

DOCUMENT RESUME

ED 119 734

IR 003 208

AUTHOR Sullivan, Joseph E., Ed.; And Others
TITLE DOTSYS III: A Portable Program for Braille Translation. Rev. 1.
INSTITUTION Mitre Corp., Bedford, Mass.
SPONS AGENCY Library of Congress, Washington, D.C. Div. for the Blind and Physically Handicapped.
REPORT NO MTR-2119
PUB DATE 2 Oct 75
NOTE 75p.

EDRS PRICE MF-\$0.83 HC-\$3.50 Plus Postage
DESCRIPTORS *Braille; *Computer Programs; *Machine Translation; Program Descriptions; Programing; Programing Languages
IDENTIFIERS Computer Translation; DOTSYS III

ABSTRACT

DOTSYS III is a COBOL program for the translation of English text into standard English braille, also known as grade 2 braille. Text in foreign languages or English may be transliterated as grade 1 braille or "computer" braille. The program's method of operation together with instructions on using the program, modifying or extending the translation heuristics, and transferring the program to a new computer environment are presented. General understanding of computer programing and braille translation would be helpful, but no special knowledge in these areas is presupposed. (Author/CH)

* Documents acquired by ERIC include many informal unpublished *
* materials not available from other sources. ERIC makes every effort *
* to obtain the best copy available. Nevertheless, items of marginal *
* reproducibility are often encountered and this affects the quality *
* of the microfiche and hardcopy reproductions ERIC makes available *
* via the ERIC Document Reproduction Service (EDRS). EDRS is not *
* responsible for the quality of the original document. Reproductions *
* supplied by EDRS are the best that can be made from the original. *

MITRE Technical Report

MTR-2119

REV. 1

ED119734

DOTSYS III: A Portable Program for Braille Translation

Joseph E. Sullivan (ed.), with contributions
by the editor, J. K. Millen & W. R. Gerhart

2 OCTOBER 1975

Library of Congress,
Div. for the Blind and
Physically Handicapped

CONTRACT SPONSOR
CONTRACT NO.
PROJECT NO.
DEPT.

1250
D71

U.S. DEPARTMENT OF HEALTH,
EDUCATION & WELFARE
NATIONAL INSTITUTE OF
EDUCATION

THIS DOCUMENT HAS BEEN REPRO-
DUCED EXACTLY AS RECEIVED FROM
THE PERSON OR ORGANIZATION ORIGIN-
ATING IT. POINTS OF VIEW OR OPINIONS
STATED DO NOT NECESSARILY REPRESENT
OFFICIAL NATIONAL INSTITUTE OF
EDUCATION POSITION OR POLICY.

Approved for public release; distribution unlimited.

IR 003 208

THE
MITRE
CORPORATION
BEDFORD, MASSACHUSETTS

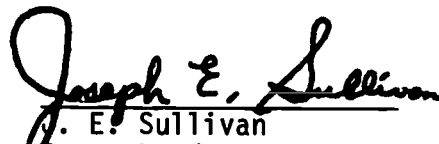
Department Approval: Judith Clapp

MITRE Project Approval: Joseph E. Sullivan

05 03 00 00

ABSTRACT

This document describes DOTSYS III, a table-driven COBOL program for the translation of English text into Standard English (American) Braille, also known as grade 2 braille. Text in any of several foreign languages and English text to be transliterated as grade 1 braille or "computer" braille may also be handled. While general acquaintance with computer programming and the subject of braille translation would be helpful in reading the document, no special knowledge in these areas is presupposed. The program's method of operation, together with detailed instructions on using the program, on modifying or extending the translation heuristics as defined in the tables, and on transferring the program to a new computer environment are presented.


J. E. Sullivan
Group Leader
Command Systems

JES/mg

ACKNOWLEDGEMENTS

It is difficult to acknowledge adequately the many different currents of direct and indirect contribution to this document, even since publication in original form in 1970. The authors have had the benefit of continued interest and ideas from R. A. J. Gildea, project leader of the original DOTSYS development, and G. Dalrymple, acting director of MIT's Sensory Aids Engineering and Development Center. We have likewise learned much from interaction with users of DOTSYS, including M. Boyles and others at the Atlanta Public Schools, the first users, P. Bagley and others at Information Engineering, Inc., J. Gill of U. Warwick, England, and D. Keeping and B. McDonald of U. Manitoba, Canada. Our thinking has been stimulated by contact with colleagues in Germany, France, Denmark and other countries, as well as the United States, who are pursuing other approaches to the problem of Braille translation as it applies to various languages.

R. Evensen of the Library of Congress, Division for the Blind and Physically Handicapped, was the project monitor and braille consultant, and M. Friedman of the Library of Congress was the computing environment consultant, for the Spanish language extensions. G. Dalrymple conducted the live tests, and was assisted by M. Scott of Perkins Institute who proofread the Spanish braille output..

M. Gallo typed this document and otherwise guided it into production.

PREFACE

This document is intended to serve the needs of many different levels of interest. Those interested only in what the program will do, for example, need read only the introduction. Persons with a general interest in the subject of braille translation should add the section on method. Those who actually wish to use the program -- transcribers, editors, and keypunchers -- will, of course, want to read the appropriate parts of the Usage Section and may or may not be interested in method. A systems programmer or other person whose main interest is to get the program running in a new environment ought to read the "Transfer" Section. Finally, a programmer who may wish to modify the program's function should read the whole document, including the final section on maintenance.

This first revision was occasioned by the addition of capabilities for Spanish and other foreign languages in uncontracted form - e.g. as text embedded in a foreign language textbook for English-speaking students. A major change in format from the previous edition has been the removal of the program and tables listing from the document, and the introduction of a procedure for version control of the program (see under "PROGRAM MAINTENANCE") and tables (see under "USAGE"). Considerable improvement has been made in the program's internal organization, for greater ease of comprehension.

This document corresponds to the MITRE 7/75 version of the program, and the MITRE 7/75 tables. For subsequent versions derived from these, comments in the program or tables themselves should indicate in what ways those versions differ.

TABLE OF CONTENTS

		<u>Page</u>
LIST OF ILLUSTRATIONS		x
SECTION I	INTRODUCTION	1
SECTION II	METHOD	3
	GENERAL	3
	THE PRIMARY INPUT PROCESSOR	3
	THE TRANSLATOR*	4
	The Buffer	4
	The Alphabet Table Search	4
	The Contraction Table Search	4
	The Buffer Shift	5
	Braille Sign Output	5
	The State Transition	5
	The Decision Table	6
	THE STACKER	6
	THE BRAILLE LINE COMPOSER	8
	THE FINAL OUTPUT WRITER	8
SECTION III	USAGE	9
	INPUT	9
	Deck Setup	9
	The Echo Option Card	9
	The Tables	10
	Table Data in General	10
	Table Version Control	11
	The Alphabetical Contraction List	11
	The Special Symbols Card	12
	The Alphabet Table	13
	The Contraction Table	13
	The Card Specifying the Number of Right-Context Classes	15
	The Right-Context Table	15
	The Card Specifying the Number of State Variables	15
	The Card Specifying the Number of Input Classes	15
	The Transition Table	15
	The Card Specifying the Number of Decision Table Columns	16
	The Decision Table	16
	The Sign Table	16

TABLE OF CONTENTS (Cont.)

	<u>Page</u>
The Run Control Cards	17
Text Input	19
General Key punching Rules	19
Capitalization	19
Italics	20
Indicating Contractions in Self-Checking Mode	20
Ordering	20
Forcing and Preventing Contractions	20
Replacement Symbols for Special Characters	21
Grade Switch (\$G)	22
Foreign Languages	22
Format and Mode Control Symbols	24
Self-Checking	26
Octal Braille	28
Computer Braille	28
Notes to the Braille Editor	29
Role of the Editor	29
Letter Sign (+)	30
Division (or Null) Symbol (\$/)	30
Forced-Contraction Symbols (/_, _/)	30
Termination Symbol (\$T)	31
OUTPUT	31
Echo Output	31
Proof Output	31
Elastomer Braille Output	32
RPQ Braille Output	33
Punched Card Output	33
MIT Brailleboss Output	33
Error Messages	33
SECTION IV	
PROGRAM TRANSFER	37
SECTION V	
PROGRAM MAINTENANCE	39
WHEN MAINTENANCE MAY BE REQUIRED	39
PROGRAM VERSION CONTROL	39
TABLE-SIZE BOUNDS	39
REPLACING THE ALPHABET TABLE SEARCH BY DIRECT INDEXING	40
CONTRACTION TABLE SEARCH ALGORITHM	42
General Rationale	42
Notation	43

TABLE OF CONTENTS (Concl.)

	<u>Page</u>
APPENDIX I	47
APPENDIX II	49
DEFINITION OF STATE VARIABLES AND INPUT CLASSES	
APPENDIX III	51
SUMMARY OF SPECIAL SYMBOLS	
APPENDIX IV	55
EQUIVALENT SIGNS, SYMBOLS AND CODES	
APPENDIX V	57
TRANSLATOR - STACKER SIGN CODES	
APPENDIX VI	59
MISCELLANEOUS CODED VARIABLES	
APPENDIX VII	61
USERS OF DOTSYS III AND ITS DERIVATIVES	
APPENDIX VIII	65
SAMPLE PROOF OUTPUT	
REFERENCES	67
DISTRIBUTION LIST	69

LIST OF ILLUSTRATIONS

<u>Figure Number</u>		<u>Page</u>
1	Decision Table Schematic	7
2	Sample of Proof Output	32
3	Elastomer Braille Output for Line in Figure 2	33
4	Punched Card Output for Line in Figure 2	34
5	Set-Up Algorithm	44
6	Look-Up Algorithm	45

LIST OF TABLES

<u>Table Number</u>		<u>Page</u>
I	Alphabetical Contraction List Card Format	11
II	Special Symbols Card Format	12
III	Alphabet Table Card Format	13
IV	Contraction Table Card Format	14
V	Right-Context Class Card Format	15
VI	Decision Table Card Format	16
VII	Sign Table Card Format	17
VIII	Suggested Number of Braille Lines Per Page	18
IX	Variable Table Capacity References	39

SECTION I

INTRODUCTION

Braille, as it is used today in almost all contexts other than primer textbooks, is almost a system of shorthand and not simply a scheme for representing individual inkprint symbols in a tactile code. This system, called Standard English Braille (American) or grade 2 braille, is defined in Reference 1. A letter-for-letter transcription is called grade 1 braille.

The chief device used to reduce the number of braille signs in grade 2 is the contraction. A contraction is the representation of an inkprint letter-group or whole word by a relatively short sequence of braille signs; for example, the word "receiving" is represented by the four braille signs for r, c, v, and g in succession. There are 189 letter-groups that may be contracted.

Unfortunately (at least from the standpoint of automating the translation process), a given contractable letter-group is not necessarily contracted wherever it appears. Moreover, the rules governing the use of contractions frequently involve such matters as pronunciation and meaning. For example, in the word "disease," the "dis" and the "ea" are normally contracted. However, when this word is used in the (now obsolete) sense "lack of ease," the "ea" should not be contracted. This example, although admittedly farfetched, illustrates that in some circumstances even a human transcriber might have difficulty applying the rules. Cases routinely arise that are easy for a human transcriber, but still difficult for a mechanical process -- for example, distinguishing the musical note "do" from the verb "do." For these reasons, automatic natural-language to braille translation algorithms tend to be heuristic, which is to say fallible.

DOTSYS III is a computer program embodying such a natural-language-to-braille translation algorithm. As its name implies, it is an outgrowth of an earlier program, DOTSYS II, described in References 2 and 3.* The basic translation algorithm remains essentially the same, but a number of new features have been added, the translation quality has been improved, and better internal processing methods have been introduced. These improvements in capability have been offset by the improved methods, so that the

* In order to make the present document as self-contained as possible, portions of Reference 2 and 3 have been incorporated with little or no change.

overall speed of DOTSYS III remains about the same as DOTSYS II (1300 wpm on an IBM 360/50, 3330 wpm on a 360/65). DOTSYS II, in turn, owes the fundamentals of its translation algorithm both to previous work in the field of braille translation by computer and to elementary concepts in the theory of automata, logic design, and formal languages. The background references and relationships discussed in Reference 2 are relevant to DOTSYS III also.

The source language used in coding DOTSYS III is ASA Standard COBOL (Reference 4). The IBM 360/370 compiler for this language (Reference 7) was used to implement later versions; nonstandard IBM extensions were avoided where possible. This should enable DOTSYS III, with relatively little modification, to be run on any computer having a COBOL compiler and a modest amount of core storage (approximately equivalent to 72,000 bytes of IBM 360 storage).

The input text is presented to DOTSYS III in the form of 80-character (punched card image) records. These can be manually keypunched directly from inkprint or produced by another program according to a fairly straightforward set of conventions. The output is the sequence of braille signs equivalent to the input text. Output can be produced in any of several forms, including "proof" output for a sighted editor and tactile (embossed) braille.

DOTSYS III is almost completely table-driven; i.e., details of the translation algorithm ~~are determined by tables read in at~~ execution time rather than by the program itself. DOTSYS III, as described in this document, comprises both the program and a standard set of tables. In principle, with modified tables, the DOTSYS III program would be capable of processing different kinds of text, such as text containing mathematical or technical notation, languages other than English*, or text containing nonstandard symbols for format control.

*The present version has provision for "grade 1" handling of certain foreign languages, which is standard for foreign text within basically English-language works, and generally for literature used by foreign language students.

SECTION II

METHOD

GENERAL

DOTSYS III comprises five cooperating processors: the primary input section, the translator, the stacker, the braille line composer, and the final output writer. It is the translator, as its name implies, that does most of the work of braille translation, and in fact, this section forms a sort of main loop or program, the others being in the form of subprograms called by the translator to do their work at the appropriate time. However, in order to follow the processing of a given piece of text from inkprint form to braille form, it is easiest to imagine the processors running sequentially, each one in turn operating on the output, or a collection of outputs, from the previous processor.

The following descriptions of these processors are idealized to some extent, so that the discussion does not become hopelessly mired in detail.

THE PRIMARY INPUT PROCESSOR

The primary input section reads the 80-character (card-image) records. Columns 73-80 are discarded, so that the first character of a record logically follows character 72 from the previous record. This stream of characters is further collapsed by deleting all instances of the vertical bar character (|)*, and by deleting all consecutive blanks after the second in a series following a period (.) and all after the first in any other context. This collapsed stream of characters, one at a time, is the output of this section.

In the "self-checking" mode (described in detail in a later section), the primary input processor prepares the correct translation words for later comparison against the translator output.

*The vertical bar, and practically all other "distinguished" symbols discussed in the text can be altered to some other symbol by changes in the tables, as discussed under "USAGE" and "PROGRAM MAINTENANCE."

THE TRANSLATOR*

The Buffer

The translation section operates on the stream of characters issued by primary input. As a first step, these are collected into a ten-character sliding window, called the "buffer," so that a group of characters can be examined.

The Alphabet Table Search

The leftmost character in the buffer is looked up in the alphabet table. In this table, there is exactly one entry for each symbol that may appear in the text, containing, among other things: (a) a code denoting the braille sign for that symbol, (b) the symbol's "input class," and (c) an index to that portion of the contraction table which pertains to this initial letter. An input class is an arbitrary numerical code with three distinct purposes, as will be seen.

The Contraction Table Search

The next step is to search that section of the contraction table indicated for this initial letter. In principle, this search may be visualized as a simple top-to-bottom entry-by-entry sequential search, although in fact a much faster tree-search algorithm (described in detail under "PROGRAM MAINTENANCE") is used. An entry in the contraction table consists of: (a) a string of up to nine characters (ten, counting the implied initial letter); (b) a "right-context class" designator; (c) an input class code; (d) a shift count; and (e) a set of up to four braille sign codes.

For a match to occur on a particular entry, three conditions must be satisfied. First, the entire string for that entry must correspond to the buffer, or left substring thereof. Secondly, the input class for the entry determines a set of "state variable" conditions that must be satisfied. A state variable is a logical switch, having a value "yes" or "no" according to its meaning and the

*The translator operates essentially as described in Section III of Reference 2, except that (1) the contraction table search has been speeded up considerably by using a modified binary search, which affects the table ordering rules slightly, (2) the STRING field delimiter in the contraction table has been changed from "\$" to "|" (vertical bar), and (3) the decision table permits a new decision symbol "F" meaning "unconditional no."

text already translated. For example, a state variable whose meaning is "after a digit" would have the value "yes" if the character immediately preceding the one leftmost in the buffer had been one of the digits 0-9 and would have the value "no" in all other circumstances, including initially. "Meaning" is, strictly speaking, defined completely by entries in another table which governs the setting of these switches, called the transition table. This table will be discussed presently. The mechanism by which the input class selects a set of state variable conditions to be tested is called the decision table; this table is discussed in more detail under "The Decision Table," below.

The third condition for a contraction table match to occur is that the right-context class be correct. If the right-context designator for the entry is blank, no right-context condition is imposed. Otherwise, the character immediately to the right of the matched string in the buffer is looked up in the alphabet table to determine its input class. Another table, called the right-context table is consulted to determine whether that input class is one of those listed as acceptable for this designator--e.g., the designator "p" (for "punctuation") may apply to input classes 2, 3, 13, and 14.

The contraction table search stops when a match occurs or the end of the table section for that particular initial letter is reached. In the latter event the shift count is taken to be 1, the input class and braille sign code revert to those found for the initial letter, and processing proceeds.

The Buffer Shift

The buffer is then "shifted" left by the amount of the shift count, with new characters from the input stream entering on the right.

Braille Sign Output

The braille sign code(s) are sent to the stacker section, constituting the output of the translator. These may directly represent braille signs, or they may be special control codes as is discussed in the section describing the stacker.

The State Transition

Finally, the input class is used to determine a set of transitions to be applied to the state variables. For each variable, the transition may be specified as "no change," "toggle" (change "yes" to "no" and "no" to "yes"), "set to yes" or "set to no."

After the state variable transition, the process is repeated, beginning with the alphabet table search.

The Decision Table

The decision table determines whether the current settings of the state variables permit the use of a given contraction table entry. It is best represented by a tableau in two parts, as depicted in Figure 1. The upper, or decision portion, has a row for each input class; the lower, or condition portion has one row per state variable. The number of columns is the same for both sections, and this number will depend upon the number of independent tests that may have to be made. The elements of the array are single characters, as given in Figure 1.

Processing of the table for a particular input class consists of a left-to-right scan of the associated row in the upper portion. If a "G" ("Go") symbol is encountered, processing stops with a "yes" decision. If an "F" ("Fail") is encountered, processing is likewise terminated, but with a "no" decision.

If one of the symbols "Y" or "N" is reached, then the corresponding column in the lower portion is compared against the current settings of the state variables. A "Y" in the *i*th row implies that the *i*th state variable must have the value "yes"; an "N" demands the value "no" and a dash is satisfied by either setting. If the specified conditions are satisfied for all the state variables, then processing is terminated with a "yes" decision if the upper portion had a "Y" symbol, and "no" if it was an "N". If one or more state variables does not have the value specified, then scanning of the decision section row is resumed.

When a dash is reached in the scan, the scan simply continues with the next column. If the scan encounters the end of the row without otherwise reaching a decision, then the decision becomes "no."

THE STACKER

The stacker operates upon the braille sign codes issued by the translator. In the simplest case, these codes are merely accumulated until the code for the blank braille sign is received--i.e., a braille word is finished. This word, constituting one entry in a stack, is normally then removed from the stack and issued as output to the braille line composer. In exceptional circumstances, two or more words may be held in a stack before release. This may be because the

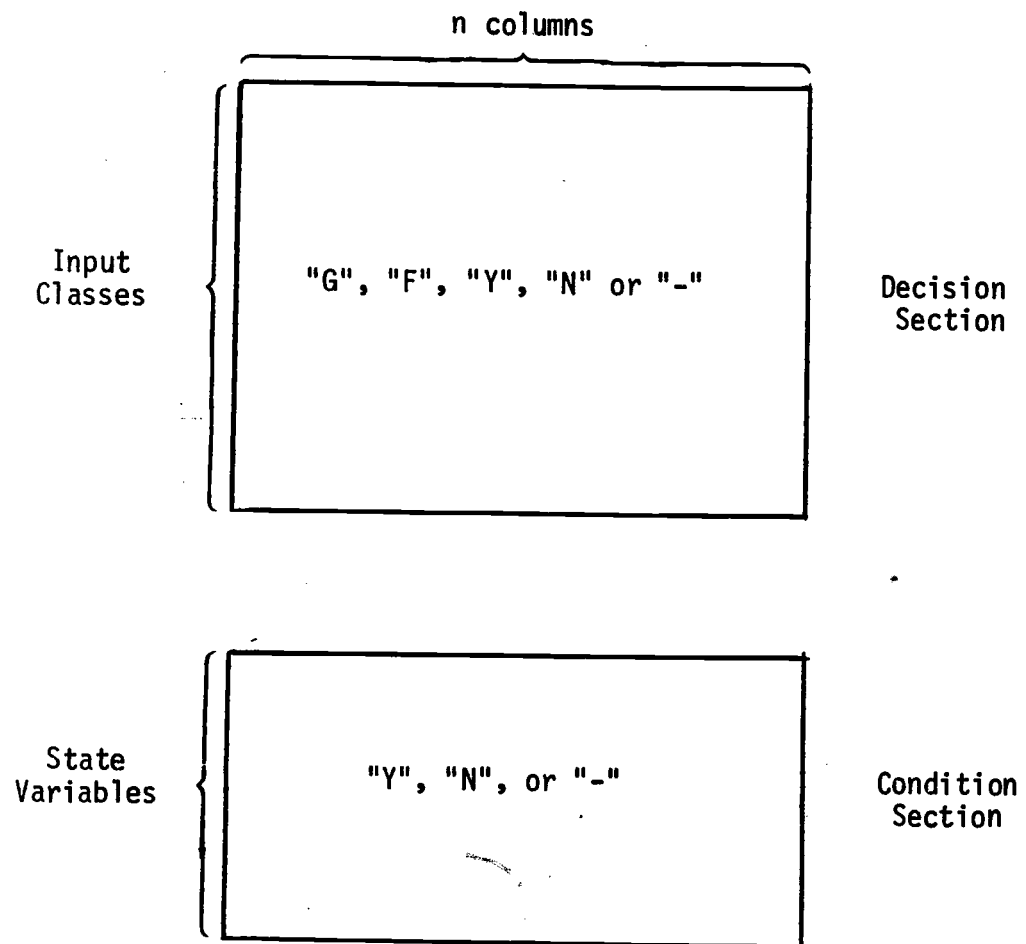


Figure 1. Decision Table Schematic

order may have to be changed (in braille, units of measure are placed before the associated numeric quantity), or because the overall length of several words must be computed before issuing the first one--as when centering headings.

All special operations by the stacker, including delayed release and interchange of order, are directed by control codes issued by the translator. These are distinguishable from ordinary sign codes in that they have a value greater than 64. (See Appendix V.)

The stacker maintains up to three distinct stacks, sharing a common area by borrowing entry fields from a linked free storage list. One stack is used for normal output; a second stack is used to collect the words in a heading (such as a chapter title); the third is used to hold the title (running head) if it is present.

In the "self-checking" mode (described in a later section), it is the stacker that compares each braille word against the correct translation prepared by primary input.

THE BRAILLE LINE COMPOSER

The line composer operates on braille words (stack entries) produced by the stacker. In each case, the length of the word will determine whether it can fit on the current braille line, an internal output buffer. If so, it is simply added thereto. Otherwise, the current line is sent to the final output writer, cleared, and started anew with the current braille word.

The line composer also concerns itself with counting lines to determine a new page condition, page numbering and titling, and similar matters.

THE FINAL OUTPUT WRITER

The output writer is actually a collection of processors, one for each distinct mode of output. For each mode selected, the associated processor produces actual output corresponding to the current braille line set up by the line composer.

SECTION III

USAGE

INPUT

Deck Setup

The complete input deck read by DOTSYS III consists of four main sections, in order:

- (1) the echo option card;
- (2) the tables;
- (3) the run control cards; and
- (4) the text.

Sections 1-3 must be sequence-numbered (in columns 73-80), in strictly increasing order, unless "NO-SEQUENCE-CHECK" is selected on the echo option card.

The Echo Option Card

The echo option card determines whether a literal listing of the tables and run control cards will be printed on the SYSPRINT (normal printer) output device, what symbols are used to flag comments in the tables, and whether sequence checking is to be performed on the input other than the text.

If column 1 of the echo card contains an "E," (for "echo") a listing of the tables and run control cards is produced; if it contains an "N", (for "no echo") it is not. For production use, the "N" option would be usual. The echo card itself is always printed; printing of the text is controlled by one of the run control cards.

Columns 9-10 contain a symbol that, when it appears in Columns 1-2 of a card within the tables or run control cards, indicates a comment card which is printed (if "echo" is on) but otherwise ignored. Comments may not appear in the text.

Column 16 should contain a "S" if checking of the sequence field (Columns 73-80) should be performed for the echo, tables and run control cards, "N" otherwise.

The Tables

Table Data in General

The tables are supplied with the DOTSYS III program and form an integral part of the process described by this document. However, circumstances may arise such that the user may wish to alter or augment the tables. One such circumstance might be a word found to be incorrectly translated by DOTSYS III, that occurs so often in a given text that it is impractical to force the correct translation by special treatment (see the "\$/", "/_" and "_/" control symbols under "Forcing and Preventing Contractions" in "Text Input", below). In such a case, an addition to the contraction table is called for.

The tables and associated dimensioning cards are to be arranged in the following order:

- (1) the alphabetical contraction list;
- (2) the special symbols card;
- (3) the alphabet table;
- (4) the contraction table;
- (5) the card specifying the number of right-context classes;
- (6) the right-context table;
- (7) the card specifying the number of state variables;
- (8) the card specifying the number of input classes;
- (9) the transition table;
- (10) the card specifying the number of decision table columns;
- (11) the decision table; and
- (12) the sign table.

The formats of the tables and cards listed are given in following paragraphs. All two-column numerical fields must contain two digits; in particular, a number less than 10 must be given a leading zero when used in a two-column field. Numerical limitations stated in the form; "this number may not exceed . . ." are appropriate for the table sizes used in the version of the DOTSYS III

program mentioned in the PREFACE. Refer to "TABLE SIZE BOUNDS" under "PROGRAM MAINTENANCE" (Section V) if changes in table sizes are contemplated.

Table Version Control

If changes are made to a table that is shared by other institutions, a new version has been created and it should be labeled according to the authoring institution and date, e.g., "MITRE 7/75." Subsequent alterations at the same institution, prior to any external distribution, need not be considered new versions. If possible, a comment should be given, at the head of the tables and following other such comments, with the version identification and information detailing the changes from the previous version, including the reasons for the changes and any important consequences for a user. For versions of the table in which comments are not permitted, it is recommended that similar information be included in a separate sheet attached to a listing and that copies be included with any distribution.

If this document is updated, the preface should be altered to show the corresponding version identification.

The Alphabetical Contraction List

This table contains exactly 189 cards, one for each contraction defined in grade 2 English braille. The order of input must be alphabetical. The table is used in support of the self-checking feature (described under "Text Input" below). The format is as given in Table I.

Table I

Alphabetical Contraction List Card Format

<u>Columns</u>	<u>Contents</u>
1-10	The inkprint contractable letter sequence, left-justified.
24-31	Four 2-digit braille sign codes (Cf. Appendix IV) representing the equivalent braille sign(s). 99's are used for filler on the right.
32-33	Presently ignored, but reserved for a fifth sign code.

The Special Symbols Card

This card defines certain symbols that are used in a special way in the tables, the program, or both. These are listed in Table II.

Table II

<u>Column</u>	<u>Contains Symbol to be used for:</u>	<u>Usual Symbol*</u>
1	(1) contraction table entry delimiter; (2) in "self-check" input text, delimiter of contracted sequences; (3) internally as an "end of text" delimiter	
2	replacing text symbols that are not found in the alphabet table (note: this replacement symbol <u>must</u> be in the alphabet table).	*
3	capitalization indicator	=
4	italics indicator	-
5	"literal" sign	+
6	accent mark	%
7	left parenthesis	(
8	right parenthesis)

*These symbols are generally referred to in their usual literal form elsewhere in this document, e.g., "%" instead of "symbol used for accent mark."

The Alphabet Table

The alphabet table identifies all legal text input characters. The order of input is arbitrary, but for the sake of efficiency should normally be by decreasing frequency of occurrence of the symbol. (Note also that the order is related to that of the contraction table, q.v.) A dummy entry, with 99 in columns 18-19, should follow the last actual alphabet table entry. Including this card, the total number may not exceed 64. The card format is given in Table III. Note that there must be an entry for "*" (see "The Special Symbols Card," above).

Table III

Alphabet Table Card Format

<u>Columns</u>	<u>Contents</u>
1	The symbol being defined.
18-19	The input class (must be in the range from 01 to the number of input classes). Note: A card with 99 in this field signals that the end of the alphabet table has been reached.
21-22	The sign code to use when the symbol occurs in a computer-braille string (see "Text Input," below and Appendix IV).
24-25	The sign code to use in normal context (Cf. Appendix IV).
30-31	01 if the symbol may be used as a prefix to, or as one of a pair of symbols bracketing, a letter denoting itself (e.g., the quote marks in "f"); 00 otherwise.

The Contraction Table

The contraction table contains not only contractions but many other sequences related to the heuristics of the translation process, and the definition of special control symbols.

Entries in the contraction table beginning with the same symbol must be grouped together, and these groups must be arranged in the

same order as the corresponding symbols in the alphabet table. Also, symbols in the alphabet table that have no corresponding section in the contraction table are grouped, in any order, at the end of the alphabet table.

Within a section of the contraction table determined by a common initial letter, all entries having a given second character must also be grouped together. The order of input of these subsections is completely arbitrary and usually will vary from section to section as a function of conditional frequency. This grouping scheme is continued right up to the 10th character. In general, if two entries agree through the nth character, then all entries between them must also agree with them through the nth character.

A dummy entry, with 99 in columns 18-19, should follow the last actual entry. The total number of cards, including the dummy, cannot exceed 1,200.

Table IV gives details of the contraction table card layout.

Table IV

Contraction Table Card Format

<u>Columns</u>	<u>Contents</u>
1-10	Character sequence, left-justified. If the string is shorter than 10 characters, then a vertical bar () should follow the last character. (Thus blanks are permitted in the string.)
16	Right-context class designation; may be specified (non-blank) only if the string is shorter than 10 characters.
18-19	Input class. Note: A card with 99 in this field signifies that the end of the table has been reached.
21-22	Shift count.
24-31	Four two-digit sign codes (Cf. Appendix IV). 99's are used for filler on the right.

The Card Specifying the Number of Right-Context Classes

As its name implies, this card contains the number of right-context classes that are to be defined. This figure is punched in columns 18-19. It cannot exceed 05.

The Right-Context Table

This table contains one card per right-context class; i.e., there are as many cards as signified on "the Number of Right-Context Classes" card, just previously described. The layout is given in Table V.

Table V

Right-Context Class Card Format

<u>Columns</u>	<u>Contents</u>
16	A single-character designator for the class being defined--e.g., "P" for "punctuation."
24-31	Up to four input class codes. All symbols having one of the listed input classes are implied to have the right context class being defined. If there are fewer than four input class codes, the last one is simply repeated to make four.

The Card Specifying the Number of State Variables

The number of state variables is punched in columns 18-19. This number may not exceed 15.

The Card Specifying the Number of Input Classes

The number of input classes is punched in columns 18-19. This number may not exceed 40.

The Transition Table

This table contains one card per state variable. On each such card, the i^{th} column contains a character signifying how the state variable is to be set when input class i is processed. An "R" signifies "reset" (set to "no"), and "S" means "set" (to "yes"), a dash (-) means "leave," and "T" means "toggle" (change to opposite state).

The Card Specifying the Number of Decision Table Columns

The number of columns in the decision table is punched in columns 18-19. This number cannot exceed 20.

The Decision Table

This table contains one card per decision table column. The layout of each card is given in Table VI.

Table VI

Decision Table Card Format

<u>Columns</u>	<u>Contents</u>
1 - nic*	The i th card column contains the upper (decision) section symbol for the input class i . It must be G, F, Y, N or -. (See "The Decision Table.")
nic - (nic+nsv)*	The (nic + j)th card column contains the symbol denoting the condition imposed on the j^{th} state variable. It must be Y, N or -.

The Sign Table

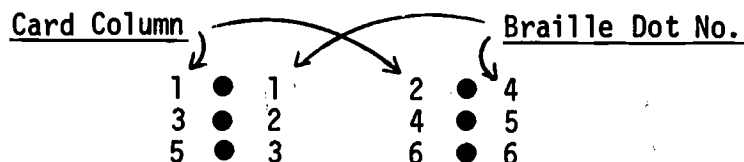
The sign table contains exactly 64 cards, one for each "ordinary" braille sign code. (Codes 00 and 64 both refer to sign 64.) The i^{th} card defines code i . The layout is given in Table VII.

* nic is the number of input classes, nsv the number of state variables.

Table VII

Sign Table Card Format

<u>Columns</u>	<u>Contents</u>
1-6	Dots, keypunched as periods, corresponding to the braille dots embossed for this character. The correspondence is:



7-9	Three characters to be printed on proof output, denoting the most common meaning for this sign.
10	Symbol to be used for this sign in Brailleboss output (see under "Run Control Cards," below).

The Run Control Cards

The run control cards select options that remain in effect throughout the translation of the text--i.e., for the entire "run," or job step.*

There are eight run control cards. The first five select the output modes (described under "output," below); any combination is permitted. On these, only column 1 is read; it should contain an "N" if the output mode is not wanted or a letter associated with the mode otherwise. Specifically, the options cards are, in order:

*On the IBM 360 under OS, it is possible to use IBM Operating System Job Control Language to run part of the text input with one set of DOTSYS III control cards, and change control cards for a succeeding part of the input. This could be done simply by using the IBM utility IEBGENER to copy the tabular input onto a temporary disk data set, say &&TABLES. Then, instead of executing the catalogued procedure COBFLG just once, do it once for each part of the text input. The SYSIN data set to use is the concatenation of &&TABLES with a DD * data set consisting of the control cards and the text of that part. Quite possibly, similar facilities exist on many other systems.

- (1) Proof output (P or N in column 1).
- (2) "Elastomer" braille (B or N in column 1).
- (3) "RPQ" braille (R or N in column 1).
- (4) Punched-card output (P or N in column 1).
- (5) MIT Brailleboss output (M or N in column 1). If selected (M), columns 16-20 should contain output characters for the following embosser control functions: turn on (16); turn off (17); idle (18); carriage return (19); page (20).

The sixth and seventh control cards select the braille page dimensions. In both cases the required number is punched in columns 18-19:

- (6) The number of braille signs per line (not less than 2 nor greater than 40).
- (7) The number of lines per 11-inch page. Table VIII gives suggested values for this item for certain kinds of output.

Table VIII

Suggested Number of Braille Lines Per Page

	Line Printer Setting Lines/Inch		
	6	8	10
Proof Output	10	13	16
Elastomer Braille Output	15	20	25

The last card determines whether or not automatic titling and page numbering is to occur:

- (8) If the top line of each braille page is to be reserved for a running title and automatically produced page number, a "P" should be punched in column 1 and the starting page number should be punched in column 1 and the starting page number should be punched in columns 32-35. Otherwise, a "N" should be punched in column 1. In this latter case, no automatic page numbering is done and running titles, while still permitted in the input text, will not appear on the output.

Text Input

General Keypunching Rules

Text is punched in columns 1-72 of each text input card using an EBCDIC keypunch (e.g., IBM 029) or the equivalent multiple punches on another model. Letters, numbers, and punctuation are reproduced as they appear in normal typewritten English text, with the exception of quotation marks, accent marks, mathematical symbols, and brackets ([,]). However, additional symbols must be added to the text to indicate capitalization, italics, and format controls, to force or prevent improper translations in exceptional cases, and to define the correct translation for self-checking purposes.

Column 72 of a card is considered to be adjacent to column 1 of the next card. In general, any number of spaces will be collapsed automatically into a single space (refer to the description of the primary input processor), except following a period, where a number of spaces greater than or equal to 2 will be interpreted as 2 spaces; (e.g.:

THE START. THE END.

is interpreted as

THE START. THE END.)

Capitalization

If the first letter of the inkprint word is capitalized, the keypunched word must be preceded immediately by an equals sign (=). If the whole inkprint word is capitalized, the key-punched word must be preceded immediately by two equals signs (==). When Roman

numerals are written as capital letters, a single equals sign must be used before a single letter and two equal signs must be used before numerals containing two or more letters.

Italics

If only one, two, or three successive inkprint words are italicized, each of the words must be preceded immediately by an (underline (_) when keypunched.

If four or more successive inkprint words are italicized, the first word must be preceded immediately by two underlines (_ _) and the last by one underline.

Indicating Contractions in Self-Checking Mode

If the self-checking feature of DOTSYS III is to be used, then all contractions which should be made must be surrounded by vertical bars (|). Refer to the section entitled "Self-Checking", below.

Ordering

When two or more special signs must be punched (e.g., an italicized capital letter), the ordering should be:

Italic sign (_)

Capital sign(s) (=, or ==)

Accent sign (%)

Vertical bar (|)

Forcing and Preventing Contractions

The form of any contractable letter group can be forced to occur, regardless of context, by surrounding the letter group with the symbols "/_" and "_/". For example:

a/_dd_/

would cause the "dd" contraction to be used even though otherwise the program would not (and should not) use it at the end of a word.

A contraction that would otherwise occur can be prevented by introducing the null replacement symbol \$/ (see below). For example,

DISE\$/ASE

will prevent the "EA" contraction.

Replacement Symbols for Special Characters

DOTSYS Reserved and Special Literal Symbols. DOTSYS uses certain symbols in a special way or as the first character of a class of special control symbols. Any one of these, viz. \$, %, /, & and #, may be forced in any particular instance to translate as itself, regardless of context, by preceding the symbol with "\$:", e.g.

\$:\$ for \$
\$:& for &

In cases where it is necessary to emphasize that the special symbol itself is intended (e.g. & is indistinguishable in braille from "and" - see Reference 1, rule VIII, Paragraph 31.a) the control \$SYM should be entered prior to the symbol, \$: combination, or spelling-out, e.g.:

\$SYM\$:& for &
\$SYM@ for @
\$SYM# or \$SYM\$:# for #

Mathematical Symbols. Symbols such as + (plus), - (minus), = (equal), > (greater than), and < (less than) must be spelled out as words, even though they appear on the keypunch.

Quotation Marks. The single quote character on the keypunch (') is always taken to mean an apostrophe; thus, special symbols must be used for single quotes.

\$' is punched for a left single quote.

\$'R is punched for a right single quote.

Ordinarily, the double quote (") on the keypunch may be used for both left and right double quotes. However, double quotes within the scope of another pair of double quotes must be represented by special symbols.

\$" is punched for a left double quote within quoted text.

\$"R is punched for a right double quote within quoted text.

Accent Marks. Any accent or other diacritical mark used with a letter (such as é, è, ê, ä, ç) is represented by preceding the keypunched letter with a percent sign (%). For example, Abbé is keypunched =ABB%E. However, see the section entitled "Foreign Languages" for proper handling of passages in foreign languages, other than anglicized words or proper names.

Brackets.

< is punched for a left bracket ([].

> is punched for a right bracket ([]).

Short Syllable Sign. To insert a short syllable sign, the special symbol \$SV is used.

Long Syllable Sign. To insert a long syllable sign, the special symbol \$LV is used.

End of Poetry Foot Sign. To insert an end of poetry foot sign, the special symbol \$FT is used.

Caesura Sign. To insert a caesura sign, the special symbol \$CS is used.

Forced Blanks. To produce multiple blanks or otherwise control their placement, the symbol \$B is provided. One blank is produced for each occurrence of the symbol (e.g., A\$B\$B\$BC produces A C).

Letter Sign, Division (or Null) Symbol, and Termination Sign. These are discussed under "Notes to the Braille Editor," below.

Grade Switch (\$G)

An occurrence of the grade switch, \$G, changes the mode of translation from grade 2 to grade 1 or vice versa. It must precede and follow any sequence of characters in which no contractions are to be used, such as a foreign word. \$GON (grade 1 on) and \$GOF (grade 1 off) may be used as more perspicuous alternatives to \$G, but caution should be used that they occur in pairs for they are merely synonyms to \$G.

Foreign Languages

General. Works incorporating passages of foreign language text must be translated uncontracted and using special braille signs for

special symbols such as accented letters (Reference 1, Rule V, Paragraph 24-26 and Appendix B).

Mode and Grade Switches. The class of language to be so translated must be declared in advance by use of a "\$FL" (foreign language) mode switch. The two presently available are:

\$FL-SPAN	Spanish
\$FL-LIFG	Latin, Italian, French, and German

The mode switch need be entered only once in the text, unless the class is to change in the midst of the text, and may be placed anywhere before the foreign language passage(s). If no switch is present, \$FL-LIFG is assumed.

Note that a mode switch is required not only because more abbreviated input symbols were chosen for Spanish, as discussed below, but also because one inkprint symbol, viz. é, is brailled differently in the two language classes and so a unified set of input symbols was not possible.

A switch into grade 1 mode (\$G. . . \$G or \$GON. . . \$GOF) should be made for the entirety of each foreign language passage.

Latin, Italian, French and German Symbols. For vowels with accents, enter "&" followed by the vowel followed by:

A	for an acute accent (´)
G	for a grave accent (`)
C	for a circumflex accent (^)
D	for a dieresis or umlaut (¨)

For example, &EA is input for é. For other special symbols:

<u>enter</u>	<u>for</u>
&CC	ç (cedilla)
&AE	ae (diphthong)
&OE	oe (diphthong)
\$LV	long vowel sign (before affected letter)
\$SV	short vowel sign (before affected letter)

Spanish Symbols. For vowels with accents, enter "&" followed by the vowel for all acute (´) accents. Enter &D for ü. For other symbols:

enter

&N
&?
&!
-- (preferred) or \$-
-- (preferred) or \$-R

for

~
ñ
¿ or ?
¡ or !
opening conversation sign (—)
closing conversation sign (—)

The conversation signs should be entered before the first quoted word or after the last with spacing conforming to inkprint.

Format and Mode Control Symbols

Keypunching Rule for Spaces Around Format Controls. Unlike replacement symbols where spaces before or after the symbol are interpreted as spaces (i.e., word dividers), any spaces before or after format symbols are ignored. However, the general rule is to provide spaces around each format symbol to avoid possible ambiguity (e.g., \$LVOICE would be interpreted as \$LV OICE even though \$L VOICE may have been intended). Otherwise, the control symbols will usually operate properly regardless of the presence or absence of blanks.

Paragraphs. The symbol \$P must be keypunched to indicate the beginning of a new paragraph. The text following the \$P is started on the next line after two spaces (that is, starting in the third cell position). Use \$P" to start a paragraph within a quotation.

New Line. The symbol \$L may be used to begin a new output line. Caution: at most one line will be skipped, even if the \$L symbol is repeated. To skip more than one line, see "Skip Multiple Lines."

Skip Multiple Lines. The symbol \$Lnnb, where b is a blank and nn is a two digit number (with a leading zero, if necessary) will skip the number of lines indicated, and output will continue from the left margin.

New Page. The symbol \$PG may be used to begin a new page.

One-Time Tabulation. If the symbol \$TABnnb is punched, where n's represent digits and b represents a blank, the text beginning immediately after the blank will be started at column nn. Tabulation is implemented by the automatic insertion of spaces into the output and will therefore proceed to the next line if nn is less than or equal to the present column number. A leading zero must be supplied if the column number is 9 or less.

Permanent Tabs. Numbered tabs can be set so that the user can right, left, or decimal point justify a word or number on any column. For example, the symbol \$STB4L03 means set tab 4 to left justify on column 3. The symbol \$STB means to set tab, followed by the one digit number of the tab to be set, followed by an L for left justification (alignment of the left most character), R for right justification or a D for decimal justification. The last two digit numbers are the column on which justification is to take place. After a tab has been set, it may be executed any number of times by inserting the symbol \$# followed by the one digit number of the tab to be executed. (Example: \$STB4L03 \$#4 LEFT--will left-justify the word LEFT on column 3. The symbol \$#4 can be used any number of times.) If a tab is called in a cell position less than or equal to the present position, output will begin on the next line.

The definition of a given tab number may be changed as often as desirable in the text. Initially, all tabs are set to left-justify on column (5 x tab number).

Titles. Titles are placed between the symbols \$TLS (Title START) and \$TLE (Title END), (e.g., \$TLS THIS IS A SAMPLE TITLE \$TLE.) After a title has been inserted, each subsequent page has a centered title as its first line (the same line which contains the page number). Pagination must be turned on for titles to appear on output (see "The Run Control Cards"). If a new title is entered, it takes the place of the old on all future pages. Entering the title does not automatically turn to a new page.

Headings. These are similar to titles except that the control symbols are \$HDS and \$HDE and that headings are a one time occurrence. Headings are centered on the next line with subsequent input beginning in column 1 of the line after the heading. Headings that overflow begin in Column 4 of successive lines.

Poetry. The symbols \$PTYS (Poetry START) and \$PTYE (Poetry END) are placed before and after all poetry text input. In this mode, the continuations of all poetry lines which exceed the physical length of the output line will be indented two spaces.

Turning the Self-Checking Mode On or Off. The symbol \$SCON\$/\$/\$/\$/\$/ is used to turn self-checking on and \$SCOFF is used to turn self-checking off. (See "Self-Checking" below.)

Self-Checking

Self-checking is a feature that allows DOTSYS III to be used to check itself (and the contraction table) against a text specially marked to indicate the correct translation. This is accomplished by automatically comparing the results of two translations:

- (1) the normal process, ignoring the special markings, and
- (2) a trivial special process, wherein the markings are used to determine where contractions occur.

Discrepancies are flagged on the output, so that the clerical effort required to check the program's correctness is greatly reduced.

A deck containing 5,808 typical and problem words and phrases, taken from the Transcribers' Guide to English Braille (Reference 6), has been prepared with the special markings necessary for self-checking.

In order to turn on self-checking, a control symbol ($\$SCON\$/\$/\$/\$/\$/$) must be included in the text. The text following this symbol is punched normally except that vertical bars (|) should immediately surround any sequence of letters that should be contracted in a word. For example,

ever v th/ing*

should be punched |EVER|Y|TH||ING|

Note: A vertical bar at the end of the word and followed by a blank may be omitted; i.e.,

|EVER|Y|TH||ING is an acceptable variation.

Those braille words not passing the comparison test are output as they ordinarily would be but flagged by appending two braille "for" symbols (all dots). In the proof output, the comment "TSK nnn" appears to the right of any line containing a flagged word. (If the first word in a line is in error, the TSK appears on the preceding line. In this context, a word is any sequence of non-blank braille symbols.) The nn is simply a sequence counter, incremented by one for each discrepancy.

*Throughout this report, an underlined sequence of two or more letters in an example denotes a group contractable in braille. Adjacent contractable letter groups are separated by a slash. This is consistent with common practice (Refs. 1 and 6).

There is only one known situation where, assuming correct input, the program will fail to flag a word that is incorrectly translated. That is when a letter sign should be added by the translator and is not; for example, the "ab" of

ab initio (italicized),

punched

AB|IN|ITIO,

should be preceded by a letter sign in braille. These cases are indicated by an asterisk in the Transcribers' Guide, so that manual checking of these is fairly easy.

More commonly, words are flagged as wrong that are in fact correct. There are two situations where this can occur.

(1) Synchronization: When, for whatever reason, the correspondence between inkprint (punched) words (sequences of non-blanks) and braille words is not one-for-one, a possibly spurious error due to synchronization will be flagged. For example, the first space in the phrase "by the by" is removed in braille; the word by will not compare with the "word" bythe, causing a synchronization error. Control inputs (\$TAB, etc.) also cause spurious errors of this class, but controls are not normally used in problem word lists. Typically, the word where the synchronization failed and the one after it are flagged before proper synchronization is reestablished.

(2) "Perceiving": For the sake of saving space, only the first four braille sign codes are read into the alphabetical contraction table, even though five are punched on the table input card. This means that "perceiving," the only 5-sign contraction, is not properly represented in the list and so words containing this contraction will be improperly flagged.

All of the spurious error flagging (or failing to flag), except possibly for the synchronization errors, could be corrected with somewhat more logic and/or storage cost in the program. At least in the case of the 5,808 problem words, these problems have not been sufficiently bothersome to warrant the expenditure.

Octal Braille

Octal braille permits the user to generate a sequence of arbitrary braille cells directly. A particular braille cell is selected by a two-digit code. This code is the sum of the numbers associated with the dots in the cell, according to the following association scheme:

Braille Cell Dots

10 . . 1
20 . . 2
40 . . 4

The codes for the desired braille cells are placed in sequence, following the special sign \$OCT. The first blank after \$OCT terminates the sequence.

The code is actually an octal representation of the braille cell. Example:

\$OCT521163107000307270

will translate into

•	••	••	•	•	•	••	•
•		••	•	•	•	••	•
•		•	•	•	•	•	•
O	C	T	A	L	B	R	L

Computer Braille

Computer braille is similar to octal braille except that it is driven by characters rather than two-digit numbers and is initiated by the special sign \$CPB. When using computer braille, a character is represented by the two-digit braille sign code punched in columns 21-22 of the corresponding alphabet table input card. This two-digit number is determined by the sum of the numbers associated with each braille dot in the table:

1 . . 8
2 . . 16
4 . . 32

Example:

Entry in alphabet table:

Column 1	18	21	24	30
↓	↓	↓	↓	↓
%	01	41	15	00

Text Input:

\$CPB%%%

will cause output

.. .. .
. . . .

Notes to the Braille Editor

Role of the Editor

This section indicates where the intervention of the editor is required and presents the tools available to the editor to implement his intervention. Generally speaking, the nineteen problem situations listed in the Memorandum on Braille translation (Reference 5) by R. L. Haynes summarize where the intervention of the braille editor is needed.

There are basically two ways in which the editor can ensure proper translation in problem situations: (1) instructing the keypunch operator to add certain symbols to the input text; and (2) modifying the tables which control DOTSYS III. Modification of tables is explained under "Tables" above; the only time it would be done under normal circumstances would be when a problem word occurs often in a particular body of text; in that case, its correct translation would be added to the contraction table.

In the remainder of this section we shall discuss the special symbols which can be inserted in the input text and the situations in which they are helpful or necessary. These symbols are the letter sign (+), the division symbol (\$/), the forced-contraction symbols (/_, _/), and the termination symbol (\$T). The editor's attention is also directed to the sections on the grade switch, octal braille, computer braille and the self-checking feature.

Letter Sign (+)

The editor must insert the letter sign when:

(1) a letter which means a letter stands alone and is not followed by a period indicating an abbreviation and is not italicized or surrounded by double quotes or parentheses.

(2) the capitalized or uncapitalized letter "a", "i", or "o" requires a letter sign in the braille, except when used after a number or as a word.

(3) combinations of letters that could be confused with short-form words appear in the input text. (It is suggested that a contraction table entry be used to insert the letter sign if such a letter combination occurs frequently. The tables already contain a number of frequently used abbreviations.)

In other situations the program inserts the letter sign automatically where necessary (where one has not already been inserted in the input).

Division (or Null) Symbol (\$/)

The division symbol is a useful and versatile tool of the braille editor, although its operation is simple: it merely prevents contraction of a letter group which crosses it. For example, for the lisped word "thentury", the "the" contraction is avoided by inserting the division symbol after the "th", thus: TH\$/ENTURY. It may be placed between the parts of compound words, such as NUT\$/HATCH. It may be placed between prefixes and stems, as in BI\$/NOMIAL, or indicate syllable division, as in PERITO\$/NEUM and SKI\$/DADDLE.

The division symbol may also be used in cases where the symbols to be translated are coincidentally DOTSYS III control symbols. For example, \$\$/TAB22 will be translated as "\$TAB22" literally, avoiding the control function "tab to column 22." Even "\$/" may be generated for output by supplying "\$\$/" as input.

Forced-Contraction Symbols (/_, _/)

These symbols are used to force a contractible letter group to be contracted. See "Text Input" for a complete description.

Termination Symbol (\$T)

This translates into the double sign dot-6, dot-3, in the output where required (Reference 1, Rule II.11a).

OUTPUT

Echo Output

When selected by the echo card (q.v.), a literal listing of the table input and run control cards is produced following the image of the echo card itself, which is always printed. This printout is placed on the SYSPRINT logical device, preceding the proof output (q.v.), if any. Error messages (q.v.) may be interspersed with the echo output.

Proof Output

Though proof output is optional (see "Run Control Cards"), it is normally called for except, perhaps, in final production runs. It is essentially a printout for the use of a sighted braille editor; the braille signs are represented as printed dots. This printout occurs on the same logical device (SYSPRINT) as the echo output and error messages (q.v.); error messages may be interspersed with the normal proof output.

The first three pages of proof output display the contents of the right context table, the decision table, and the transition table in a form much easier to read than the literal echo listing.

The remainder of the output is primarily a page-by-page, line-by-line representation of the braille output, using periods for braille dots. Below the braille signs are up to three characters intended to help identify the sign. If the sign represents a letter, for example, the letter itself is printed below the sign. The identification characters depend only on the sign (as determined by the sign table, q.v.) and not its context. For example, the sign for "K" has the identification character "K" even when it represents the word "knowledge." Below the identification characters is printed the braille sign code in the range 0 through 63, which may be used to check the punched output. Figure 2 contains a sample of proof output.



Figure 3. Elastomer Braille Output for Line in Figure 2

RPQ Braille Output

RPQ braille is similar to elastomer braille (q.v.) in most respects, with an additional modification to the print chain (IBM RPQ F19229) so that spacing of the dots is closer to the braille standard. This output, if selected, is placed on logical device SYSRPQ.

Punched Card Output

Punched card output makes possible the processing of DOTSYS III's braille output by other programs--for example, to produce tactile braille on a computer system which can drive an embosser but cannot support a suitability modified DOTSYS III.

Each card represents one braille line; page boundaries are not represented. Forty two-digit braille sign codes are punched on each card, with 00's filling out the line on the right. Figure 4 illustrates part of a card containing punched output.

Punched output is placed on logical device SYSPUNCH. Of course, on systems supporting a logical file concept, this device need not be physically a card punch but could be, for instance, a tape containing equivalent card images.

MIT Braillemboss Output

This is a sequence of characters that is usually put out through a teletype directly attached to a Braillemboss. Alternatively, a paper tape may be punched as an intermediary. The lines of output in this mode contain padding (idles) and other controls suitable for this device.

Error Messages

Error messages are unconditionally printed on SYSPRINT, thereby appearing intermixed with echo and proof output, if any. The

(3) **FOLLOWING CARD(S) OUT OF SEQUENCE

- A. An out-of-order card was found in the section of the input containing the echo card, the tables and the run control cards.
- B. Processing of this section is continued but processing of the text is suppressed.
- C. Reestablish correct order or correct the sequence number.

SECTION IV
PROGRAM TRANSFER

This section contains a checklist of steps required to bring DOTSYS III into operation on any computer having a Standard COBOL compiler and sufficient core storage to execute DOTSYS III. 'Standard COBOL' means COBOL according to Reference 4 having the level 1 nucleus, table handling, and sequential access. The core storage required depends on the compiler and on the size of the contraction table. With the IBM 360 level F compiler and 1,200 contraction table entries, about 72,000 bytes are required.

The program can be made about fifteen per cent faster, without affecting its translation capability, by replacing the alphabet search by a machine-dependent indexing scheme, at the expense of reducing further transferability and increasing the program size. The method of doing this is explained under "Maintenance."

Here is the checklist of steps to transfer DOTSYS III:

(1) Rewrite the environment division of the program in accordance with the COBOL manual for your computer and peripheral units. Keep in mind that SYSINPUT is the input file, and SYSPRINT, SYSPUNCH, SYSBRL, SYSRPQ and SYSMEMB are output files. Other information about them is in the file description (FD) entries in the data division.

(2) Check the file description entries (in the file section of the data division) to ensure that the block size and label records are specified correctly for your installation.

Punched output records may or may not begin with a control character (for pocket selecting), and the control character, if present, varies with the installation. DOTSYS III is set up to place a control character in the beginning of each punched output record; the control character used is the value of the data item POCKET-SELECT in the working storage section. If no control character is to be used, remove the 02-level data item FILLER in CODED-OUTPUT, and change the sentence

WRITE CODED-OUTPUT AFTER POCKET-SELECT

in the BREAK41 paragraph of the OUT-OUT section to

WRITE CODED-OUTPUT

(3) Check the COBOL character set at your installation against the characters in the program. Some characters, such as the single quote ('), greater than (>), less than (<), and equals (=), may have to be replaced by other characters or expressions.

(4) Check the character set of your computer and keypunch against the characters in the tables. Remove or replace table entries containing illegal characters. Inkprint characters having no single-character machine-readable form may be represented by multiple-character symbols, just as in the case of single quotation marks. They are implemented via contraction table entries. Be sure to inform the keypunch operator of your choices in this matter.

(5) If your COBOL does not permit the COMPUTE verb, replace each occurrence of it by an equivalent sequence of individual arithmetic operations. For example, the sentence "COMPUTE I = N * M + J." may be replaced by the two sentences

MULTIPLY N BY M GIVING I.

ADD J TO I.

(6) Update the comments in the tables or program, as appropriate, to label and describe the new version. (See PROGRAM VERSION CONTROL in the next section and "Table Version Control" in the USAGE section.)

(7) Compile the program to obtain an object deck. If the object program is too large, try leaving out the table display routine in INITIALIZATION, from TABLE-DISPLAY through SKIP-DISPLAY. If the self-checking feature is not to be used, another possibility would be to shorten the alphabetic contraction table, or even to remove it and all associated code. As a last resort, decrease the contraction table bound (and shorten the table input accordingly).

(8) Execute the program, using standard test case input if possible.

SECTION V
PROGRAM MAINTENANCE

WHEN MAINTENANCE MAY BE REQUIRED

This section presents certain details of the COBOL implementation of DOTSYS III. It is for use in making changes in the tabular input, and in the program itself, that may be made necessary or desirable by the transfer of DOTSYS III to another installation, a change in peripheral units, a change in the braille translation rules, or a desire to improve the conformity of the translation with the braille ideal.

PROGRAM VERSION CONTROL

When any change is made to a version of the program that is shared by other institutions, a new version has been created and it should be labeled according to the authoring institution and data, and information on the change should be incorporated in a comment at the head of the program, just as is done for the tables. See "Table Version Control" under "INPUT," above.

TABLE-SIZE BOUNDS

The maximum capacities for a number of tables are conceptually variable, in that only a few OCCURS clauses or other references need be changed and the program recompiled, to effect an increase or decrease. These items are listed in Table IX.

Table IX

Variable Table Capacity References

<u>Table</u>	<u>Item or (Paragraph) Name</u>
Alphabetic Contraction	LIST-ENTRY, (LOOP2A), (LKUP-CC)
Alphabet	ALPHABETIC-ENTRY
Contraction	TABLE-ENTRY
Right Context	RIGHT-CONTEXT-TABLE-ENTRY

Table IX (Concl.)

Variable Table Capacity References

<u>Table</u>	<u>Item or (Paragraph) Name</u>
State Variables	TRANSITION-TABLE-COLUMN, CONDITION
Input Classes	DECISION, TRANSITION
Decision Table (width)	DECISION-TABLE-COLUMN
Stacks (no. of entries)	STACK, (LOOP1-twice)
Stack (entry width)	ELEMENT, (ORD-CHAR-OUT), (CARRIAGE-CONTROL), (MOVE-ERROR- PARAGRAPH)
Self-Checking Ring (entry width)	ELEMENT-TRANS, (NOT-IN-CONTRACT), (C-L-NOT-FOUND), (BRL-E-MOVE)

Note that increasing the size of a table in such a way that the number of digits required to index it is increased (e.g., a change from 99 to 101) may require that the PICTURE clause for data items used as indices (subscripts) to that table may have to be changed (e.g., from S99 to S999). Note also that in some cases a dummy (terminator) entry is actually stored in the table.

REPLACING THE ALPHABET TABLE SEARCH BY DIRECT INDEXING

The first step in translating the current contents of the buffer is to find the alphabet table entry for the leftmost character in the buffer. In order to preserve machine-independence, DOTSYS III does this by searching the alphabet table linearly from the top for a match with the leftmost character in the buffer. If the alphabet table has been ordered with the higher-frequency items nearer the top, an average of about nine entries is tested. On the IBM 360/50, that takes about one millisecond, or roughly fifteen per cent of the average time spent per character.

On some machines, including the IBM 360, it is possible to reduce drastically the time required to find the appropriate alphabet table entry, by using the test character itself as an index. This is done by moving the character to a data item defined as USAGE DISPLAY (the default case) but redefined as USAGE COMPUTATIONAL. For example,

on the IBM 360, the data item below describes a half-word integer whose value is determined by the character moved into its right-hand byte; its left-hand byte is binary zeroes.

```
01 CHARACTER-INDEX.  
02 LEFT          PICTURE X, VALUE LOW-VALUE  
02 RIGHT        PICTURE X.  
01 N REDEFINES CHARACTER-INDEX,  
                PICTURE S999, USAGE COMPUTATIONAL.
```

Thus, if a character is moved to RIGHT, then N is a number determined by that character, and can be used as an index into an array of pointers to the appropriate places in the alphabet table. For example, suppose the array of pointers is called ARRAY-OF-POINTERS and the left most character in the buffer is a space, whose alphabet table entry comes first. Moving the space to RIGHT gives N a value of hexadecimal 40 or decimal 64; the 64th entry in ARRAY-OF-POINTERS should be 1 since the space entry is first in the alphabet table. Thus, the two sentences,

```
MOVE RL1 TO RIGHT.
```

```
MOVE ARRAY-OF-POINTERS (N) TO LETTER.
```

replace the alphabet table search, provided that ARRAY-OF-POINTERS has been properly initialized, which takes only two sentences in the READ-ALPHABET loop during initialization. The same thing can be done to replace the search occurring in the paragraphs following TEST-NON-TERMINAL (which check right context) in the translation section.

Note that with indexing, there is no longer any need to order the alphabet table according to frequency; however, the order of contraction table sections must still match the order of the associated alphabet table entries.

The price paid for the reduced search time is an increased storage requirement, since ARRAY-OF-POINTERS must have 2^k entries, where k is the number of bits in a character. On the IBM 360, ARRAY-OF-POINTERS would take up $2^8 \times 2 = 512$ bytes.

CONTRACTION TABLE SEARCH ALGORITHM

General Rationale

A form of tree search has been implemented for comparison of a string against the contraction table. This algorithm is inherently much more efficient than a linear search for any size table. Moreover, the proportionate cost (in time) for new entries is reduced: the time increases only logarithmically, rather than directly.

The method takes advantage of the fact that often a comparison failure on, say, the third character implies that quite a few additional entries (with the identical first three characters) can be skipped around. The method also conserves space in that only one numeric field must be added to each table entry to support the algorithm.

The basic idea is quite simple. In a given initial-letter section of the table, all the entries which start a new subsection (wherein the first two letters are identical) are chained together, using the added "branch" field as a forward link. This forms the "level 1" chain; a level 2 chain may be formed and so forth. The implicit structure is that of a binary tree. During search, one either follows the branch (continues on the current chain) if comparison failure occurs at the character whose index equals the current level, or else goes to the next entry and one level higher.

The main complication with this process is that, having reached the highest point in the tree and still not having found a match, it still may be necessary to return to the lower level. For example, "AB" may be at level 2 and would probably follow "ABCDE"--at, say, level 4--in the table. The key "ABCDF" would thus reach at least level 4 and yet may not actually match until the "AB" is encountered. The handling of this situation in the algorithm is facilitated by indicating the level of the next entry whenever the current level chain ends. This level is entered in the branch field, in negative form so as not to be confused with a forward pointer and also to indicate that the forward pointer is null.

It should be noted that the table must be properly ordered on input in order for the algorithm to work. The proper ordering condition is that entries whose first p letters correspond must be together in the table; formally, using the notation defined below: if there exist two entries, with indices q and r ($q < r$) and an integer $p > 0$ such that $C_{qj} = C_{rj}$ for all j in the range $1 \leq j \leq p$, then for all t in the range $q \leq t \leq r$ it is also true that $C_{qj} = C_{tj}$ for all j in the range $1 \leq j \leq p$.

Notation

The notation used in the flow charts (Figures 5 and 6) is as follows:

LET

- C_{ij} be the j^{th} character in the i^{th} entry of the contraction table.
- b_i be the value of the branch field for the i^{th} entry.
- s_k be the index of the first entry in the contraction table for the k^{th} initial letter.
- e_k be the index of the last entry for the k^{th} initial letter.
- L be the index for the lowest entry to be considered at the current level.
- H be the index for the highest entry to be considered at the current level.
- V be the highest index for the current initial letter.
- n be the current level.
- A_n be the upper index bound for the n^{th} level.
- m be the number of entries in the contraction table, not counting the extra "end of table" entry.
- M be the number of initial letters.
- w_j be the j^{th} character in the current input string (string being looked up).

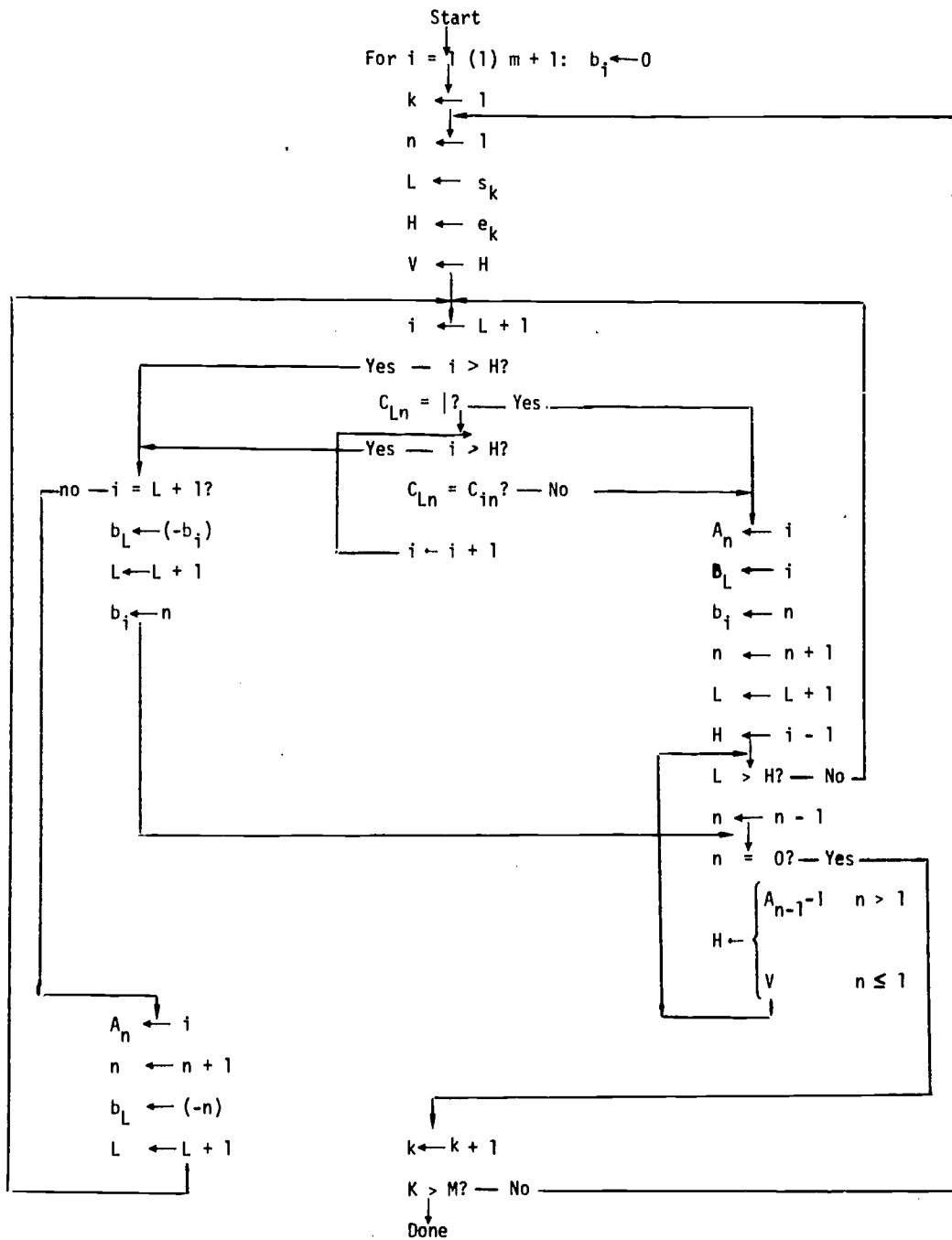


Figure 5. Set-Up Algorithm

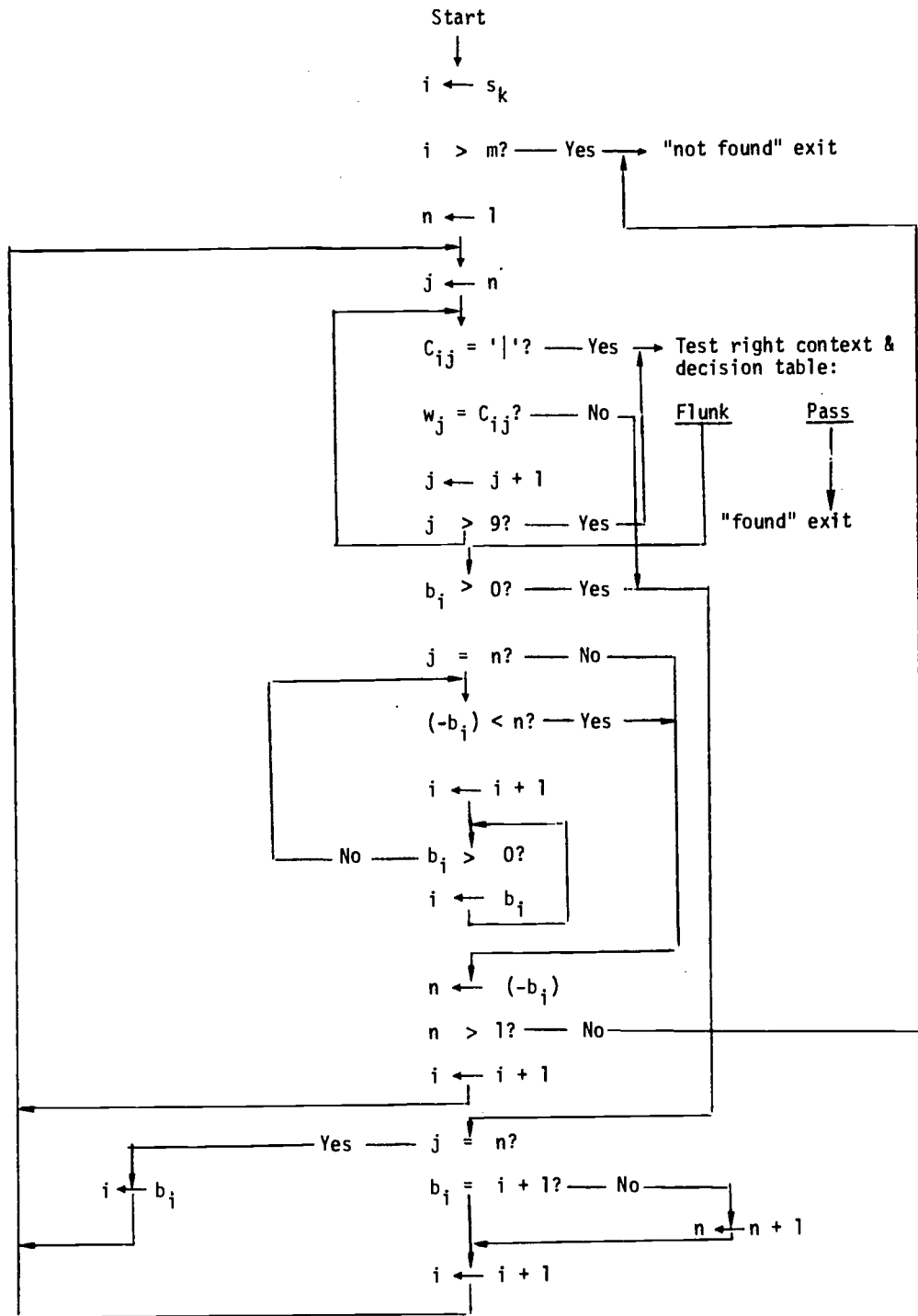


Figure 6. Look-Up Algorithm

APPENDIX I

DOTSYS III (TABLES 10/70)	TRANSCRIBERS' GUIDE	DOTSYS III (TABLES 10/70)	TRANSCRIBERS' GUIDE
<u>Holin/sh/ed</u>	<u>Holinshed</u>	pandemonism	pandemonism
hornblende	hornblende	pedometer	pedometer
hyaena	hyaena	Persephone	Persephone
<u>in/en/arrable</u>	<u>inenarrable</u>	<u>persever/ance</u>	<u>persever/ance</u>
<u>inglenook</u>	<u>inglenook</u>	<u>pigh/eaded</u>	<u>pigheaded</u>
<u>insof/ar</u>	<u>insofar</u>	pokeroot	pokeroot
Ione	Ione	potherb	potherb
Iredell	Iredell	praenomen	praenomen
isinglass	isinglass	pronephros	pronephros
jibboom	jibboom	pros and cons	pros and cons
krone	krone	protonema	protonema
<u>Letter/er</u>	<u>Letter/er</u>	<u>rar/eripe</u>	<u>rareripe</u>
Letterman	Letterman	rawhide	rawhide
Lever (bros.)	Lever	Reno	Reno
<u>lin/eage</u> (alignment)	<u>lineage</u>	reredos	<u>rer/edos</u>
<u>ling/erie</u>	<u>lingerie</u>	retroflex	retroflex
Littleton	Littleton	riboflavin	riboflavin
locoweed	locoweed	<u>roped/ancer</u>	ropedancer
<u>Loffler</u>	<u>Loffler</u>	saintonge	saintonge
maenad	maenad	sawhorse	sawhorse
Maugham	Maugham	Sheean	Sheean
<u>men/ingeal</u>	<u>men/ingeal</u>	shorthand	shorthand
mesitylene	mesitylene	shorthorn	shorthorn
Micronesian	Micronesian	<u>Shosh/one</u>	<u>Shoshone</u>
midwifery	midwifery	skedaddle	skedaddle
<u>minestrone</u>	<u>minestrone</u>	snakeroot	snakeroot
mistitled	<u>mistitled</u>	so (musical note)*	so
More's (name)	<u>More's</u>	Somes	Somes
muraena	muraena	Soong	Soong
<u>nea/therd</u>	<u>neatherd</u>	<u>sou'ea/st/er</u>	<u>sou'east/er</u>
newsletter	newsletter	spathose	spathose
<u>nuthatch</u>	<u>nuthatch</u>	speakeasy	speakeasy
Oenone	Oenone	spikenard	spikenard
<u>oer/st/ed</u>	<u>oerst/ed</u>	spumone	spumone
optime	optime	<u>stagh/ound</u>	staghound
Osgood	Osgood	<u>st/ereoisomer</u>	<u>st/ereoisomer</u>
<u>oughtlins</u>	<u>ou/ghtlins</u>	stirabout	stirabout
<u>ou/thaul</u>	<u>outhaul</u>	<u>stringendo</u>	<u>string/endo</u>
padrone	padrone	suede	suede
<u>paleaceous</u>	<u>paleaceous</u>	super/erogatory	supererogatory

*misspelled

APPENDIX I (Concl.)

DOTSYS III
(TABLES 10/70)

TRANSCRIBERS'
GUIDE

<u>suprar/enal</u>	suprarenal
<u>swastikaed</u>	swastikaed
<u>taenia</u>	taenia
<u>learoom</u>	tearoom
<u>tearose</u>	tearose
<u>teleran</u>	teleran
<u>Theresa</u>	Theresa
<u>thiourea</u>	thiourea
<u>toadeater</u>	toadeater
<u>tonelada</u>	tonelada
<u>transmental</u>	transmental
<u>treenail</u>	treenail
<u>trenal</u>	trenal
<u>treponema</u>	treponema
<u>tuberose</u>	tuberose
<u>tufthunter</u>	tufthunter
<u>tweedledee</u>	tweedledee
<u>tweedledum</u>	tweedledum
<u>'twouldn't</u>	'twouldn't
<u>undenomin/ational</u>	undenomin/ational
<u>underogating</u>	underogating
<u>us'n</u>	us'n
<u>vainglorious</u>	vainglorious
<u>Vandyke</u>	Vandyke
<u>violone</u>	violone
<u>wakerife</u>	wakerife
<u>Wenceslaus</u>	Wenceslaus
<u>whaddaya</u>	whaddaya
<u>Wingate</u>	Wingate
<u>wired/ancer</u>	wiredancer
<u>wiredrawn</u>	wiredrawn
<u>wiseacre</u>	wiseacre

APPENDIX II
DEFINITION OF STATE VARIABLES AND INPUT CLASSES

State Variable	1 after the start of a number
	2 after the start of a word
	3 grade 1 translation
	4 in a quotation
	5 in italicized text
	6 not at the start of a prefix or stem
	7 part way through a word or phrase too long for one entry
	8 just after a space (or A-J), following a number
	9 Foreign language passages are Spanish
Input Class	1 contractions always used in grade 2
	2 digits
	3 most punctuation and control (\$) symbols
	4 contractions used after the start of a word
	5 \$G (grade switch)
	6 contractions used at the start of a word
	7 isolated full-word contractions
	8 \$P" (start paragraph in quotation)
	9 \$P (start paragraph in italics)
	10 " (left quote)
	11 " (right quote)
	12 __ (begin italics)

- Input Class**
- 13 _ (last word of italics)
 - 14 (space), certain control (\$) symbols
 - 15 A to J or space occurring in a number
 - 16 contractions always used in grade 2 containing terminal punctuation
 - 17 prefix or first word of compound word
 - 18 non-prefix beginning of word
 - 19 first entry of double entry
 - 20 second entry of double entry
 - 21 "forced" contraction begin sign (/_)
 - 22 units of measure and numbers following numbers
 - 23 Foreign language special symbols
 - 24 Spanish special symbols
 - 25 \$FL-SPAN
 - 26 \$FL-LIFG
 - 27 Decimal point (.) within a number

APPENDIX III
SUMMARY OF SPECIAL SYMBOLS

<u>Symbol</u>	<u>Input Representation</u>	<u>Text Reference Page(s)</u>
capitalize letter	=	12, 19, 20
capitalize word	==	12, 19, 20
italicize word	_	12, 20
italics start	--	12, 20
left single quote	\$'	21
right single quote	\$'R	21
left double quote	\$"	21
right double quote	\$"R	21
left conversation (Spanish-old form)	\$-	24
right conversation (Spanish-old form)	\$-R	24
dash or conversation mark	--	24
accent marks	%	12, 20, 22
left bracket ([)	<	22
right bracket (])	>	22
start paragraph	\$P	24
begin new line	\$L	24
begin new page	\$PG	24
skip to column nn	\$TABnn	24
letter sign	+	12, 30

<u>Symbol</u>	<u>Input Representation</u>	<u>Text Reference Page(s)</u>
change grade	\$G, \$GON or \$GOF	22, 23
divide word	\$/	20, 30
termination sign	\$T	31
long vowel sign	\$LV	22, 23
start poetry	\$PTYS	25
end poetry	\$PTYE	25
start paragraph within quotation	\$P"	24
start title	\$TLS	25
end title	\$TLE	25
forced blank	\$B	22
start heading	\$HDS	25
end heading	\$HDE	25
skip lines	\$SLnn	24
short vowel sign	\$SV	22, 23
set tab t to col. nn	\$STBt[L,R or D]nn	25
turn self-checking on	\$SCONS\$/\$/\$/\$/	25, 26
turn self-checking off	\$SCOFF	25, 26
end of poetry foot sign	\$FT	22
computer braille	\$CPB	28
caesura sign	\$CS	22
special literal symbol	\$SYM	21
symbol \$:\$	21
symbol %	:%	21

<u>Symbol</u>	<u>Input Representation</u>	<u>Text Reference Page(s)</u>
symbol /	\$/	21
symbol &	\$:&	21
symbol #	\$:#	21
call (permanent) tab t	\$/t	25
octal braille	\$OCT	28
Latin, Italian, French or German language mode	\$FL-LIFG	23
Spanish language mode	\$FL-SPAN	23
special foreign language symbols	&x or &xx	23, 24
start forced contraction	/_	20, 30
end forced contraction	_/	20, 30
contraction indicator for self-checking (also general delimiter)		12, 14, 20, 26, 27

APPENDIX IV

EQUIVALENT SIGNS, SYMBOLS AND CODES

	0	1	2	3	4	5	6	7
		·	·	∴	·	∴	∴	∴
0		%	5	45	=	—	+	456
	00	08	16	24	32	40	48	56
		·	·	∴	·	∴	∴	∴
1	A	C	E	D	CH	SH	WH	TH
	01	09	17	25	33	41	49	57
		·	·	∴	·	∴	∴	∴
2	,	I	:	J	EN	OW	.	W
	02	10	18	26	34	42	50	58
		·	·	∴	·	∴	∴	∴
3	B	F	H	G	GH	ED	OU	ER
	03	11	19	27	35	43	51	59
		·	·	∴	·	∴	∴	∴
4	'	ST	IN	AR	-	ING	"	#
	04	12	20	28	36	44	52	60
		·	·	∴	·	∴	∴	∴
5	K	M	O	N	U	X	Z	Y
	05	13	21	29	37	45	53	61
		·	·	∴	·	∴	∴	∴
6	;	S	TO	T	?	THE)(WTH
	06	14	22	30	38	46	54	62
		·	·	∴	·	∴	∴	∴
7	L	P	R	Q	V	AND	OF	FOR
	07	15	23	31	39	47	55	63

first octal digit	second octal digit
	Braille sign
	Proof character(s)
	Sign code

FORMAT

APPENDIX V
TRANSLATOR - STACKER SIGN CODES

<u>Code</u>	<u>Meaning</u>
00	required blank
01-63	braille sign codes
64	end of braille word (blank or end of line)
65	new line
66	unused
67	paragraph start
68	one-time tab (skip to col. nn)
69	new page
70	unused
71	skip (multiple) lines
72	tab (skip according to permanent tab)
73-80	unused
81	start heading input
82	end heading input
83	unused
84	set continuous text stacking mode
85	idle (reserved for: reset continuous text stacking mode)
86	unit of measure
87	unused

<u>Code</u>	<u>Meaning</u>
88	start running title input
89	end of running title input
90	unused
91	self-checking mode off
92	self-checking mode on
93	set tab
94	start octal braille
95	start poetry mode
96	end poetry mode
97	start computer-braille
98	end of run
99	(filler in contraction table)

APPENDIX VI
MISCELLANEOUS CODED VARIABLES

<u>Variable</u>	<u>Code</u>	<u>Meaning</u>
STACK-INDICATOR	2	Normal text; clear stack after each entry
	4	Continuous stacking of title or heading
	5	Continuous stacking of normal text
CURRENT-TYPE	1	Normal text or heading stack
	2	Title stack

Note: logical indicator variables are generally coded according to the convention 0 = "off" = "no" = "false", 1 = "on" = "yes" = "true".

APPENDIX VII

USERS OF DOTSYS III AND ITS DERIVATIVES

Since its original release in August of 1970, DOTSYS III has been widely distributed and in the process has been considerably improved upon to suit various needs. It has been recast in at least two other languages, PL/I and FORTRAN, in each case with considerable performance improvement. It has been cut in two, so that the time-consuming table interpretation and assembly in internal form need not be repeated each run. The program and tables have been altered and augmented to improve the English translation, to process Nemeth Code and some foreign languages, and to allow additional kinds of formatting. All of these changes are consistent with the original purpose of DOTSYS, which was to be a widely portable working prototype, a basic starter system and standard that would be adaptable to local needs.

The following is a list of all institutions that are known to have received DOTSYS or its derivatives. Most use it in its native IBM 360-370 environment. In some cases the program is not presently in active use. Where additional pertinent information is available, it is listed under "Notes."

<u>Institution and Cognizant Individual(s)*</u>	<u>Notes**</u>
Argonne National Laboratory 9700 S. Cass Ave. Argonne, Illinois 60439 - Arnold Grunwald	
Arkansas Enterprises for the Blind, Inc. 2811 Fair Park Blvd. Little Rock, Arkansas 72204 - Elmo Knoch	Transferred to CDC 3300; also using Hatfield Polytechnic's PDP-10 version.

*This list is derived for the most part from a memorandum by G. Dalrymple of MIT.

**Based in part on a user survey taken in August 1973, and in part on other informally received information.

Institution and
Cognizant Individual(s)

Notes

Atlanta Public Schools
210 Pryor Street, S.W.
Atlanta, Georgia 30303
- Dr. Marion P. Boyles

Rewrote in PL/I (program
called BRAILLEMMASTER). Added
preprocessor for IBM's ATMS
text editing system, and text-
book formatting features.

Bell Telephone Co. of Philadelphia
One Parkway, 11th Floor
Philadelphia, Pennsylvania 19103
- Frank Benner

Bell Telephone Labs
Whippany Rd.
Whippany, New Jersey 07981
- G. L. Calessio

Board of Education
Nassau County, N.Y., via
Bradford Computer & Systems Inc.
220 E. 42nd Street
New York, New York 10017
- Richard Snipas (now of
Triformation Systems)

Canadian National Institute for
the Blind
1929 Bayview Avenue
Toronto, Ontario
Canada M4G 3E8
- David Brown

Dartmouth College
Kiewit Computation Center
Hanover, New Hampshire
- R. F. Hargraves

Transferred to GE-635

Duxbury Systems
49 Soule Avenue
Duxbury, Massachusetts
- Dr. Steven M. Simpson, Jr.

Planning to reimplement on
a minicomputer

Institution and
Cognizant Individual(s)

Notes

The Hatfield Polytechnic
Computer Science
P.O. Box 109
Hatfield, Hartfordshire
AL10 9AB
England
- J. M. Jenken

Transferred program to a
PDP-10

Honeywell, Inc.
200 South Street
Waltham, Massachusetts

Transferred to H-3200

Information Engineering
3401 Market Street
Philadelphia, Pennsylvania 19104
- Philip R. Bagley, President

Library of Congress
Division for the Blind and
Physically Handicapped
1921 Taylor Street, N.W.
Washington, D.C. 20542
- Morton Friedman

For production, using
Atlanta's BRAILLEMATER

Ohio State University
Mechanized Information Center
1827 Neil Avenue
Columbus, Ohio 43210
- Ronald J. Beaton

Southern Illinois University
Computer Sciences Dept.
Carbondale, Illinois 62901
- Peter Baum

State Services for the Blind (Minn.)
Communication Center
1745 University Avenue
St. Paul, Minnesota 55104
- Robert D. Watson

Institution and
Cognizant Individual(s)

Notes

University of Connecticut
Social Science Data Center
U-164
Storrs, Connecticut
- William D. Slysz

University of Manitoba
Computer Center
624 Engineering Building
Winnipeg, Manitoba
Canada
- Donald Keeping
- Bonnie MacDonald

University of Warwick
Coventry, Warwickshire CV47AL
England
- John M. Gill

Worcester Polytechnic Institute
Worcester, Massachusetts
- William Swiger

Xerox Corporation
Xerox Square, W-128
Rochester, New York 14644
- David B. Coblitz

Improvement to program and
tables; handle Nemeth Code
and grade 1 French

Rewrote in FORTRAN

Transferred to PDP-10

APPENDIX VIII

SAMPLE PROOF OUTPUT

The following page contains a sample of output, corresponding to a page of braille, in "proof" form. Note that original input lines, characterized by sequence numbers 1312 - 1317 at the right end of each line, are interspersed with the braille output. Note also that each cell on a line of braille is expressed in three forms: a dot configuration, a "mnemonic" for the configuration that may or may not express the meaning in a particular context (e.g., "THE" after "45" means "THESE"), and a numeric code. The output generally lags the associated input by up to several lines.

REFERENCES

1. English Braille, American Edition, 1959 (Revised 1972), American Printing House for the Blind, Louisville, Kentucky.
2. J. K. Millen, DOTSYS II: Finite-State Syntax-Directed Braille Translation, MTR-1829, The MITRE Corporation, (1970).
3. J. K. Millen, DOTSYS II: User's Guide and Transfer and Maintenance Manual, MTR-1853, The MITRE Corporation, (1970).
4. U.S.A. Standard COBOL, American National Standards Institute X3.23-1968.
5. R. L. Haynes, "Computer Translation of Grade II Braille", Proceedings Conference on New Processes for Braille Manufacture, 1968, American Printing House for the Blind, Louisville, Kentucky, pp. 1-4.
6. B. M. Krebs, Transcribers' Guide to English Braille, The Jewish Guild for the Blind, New York, 1967.
7. IBM System/360 Operating System, Full American National Standard COBOL. IBM document No. GC28-6396-2 (June 1970).

DISTRIBUTION LIST

INTERNAL

D-71

F. Chess
J. A. Clapp
R. A. J. Gildea (10)
A. D. McKersie
J. E. Sullivan (20)

E. L. Glaser
Computation Center
Case Western Reserve University
University Circle
Cleveland, Ohio 44106

D-73

N. W. Anschuetz
W. R. Gerhart
J. K. Millen

Dr. C. E. Hallenbeck
Department of Psychology
University of Kansas
Lawrence, Kansas 66044

W-50

K. J. Stetten

R. Haynes
American Printing House for
the Blind
1839 Frankfort Avenue
Louisville, Kentucky 40206

PROJECT

G. Dalrymple, MIT/SAEDC (40)
R. Evensen, L.O.C. (60)

Dr. K. R. Ingham
American Systems, Inc.
123 Watertown Avenue
Watertown, Massachusetts 02172

EXTERNAL

P. R. Bagley
President
Information Engineering
3401 Market Street
Philadelphia, Pennsylvania 19104

E. Knoch
Arkansas Enterprises for
the Blind, Inc.
2811 Fair Park Blvd.
Little Rock, Arkansas 72204

Dr. M. P. Boyles
Director, Computer-Braille Project
Instructional Services Center
Atlanta Public Schools
2930 Forrest Hill Drive, S. W.
Atlanta, Georgia 30315

R. E. LaGrone
IBM Corporation
Federal Systems Division
Department PC4, Room 2P25
18100 Frederick Pike
Gaithersburg, Maryland 20760

L. L. Clark
Director, IRIS
American Foundation for the Blind
15 West 16th Street
New York, New York 10011

Dr. L. Leffler
3001 St. Mary's
Midland, Michigan 48640

EXTERNAL (Concluded)

Prof. A. Nemeth
Mathematics Department
University of Detroit
4001 W. McNichols
Detroit, Michigan 48821

A. Schack
Schack Associates
127 West 12th Street
New York, New York 10011

J. Siems
American Printing House for
the Blind
1839 Frankfort Avenue
Louisville, Kentucky 40206

R. Snipas
Triformation Systems, Inc.
P. O. Box 127
Wall Street Station
New York, New York 10005

Dr. E. J. Waterhouse
Director
Perkins School for the Blind
175 North Beacon Street
Watertown, Massachusetts 02172

V. Zickle
1150 Standford Avenue
Louisville, Kentucky 40206

FOREIGN

S. Becker
KTH
Fack 10.044
Stockholm 70
Sweden

P. W. F. Coleman
University of Warwick
Coventry, Warwickshire, CV4 7AL
United Kingdom

C. W. Garland
Royal National Institute for
the Blind
224-6 Great Portland Street
London, W. 1
United Kingdom

J. M. Gill
University of Warwick
Coventry, Warwickshire, CV4 7AL
United Kingdom

J. J. Jenken
The Hatfield Polytechnic
Computer Science
P. O. Box 109
Hatfield, Hartfordshire, AL10 9AB
United Kingdom

Hans Karlgren
Research Group for Quantitative
Linguistics
Sodermalmstorg 8,
11645 Stockholm
Sweden

D. Keeping
University of Manitoba
Computer Center
624 Engineering Building
Winnipeg, Manitoba
Canada

B. Lindqvist
De Blindas Forening
Utvechlingsavdelningen
Gotlandsgatan 46
116 65 Stockholm
Sweden

FOREIGN (Concluded)

J. Lindstrom
Handikappinstitutet
Pa Iach, 161 03 Bromma 3
Sweden

V. Mokleby
Husby OFF Skole for Blinde
Hovseterveien 3
Oslo 7
Norway

W. Slaby
Rechenzentrum der Universität Münster
4400 Münster
Roxeler Strasse 60
GFR

W. Sorke
Deutsche Blindenstudienanstalt
355 Marburg/Lahn
Am Schlag 8
GFR

M.M.M. Truquet
9, Rue Alain Lesage
Residence SEMIRAMIS
31400 Toulouse
France

J. Vinding
Christian Rovsing A/S
Marielundvej 46B
2730 Herlev
Denmark

M. J. Vliegenhart
Philips Gloeilampenfabrieken
Research Lab.
Building WAA
Eindhoven
Netherlands

Prof. Dr. H. Werner
Rechenzentrum der Universität
Münster
4400 Münster
Roxeler Strasse 60
GFR