algorithm encodable in L_1 by S is also encodable in L_2 by T. To show this, it is sufficient to show that Q also satisfies the hypotheses of the lemma.

Let m be a program in T and i be an input to m. It is necessary to show that $m(i) = Qm(i)$; that is, the output from Qm when i is the input is the same as the output from m.

$$m(i) = (PQm)(i) = [P(Qm)](i) = Qm(i)$$

The last equality holds because P satisfies the hypotheses of the lemma.

QED

THEOREM 2.

Any algorithm may be encoded in the Turing machine processor language.

Proof.

Consider S as the set of all programs which can be written in the Turing machine processor language. Let T be the set of all Turing machines whose alphabet is a subset of the 48 keypunch characters. Basing ourselves on the fact that an algorithm must be performable by some Turing machine which stops, T is the set of all algorithms, since Shannon (11) has shown that the restriction to 48 characters does not restrict the set of algorithms[3]. Let $P: S \to T$ be the operation which transforms programs written in the Turing machine language into Turing machines. P is the Turing machine processor. The hypotheses of the lemma are satisfied for P, because the TASP processor acts as a Turing machine simulator; i.e., the operation of a program $a \in S$ is defined to be the operation Pa as a Turing machine.

So by Theorem 1, it is sufficient to find a map $Q: T \to S$ which acts as a right inverse for P. The remainder of the proof will be the construction of the map Q. Suppose $m \in T$ then $m = (s_1,\ldots,s_\alpha)$ where α is the number of states and s_j are the states. Each $s_j = (q_j,\ldots,q_{j\beta})$ and the q_{jk} the quintuples of the j^{th} state and β is the total number of symbols. Let

---

3. Actually reference (11) establishes that two symbols are sufficient.

$\{\gamma_1,\ldots,\gamma_\beta\}$ be the set of symbols in the alphabet of m.

Then $q_{jk} = (s_j,\gamma_k,\delta_{jk},\gamma_{jk},s_{jk})$ where

s_j is the state in which the Turing machine is in

γ_k is the symbol observed under the Turing head

δ_{jk} is the direction $\in$ {RIGHT, LEFT, PLACE}

γ_{jk} is the new state in which the Turing machine goes

s_{jk} is the new symbol to be written on the tape

Let $Qm = (h,b_1,\ldots,b\ \ ,t)$ where $h = \begin{Bmatrix}\text{TURING MACHINE NO. } m \\ \text{ALPHABET } \gamma_1,\ldots,\gamma_\beta\end{Bmatrix}$

And $b_j = (1_{j1},\ldots,1_{j}\ {+2})$ where

$$\ell_{j1} = \text{STATE } j$$

$$\ell_{j2} = \text{SEARCH LEFT FOR } \gamma_1,\ldots,\gamma_\beta$$

For $k > 2$ $$\ell_{jk} = \text{CHANGE TO } \gamma_{jk-2} \text{ AND GO } \delta_{jk-2} \text{ TO } s_{jk-2}$$

$$T = \text{END}$$

h is the heading block

b_j are the state blocks and

t is the termination block.

Clearly, $Qm \in S$, because Qm satisfies the rules of syntax explained in section 1.

To conclude this proof, it would be sufficient to show that P applied to Qm yields m again. But this can be verified by following the Editor flow chart (see Appendix B). The Editor processes each block independently of the processing of any other block. If one assumes that the Editor is confronted with b_j then it is sufficient to show that the processor outputs $s_j = (q_{j1},\ldots,q_{k\beta})$. The program works in the following manner. Given the input $b_j = (\ell_{j1},\ldots,\ell_{j\beta+2})$ where

$$\ell_{j1} = \text{STATE } j$$

$$\ell_{j2} = \text{SEARCH LEFT FOR}$$

For $k > 2$

$$\ell_{jk} = \gamma_{k-2} \text{ CHANGE TO } \gamma_{jk-2} \text{ AND GO } \delta_{jk-2} \text{ TO } s_{jk-2}$$

it will process each b_j separately.

In the memory of the computer is set up a two dimensional matrix called ALPHABET. This array has four rows: ALPHABET, NEWALPHA, DIRECT, STATE, and β columns. The contents of ALPHABET are $\gamma_1,\ldots,\gamma_\beta$. When the program reads a state card ℓ_{j1} it stores j in the memory location called STATENO. This is done in the main program beginning at the location labeled PICKNO. After ℓ_{j1} is processed ℓ_{j2} is read, within the Editor, and then ℓ_{j2} is stored in the 20-word array labeled CARD. The card area is then searched for the word 'SEARCH'. When this is found, the direction following the string 'SEARCH' is put in storage location 'DIRCTION'. Then for $k > 2$, each ℓ_{jk} is successively read into the CARD array. That area is searched for $\gamma_{jk-2},\delta_{jk-2},s_{jk-2}$ which are then placed in the j^{th} columns of NEWALPHA, DIRECT, and STATE respectively. When a b_j has been processed the program jumps to the subroutine called CODEGEN. In this subroutine a check is made to see if every column of NEWALPHA for missing quintuples (an empty column indicates a missing quintuple). Then, if no errors have been detected, representing the j^{th} row, the following is outputted onto the disk file:

> Contents of STATENO, contents of the j^{th} column of ALPHABET, contents of the j^{th} column of NEWALPHA, contents of the j^{th} column of DIRECT, contents of the j^{th} column of STATE respectively.

At this time STATENO contains j, the j^{th} column of ALPHABET contains γ_j, the j^{th} column of NEWALPHA contains $\gamma_{j\beta}$, the j^{th} column of DIRECT contains $\delta_{j\beta}$, the j^{th} column of STATE contains $s_{j\beta}$. This is done for each b_j. Thus the output of the processor consists of $(q_{j1},\ldots,q_{j\beta})$ where each $q_{jk} = j,\gamma_j,\delta_{jk},s_{jk}$.

QED

EPILOGUE

The language as it now stands seems quite satisfactory in clarifying and simplifying the writing of most Turing machines. However, for very long Turing machines there is an addition to the language which would further clarify and reduce the structure of complicated algorithms, a modification which has not as yet been implemented for use on the CDC 3300, namely the patch. The patch would enable one to introduce a subroutine feature into Turing machine programming. The TASP processor does contain a feature whereby a collection of Turing machines may be operated serially, as in the composition of functions. However, this feature, which would allow the use of two short Turing machines in place of one long Turing machine, does not seem to truly reduce the structure of the resulting algorithm. A patch is a method to allow parallel placement of Turing machines instead of sequential placement.

The modification to the description of Chapter 2 necessary for this addition to the language is obtained by replacing paragraph 4.3.3 by the following:

<state block> :: = <block label><basic statement> <statement>|<state block><statement>|<patch block> <patch statement>

<patch block> :: = <block label> <u>patch</u> <number>

<patch statement> :: = <external alphabet character>

<u>is</u> <external alphabet character>|<patch statement>

<external alphabet character> <u>is</u> <external alphabet character>

A typical patch block would be:

STATE j

PATCH n

A_1 is B_1

⋮

A_m is B_m

where j is the state number of the patch state block, n is the Turing machine to be patched to the Turing machine at that stage. $A_1 \ldots A_m$ are the external alphabet characters in the original Turing machine which are to play the role of the external alphabet characters $B_1 \ldots B_m$ in the Turing machine n. When the processor encounters a patch block, it starts to execute the Turing machine n, substituting characters as described above, until it reaches a STOP command in Turing machine n. At that time it returns to the calling Turing machine at the state block following the patch block.

BIBLIOGRAPHY

1. Bachelor, G. OS3 user's manual. Corvallis, 1967. 19p. (Oregon State University. Computer Center. Manual cc no. 67-15).

2. Backus, J.W. The syntax and semantics of the proposed international algebraic language of the Zurich ACM-GAMM conference. In: Proceedings of the International Conference of Information Processes, Paris. Paris, UNESCO, 1959. (Cited in: Control Data Corporation. ALGOL. Minneapolis, 1968. p. 13).

3. Curtis, M.W. A Turing machine simulator. Journal of the Association of Computing Machines. 12: 1-13. 1965.

4. Davis, Martin. Computability and unsolvability. New York, McGraw-Hill, 1958. 210p.

5. Hermes, H. Enumerability decidability computability. New York, Academic, 1965. 245p.

6. Kleene, S. Introduction to metamathematics. New York, Van Nostrand, 1952. 550p.

7. Knuth, D.E. and R.H. Bigelow. Programming language for automata. Journal of the Association of Computing Machines. 14: 615-635. 1967.

8. Lukasiewiez, Jan. Elements of mathematical logic. New York. Macmillian, 1963. 124p.

9. Minsky, M.L. Recursive unsolvability of Post's problem of "tag" and other topics in the theory of Turing machines. Annals of Mathematics. 74: 437-454. 1961.

10. Shannon, Claude E. A universal Turing machine with two internal states. In: Automata studies, ed. by C.E. Shannon and J. McCarthy. Princeton, New Jersey, Princeton University, 1956. p. 156-183. (Annals of Mathematical Studies no. 34).

11. Stahl, Walter. Working notes on the algorithmic simulation of a biological cell. Corvallis, Oregon State University, Dept. of Mathematics, n.d. n.p. (Unpublished report).

12. Watanabe, Shigeru. On a minimal universal Turing machine. Journal of the Association of Computing Machines. 8: 476-483. 1962.

APPENDIX 1

Sample Problem

Input and Output

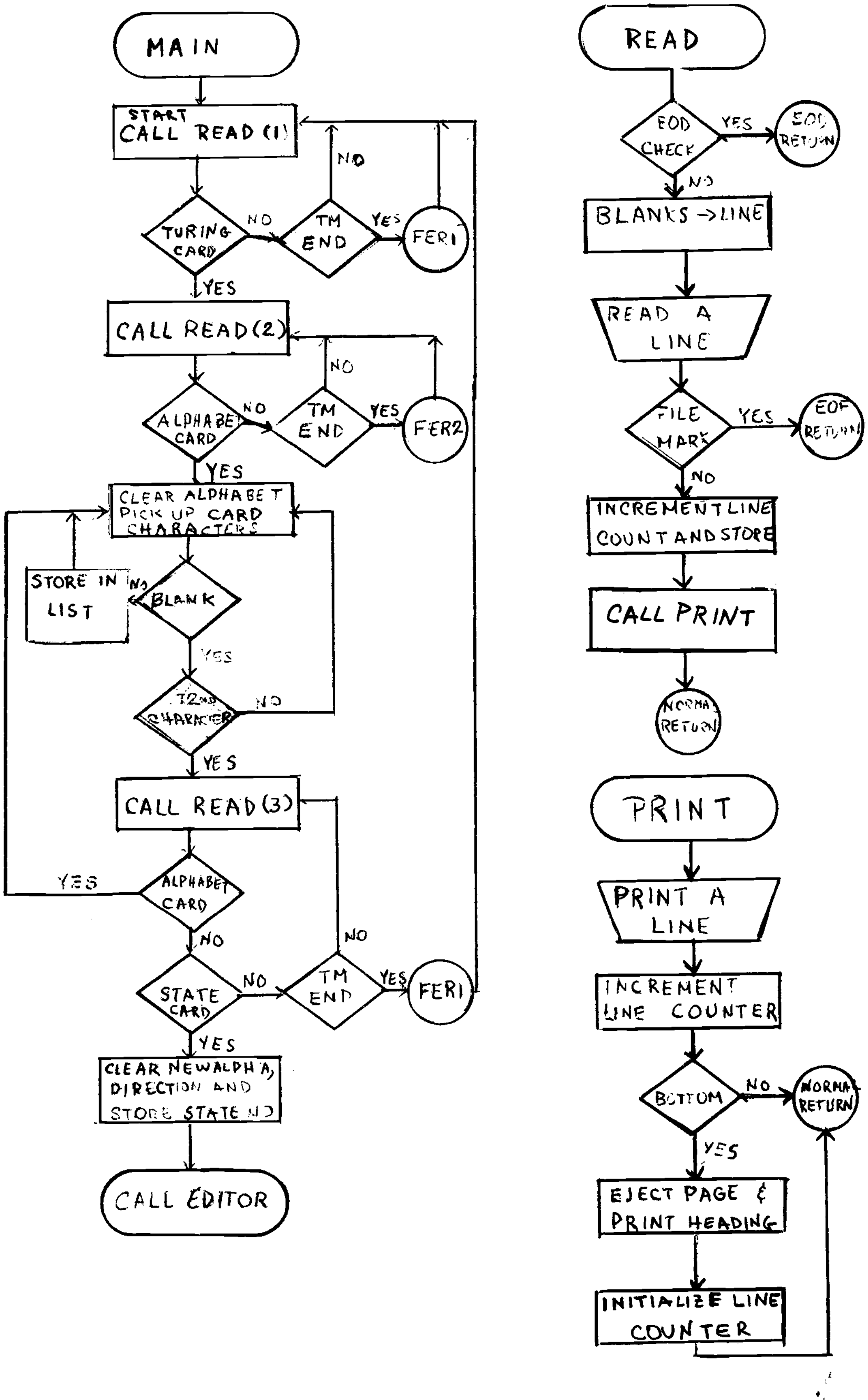
```
TURING MACHINE TRANSLATOR  1219        11/24/69       UNIVERSAL TURING MACHINE
        TURING MACHINE 1
        ALPHABET 0 1 * A B
        STATE 1
                   SEARCH RIGHT FOR 0 AND 1 AND A AND B
                         0 CHANGE TO * AND GO LEFT TO 2
                         1 CHANGE TO * AND GO LEFT TO 3
                         A CHANGE TO 0 AND GO LEFT TO 5
                         B CHANGE TO 1
        STATE 2
                   SEARCH LEFT FOR 0 AND 1 AND * AND A AND B
                         0 CHANGE TO A
                         1 CHANGE TO B
                         * CHANGE TO A AND GO L TO 4
                         A CHANGE TO 0 AND GO R TO 7
                         B CHANGE TO 1 AND GO R TO 8
        STATE 3
                   SEARCH LEFT FOR 0 AND 1 AND * AND A AND B
                         0 CHANGE TO A
                         1 CHANGE TO B
                         * CHANGE TO A AND GO L TO 2
                         A GO R TO 7
                         B GO R TO 8
        STATE 4
                   SEARCH LEFT FOR 0 AND 1 AND * AND A
                         0 GO R TO STATE 3
                         1 CHANGE TO B AND GO R TO 5
                         * GO R TO 1
                         A CHANGE TO *
        STATE 5
                   SEARCH RIGHT FOR 0 AND 1 AND * AND A
                         0 GO R TO 4
                         1 CHANGE TO B AND GO LEFT TO 6
                         * CHANGE TO A AND GO R TO 5
                         A CHANGE TO * AND GO L TO 4
        STATE 6
                   SEARCH LEFT FOR 0 1 * A B
                         0 CHANGE TO A
                         1 CHANGE TO B
                         * CHANGE TO 0 AND GO R TO 7
                         A CHANGE TO 0 AND GO R TO 2
                         B CHANGE TO 1 AND GO R TO 2
        STATE 7
                   SEARCH RIGHT FOR 0 1 * A B
                         0 GO R TO 1
                         1 GO R TO 1
                         * CHANGE TO 0 AND GO L TO 6
                         A CHANGE TO 0
                         B CHANGE TO 1
        STATE 8
                   SEARCH R FOR 0 1 * A B
                         0 GO L TO 1
                         1 GO L TO 1
                         * CHANGE TO 1 AND GO L TO 6
                         A CHANGE TO 0
                         B CHANGE TO 1
        END
```

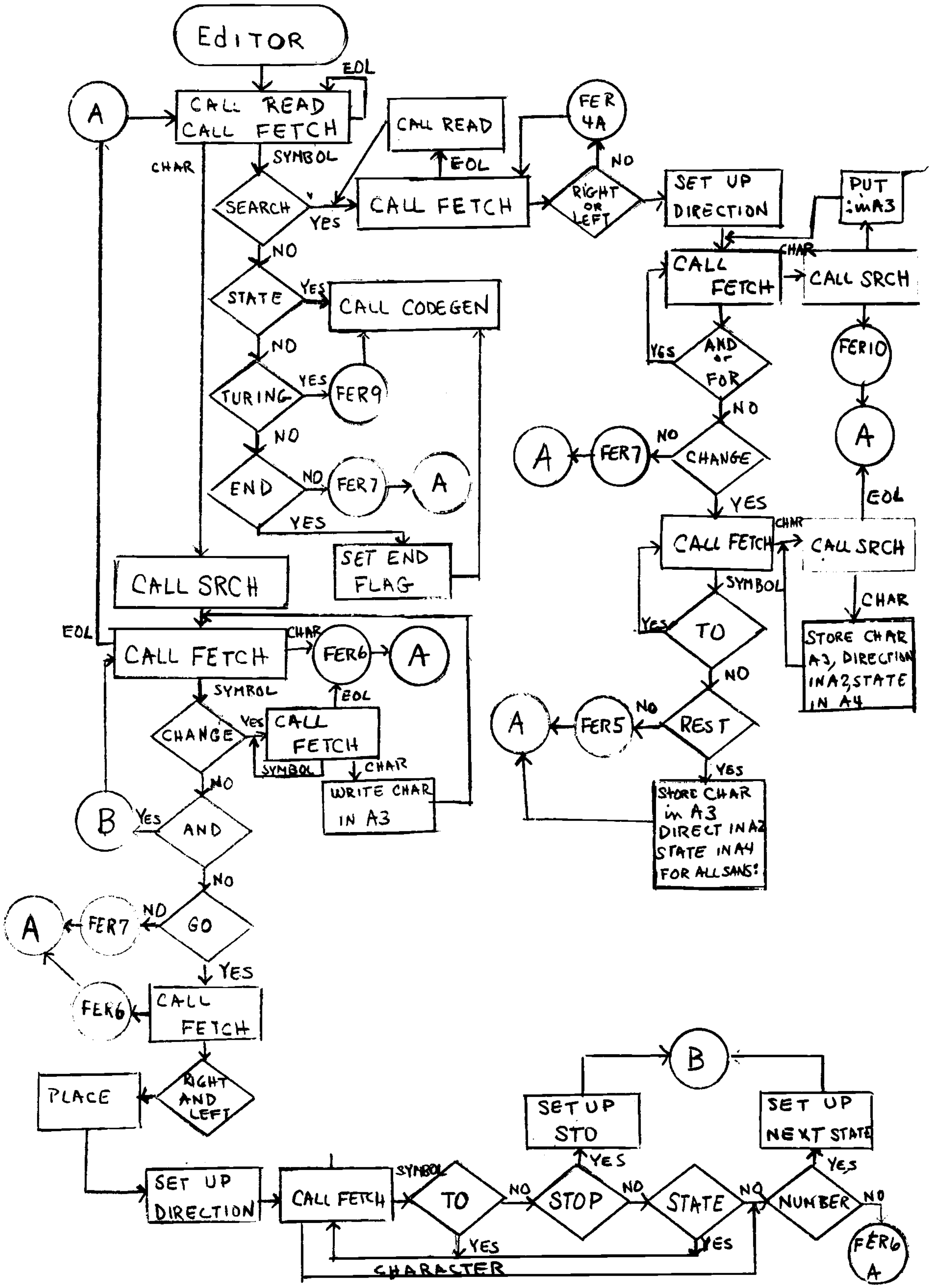
```
TURING MACHINE TRANSLATOR  1219       11/24/69       UNIVERSAL TURING MACHINE
          0          1          *          A          B

001   *L0002     *L0003     *R0001     0L0005     1R0001

002   AL0002     BL0002     AL0004     0R0007     1R0008

003   AL0003     BL0003     AL0002     AR0007     BR0008

004   0R0003     BR0005     *R0001     *L0004     BL0004

005   0R0004     BL0006     AR0005     *L0004     BR0005

006   AL0006     BL0006     0R0007     0R0002     1R0002

007   0R0001     1R0001     0L0006     0R0007     1R0007

008   0L0001     1L0001     1L0006     0R0008     1R0008

010+02*0011+03*001*-01*001A+050001B-011

020+02A0021+02B002*+04A002A-070002B-081

030+03A0031+03B003*+02A003A-07A003B-08B

040-0300041-05B004*-01*004A+04*004B+04B

050-0400051+06B005*-05A005A+04*005B-05B

060+06A0061+06B006*-070006A-020006B-021

070-0100071-011007*+060007A-070007B-071

080+0100081+011008*+061008A-080008B-081

END OF PROGRAM EXECUTION
```

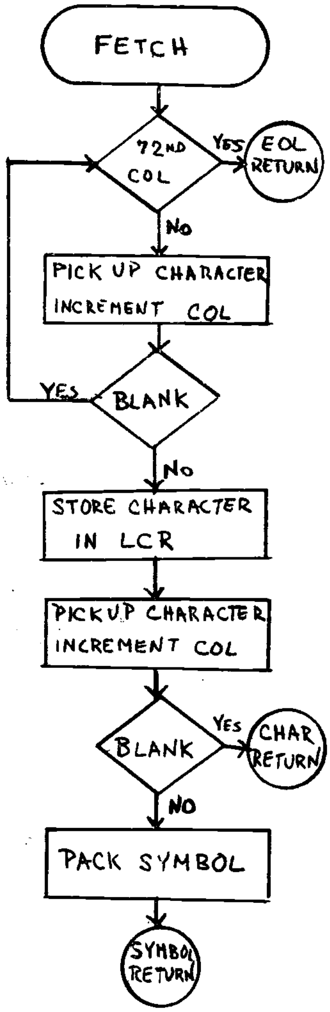
APPENDIX 2

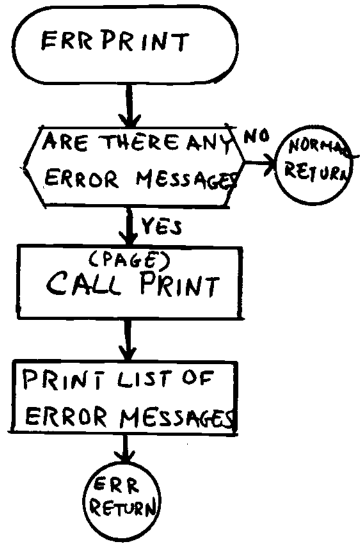
Flow Charts of

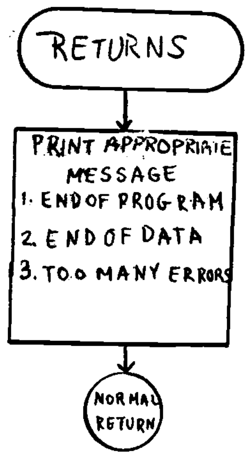
the Turing Machine Processor

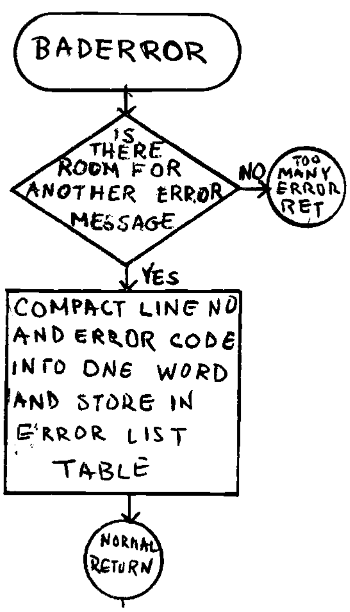

MAIN
START
CALL READ (1)
TURING CARD
NO
TM END
YES
FER1
CALL READ(2)
ALPHABET CARD
TM END
FER2
CLEAR ALPHABET
PICK UP CARD CHARACTERS
STORE IN LIST
BLANK
72ND CHARACTER
CALL READ (3)
ALPHABET CARD
STATE CARD
CLEAR NEWALPHA, DIRECTION AND STORE STATE NO
CALL EDITOR
READ
EOD CHECK
EOD RETURN
BLANKS → LINE
READ A LINE
FILE MARK
EOF RETURN
INCREMENT LINE COUNT AND STORE
CALL PRINT
NORMAL RETURN
PRINT
PRINT A LINE
INCREMENT LINE COUNTER
BOTTOM
EJECT PAGE & PRINT HEADING
INITIALIZE LINE COUNTER

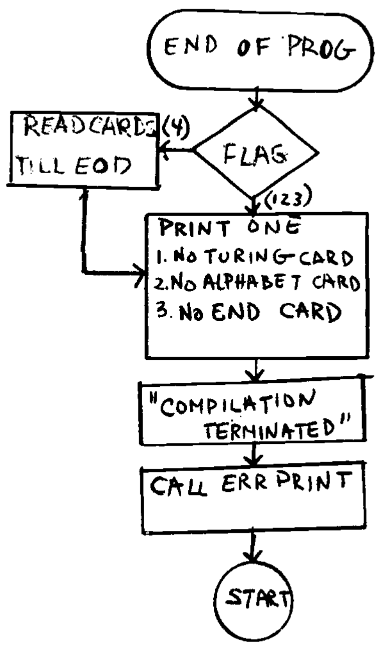

EditOR
CALL READ
CALL FETCH
EOL
SYMBOL
CHAR
SEARCH
YES
NO
CALL READ
CALL FETCH
FER 4A
RIGHT OR LEFT
SET UP DIRECTION
PUT :inA3
CALL FETCH
CALL SRCH
STATE
CALL CODEGEN
TURING
FER9
END
FER7
SET END FLAG
AND or FOR
FER10
CHANGE
CALL FETCH
CALL SRCH
STORE CHAR A3, DIRECTION IN A2, STATE IN A4
TO
REST
FER5
STORE CHAR in A3 DIRECT IN A2 STATE IN A4 FOR ALL SANS!
CALL SRCH
CALL FETCH
FER6
CHANGE
CALL FETCH
WRITE CHAR IN A3
AND
GO
CALL FETCH
RIGHT AND LEFT
PLACE
SET UP DIRECTION
CALL FETCH
TO
STOP
STATE
NUMBER
SET UP STO
SET UP NEXT STATE
CHARACTER
FER6 A

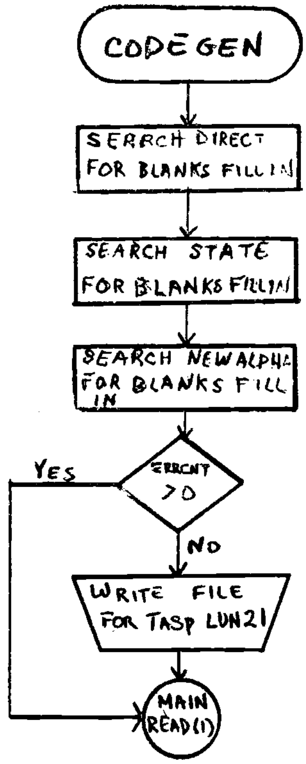

FETCH
72ND COL
YES
EOL RETURN
NO
PICK UP CHARACTER
INCREMENT COL
YES
BLANK
NO
STORE CHARACTER IN LCR
PICKUP CHARACTER
INCREMENT COL
BLANK
YES
CHAR RETURN
NO
PACK SYMBOL
SYMBOL RETURN


ERRPRINT
ARE THERE ANY ERROR MESSAGES
NO
NORMAL RETURN
YES
(PAGE)
CALL PRINT
PRINT LIST OF ERROR MESSAGES
ERR RETURN


RETURNS
PRINT APPROPRIATE MESSAGE
1. END OF PROGRAM
2. END OF DATA
3. TOO MANY ERRORS
NORMAL RETURN


BADERROR
IS THERE ROOM FOR ANOTHER ERROR MESSAGE
NO
TOO MANY ERROR RET
YES
COMPACT LINE NO AND ERROR CODE INTO ONE WORD AND STORE IN ERROR LIST TABLE
NORMAL RETURN


END OF PROG
FLAG
(4)
READ CARDS TILL EOD
(123)
PRINT ONE
1. No TURING CARD
2. No ALPHABET CARD
3. No END CARD
"COMPILATION TERMINATED"
CALL ERR PRINT
START


CODEGEN
SEARCH DIRECT FOR BLANKS FILL IN
SEARCH STATE FOR BLANKS FILL IN
SEARCH NEWALPHA FOR BLANKS FILL IN
ERRCNT >0
YES
NO
WRITE FILE FOR TASP LUN21
MAIN READ (1)

APPENDIX 3

Program Listing

```
          IDENT     MAIN
          ENTRY     START,BEGIN,NORUN,INTIN,READ3A,READ1A
          EXT       READ,ERRPRINT,EOFRET
          EXT       INUNIT,OUTUNIT,COL,LCR,EDITOR
          EXT       ERRABRT,NORMEND,NOEND,NOEND1
          EXT       IOLUNS
          EXT       HEADING,HEDSTRT,PAGE
SCRATCH   EQU       20
SCRATCH2  EQU       21
EQUIP     EQU       3
UNEQUIP   EQU       2
BEGIN     UJP       *+1
          ENI       UNEQUIP,1
          XREQ      59
          ENI       UNEQUIP,1
          XREQ      SCRATCH
          ENI       EQUIP,1
          LDAQ      FILE
          XREQ      SCRATCH
          ENI       UNEQUIP,1
          XREQ      SCRATCH2
          ENI       EQUIP,1
          LDAQ      FILE
          XREQ      SCRATCH2
          RTJ       IOLUNS
          SWA       INUNIT
          SHAQ      24
          SWA       OUTUNIT
          ENA       ERRABRT
          ADA       =O01000000
          STA       77777B
START     ENA,S     -0
          STA       LCR
          ENA       0
          SWA       COL
          STA       ERRCNT
          ENQ       0
          LDA       BLANK
          SHAQ      12
          STAQ      LINE
          ENI       19,1
          LDA       BLANK
          STA       HEADING,1
          IJD       *-1,1
          ENA       HEADING
          ENQ       20
          READ,I    INUNIT
          RTJ       HEDSTRT
          RTJ       PAGE
READ1     RTJ       READ
          UJP       NOEND
READ1A    LDA       CARD
          LDQ       TMASK
          AQJ,EQ    READ2
          LDQ       ENDMK
          AQJ,EQ    FER1
          UJP       READ1
READ2     RTJ       READ
          UJP       NOEND
          LDA       CARD
          LDQ       ALPHAMK
          AQJ,EQ    CLEAR
          LDQ       ENDMK
          AQJ,EQ    FER2
          UJP       READ2
CLEAR     ENI       48,2
CLE       LDA       BLANK
          STA       ALPHABET,2
          IJD       CLE,2
          ENI       0,2
ALPHA     ENI       9,1
ALPHA1    LACH      CARD,1
          INI       1,1
          STA       ALPHABET,2
          INI       1,2
          LDQ       BLANKMK
          AQJ,NE    ALPHA1
          INI       -1,2
          ISE       72,1
          UJP       ALPHA1
READ3     RTJ       READ
          UJP       NOEND
          LDA       CARD
          LDQ       ALPHAMK
          AQJ,EQ    ALPHA
          STI       NALPHA,2
READ3A    LDA       CARD
          LDQ       STATEMK
          AQJ,EQ    PICKNO
          LDQ       ENDMK
          AQJ,NE    READ3
          RTJ       ERRPRINT
          UJP       NORMEND
          UJP       EOFRET
PICKNO    RTJ       INTIN
          STA       STATENO
          ENI       143,2
CLEAR1    LDA       BLANK
          STA       NEWALPHA,2
          IJD       CLEAR1,2
          UJP       EDITOR
FER1      ENA       1
          UJP       NOEND1
FER2      ENA       2
          UJP       NOEND1
INTIN     UJP       **
          ENI       -4,2
          ENI       6,1
PI        LACH      CARD,1
          SACH      STATENO+4,2
          ASE       60B
          UJP       *+2
          UJP       INTIN
          INI       1,1
          IJI       INTIN,2
          IJI       PI,2
          UJP       INTIN
TMASK     BCD       1,TURI
ENDMK     BCD       1,END
ALPHAMK   BCD       2,ALPHABET
BLANKMK   OCT       00000060
STATEMK   BCD       1,STAT
NORUN     BSS       1
FILE      BCD       2,FILE
```

```
BLANK    BCD      1,
         DATA
LINE     BSS      23
CARD     EQU      LINE+2
ERRCNT   BSS      1
ERLIST   BSS      128
ALPHABET BSS      192
NEWALPHA EQU      ALPHABET+48
DIRECT   EQU      ALPHABET+96
STATE    EQU      ALPHABET+144
STATENO  BSS      1
NALPHA   BSS      1
DIRCTION BSS      1
         END      BEGIN
         IDENT    READ
         ENTRY    READ,INUNIT
         EXT      STATUS,PRINT,EOD
READ     UJP      **
         ENQ      STATUS
         CNTL,I   INUNIT
         SHA      2
         AZJ,LT   EOD
         STI      S.XR1,1
         ENI      20,1
         LDA      BLANK
         STA      CARD,1
         IJD      *-1,1
S.XR1    ENI      **,1
         ENA      CARD
         ENQ      20
INUNIT   READ     **
         SHA      3
         AZJ,LT   READ
         LDAQ     LINE
         SHAQ     12
         ADA      =O66666667
         STA      TEMP1
         LPA      =O60606060
         STA      TEMP2
         SHA      21
         ADA      TEMP2
         XOA,S    -0
         ADA      TEMP1
         SHAQ     36
         STAQ     LINE
         LDAQ     CARD+19
         SHAQ     24
         STAQ     CARD+19
         RTJ      PRINT
         LDA      READ
         INA      1
         SWA      READ
         UJP      READ
TEMP1    BSS,     1
TEMP2    BSS      1
BLANK    BCD      1,
         DATA
LINE     BSS      23
CARD     EQU      LINE+2
ERRCNT   BSS      1
ERLIST   BSS      128
ALPHABET BSS      192
NEWALPHA EQU      ALPHABET+48
DIRECT   EQU      ALPHABET+96
STATE    EQU      ALPHABET+144
STATENO  BSS      1
NALPHA   BSS      1
DIRCTION BSS      1
         END
         IDENT    PRINT
         EXT      DATE,TIME
         ENTRY    PRINT,OUTUNIT,PAGE,HEADING,HEDSTRT
PRINT    UJP      **
         ENQ      23
         ENA      LINE
OUTUNIT  WRITE    **
         ENA,S    -1
         ADA      LINCNT
         STA      LINCNT
         AZJ,GE   PRINT
         RTJ      PAGE
         UJP      PRINT
PAGE     UJP      **
         ENA      HEAD
         ENQ      26
         WRITE,I  OUTUNIT
         ENA      55
         STA      LINCNT
         UJP      PAGE
LINCNT   BSS      1
HEAD     BCD      12,1TURING MACHINE TRANSLATOR
HEADING  BSS      20
HEDSTRT  UJP      **
         ECHA     DAT
         RTJ      DATE
         ECHA     TIM
         RTJ      TIME
         UJP      HEDSTRT
TIM      EQU      HEAD+7
DAT      EQU,C    HEAD+38
         DATA
LINE     BSS      23
CARD     EQU      LINE+2
ERRCNT   BSS      1
ERLIST   BSS      128
ALPHABET BSS      192
NEWALPHA EQU      ALPHABET+48
DIRECT   EQU      ALPHABET+96
STATE    EQU      ALPHABET+144
STATENO  BSS      1
NALPHA   BSS      1
DIRCTION BSS      1
         END
         IDENT    RETS
         EXT      START,OUTUNIT
         ENTRY    NORMEND,EOD,ERRABRT
NORMEND  ENA      MESS1
WRITE    ENQ      8
         WRITE,I  OUTUNIT
         UJP      START
EOD      ENA      MESS2
         UJP      WRITE
ERRABRT  ENA      MESS3
         UJP      WRITE
```

```
MESS1    BCD      8,-END OF PROGRAM EXECUTION
MESS2    BCD      8,- END OF DATA ENCOUNTERED
MESS3    BCD      8,- TOO MANY ERRORS (ABORT)
         END
         IDENT    BADERROR
         EXT      TMERR,NORUN
         ENTRY    BADERROR,ERROR
BADERROR UJP      **
         STA      TEMP
         ENA,S    -0
         STA      NORUN
         STA      FLAG
ERRER    STI      S1,1
         LDI      ERRCNT,1
         ISG      129,1
         UJP      OK
         UJP      TMERR
OK       LDAQ     LINE
         SHAQ     -12
         ENA      0
         STI      S2,2
         ENI      3,2
LOOP     SHQ      3
         SHAQ     3
         IJD      LOOP,2
         SHA      12
         ADA      TEMP
S2       ENI      **,2
         STA      ERLIST,1
         INI      1,1
         STI      ERRCNT,1
S1       ENI      **,1
         LDA      FLAG
         AZJ,LT   BADERROR
         UJP      ERROR
ERROR    UJP      **
         STA      TEMP
         ENA      0
         STA      FLAG
         UJP      ERRER
FLAG     BSS      1
TEMP     BSS      1
         DATA
LINE     BSS      23
CARD     EQU      LINE+2
ERRCNT   BSS      1
ERLIST   BSS      128
ALPHABET BSS      192
NEWALPHA EQU      ALPHABET+48
DIRECT   EQU      ALPHABET+96
STATE    EQU      ALPHABET+144
STATENO  BSS
NALPHA   BSS      1
DIRCTION BSS      1
         END
         IDENT    FETCH
         ENTRY    FETCH,ITEM,COL,LCR
FETCH    UJP      **
         STI      XR1,1
         STI      XR2,2
         STI      R1,1
         STI      R2,2
         STI      X1,1
         STI      X2,2
COL      ENI      **,1
         UJP      *+2
CON2     INI      1,1
         ISG      72,1
         UJP      CON
         LDA      FETCH
         INA      1
         SWA      FETCH
         ENI      0,1
         STI      COL,1
XR1      ENI      **,1
XR2      ENI      **,2
         UJP      FETCH
CON      LACH     CARD,1
         ASE      60B
         UJP      CON1
         UJP      CON2
CON1     SACH     LCR
         INI      1,1
         ISG      72,1
         UJP      CONT
CRET     LDA      FETCH
         INA      2
         SWA      FETCH
         LACH     LCR
         STI      COL,1
X1       ENI      **,1
X2       ENI      **,2
         UJP      FETCH
CONT     LACH     CARD,1
         ASE      60B
         UJP      PACK
         UJP      CRET
PACK     LDA      BLANK
         STA      ITEM
         STA      ITEM+1
         ENI      -7,2
         LACH     LCR
PS       SACH     ITEM+7,2
         LACH     CARD,1
         INI      1,1
         ISG      72,1
         UJP      *+2
         UJP      DONE
         ASE      60B
DONE     IJI      PS,2
         LDAQ     ITEM
         STI      COL,1
R1       ENI      **,1
R2       ENI      **,2
         UJP      FETCH
ITEM     BSS      2
LCR      BSS      1
BLANK    BCD      1,
         DATA
LINE     BSS      23
CARD     EQU      LINE+2
ERRCNT   BSS
ERRLIST  BSS      128
```

```
ALPHABET BSS       192
NEWALPHA EQU       ALPHABET+48
DIRECT   EQU       ALPHABET+96
STATE    EQU       ALPHABET+144
STATENO  BSS       1
NALPHA   BSS       1
DIRCTION BSS       1
         END
         IDENT     EDITOR
         ENTRY     EDITOR,ENDFLAG
         EXT       READ,FETCH,CODEGEN,BADERROR,INTIN
         EXT       NOEND
EDITOR   UJP       *+1
A        RTJ       READ
         UJP       NOEND
         RTJ       FETCH
         UJP       SYM
         UJP       A
         UJP       CHAR
SYM      LDQ       SERCHMK
         AQJ,NE    NEXT
FETC     RTJ       FETCH
         UJP       SYMRL
         UJP       EOL1
         UJP       SYMRL2
EOL1     RTJ       READ
         UJP       NOEND
         UJP       FETC
SYMRL    LDQ       RIGHTMK
         AQJ,NE    LEFT
         ENA       51B
         STA       DIRCTION
         UJP       NEXT1
LEFT     LDQ       LEFTMK
         AQJ,NE    FER4
         ENA       43B
         STA       DIRCTION
         UJP       NEXT1
SYMRL2   LDQ       RMASK
         AQJ,NE    LEFT1
         ENA       51B
         STA       DIRCTION
         UJP       NEXT1
LEFT1    LDQ       LMASK
         AQJ,NE    FER4
         ENA       43B
         STA       DIRCTION
NEXT1    RTJ       FETCH
         UJP       SYM2
         UJP       A
         LDI       NALPHA,1
         ENQ,S     -0
         RTJ       SRCH
         UJP       FER10
         UJP       NEXT1
SYM2     LDQ       ANDMK
         AQJ,EQ    NEXT1
         LDQ       FORMK
         AQJ,EQ    NEXT1
         LDQ       CHNGMK
         AQJ,NE    FER7
FE       RTJ       FETCH
         UJP       SYN
         UJP       FEN
         RTJ       SRCH
         UJP       FER9
         RTJ       FETCH
         UJP       SYN
         UJP       FEN
         STA       NEWALPHA,1
         LDA       DIRCTION
         STA       DIRECT,1
         LDA       DIRCTION
         LDA       STATENO
         STA       STATE,1
         UJP       FE
FEN      RTJ       READ
         UJP       NOEND
         UJP       FE
SYN      LDQ       TOMK
         AQJ,EQ    FE
         LDQ       RESTMK
         AQJ,NE    FER5
         RTJ       FETCH
         UJP       SYM
         UJP       A
         STA       NEWLET
         LDI       NALPHA,1
MEQ      LDA       BLANK
         ENQ,S     -0
         MEQ       NEWALPHA,1
         UJP       NFOUND
         LDA       DIRCTION
         STA       DIRECT,1
         LDA       STATENO
         STA       STATE,1
         UJP       MEQ
NFOUND   UJP       A
NEXT     LDQ       STATEMK
         AQJ,EQ    CODEGEN
         LDQ       TURIMK
         AQJ,EQ    F8
         LDQ       ENDMK
         AQJ,NE    FER7
         ENA,S     -0
         STA       ENDFLAG
         UJP       CODEGEN
F8       ENA       1
         STA       ENDFLAG
         UJP       CODEGEN
CHAR     LDI       NALPHA,1
         ENQ,S     -0
         RTJ       SRCH
         UJP       FER9
B        RTJ       FETCH
         UJP       SYMB
         UJP       A
         UJP       FER6
SYMB     LDQ       CHNGMK
         AQJ,NE    AN
F        RTJ       FETCH
         UJP       F
         UJP       FER6
         STA       NEWALPHA,1
```

```
         UJP       B
AN       LDQ       ANDMK
         AQJ,EQ    B
         LDQ       GOMK
         AQJ,NE    FER7
         RTJ       FETCH
         UJP       SKIP
         UJP       FER6
         LDQ       RMASK
         AQJ,NE    LEFT2
         ENA       51B
         STA       DIRECT,1
         UJP       NEXTA
LEFT2    LDQ       LMASK
         AQJ,NE    FER4
         ENA       43B
         STA       DIRECT,1
         UJP       NEXTA
LEFTS    LDQ       LEFTMK
         AQJ,NE    PL
         ENA       43B
         STA       DIRECT,1
         UJP       NEXTA
SKIP     LDQ       RIGHTMK
         AQJ,NE    LEFTS
         ENA       51B
         STA       DIRECT,1
         UJP       NEXTA
         ENA       43B
         STA       DIRECT,1
         UJP       NEXTA
PL       ENQ       47B
         STQ       DIRECT,1
         UJP       SYMBO
NEXTA    RTJ       FETCH
         UJP       SYMBO
         UJP       FER6
         UJP       ST2
SYMBO    LDQ       TOMK
         AQJ,EQ    NEXTA
         LDQ       STOPMK
         AQJ,NE    ST
         LDA       STO
         STA       STATE,1
         UJP       B
ST       LDQ       STATEMK
         AQJ,EQ    NEXTA
ST2      STA       TEMP
         LACH      TEMP
         ASG       12B
         UJP       SETUP
         UJP       FER6
SETUP    LDA       TEMP
         STA       STATE,1
         UJP       B
FER4     ENA       4
         RTJ       BADERROR
         UJP       NEXTA
FER6     ENA       6
         RTJ       BADERROR
         UJP       A
FER5     ENA       5
         RTJ       BADERROR
         UJP       A
FER7     ENA       7
         RTJ       BADERROR
         UJP       A
FER9     ENA       9
         RTJ       BADERROR
         UJP       CODEGEN
FER10    ENA       10
         RTJ       BADERROR
         UJP       A
SRCH     UJP       **
         MEQ       ALPHABET,1
         UJP       SRCH
         LDA       SRCH
         INA       1
         SWA       SRCH
         UJP       SRCH
BLANK    BCD       1,
SERCHMK  BCD       2,SEARCH
RIGHTMK  BCD       2,RIGHT
LEFTMK   BCD       1,LEFT
RMASK    BCD       1,000R
LMASK    BCD       1,000L
ANDMK    BCD       1,AND
FORMK    BCD       1,FOR
CHNGMK   BCD       1,CHAN
STO      BCD       1, STO
TOMK     BCD       1,TO
RESTMK   BCD       1,REST
STATEMK  BCD       1,STAT
TURIMK   BCD       1,TURI
ENDMK    BCD       1,END
ENDFLAG  BCD       1,
GOMK     BCD       1,GO
TEMP     BSS       1
NEWLET   BSS       1
STOPMK   BCD       1,STOP
         DATA
LINE     BSS       23
CARD     EQU       LINE+2
ERRCNT   BSS       1
ERRLIST  BSS       12B
ALPHABET BSS       192
NEWALPHA EQU       ALPHABET+48
DIRECT   EQU       ALPHABET+96
STATE    EQU       ALPHABET+144
STATENO  BSS       1
NALPHA   BSS       1
DIRCTION BSS       1
         END
         IDENT     END OF PROGRAM
         EXT       OUTUNIT,ERRPRINT,START,READ
         ENTRY     NOEND,NOEND1,THERR,EOFRET
THERR    RTJ       READ
         UJP       EOFRET
         UJP       *-2
EOFRET   UJP       NOEND1
         ENA       4
         UJP       NOEND1
NOEND    ENA       3
NOEND1   STA       EM
```

```
         ENI     3,1
         LDA     EM
         INA,S   -1
         MUA     =D4
         INA     EMS
         SWA     LOOP
LOOP     LDA     **,1
         STA     ERRM,1
         IJD     LOOP,1
         ENA     ERMS
         ENQ     11
         WRITE,I OUTUNIT
         ENA     MESS
         ENQ     5
         WRITE,I OUTUNIT
         RTJ     ERRPRINT
         NOP
         UJP     START
EM       BSS     1
ERMS     BCD     7,0COMPILATION TERMINATED
ERRM     BCD     4,
EMS      BCD     4, NO TURING CARD
         BCD     4, NO ALPHABET
         BCD     4, NO END
         BCD     4, TOO MANY ERRORS
MESS     BCD     5,-EXECUTION DELETED
         END
         IDENT   ERRPRINT
         EXT     OUTUNIT,PAGE
         ENTRY   ERRPRINT
ERRPRINT UJP     **
         STI     S1,1
         STI     S2,2
         STI     S3,3
         LDA     ERRCNT
         AZJ,EQ  ERRPRINT
         ENA     HD
         ENQ     4
         WRITE,I OUTUNIT
         ENA     HD1
         ENQ     1
         WRITE,I OUTUNIT
         LDA     ERRCNT
         SWA     LOOPEND
         ENI     0,1
LOOP     LDQ     ERLIST,1
         ENA     0
         ENI     3,3
LP       SHA     3
         SHAQ    3
         IJD     LP,3
         STA     ERLINE+2
         LDQ     ERLIST,1
         SHAQ    12
         ENA     0
         SHAQ    12
         INA,S   -1
         MUA     =D9
         INA     M1
         SWA     PLP
         ENI     8,2
PLP      LDA     **,2
```

```
         STA     LCONT,2
         IJD     PLP,2
         ENA     ERLINE
         ENQ     13
         WRITE,I OUTUNIT
         INI     1,1
LOOPEND  ISG     **,1
         UJP     LOOP
         ENA     HD1
         ENQ     1
         WRITE,I OUTUNIT
S1       ENI     **,1
S2       ENI     **,2
S3       ENI     **,3
         LDA     ERRPRINT
         INA     1
         SWA     ERRPRINT
         UJP     ERRPRINT
HD       BCD     4,0ERROR MESSAGES
HD1      BCD     1,
ERLINE   BCD     4,  LINE
LCONT    BCD     9,
M1       BCD     9,
M2       BCD     9,
M3       BCD     9,
M4       BCD     9, NO DIRECTION
M5       BCD     9, ILLEGAL STATEMENT AFTER CHANGE
M6       BCD     9, SYNTAX ERROR
M7       BCD     9, UNDEFINED TASK
M8       BCD     9,
M9       BCD     9, NOT IN THE ALPHABET
M10      BCD     9, ERROR IN SYMBOL LIST
         DATA
LINE     BSS     23
CARD     EQU     LINE+2
ERRCNT   BSS     1
ERLIST   BSS     128
ALPHABET BSS     192
NEWALPHA EQU     ALPHABET+48
DIRECT   EQU     ALPHABET+96
STATE    EQU     ALPHABET+144
STATENO  BSS     1
NALPHA   BSS     1
DIRCTION BSS     1
         END
         IDENT   CODEGEN
         ENTRY   CODEGEN
         EXT     READ3A,READ1A,ENDFLAG,NORMEND,OUTUNIT,REWIND
         EXT     ERRPRINT
CODEGEN  NOP
         LDI     NALPHA,1
MEQ      LDA     BLANK
         ENQ,S   -0
         MEQ     DIRECT,1
         UJP     NFOUND
         LDA     DIRCTION
         STA     DIRECT,1
         UJP     MEQ
NFOUND   LDI     NALPHA,1
MEQ2     ENQ,S   -0
         LDA     BLANK
         MEQ     STATE,1
```

```
         UJP     NFOUND2
         LDA     STATENO
         STA     STATE,1
         UJP     MEQ2
NFOUND2  LDI     NALPHA,1
MEQ3     ENQ,S   -0
         LDA     BLANK
         MEQ     NEWALPHA,1
         UJP     DONE
         LDA     ALPHABET,1
         STA     NEWALPHA,1
         UJP     MEQ3
DONE     ENQ     192
         ENA     ALPHABET
         WRITE   SCRATCH
         LDA     ENDFLAG
         ENQ,S   -0
         AQJ,EQ  WROTE
         ENQ     1
         AQJ,EQ  READ1A
         UJP     READ3A
WROTE    ENQ     REWIND
         CNTL    SCRATCH
         RTJ     ERRPRINT
         UJP     *+2
         UJP     NORMEND
         ENA     0
         STA     NUMB
BEG      ENQ     192
         ENA     ALPHABET
         READ    SCRATCH
         RTJ     BLNK
BEGA     LDA     NUMB
         ENQ     0
         AQJ,EQ  WALPHA
         STA     LINE
         LDI     PLACE,1
         ENI     6,2
         ENI     14,3
ROTA     LACH    NEWALPHA,1
         SACH    LINE,2
         INI     1,2
         LACH    DIRECT,1
         SACH    LINE,2
         ENA     4
         STA     TEMP
         INI     -4,1
DOA      INI     1,2
         INI     1,1
         LACH    STATE,1
         SACH    LINE,2
         LDA     TEMP
         SBA     ONE
         STA     TEMP
         AZJ,NE  DOA
         INI     4,2
         INI     4,1
         IJD     ROTA,3
         ENQ     33
         ENA     LINE
         WRITE,I OUTUNIT
         LDA     NUMB
```

```
         ADA     ONE
         STA     NUMB
         LDQ     STATENO
         AQJ,EQ  BEG
         AQJ,GE  NEXT
         UJP     BEG
WALPHA   LDI     PLACE,1
ROT      ENI     8,2
         ENI     16,3
ROTS     LACH    ALPHABET,1
         SACH    LINE,2
         INI     4,1
         INI     9,2
         IJD     ROTS,3
         ENQ     33
         ENA     LINE
         WRITE,I OUTUNIT
         RTJ     BLNK
         LDA     NUMB
         INA     1
         STA     NUMB
         UJP     BEGA
NEXT     LDA     NALPHA
         SBA     FORTEEN
         AQJ,LT  NEW
         ENQ     1
         LDA     PLACE
         INA     14
         STA     PLACE
         UJP     WROTE
BLNK     UJP     **
         LDA     BLANK
         ENI     33,1
CLEAR    STA     LINE,1
         IJD     CLEAR,1
         UJP     BLNK
ZERO     UJP     **
         LDA     ZER
         ENI     33,1
         STA     LINE,1
         IJD     *-1,1
         UJP     ZERO
NEW      ENQ     REWIND
         CNTL    SCRATCH
         ENA     0
         STA     NUMB
BETA     ENI     0,2
         LDA     NUMB
         ADA     ONE
         STA     NUMB
         LDQ     STATENO
         AQJ,EQ  *+2
         AQJ,GE  DID
         ENQ     192
         ENA     ALPHABET
         READ    SCRATCH
         RTJ     ZERO
         LDI     NALPHA,3
         ENI     0,1
         LDA     NUMB
         RTJ     VALQ
         STA     BEAT
```

```
BET      LDA      BEAT
         STA      LINE,2
         LDA      ALPHABET,1
         RAD      LINE,2
         INI      1,1
         INI      2,2
         IJD      BET,3
         ENI      0,1
         ENI      1,2
         LDI      NALPHA,3
BES      LDA      DIRECT,1
         LDQ      RMASK
         AQJ,EQ   R
         LDQ      LMASK
         AQJ,EQ   L
         LDA      PCODE
         RAD      LINE,2
         UJP      Q
L        LDA      LCODE
         RAD      LINE,2
         UJP      Q
R        LDA      RCODE
         RAD      LINE,2
Q        LDA      NEWALPHA,1
         RAD      LINE,2
         LDA      STATE,1
         RTJ      VALQ
         RAD      LINE,2
         INI      1,1
         INI      2,2
         IJD      BES,3
         ENA      LINE
         LDQ      NALPHA
         SHQ      1
         WRITE    61
         WRITE    SCRATCH2
         UJP      BETA
DID      UJP      NORMEND
VALQ     UJP      **
         RTJ      BCD
         ENQ      0
         SHAQ     6
         UJP      VALQ
BCD      UJP      **
         STI      S1,1
         ENQ      0
         ENI      S,1
BCD1     SBA      BK-1,1
         AZJ,LT   BCD2
         SHAQ     24
         ADA      BK+4,1
         SHAQ     24
         UJP      BCD1
BCD2     ADA      BK-1,1
         IJD      BCD1,1
         SHAQ     24
S1       ENI      **,1
         UJP      BCD
BK       OCT      1
         OCT      100
         OCT      10000
         OCT      100000
         OCT      1000000
         OCT      1
         OCT      12
         OCT      144
         OCT      1750
BEAT     BSS      1
RMASK    BCD      1,000R
LMASK    BCD      1,000L
PCODE    OCT      0
RCODE    OCT      40000000
LCODE    OCT      20000000
SCRATCH2 EQU      21
ZER      BCD      1,0000
SCRATCH  EQU      20
PLACE    BCD      1,0003
TEMP     BSS      1
ONE      BCD      1,0001
NUMB     BSS      1
FORTEEN  BCD      1,0014
BLANK    BCD      1,
         DATA
LINE     BSS      23
CARD     EQU      LINE+2
ERRCNT   BSS      1
ERRLIST  BSS      128
ALPHABET BSS      192
NEWALPHA EQU      ALPHABET+48
DIRECT   EQU      ALPHABET+96
STATE    EQU      ALPHABET+144
STATENO  BSS      1
NALPHA   BSS      1
DIRCTION BSS      1
         END
```