DESIGN AND EVALUATION OF A SYSTEM FOR THE PRODUCTION OF SHORT DOCUMENTS IN CONTRACTED BRAILLE.

HV1703
.D460
1975


M.C. MIGEL MEMORIAL LIBRARY
American Foundation for the Blind
15 West 16th Street, New York, New York
10011

DESIGN AND EVALUATION OF A SYSTEM FOR THE PRODUCTION OF SHORT DOCUMENTS IN CONTRACTED BRAILLE

Warwick Research Unit for the Blind
University of Warwick
Coventry CV4 7AL
England.

1975

HV1703
D460
Copy 2

CONTENTS

Introduction

This report covers a one-year period during which a computer-based system for the production of short documents in braille has been initiated. A braille translation program has been implemented and basic problems of input, editing and output have been solved such that an overall system has been satisfactorily constructed. A sufficient quantity of braille has been produced to permit the beginning of a thorough evaluation into the acceptability, type and quantity of demand for this form of braille document.

Background

The University of Warwick started studying aids for the visually handicapped in 1971 when a research study in the discriminability of embossed symbols for points, lines and areas was initiated in close liaison with the Blind Mobility Research Unit at Nottingham University. From this work, the design and evaluation of maps and diagrams evolved, using a unique computer-aided design and production facility. As maps were made available to blind users, the necessity for a considerable amount of accompanying braille text became apparent. Also the recognition of the problem of obtaining single copies of short documents in contracted braille encouraged the University to study computer-based methods for producing braille.

The Sensory Aids Evaluation and Development Center provided Warwick with a copy of DOTSYS III and the Royal National Institute for the Blind financed a one-year project from September 1974. The R.N.I.B. lent the University a Braillemboss and provided funds to employ a programmer, typist/secretary and a part-time braillist. This report summarises the progress attained during this one-year period.

Basic system

The basic system for producing short documents in contracted braille is:

(i) A typist, with no computing knowledge, inputs the text on punched cards, paper tape or directly on a visual display unit. Control characters, for new paragraph etc., are also added by the typist as she inputs the material (i.e. the text is not annotated by someone else).

(ii) A line printer listing of the text is produced in order to proof-read for typing errors.

(iii) The text is interactively edited on a visual display unit with a program designed specifically for this purpose. This program has been designed for speed of operation, minimal computing requirement and for ease of use by operators with no experience of computing.

(iv) The text is translated to a good approximation to Grade II standard English Braille, and the translation is stored on magnetic tape.

(v) The braille is output on an on-line embosser.

Only the translation phase requires extensive central processor time; all other computer operations use less than 1% of the central processor time, and can be time shared with other unrelated programs.

The current output of the system is about 20,000 braille cells (circa 30 pages) per day. Allowing for multiple copies, such that on average about two copies of each document are produced, this means that 15,000 braille cells are translated per day, requiring about one to two minutes of central processor time for translation. It is not envisaged that this can be increased with existing facilities (particularly staff) since the typist is currently the only full time worker on this project, and she also undertakes all proof reading and routine secretarial duties associated with the project.

Translation program

The Sensory Aids Evaluation and Development Center provided Warwick University with a copy of DOTSYS III which is a program, written in Cobol, to translate text to a good approximation to Grade II standard American-English Braille. The version currently in use at Warwick uses 13k words of store with initialisation overlaid and translates at 5000 words per minute to a good approximation to Grade II standard English Braille (see Appendix 1).

Evaluation

In order to evaluate the system a small scale pilot service was operated for producing single copies of short documents for blind subjects. In the first few months of operation the system was undergoing almost continual modification based on informal feedback from the blind subjects which made it impractical to start a formal evaluation programme. However a questionnaire was circulated to subjects who had used the most recent system; the results are shown in Appendix 2.

Firm conclusions cannot be drawn from the survey due to the small sample size. However these results show that both the number of mis-contractions and the use of single-sided braille are acceptable for this application. The lowest score was for turn-round time which varied from a few hours to two weeks for short documents.

Conclusions

This project has demonstrated that there is a considerable demand for short documents in braille, and that computer-based systems can potentially satisfy a significant proportion of this demand.

It is important to continue this evaluation and to carry out the projects listed in Appendix 3. It is recommended that a computer-based service, or services, should be established as soon as possible to meet the unsatisfied demand of the braille-reading population who require short documents in braille.

Acknowledgements

The project is being undertaken with financial assistance from the Royal National Institute for the Blind. The Science Research Council has provided the computing facilities for this project. The Department of Health and Social Security are funding the Senior Research Fellow responsible for the day to day running of the research unit. The aid of these organisations is gratefully acknowledged. We would also like to thank the staff of the Department of Psychology at the University of Warwick who have given considerable help and advice.

Relevant Publications and Reports

Bagley P.R. "A revised system for computer-assisted braille production based on DOTSYS". Information Engineering, USA, April 1973, 15pp.

Coleman P.W.F. "The search for a braille translation program". Computer Weekly, 14 August 1975, p.6.

Coleman P.W.F. "Braille programs - part 2: Defining the language". Computer Weekly, 21st August 1975, p.6.

Douce J.L. "Some developments in computer-aided information services for the blind". Proceedings of the Institution of Electrical Engineers, to be published.

Gerhart, W.R., Millen J.K. & Sullivan J.E. "DOTSYS III: a portable program for grade 2 braille translation". MITRE Corporation, USA, 1971, 139 pp.

Gill J.M. "Methods of increasing the accessibility of reading material by the blind". The Louis Braille Conference, Cambridge, England, Jan 1975, pp 155-162.

Gill J.M. "Non-visual computer peripherals". Research Bulletin of the American Foundation for the Blind, No.29, pp 197-212.

Gill J.M. "The international register of research on blindness and visual impairment". Warwick Research Unit for the Blind, University of Warwick, England, 1975.

Goldish L.H. "Braille in the United States: its production, distribution and use". I.R.I.S., American Foundation for the Blind, New York.

Lawes W.F. "The feasibility study into the usage of computers by and for the blind". Royal National Institute for the Blind, London, Jan 1975.

Mann R.W. "Technology and human rehabilitation: Prostheses for sensory rehabilitation and/or sensory substitution". Advances in Biomedical Engineering, Vol.4, Academic Press, 1974, pp 209-353.

Proceedings of the workshop "Towards the communality of algorithms among braille transcription systems for multilingual usage." University of Munster, Germany, March 1973.

Proceedings of the workshop "Computerised braille production". Danish Association of the Blind, Copenhagen, Sept 1974, to be published.

"A restatement of the lay-out definitions and rules of the standard English braille system". Royal National Institute for the Blind, London, 1969 & 1971.

Appendix 1 Computer programs

Section I

Introduction

This appendix described DOTSYS III-F, a table-driven Fortran program for the translation of English text into a good approximation to grade 2 braille. The program is a translation, with some additions, omissions and modifications, of a Cobol program, DOTSYS III (Gerhart, Millen and Sullivan, 1971). This appendix is based on their report, from which large portions have been quoted with little or no change.

DOTSYS III is a computer program embodying a natural-language-to-braille translation algorithm. 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 either of two forms, "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.

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-characters (card-image) records, The first character of a record logically follows character 80 from the previous record. This stream of characters is collapsed by deleting all instances of the vertical bar character (|), and by deleting all consecutive blanks after the first. This collapsed stream of characters, one at a time, is the output of this section.

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 in Section IV 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 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 also discussed in more detail in a separate section.

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 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 braille sign codes in that they have a value greater than 64. (See Appendix 1.4)

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.

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.

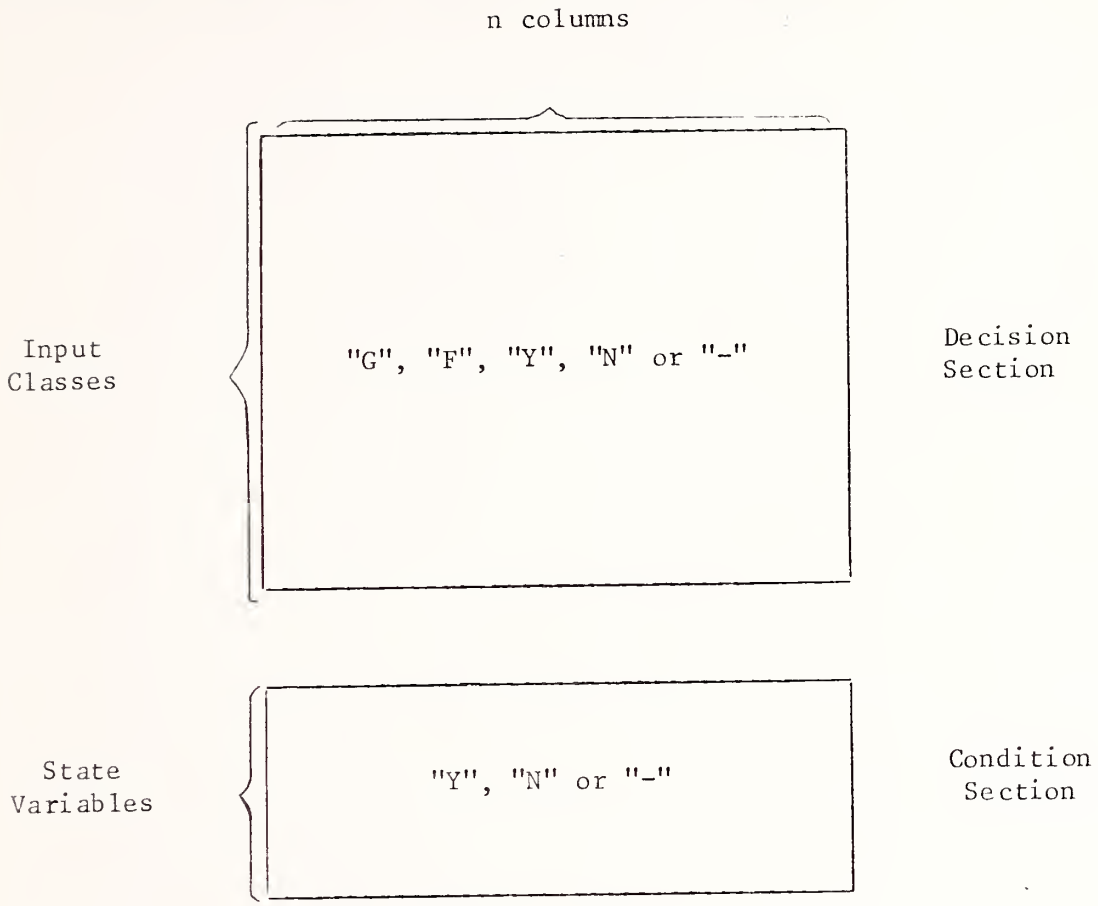


Figure 1. Decision Table Schematic

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 read by DOTSYS III consists of three main sections, in order:

(1) the tables;
(2) the run control cards; and
(3) the text.

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

The formats of the tables and cards listed above follow. 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.

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 I.

Table I

Alphabet Table Card Format

Columns	Contents
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 1.3).
24-25	The sign code to use in normal context (Cf. Appendix 1.3).
30-31	01 if the symbol may be used as one of a pair of symbols bracketing a letter denoting itself; 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 dtermined 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 up to the nth character, then all entries between them must also agree with them up to 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, must not exceed 1,000.

Table II gives details of the contraction table card layout.

Table II

Contraction Table Card Format

Columns	Contents
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.)
13	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 1.3). 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 02.

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 III.

Table III

Right-Context Class Card Format

Columns	Contents
13	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 10.

The Card Specifying the Number of Input Classes

The number of input classes is punched in columns 18-19. This number may not exceed 25.

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"), an "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 15.

The Decision Table

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

Table IV

Decision Table Card Format

Columns	Contents
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 + 1)-(nic+nsv)*	The (nic + j)th card column contains the symbol denoting the condition imposed on the jth state variable. It must be Y, N or -.

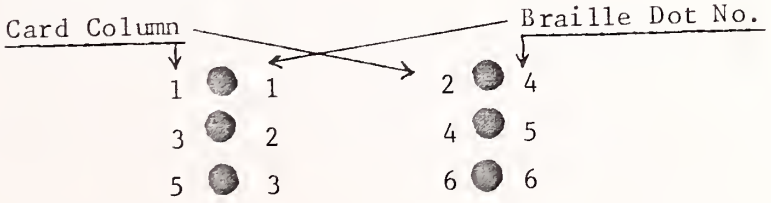
The Sign Table

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 V.

Table V

Sign Table Card Format

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



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

Columns	Contents
7-9	Up to three characters to be printed on proof output, denoting the most common meaning for this sign.

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 5 run control cards. The first two 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:

(1) Proof output (P or N in column 1).

T in column 16 indicates that a listing of the logic tables is also required.

(2) MIT embosser output (M or N in column 1).

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

(3) The number of braille signs per line (not less than 2 nor greater than 40).

(4) The number of lines per page.

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

(5) 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 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 - 80 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 italics and format controls, and to force or prevent improper translations in exceptional cases.

Column 80 of a card is considered to be adjacent to column 1 of the next card. In general, any number of consecutive spaces will be collapsed automatically into a single space (refer to the description of the primary input processor).(e.g.

THE START. THE END.

is interpreted as

THE START. THE END.)

Capitalisation

Where a capital sign is required in braille, a logical-not sign (¬) should be punched. Two logical-not signs (¬¬) should be punched if a double capital sign is required. The capital sign is not normally used.

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.

Ordering

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

Italic sign (_)

Capital sign(s) (¬ or ¬ ¬)

Accent sign (ȼ)

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

Mathematical Symbols. The addition sign (+) is represented by $+, the multiplication sign (×) is represented by $X, the subtraction sign (-), is represented by $-, and the division sign (÷) is represented by $DIV. In each case the symbol should be preceded and followed by a space. Other mathematical signs such as > (greater than) and < (less than) must be spelt 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.

$' followed by a space 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.

$" followed by a space 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 cents sign (¢). For example, Abbé is keypunched ABB¢E.

Currency. The pound sign (£) should be represented by a letter L. Where the abbreviation P is used for pence, PENCE should be written out in full.

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.

Caesure Sign. To insert a caesure sign, the special symbol $CS is used.

Null Symbol. The symbol $/ is a null replacement symbol (not a blank) and is generally used to prevent contractions (see above).

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

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).

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 $SLnnb, 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 (2 x tab number).

Margin Indentation. The symbol $Mnn will cause the left-hand margin to be set to column nn, so that text will start in column nn+1 of each line. Format control symbols override any margin indentation so that, for example, $L will still cause a new line to start in column 1. Margin indentation may be used to indent continuation lines in an alphabetical list, or for line by line poetry. The symbol $M00 will restore the left-hand margin to the left-hand edge of the page.

Double Spacing. The symbol $DOUBLE will cause subsequent output to appear with a blank line between each line of output. A further $DOUBLE will cause subsequent output to resume single-spacing.

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 centred 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 centred 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. Two alternative forms of poetry input may be used.

(i) If line by line poetry is required, that is, each line of poetry starts a new line of braille: the poetry text should be preceded by the symbol $M04 and followed by the symbol $M00, the first line of each verse should be preceded by the symbol $P and each other line by $L.

(ii) If continuous poetry is required: the poetry text should be preceded by the symbols $P $PY, each verse except the first should be preceded by $P, and at the end of each line except the last line of the poem the symbol $PS should be added without any preceding space.

Computer Braille. An occurrence of the computer braille switch, $C, changes the mode of translation from grade 2 to computer braille or vice versa. It must not be used during grade 1 translation.

Processing several independent documents together. The symbol $T; will reset page numbering to 1, reset state-variables to their initial values, and cancel margin-indentation, double spacing, computer braille, and running title if any of these are in effect.

If several documents are to be processed together it is recommended that the following be inserted at the beginning of the text for each document:

$PG $%$%$%$% $%$%$%$% $SL01 <identification>

$SL01 $%$%$%$% $%$%$%$% $T $PG

where <identification> is a short description or name for the document. This will produce a distinctive header page for each document in the output, containing the braille translation of the <identification> surrounded by two rows of braille FOR signs (all six dots).

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.

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 number sign (#), the grade switch ($G), the division symbol ($/) and the forced-contraction symbols (/_, _/).

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 capitalised or uncapitalised 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.)

(4) Roman numerals appear in the text.

(5) abbreviations are used without periods.

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

Number Sign (#)

The number sign should be punched instead of a period where there may be confusion with a decimal point and the number sign is required in braille. This applies to currency and times of the day, e.g. L6#99, 10#30 A.M.

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.

Division Symbol ($/)

The division symbol is the most 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.

It must be used in a time interval after the hyphen separating minutes from hours, as in 9#30 - $/10#30, in order to produce a number sign correctly.

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.

OUTPUT

Proof Output

Proof output is optional (see "Run Control Cards"), and is not normally called for in 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 as the error messages (q.v.); error messages may be interspersed with the normal proof output.

If table-display is called for, the first pages of proof output display the contents of the right context table, the decision table, and the transition table in an easily readable form.

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 sign 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 to 63. Figure 2 contains a sample of proof output.

A literal listing of the text input cards, as read, is also produced between the lines of braille output (where a "braille output line" is actually a group of five printed lines). These card images will generally occur somewhat sporadically, with two or more occurring together at times and none between pairs of braille lines in other cases. (For separation, a blank line is printed in the latter case.) Typically, a given input word and the corresponding output are separated by about two braille lines.

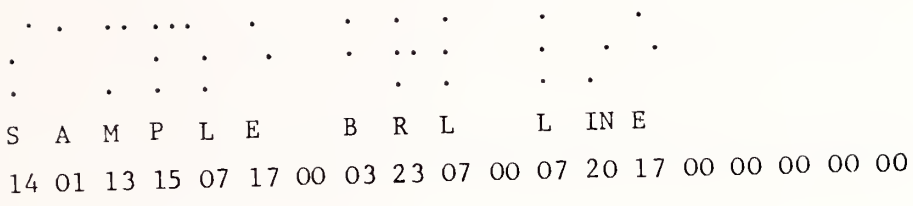


Figure 2. Sample of Proof Output

Error Messages

Error messages are unconditionally printed on the proof-output device, thereby appearing intermixed with proof output, if any. The following is a list of the possible messages. For each entry, "A" is an explanation, "B" is the program's automatic action, and "C" is the suggested user response.

(1) NEW CHAR X

A. The character printed (X) has appeared in the input text but does not have an alphabet table entry.

B. The character is replaced by an asterisk and processing proceeds.

C. Correct the text input or provide an alphabet table entry.

(2) ** TABLE SEQUENCE ERROR

A. A contraction table section is out-of-sequence with respect to the alphabet table order.

B. Reading of the tables is continued but processing of the text is suppressed.

C. Refer to "Input" for a discussion of the relationship between the alphabet table arrangement and that of the contraction table, and rearrange accordingly.

SECTION IV

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 (continued 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 "BCDF" 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 algorithms 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 3 and 4) 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).

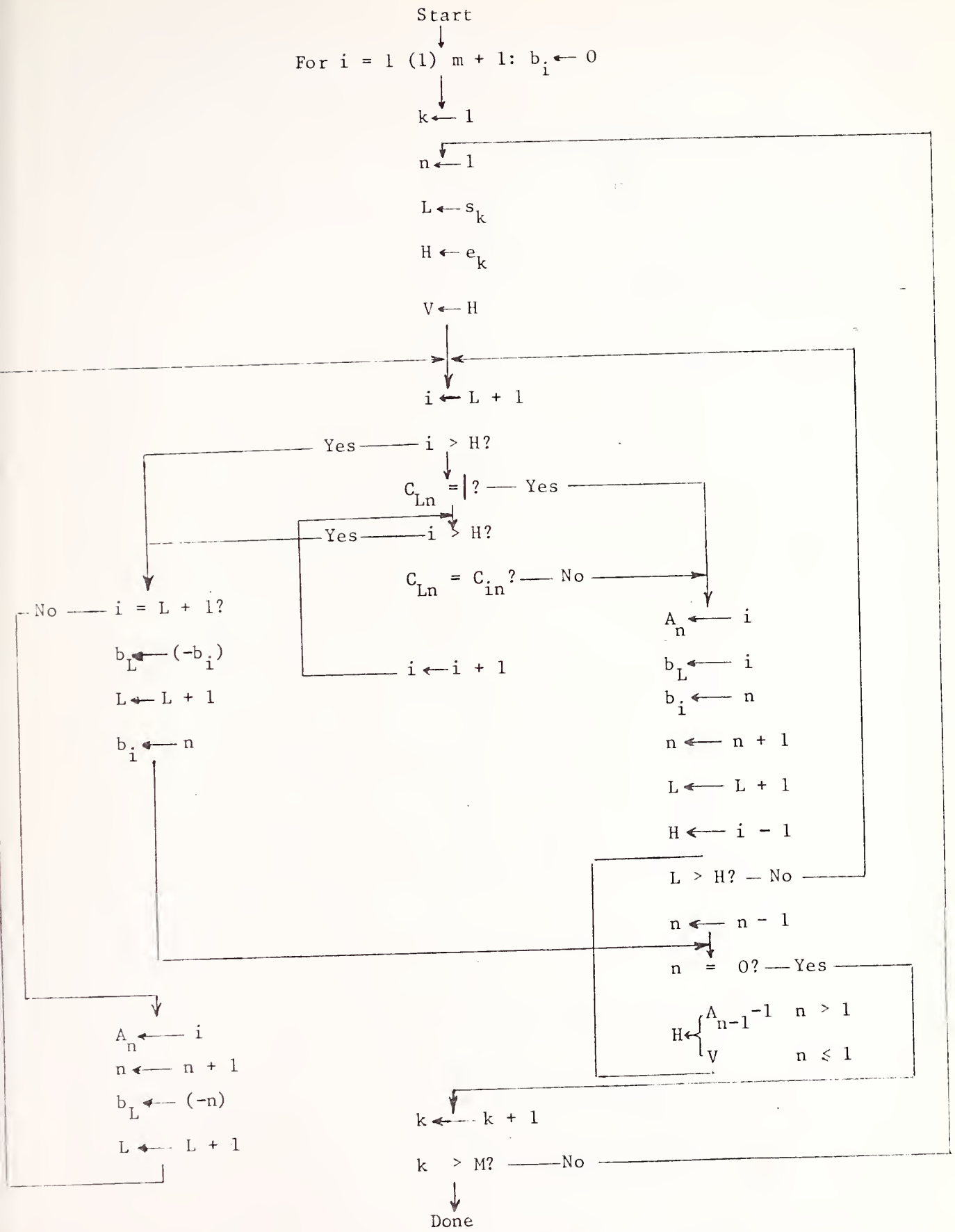


Figure 3. Set-Up Algorithm

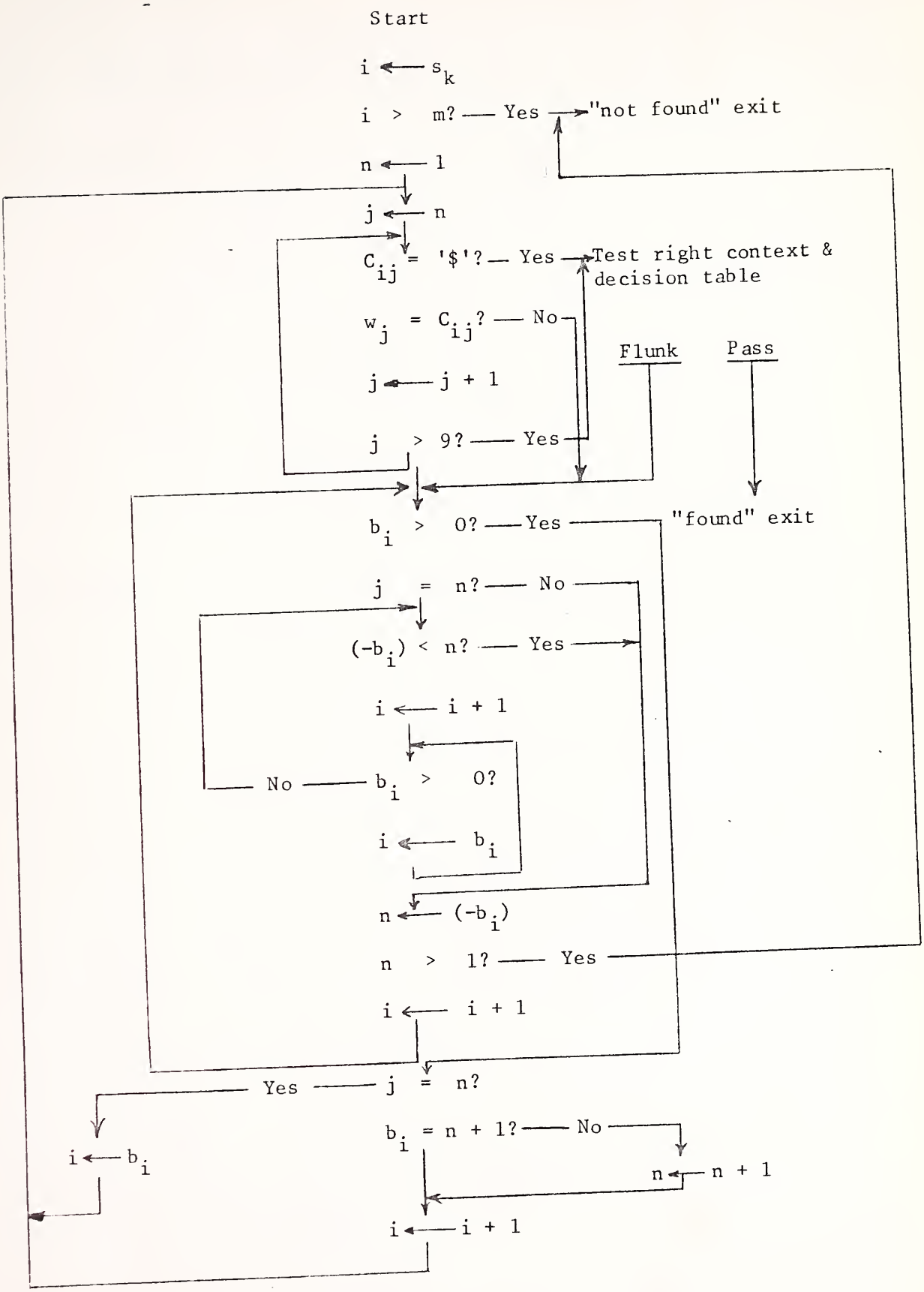


Figure 4. Look-Up Algorithm

APPENDIX 1.1

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
Input Class	1	contractions always used in grade 2
	2	digits
	3	most punctuation
	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)
	13	_ (last word of italics)
	14	(space)

APPENDIX 1.2

SUMMARY OF SPECIAL SYMBOLS

Symbol	Input Representation
capitalize letter	¬
capitalize word	¬ ¬
italicize word	_
italics start	_ _
left single quote	$'
right single quote	$'R
left double quote	$"
right double quote	$"R
accent marks	¢
left bracket ([)	<
right bracket (])	>
start paragraph	$P
begin new line	$L
begin new page	$PG
skip to column nn	$TABnn
letter sign	=
number sign	≠
change grade	$G
divide word	$/
long vowel sign	$LV
start paragraph within quotation	$P"
start title	$TLS

Symbol	Input Representation
end title	$TLE
forced blank	$B
start heading	$HDS
end heading	$HDE
skip lines	$$Lnn
short vowel sign	$SV
set tab t to col. nn	$STBt[L,R or D]nn
end of poetry foot sign	$FT
computer braille switch	$C
caesura sign	$CS
call (permanent) tab t	$#t
start forced contraction	/_
end forced contraction	_/
introductory poetry sign	$PY
poetry sign	$PS

APPENDIX 1.3

EQUIVALENT SIGNS, SYMBOLS AND CODES

	0	1	2	3	4	5	6	7
0		c (@)	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	, 1	I	: 3	J	EN 5	OW ,	. 4	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 9	AR >	-	ING (+)	" 0 (zero)	#
	04	12	20	28	36	44	52	60
5	K	M	O	N	U	X	Z	Y
	05	13	21	29	37	5	53	61
	; 2	S	TO 6	T	? 8	THE !	)(7	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

second octal digit

first octal digit

Braille sign	
Proof character(s)	Computer braille graphic
Sign code	

FORMAT

Included only where different from proof character. Parentheses around the computer braille graphic indicate that it is not implemented in the tables listed in Appendix I (and will be translated as a blank)

APPENDIX 1.4

TRANSLATOR - STACKER SIGN CODES

Code	Meaning
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	initialise state-variables and page-numbering
84	set continuous text stacking mode
85	idle (reserved for: reset continuous text stacking mode)
86	unit of measure
87	unused
88	start running title input
89	end of running title input
90	switch double-spacing of lines on or off
93	set tab
95	set margin

Code	Meaning
97	start computer-braille
98	end of run
99	(filler in contraction table)

APPENDIX 1.5

MISCELLANEOUS CODED VARIABLES

Variable	Code	Meaning
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 "off" = "no" = "false", "on" = "yes" = "true".

APPENDIX 1.6

RJE TAPE FILE HANDLING PROGRAM

This program incorporates a modified version of DOTSYS III-F which by the inclusion of assembly code achieves a considerably faster translation speed and uses less core storage than the pure Fortran version, together with several subprograms concerned with preparation of data for DOTSYS and embossing of the braille output from DOTSYS.

The program operates on a magnetic tape containing files in Remote Job Entry (RJE) format. The files on this tape may be edited on a visual display unit using the interactive utility text-editing program TED. (See "Summary of Text Editor commands".)

The specific functions of the RJE tape file handling program are as follows:

1. to create a tape file from punched cards
2. to list one or more tape files on the lineprinter
3. to convert the DOTSYS tables (which are stored as a file on the RJE tape) and one or more files of text to card-image format for processing by DOTSYS
4. to store files on a separate archive tape for future reference so that they may be deleted from the RJE tape once they have been translated
5. (DOTSYS) to produce a file containing the braille translation of one or more files of text
6. to emboss a file containing braille on the MIT Braillemboss.

Summary of Text Editor (TED) Commands

The symbols < and > are used below for the purpose of description only and are not actually typed. Underlinings show which character sequences are typed by the user. <CR> refers to the carriage-return key.

File handling commands

#BUILD<file name><CR>
<line 1><CR>
<line 2><CR>
⋮
<line m><CR>
<CR>

creates a new file with a name given by the string <file name>; the file will contain the lines following the command. Termination of the file content and the BUILD command occurs if two consecutive new-lines are requested.

#LIST <file name><CR>

will be followed by the listing of the entire named file. This command may be used without an argument, then the names of all files on the loaded tape will be listed.

#DELETE <file name><CR>

Deletes the named file.

#EDIT <file name><CR>

Calls in the edit overlay and prepares the named file for editing.

#BYE

causes the computer to execute some housekeeping on the user files then releases the program. The terminal must not be used again until the program is released, as indicated by a message on the operator's console.

Edit Subcommands

Find commands

*FI<integer><CR>

Find the line <integer> further down the text than the current line of interest and modify the pointer to create this new line of interest.

*FI,<text string><CR>

Find the next line that begins with text string and set the pointer to indicate this new line. If the end of the file is reached before the line is found, the search continues from the beginning of the file.

*FI<CR>

Sets the pointer to the end of the file.

Delete commands

*DE<integer><CR>

Delete <integer> lines - the current line and <integer>-1 subsequent lines - the pointer is set to the line immediately following the deletion.

*DE,<text string><CR>

Deletes all lines up to and including the line beginning with <text string>, deletes the current line and moves the pointer to the line following the delection.

*DE<CR>

Deletes the current line and increments the pointer by one.

Insertion commands

*IN CR
<line 1><CR>
<line 2><CR>
:

⋮

<line m><CR>

<CR>

Inserts the lines following the command before the line of current interest; the pointer is not altered. This command is terminated by two consecutive carriage return characters.

Typing commands

*TY<integer><CR>

Types the current line and the following <integer>-1 lines; the pointer is not changed.

*TY,<text string><CR>

Types all lines between and including the current line and the line beginning with <text string>; the pointer is unaltered.

*TY<CR>

Types the current line.

Search limiting commands

*LT<integer><CR>

Set the limits of searching to <integer> lines. From the current line to the (<integer>-1)th following line.

*LT,<text string><CR>

Set the limits of searching from the current line to the line beginning with <text string> inclusive.

*LT<CR>

Reset the limits to the entire file and sets the pointer to the first line in the file.

Substitute command

*SU,<text string 1>\<text string 2><CR>

Replace the first occurrence of <text string 1> on the current line by <text string 2>. If <text string 1> is omitted, <text string 2> will be inserted at the beginning of the line. If <text string 2> is omitted, <text string 1> will be deleted.

Repeat command

*RE<CR>

Repeat the last typed editing subcommand.

End of Edit Commands

*END<CR>

Terminate the edit, updates the file according to the commands given and returns control to the TED command decoder.

*<INTERRUPT>

terminates the edit and leaves the user file unaltered - as though the most recent sequence of edit commands had not been sent.

Compound commands

Insert and delete

*ID<integer><CR>

<line 1><CR>

<line 2><CR>

<line 3><CR>

⋮

<line m><CR>

<CR>

Deletes <integer> lines including the current line and inserts the lines following the text in that place; the pointer indicates the

line after the insertion on completion of the command. The command is terminated by two consecutive carriage return characters.

*ID,<text string><CR>
<line 1><CR>
<CR>

Similar to the previously described command but deletion is up to and including the line beginning with <text string>.

*ID<CR>
<line 1><CR>
<line 2><CR>
⋮
<line m><CR>
<CR>

Deletes the current line and inserts the text following the command.

Find and type commands

*FT<integer><CR>

Sets the pointer <integer> lines further down the text and types the line when it is found.

*FT,<text string><CR>

Sets the pointer to the next line beginning with <text string> and types the entire line.

NOTES

1. The 'BREAK' or 'INTERRUPT' character halts any listing after a further line. The listing may be terminated by pressing 'INTERRUPT' again, or continued by typing <CR>.
2. In an EDIT command a '?' is 'wild' and can be substituted for any character.
3. 'DEL' deletes the last letter typed in; 'CAN' deletes the current line.

APPENDIX 1.7

USE OF MACROS IN CARD INPUT TO RJE TAPE

(these can save time in typing documents in which a phrase or long word occurs many times.)

A MACRO DEFINITION has the following form:

@@<character-pair>=<text>@

Where <character-pair> is any pair of characters not including an @ sign

and <text> is any sequence of characters (of any length) not including an @ sign.

e.g. @@CV=$P COVENTRY (POOL MEADOW)@
@@LL=$P LONG LAWFORD (GREEN)@ @@BT=BRETFORD TURN@
@@XX=$P BRETFORD TURN 05 $P CHURCH LAWFORD (CROSSROADS) 09 $P LONG LAWFORD (GREEN) 13 $P RUGBY (SHEEP STREET) 20@
@@RC=$P RUGBY CENTRAL@

Any number of macro definitions may occur in a document and they may occur anywhere in the input without affecting adjacent text.

A MACRO has the following form:

@<character-pair>

Where <character-pair> is the same as in one of the macro-definitions in the same document.

A macro may occur any number of times anywhere in the input (except in a macro definition) after it has been defined. Each time a macro occurs, the program which copies the cards to tape will replace it with the corresponding text.

e.g. @CV 0805 @LL 15 $P @BT 20 ...
or @CV 1300 @XX @RC 25 ....

```
      C     PROGRAM-TITLE:  DOTSYS III-F
      C     ----------------------------
      C
      C     THIS PROGRAM IS A TRANSLATION, WITH MODIFICATIONS, INTO RXDS EXTENDED
      C     FORTRAN IV-H OF THE COBOL PROGRAM DOTSYS III.
      C     REFERENCE:  DOTSYS III, A PORTABLE PROGRAM FOR GRADE II BRAILLE
      C     TRANSLATION, BY W.R. GERHART, J.K. MILLEN, AND J.E. SULLIVAN,
      C     MITRE TECHNICAL REPORT MTR-2119, THE MITRE CORPORATION, BEDFORD,
      C     MASSACHUSETTS.
      C
      C
      C     MODIFICATIONS MADE TO ORIGINAL PROGRAM
      C     --------------------------------------
      C
      C     THE FOLLOWING HAVE BEEN OMITTED:
      C     SELF-CHECKING AND ALPHABETICAL CONTRACTION LIST; ECHO OF TABLES; DOUBLE
      C     SPACE PERMITTED AFTER FULL STOP; OCTAL BRAILLE; ELASTOMER BRAILLE
      C     OUTPUT; PPO BRAILLE OUTPUT; PUNCHED CARD OUTPUT; LINE BY LINE POETRY.
      C
      C     THE FOLLOWING HAVE BEEN MODIFIED:
      C     COMPUTER BRAILLE NOT TERMINATED BY A SPACE; TABLE FORMAT: RIGHT CONTEXT
      C     CLASS DESIGNATOR IS IN COLUMN 13 INSTEAD OF 16 IN CONTRACTION TABLE AND
      C     RIGHT CONTEXT CLASS INPUT.
      C
      C     THE FOLLOWING HAVE BEEN ADDED:
      C     DOUBLE-SPACING OF OUTPUT; MARGIN INDENTATION; RESET FACILITY FOR
      C     PROCESSING SEPARATE DOCUMENTS CONSECUTIVELY.
      C
      C
      C     USE OF VARIABLES
      C     ----------------
      C
      C     ALPHABET TABLE
      C     SYMBOL    ACCEPTABLE INPUT CHARACTERS
      C     CCLASS    CHARACTER CLASS FOR SYMBOL
      C     SINGSG    SINGLE-SIGN TRANSLATION FOR SYMBOL
      C     EXTENT    INDEX OF LAST CONTRACTION TABLE ENTRY STARTING WITH GIVEN SYMBOL
      C     COMPB     COMPUTER BRAILLE TRANSLATION FOR SYMBOL
      C     ISOLET    "IS-ROUND-LETTER"
      C     NALPH     NUMBER OF ALPHABET TABLE ENTRIES
```

```
41          C
42          C     CONTRACTION TABLE
43          C     CTABLE   CONTRACTION TABLE.  EACH COLUMN CONTAINS THE CHARACTER-STRING IN
44          C              BYTES 1-9 AND THE RIGHT CONTEXT CLASS IN BYTE 12
45          C     ICLASS   THE INPUT CLASSES FOR THE CONTRACTION TABLE ENTRIES
46          C     SIGNS    SHIFT COUNT AND FOUR BRAILLE SIGN CODES FOR EACH CONTRACTION
47          C                 TABLE ENTRY, PACKED ARITHMETICALLY INTO A SINGLE INTEGER
48          C     BRANCH   INDEX OF NEXT ENTRY TO BE EXAMINED IN CONTRACTION TABLE SEARCH
49          C     NCT      NUMBER OF CONTRACTION TABLE ENTRIES
50          C     CTMAX    ANY NUMBER GREATER THAN THE DIMENSION OF CTABLE
51          C
52          C     LOGIC TABLES
53          C     DECIS    DECISION TABLE
54          C     COND     CONDITION TABLE
55          C     TRANS    TRANSITION TABLE
56          C     NONTER   RIGHT-CONTEXT DESIGNATORS
57          C     RCCLAS   INPUT CLASS CODES CORRESPONDING TO NONTER
58          C     NDTC     NUMBER OF DECISION TABLE ENTRIES
59          C     NIC      NUMBER OF INPUT CLASSES
60          C     NNT      NUMBER OF NON-TERMINAL ENTRIES (RIGHT CONTEXT CLASSES)
61          C     STVAR    STATE VARIABLES
62          C     NSV      NUMBER OF STATE VARIABLES
63          C
64          C     STACKS
65          C     CURTYP   TYPE OF STACK IN CURRENT USE:
66          C              1 = NORMAL TEXT OR HEADING,  2 = RUNNING TITLE
67          C     CS       POINTER TO ROW OF CURRENT STACK CURRENTLY IN USE
68          C     HS       POINTER TO HEAD OF CURRENT STACK
69          C     TS       POINTER TO TAIL OF CURRENT STACK
70          C     CURRS    POINTERS TO CURRENT ROWS OF STACKS
71          C     HEADS    POINTERS TO HEADS OF STACKS
72          C     TAILS    POINTERS TO TAILS OF STACKS
73          C     STKIND   STACK INDICATOR, TAKES ONE OF THE FOLLOWING VALUES:
74          C              2   NORMAL TEXT; CLEAR STACK AFTER EACH ENTRY
75          C              4   CONTINUOUS STACKING OF TITLE OR HEADING
76          C              5   CONTINUOUS STACKING OF NORMAL TEXT
77          C     HFSTAK   POINTER TO HEAD OF FREE STACK
78          C     ELT      19 ROWS OF STORAGE FOR STACKS
79          C     SPCONT   SPECIAL CONTROL SIGN IF ANY FOR GIVEN ROW OF STACK
80          C     NELTS    NUMBER OF SIGNS HELD IN GIVEN ROW OF STACK
```

```
      C     PVSTAK   BACKWARD POINTER TO PREVIOUS ROW IN STACK (0 FOR FIRST ROW)
      C     NXSTAK   FORWARD POINTER TO NEXT ROW IN STACK (0 FOR LAST ROW)
      C
      C     SIGN TABLE
      C     DOTS14, DOTS25, DOTS36   BRAILLE SIGN FOR LINE-PRINTER PROOF OUTPUT
      C                 (SPACE OR DOT)
      C     PCHAR    UP TO THREE CHARACTERS ASSOCIATED WITH SIGN FOR PROOF OUTPUT
      C     MCODE    SINGLE CHARACTER FOR OUTPUT TO EMBOSSER
      C
      C     TABULATION
      C     TABTYP   TYPES OF TABS (LEFT, RIGHT OR DECIMAL JUSTIFIED)
      C     TABCLM   COLUMNS ON WHICH TABS ARE BASED
      C     TABULA   COLUMN AT WHICH TABULATED TEXT IS TO BEGIN IN GIVEN CASE
      C
      C     EMBOSSER OUTPUT
      C     EMBOUT   OUTPUT BUFFER FOR EMBOSSER OUTPUT
      C     MEMBCR   CARRIAGE-RETURN CHARACTER FOR EMBOSSER
      C     MEMBPG   PAGE-FEED CHARACTER FOR EMBOSSER
      C     MIDLE    IDLE CHARACTER FOR EMBOSSER
      C
      C     PROOF OUTPUT
      C     BRAILL   OUTPUT BUFFER FOR PROOF BRAILLE SIGNS
      C     PROOF    OUTPUT BUFFER FOR PROOF CHARACTERS
      C     CODE     OUTPUT BUFFER FOR CODE NUMBER ASSOCIATED WITH BRAILLE SIGN
      C     BTEMP, CTEMP, PTEMP   TEMPORARY STORAGE FOR PROOF OUTPUT BUFFER DURING
      C                 OUTPUT OF PAGE HEADING
      C     CPRINT   TRUE IF INPUT CARD IMAGE HAS BEEN PRINTED SINCE LAST PROOF-LINE
      C
      C     INPUT
      C     TEXT     INPUT BUFFER
      C     INPTR    POINTER TO CURRENT POSITION IN INPUT BUFFER
      C     PRIORS, SCOUNT   USED FOR CONDENSING CONSECUTIVE SPACES IN INPUT
      C
      C     OUTPUT
      C     OUTPTR   POINTER TO CURRENT POSITION IN OUTPUT LINE
      C     OUTL     NUMBER OF CHARACTERS PER LINE
      C     LPG      NUMBER OF LINES PER PAGE
      C     LCOUNT   CURRENT LINE
      C
      C     LOGICAL VARIABLES
```

```
121         C      COMPUT   COMPUTER BRAILLE TRANSLATION
122         C      EMBOPT   EMBOSSER OUTPUT REQUIRED
123         C      PFOUT    PROOF OUTPUT REQUIRED
124         C      PAGINA   PAGINATION REQUIRED
125         C      DOUBLE   DOUBLE SPACING REQUIRED (BLANK LINE BETWEEN EACH LINE OF OUTPUT)
126         C      INHEAD   IN HEADING
127         C
128         C      GENERAL
129         C      CDIGIT   FOUR DIGITS OF CURRENT PAGE
130         C      DIGIT    BRAILLE SIGN CODES FOR DIGITS
131         C      CCHIGH   HIGHEST VALUE OF CARRIAGE CONTROL CODES
132         C      SCHIGH   HIGHEST VALUE OF STACK CONTROL CODES
133         C
134         C
135               IMPLICIT INTEGER(A-Z)
136               COMMON /BUFF/ BUFFER(10)
137               EQUIVALENCE (BUFFER(1),BUFF1),(BUFFER(2),BUFF2,RBUFF(1))
138               DIMENSION RBUFF(9)
139               COMMON /CTAB/ CTABLE(3,1000),ICLASS(1000),SIGNS(1000),
140              *   BRANCH(1000),NCT,TSIGN(4)
141               COMMON /ALPH/ SYMBOL(64),CCLASS(64),SINGSG(64),EXTENT(64),
142              *   COMPB(64),ISOLET(64),NALPH
143               COMMON /LOGIC/ DECIS(15,25),COND(15,10),TRANS(10,25),NONTER(2),
144              *   RCCLAS(4,2),NDTC,NIC,NNT
145               COMMON /STV/ STVAR(10),NSV
146               COMMON /SPAC/ SPACE
147               COMMON /S/ LETS
148               COMMON /N/ LETN
149               COMMON /COMP/ COMPUT
150               LOGICAL COMPUT
151               COMMON /RCC/ RCC
152               COMMON /FILES/ TBDEV,INDEV,PRDEV,BRDEV
153         C     FOLLOWING COMMONS NOT USED IN MAIN PROG BUT NEEDED FOR OVERLAY
154               COMMON /STAX/ TS,HS,CS,PS,TAILS(2),HEADS(2),CURRS(2),CURTYP,STKIND
155              *,HFSTAK,ELT(19,30),SPCONT(19),NOELTS(19),PVSTAK(19),NXSTAK(19)
156               COMMON /IN/ INPTR,PRIORS,SCOUNT
157               COMMON /OUT/ OUTPTR,OUTL,LINES,LPG
158               COMMON /TAB/ TABTYP(9),TABCLM(9),TABULA,LETR,LETD
159               COMMON /OPTS/ EMBOPT,PFOUT
160               COMMON /SIGNS/DOTS14(64),DOTS25(64),DOTS36(64),PCHAR(64),MCODE(64)
```

```
      COMMON /MEMR/ EMBOUT(11),MEMBCR,MEMBPG,MIDLE,MEMBON,MEMBOF
      COMMON /CPR/ CPRINT
      COMMON /PAG/ CDIGIT(4),DIGIT(10),PAGINA
      COMMON /LC/ LCOUNT
      LOGICAL EMBOPT,PFOUT,CPRINT
C
      DATA LETA,LETF,LETG,LETI,LETN,LETO,LETR,LETS,LETT,LETY,STAR,
     *   HYPHEN,ITALIC,SPACE,LETSIG,CAPSIG,ACCENT,OPBRAK,CLBRAK/
     *   'A','F','G','I','N','O','R','S','T','Y','*','-','~',' ','=',
     *   '¬','¢','(',')'/
      DATA DIGIT /26,1,3,9,25,17,11,27,19,10/
      DATA TBDEV,INDEV,BRDEV,PRDEV /131,131,132,108/
      DATA RCC / '|' /
    1 FORMAT(' NEW CHAR ',A1)
C
C     SEGLOD LOADS SPECIFIED SEGMENT OF OVERLAY
      CALL SEGLOD(3)
      CALL INIT
      CALL SEGLOD(4)
      CALL SHIFT(10)
      CALL PCLEAR
      CALL RESET
C
C     IDENTIFY LEFTMOST CHARACTER OF BUFFER
  200 DO 400 LETTER=1,NALPH
  400 IF (SYMBOL(LETTER) .EQ. BUFF1) GO TO 600
      IF (BUFF1 .NE. RCC .AND. BUFF1 .NE. STAR) GO TO 9800
      CALL OPSIGN(98)
C     RUN WILL STOP WITH ABOVE CALL
C
  600 IF (ISOLET(LETTER) .NE. 1) GO TO 3200
C
C     INSERT LETTER SIGN BEFORE SINGLE LETTER IN BRACKETS, QUOTES, ETC.
      IF (BUFF1 .EQ. ITALIC .AND. (BUFF2 .EQ. LETA .OR. BUFF2 .EQ. LETI
     *   .OR. BUFF2 .EQ. LETO .OR. BUFF2 .EQ. ITALIC)) GO TO 3200
      N = 1
      K = 1
  800 T = RBUFF(N)
      IF (T .EQ. LETSIG) GO TO 3200
      IF (T .EQ. CAPSIG .OR. T .EQ. ACCENT) GO TO 2200
```

```
201             IF (T .EQ. ITALIC) GO TO 2400
202             DO 1000 X=1,NALPH
203          1000 IF (SYMBOL(X) .EQ. T) GO TO 1200
204             WRITE(PRDEV,1) T
205          1200 IF (CCLASS(X) .NE. 1) GO TO 3200
206             M = N + 1
207             IF (BUFF1 .EQ. ITALIC) GO TO 2600
208             IF (SYMBOL(LETTER) .EQ. OPBRAK) GO TO 1400
209             IF (SYMBOL(LETTER) .EQ. RBUFF(M)) GO TO 1600
210             GO TO 3200
211          C
212          1400 IF (RBUFF(M) .NE. CLBRAK) GO TO 3200
213          1600 DO 1800 I=1,K
214             RBUFF(M) = RBUFF(N)
215             N = N - 1
216          1800 M = M - 1
217             CALL SHIFT(M)
218             BUFF1 = LETSIG
219             GO TO 200
220          C
221          2200 K = K + 1
222          2400 N = N + 1
223             GO TO 800
224          C
225          2600 DO 2800 X=1,NALPH
226          2800 IF (SYMBOL(X) .EQ. RBUFF(M)) GO TO 3000
227          3000 IF (CCLASS(X) .EQ. 1) GO TO 3200
228             IF (N .GT. K) CALL SHIFT(1)
229             BUFF1 = LETSIG
230             GO TO 200
231          C
232          C     CONTRACTION TABLE SEARCH
233          C     INDEX POINTS TO CONTRACTION TABLE ENTRY UNDER CONSIDERATION
234          3200 INDEX = EXTENT(LETTER)
235             IF (INDEX .GT. NCT) GO TO 9200
236             NXTLEV = 1
237          C
238          C     TEST FOR MATCH BETWEEN CURRENT TABLE ENTRY AND BUFFER
239          3400 DO 3600 POINTR=NXTLEV,9
240             T = CHAR(CTABLE,INDEX,POINTR)
```

```
241               IF (T .EQ. RCC) GO TO 5000
242          3600 IF (RBUFF(POINTR) .NE. T) GO TO 3800
243               GO TO 5000
244         C
245         C     MISMATCH; OBTAIN INDEX FOR NEXT ENTRY TO BE COMPARED
246          3800 BRI = BRANCH(INDEX)
247               IF (BRI .GT. 0) GO TO 4600
248               IF (POINTR .NE. NXTLEV) GO TO 4400
249          4000 IF (-BRI .LT. NXTLEV) GO TO 4400
250               INDEX = INDEX + 1
251          4200 BRI = BRANCH(INDEX)
252               IF (BRI .LE. 0) GO TO 4000
253               INDEX = BRI
254               GO TO 4200
255         C
256          4400 NXTLEV = -BRI
257               IF (NXTLEV .LE. 1) GO TO 9200
258               INDEX = INDEX + 1
259               GO TO 3400
260          4600 IF (POINTR .EQ. NXTLEV) GO TO 4800
261               IF (BRI .NE. INDEX+1) NXTLEV = NXTLEV + 1
262               INDEX = INDEX + 1
263               GO TO 3400
264          4800 INDEX = BRI
265               GO TO 3400
266         C
267         C     CHARACTER STRING MATCHES LEFT SUBSTRING OF BUFFER
268         C     NOW TEST NEXT CHARACTER IN BUFFER TO CHECK THAT RIGHT CONTEXT IS O.K.
269          5000 T = CHAR(CTABLE,INDEX,12)
270               IF (T .EQ. SPACE) GO TO 6200
271               DO 5200 N=1,NALPH
272          5200 IF (SYMBOL(N) .EQ. RBUFF(POINTR)) GO TO 5400
273               GO TO 3800
274         C
275          5400 DO 6000 K=1,NNT
276               IF (T .NE. NONTER(K)) GO TO 6000
277               DO 5800 J=1,4
278               IF (CCLASS(N) .EQ. RCCLAS(J,K)) GO TO 6200
279          5800 IF (RCCLAS(J,K) .EQ. 99) GO TO 3800
280               GO TO 3800
```

```
 6000 CONTINUE
      GO TO 3800
C .
C     RIGHT CONTEXT IS O.K. - NOW CHECK INPUT CLASS FOR COMPATIBILITY
C     WITH CURRENT VALUE OF STATE VARIABLES
 6200 TCLASS = ICLASS(INDEX)
      DO 7000 K=1,NDTC
      DKT = DECIS(K,TCLASS)
      IF (DKT .EQ. HYPHEN) GO TO 7000
      IF (DKT .EQ. LETG) GO TO 7600
      IF (DKT .EQ. LETF) GO TO 3800
      DO 6800 N=1,NSV
      CKN = COND(K,N)
 6800 IF (CKN .NE. HYPHEN .AND. CKN .NE. STVAR(N)) GO TO 7400
      GO TO 7200
 7000 CONTINUE
C
 7200 IF (DKT .EQ. LETY) GO TO 7600
      GO TO 3800
 7400 IF (DKT .EQ. LETY) GO TO 3800
C
C     CONTRACTION TABLE ENTRY IS ACCEPTED
C     SEND APPROPRIATE SIGNS TO STACKER AND SHIFT BUFFER LEFT BY
C     NUMBER OF CHARACTERS INDICATED IN CONTRACTION TABLE ENTRY
 7600 T1 = SIGNS(INDEX)
      T = T1 / 11
      CALL SHIFT(T1 - 11 * T)
      DO 8200 J=1,4
      T1 = T / 100
      OUTSIG = T - 100 * T1
      IF (OUTSIG .EQ. 99) GO TO 8400
      T = T1
 8200 CALL OPSIGN(OUTSIG)
C
C     ADJUST STATE VARIABLES ACCORDING TO TRANSITION TABLES
 8400 I = TCLASS
      DO 9000 N=1,NSV
      TNI = TRANS(N,I)
      IF (TNI .EQ. HYPHEN) GO TO 9000
      IF (TNI .NE. LETS .AND. (TNI .NE. LETT .OR. STVAR(N) .NE. LETN))
```

```
321             *    GO TO 8800
322                  STVAR(N) = LETY
323                  GO TO 9000
324              8800 STVAR(N) = LETN
325              9000 CONTINUE
326                  GO TO 200
327             C
328             C    NO CONTRACTION TABLE ENTRY IS ACCEPTABLE - OUTPUT SINGLE CHARACTER
329             C    OBTAINED FROM ALPHABET TABLE
330              9200 IF (.NOT. COMPUT) GO TO 9400
331                  OUTSIG = COMPB(LETTER)
332                  GO TO 9600
333              9400 OUTSIG = SINGSG(LETTER)
334              9600 CALL SHIFT(1)
335                  CALL OPSIGN(OUTSIG)
336                  TCLASS = CCLASS(LETTER)
337                  GO TO 8400
338             C
339             C    LEFTMOST CHARACTER OF BUFFER DOES NOT OCCUR IN ALPHABET TABLE
340             C    OUTPUT ERROR MESSAGE
341              9800 WRITE(PRDEV,1) BUFF1
342                  BUFF1 = STAR
343                  GO TO 200
344                  END


345                  INTEGER FUNCTION CHAR(ARRAY,I,J)
346             C    VALUE OF CHAR IS CHARACTER IN J TH ROW OF I TH COLUMN OF ARRAY,
347             C    LEFT JUSTIFIED
348                  IMPLICIT INTEGER(A-Z)
349                  DATA ADDEND /6Z404040/
350                  CHAR = ISL(ICH(ARRAY,12*(I-1)+J),24) + ADDEND
351                  END


352                  SUBROUTINE STCHAR(CHAR,ARRAY,POSN)
353                  IMPLICIT INTEGER(A-Z)
354             C    STORES LEFT-HAND BYTE OF CHAR IN POSN TH BYTE OF ARRAY
355             S       LB,9      *CHAR
356             S       LW,7      *POSN
```

```
      S         AI,7       -1
      S         STB,9      *ARRAY,7
            END


            SUBROUTINE INIT
      C
      C     INIT - INITIALISATION.  READ TABLES AND CONTROL CARDS.  SET UP CONTRACTION
      C        TABLE SEARCH POINTERS. SET UP STACK POINTERS, TABS ETC. INITIALISE
      C        VARIABLES.
      C
            IMPLICIT INTEGER(A-Z)
            COMMON /STAX/ TS,HS,CS,PS,TAILS(2),HEADS(2),CURRS(2),CURTYP,STKIND
           *,HFSTAK,ELT(19,30),SPCONT(19),NOELTS(19),PVSTAK(19),NXSTAK(19)
            COMMON /IN/ INPTR,PRIORS,SCOUNT
            COMMON /OUT/ OUTPTR,OUTL,LINES,LPG
            COMMON /TAB/ TABTYP(9),TABCLM(9),TABULA,LETR,LETD
            COMMON /OPTS/ EMBOPT,PFOUT
            COMMON /SIGNS/DOTS14(64),DOTS25(64),DOTS36(64),PCHAR(64),MCODE(64)
            COMMON /MEMB/ EMBOUT(11),MEMBCR,MEMBPG,MIDLE,MEMBON,MEMBOF
            COMMON /CPR/ CPRINT
            COMMON /PAG/ CDIGIT(4),DIGIT(10),PAGINA
            COMMON /SPAC/ SPACE
            COMMON /CTAB/ CTABLE(3,1000),ICLASS(1000),SIGNS(1000),
           *   BRANCH(1000),NCT,TSIGN(4)
            COMMON /ALPH/ SYMBOL(64),CCLASS(64),SINGSG(64),EXTENT(64),
           *   COMPB(64),ISOLET(64),NALPH
            COMMON /LC/ LCOUNT
            COMMON /LOGIC/ DECIS(15,25),COND(15,10),TRANS(10,25),NONTER(2),
           *   RCCLAS(4,2),NDTC,NIC,NNT
            COMMON /STV/ STVAR(10),NSV
            COMMON /RCC/ RCC
            LOGICAL EMBOPT,PFOUT,SEQERR,PAGINA,CPRINT
            COMMON /FILES/ TBDEV,INDEV,PRDEV,BRDEV
            DIMENSION LARRAY(10)
            DATA LETE,LETL,LETM,LETP,LETT /'E','L','M','P','T'/
            DATA CTMAX /1000/
      C
          1 FORMAT(80X)
          2 FORMAT(' ** TABLE SEQUENCE ERROR')
```

```
395              3 FORMAT(A1,71X)
396              4 FORMAT(A1,16X,3(I2,X),3X,I2)
397              5 FORMAT(A1,3A4,4X,2(I2,X),4I2)
398              6 FORMAT(17X,I2)
399              7 FORMAT(12X,A1,10X,4I2)
400              8 FORMAT(25A1)
401              9 FORMAT(NA1,NA1)
402             10 FORMAT(3A2,A3,A1)
403             11 FORMAT(A1,14X,5A1)
404             12 FORMAT(A1,30X,4I1)
405             13 FORMAT(' TOO MANY CONTRACTION TABLE ENTRIES')
406          C
407                INPTR = 81
408                SEQERR = .FALSE.
409          C
410          C     SET UP STACKS
411                DO 200 N=1,19
412                NOELTS(N) = 0
413                SPCONT(N) = 0
414                PVSTAK(N) = N - 1
415            200 NXSTAK(N) = N + 1
416                NXSTAK(19) = 0
417                NXSTAK(1) = 0
418                PVSTAK(2) = 0
419                HFSTAK = 3
420                CURTYP = 1
421                HEADS(1) = 1
422                TAILS(1) = 1
423                CURRS(1) = 1
424                HEADS(2) = 2
425                TAILS(2) = 2
426                CURRS(2) = 2
427                HS = 1
428                TS = 1
429                CS = 1
430                PVSTAK(3) = 0
431                NXSTAK(2) = 0
432           C
433                DO 600 N = 1,9
434                TABTYP(N) = LETL
```

```
435            600 TABCLM(N) = 2 * N + 1
436                LCOUNT = 0
437                OUTPTR = 1
438                TABLLA = 0
439                SCOUNT = 0
440                TEMPL = 0
441                STKIND = 2
442                CPRINT = .FALSE.
443                PRIORS = 0
444          C
445          C     READ ALPHABET TABLE
446                DO 800 N=1,65
447                EXTENT(N) = 999
448                READ(TBDEV,4) SYMBOL(N),CCLASS(N),CMPB(N),SINGSG(N),ISOLET(N)
449             800 IF (CCLASS(N) .EQ. 99) GO TO 1000
450            1000 NALPH = N
451                I = 1
452          C
453          C     FILL CONTRACTION TABLE
454                DO 1800 N=1,CTMAX
455                READ(TBDEV,5) LET1,(CTABLE(J,N),J=1,3),ICLASS(N),SHIFNO,
456               *    (TSIGN(J),J=1,4)
457          C     PACK SIGNS AND SHIFT COUNT INTO SIGNS(N)
458                SIGNS(N) = 0
459                DO 1100 J=4,1,-1
460            1100 SIGNS(N) = 100 * SIGNS(N) + TSIGN(J)
461                SIGNS(N) = 11 * SIGNS(N) + SHIFNO
462                IF (LET1 .EQ. TEMPL) GO TO 1400
463                TEMPL = LET1
464                EXTENT(I) = N
465                IF (SYMBOL(I) .EQ. TEMPL .OR. ICLASS(N) .EQ. 99) GO TO 1200
466                SEQERR = .TRUE.
467                WRITE(PRDEV,2)
468            1200 I = I + 1
469            1400 CONTINUE
470            1800 IF (ICLASS(N) .EQ. 99) GO TO 2000
471          C
472                WRITE(PRDEV,13)
473                STOP 'ERROR'
474          C
```

```
      2000 NCT = N - 1
           J = I - 1
           EXTENT(I) = EXTENT(J) + 1
           MAXEXT = J
C
C          CONTRACTION TABLE SEARCH SET-UP ALGORITHM
           T = NCT + 1
           DO 2200 I=1,T
      2200 BRANCH(I) = 0
           DO 4000 I=1,MAXEXT
           N = 1
           LOWNO = EXTENT(I)
           J = I + 1
           HIGHNO = EXTENT(J) - 1
           VHIGH = HIGHNO
      2600 II = LOWNO + 1
           IF (II .GT. HIGHNO) GO TO 3600
           IF (CHAR(CTABLE,LOWNO,N) .EQ. RCC) GO TO 3000
      2800 IF (II .GT. HIGHNO) GO TO 3600
           IF (CHAR(CTABLE,LOWNO,N) .NE. CHAR(CTABLE,II,N)) GO TO 3000
           II = II + 1
           GO TO 2800
      3000 LARRAY(N) = II
           BRANCH(LOWNO) = II
           BRANCH(II) = N
           N = N + 1
           LOWNO = LOWNO + 1
           HIGHNO = II - 1
      3200 IF (LOWNO .LE. HIGHNO) GO TO 2600
           N = N - 1
      3400 IF (N .EQ. 0) GO TO 4000
           IF (N .LE. 1) GO TO 3500
           HIGHNO = LARRAY(N-1) - 1
           GO TO 3200
      3500 HIGHNO = VHIGH
           GO TO 3200
      3600 IF (II .NE. LOWNO + 1) GO TO 3800
           BRANCH(LOWNO) = - BRANCH(II)
           LOWNO = LOWNO + 1
           BRANCH(II) = N
```

```
      GO TO 3400
 3800 LARRAY(N) = II
      N = N + 1
      BRANCH(LOWNO) = -N
      LOWNO = LOWNO + 1
      GO TO 2600
 4000 CONTINUE
C
C     READ RIGHT-CONTEXT TABLE
      READ(TBDEV,6) NNT
      DO 4200 N=1,NNT
 4200 READ(TBDEV,7) NONTER(N),(RCCLAS(I,N),I=1,4)
C
C     READ STATE TABLES
      READ(TBDEV,6) NSV
      READ(TBDEV,6) NIC
      DO 4400 I=1,NSV
 4400 READ(TBDEV,8) (TRANS(I,J),J=1,25)
      READ(TBDEV,6) NDTC
      DO 4600 N =1,NDTC
 4600 READ(TBDEV,9) NIC,(DECIS(N,I),I=1,NIC),NSV,(COND(N,I),I=1,NSV)
C
C     READ SIGN TABLE
      DO 5000 N = 1,64
 5000 READ(TBDEV,10) DOTS14(N),DOTS25(N),DOTS36(N),PCHAR(N),MCODE(N)
C
C     READ CONTROL CARDS
      READ(TBDEV,11) T,T1
      PFOUT = T .EQ. LETP
      IF (PFOUT .AND. T1 .EQ. LETT) CALL TABDIS
      READ(TBDEV,11) T,MEMBON,MEMBOF,MIDLE,MEMBCR,MEMBPG
      EMBOPT = T .EQ. LETM
      CALL STCHAR(MIDLE,EMBOUT,1)
      CALL STCHAR(MEMBON,EMBOUT,2)
      CALL STCHAR(MIDLE,EMBOUT,41)
      CALL STCHAR(MEMBOF,EMBOUT,42)
      READ(TBDEV,6) OUTL
      READ(TBDEV,6) LPG
      READ(TBDEV,12) T,(CDIGIT(I),I=1,4)
      PAGINA = T .EQ. LETP
```

```
      IF (SEQERR) STOP 'SEQUENCE ERROR'
      END


      SUBROUTINE TABDIS
C
C     TABDIS - TABLE DISPLAY FOR LOGIC TABLES
C
      IMPLICIT INTEGER(A-Z)
      COMMON /STV/ STVAR(10),NSV
      COMMON /LOGIC/ DECIS(15,25),COND(15,10),TRANS(10,25),NONTER(2),
     *   RCCLAS(4,2),NDTC,NIC,NNT
      COMMON /FILES/ TBDEV,INDEV,PRDEV,BRDEV
   10 FORMAT(///16X,'DECISION TABLE'//'-COLUMN')
   11 FORMAT(///16X,'RIGHT CONTEXT TABLE'//' NON-TERMINAL   INPUT CLASSE
     *S'//)
   12 FORMAT(6X,A1,8X,4(I2,2X))
   13 FORMAT(19X,20(2X,I2))
   14 FORMAT(/'-INPUT CLASS')
   15 FORMAT(17X,I2,3X,20(A1,3X))
   16 FORMAT(/'-STATE VARIABLE')
   17 FORMAT(///16X,'TRANSITION TABLE'//'-STATE VARIABLE')
   18 FORMAT(//)
      WRITE(PRDEV,11)
      DO 6200 K=1,NNT
 6200 WRITE(PRDEV,12) NONTER(K),(RCCLAS(I,K),I=1,4)
      WRITE(PRDEV,10)
      WRITE(PRDEV,13) (K,K=1,NDTC)
      WRITE(PRDEV,14)
      DO 6400 N=1,NIC
 6400 WRITE(PRDEV,15) N,(DECIS(K,N),K=1,NDTC)
      WRITE(PRDEV,16)
      DO 6600 N=1,NSV
 6600 WRITE(PRDEV,15) N,(COND(K,N),K=1,NDTC)
      WRITE(PRDEV,17)
      WRITE(PRDEV,13) (K,K=1,NSV)
      WRITE(PRDEV,14)
      DO 6800 N=1,NIC
 6800 WRITE(PRDEV,15) N,(TRANS(K,N),K=1,NSV)
      WRITE(PRDEV,18)
```

```
      END


      SUBROUTINE SHIFT(NCHARS)
C
C     SHIFT(NCHARS) - INPUT PROCESSOR.  OBTAINS INPUT FROM INPUT DEVICE, WITH
C        ECHO ON PROOF DEVICE IF PROOF OUTPUT REQUESTED. CONDENSES CONSECUTIVE
C        SPACES. REMOVES ALL OCCURENCES OF RCC CHARACTER (VERTICAL BAR). INSERTS
C        RCC CHARACTER AT END OF INPUT. SHIFTS BUFFER LEFT BY NCHARS AND FILLS
C        RIGHT HAND SIDE WITH RESULT OF ABOVE PROCESS.
C
      IMPLICIT INTEGER(A-Z)
      COMMON /IN/ INPTR,PRIORS,SCOUNT
      COMMON /OUT/ OUTPTR,OUTL,LINES,LPG
      COMMON /BUFF/ BUFFER(10)
      COMMON /OPTS/ EMBOPT,PFOUT
      LOGICAL EMBOPT,PFOUT
      COMMON /SPAC/ SPACE
      COMMON /CPR/ CPRINT
      LOGICAL CPRINT
      COMMON /RCC/ RCC
      COMMON /FILES/ TODEV,INDEV,PRDEV,BRDEV
      DIMENSION TEXT(20)
    1 FORMAT(20A4)
    2 FORMAT(X,20A4)
C
      IF (NCHARS .EQ. 0) RETURN
      DO 1400 I=1,NCHARS
      DO 50 N=1,9
   50 BUFFER(N) = BUFFER(N+1)
  100 IF (INPTR .LE. 80) GO TO 200
      IF (SCOUNT .GT. 0) GO TO 800
      READ(INDEV,1,END=800) TEXT
      INPTR = 1
      IF (.NOT. PFOUT) GO TO 200
      WRITE(PRDEV,2) TEXT
      CPRINT = .TRUE.
  200 NXTCHR = CHAR(TEXT,1,INPTR)
      INPTR = INPTR + 1
      IF (NXTCHR .NE. SPACE) GO TO 400
```

```
631               IF (PRIORS .EQ. 0) GO TO 100
632               PRIORS = PRIORS - 1
633               GO TO 600
634           400 PRIORS = 1
635           600 IF (NXTCHR .EQ. RCC) GO TO 100
636               GO TO 1200
637           800 SCOUNT = SCOUNT + 1
638               IF (SCOUNT .EQ. 1 .AND. PRIORS .GT. 0) GO TO 1000
639               NXTCHR = RCC
640               GO TO 1200
641          1000 NXTCHR = SPACE
642          1200 BUFFER(10) = NXTCHR
643          1400 CONTINUE
644               END


645               SUBROUTINE OPSIGN(OUTSIG)
646         C
647         C     OPSIGN(OUTSIG) - STACKER.  TAKES OUTSIG AS INPUT. CLEARS CURRENT STACK IF
648         C        REQUIRED (SPACE, END HEADING, END TITLE), CHANGES CURRENT STACK IF
649         C        REQUIRED (START OR END HEADING OR TITLE), CHANGES LOGICAL VARIABLES IF
650         C        REQUIRED (DOUBLE-SPACING, COMPUTER BRAILLE), OTHERWISE STORES SIGN ON
651         C        CURRENT STACK. INTERCHANGES LAST TWO ROWS OF STACK FOR UNITS OF MEASURE
652         C
653               IMPLICIT INTEGER(A-Z)
654               COMMON /STAX/ TS,HS,CS,PS,TAILS(2),HEADS(2),CURRS(2),CURTYP,STKIND
655              *,HFSTAK,ELT(19,30),SPCONT(19),NOELTS(19),PVSTAK(19),NXSTAK(19)
656               COMMON /OUT/ OUTPTR,OUTL,LINES,LPG
657               COMMON /PAG/ CDIGIT(4),DIGIT(10),PAGINA
658               COMMON /BUFF/ BUFFER(10)
659               COMMON /OPTS/ EMBOPT,PFOUT
660               COMMON /S/ LETS
661               COMMON /TTL/ TTLENG
662               COMMON /COMP/ COMPUT
663               COMMON /INDENT/ MARGIN,INHEAD
664               COMMON /TAB/ TABTYP(9),TABCLM(9),TABULA,LETR,LETD
665               COMMON /DUB/ DOUBLE
666               COMMON /FILES/ TBDEV,INDEV,PRDEV,BRDEV
667               LOGICAL PAGINA,EMBOPT,PFOUT,INHEAD,COMPUT,DOUBLE
668               DATA CCHIGH,SCHIGH/80,89/
```

```
669         C
670         C       CHARACTER TEST
671               IF (OUTSIG - 64) 800,400,200
672           200 IF (OUTSIG .LE. CCHIGH) GO TO 1400
673               IF (OUTSIG .LE. SCHIGH) GO TO 1800
674               GO TO 2000
675         C
676         C       OUTPUT SPACE
677           400 IF (STKIND .NE. 2) GO TO 600
678               CALL CLEAR
679               RETURN
680           600 CALL STAKTB
681               RETURN
682         C
683         C       OUTPUT ORDINARY CHARACTER
684           800 IF (NOELTS(CS) .NE. 30) GO TO 1000
685               CALL STAKTB
686               NOELTS(CS) = 1
687               GO TO 1200
688          1000 NOELTS(CS) = NOELTS(CS) + 1
689          1200 ELT(CS,NOELTS(CS)) = OUTSIG
690               RETURN
691         C
692         C       CARRIAGE CONTROL
693          1400 IF (NOELTS(CS) .NE. 0) CALL STAKTB
694               NOELTS(CS) = 30
695               SPCNT(CS) = OUTSIG
696               IF (OUTSIG .NE. 72) GO TO 1600
697               CALL DECODE(1,ELT(CS,3))
698               CALL SHIFT(1)
699          1500 IF (STKIND .EQ. 2) STKIND = 5
700               RETURN
701          1600 IF (OUTSIG .NE. 68 .AND. OUTSIG .NE. 71) RETURN
702          1700 CALL DECODE(2,ELT(CS,3))
703          1750 CALL SHIFT(2)
704               RETURN
705         C
706         C       STACK CONTROL
707          1800 T = OUTSIG - 80
708               GO TO (2400,2600,5000,1500,1900,4600,1900,3600,4000),T
```

```
      1300 RETURN
C
C     MODE CONTROL
 2000 IF (OUTSIG .EQ. 93) GO TO 5800
      IF (OUTSIG .EQ. 95) GO TO 4800
      IF (OUTSIG .EQ. 98) GO TO 5600
      IF (OUTSIG .EQ. 97) COMPUT = .NOT. COMPUT
      IF (OUTSIG .EQ. 30) DOUBLE = .NOT. DOUBLE
      RETURN
C
C     BEGIN HEADING
 2400 IF (NOELTS(CS) .NE. 0) CALL CLEAR
      CALL CARRIJ(65)
      STKIND = 4
      RETURN
C
C     END HEADING
 2600 N=HEADS(1)
      M = 0
 2800 M = M + NOELTS(N)
      IF (SPCONT(N) .EQ. 0 .AND. NOELTS(N) .GT. 0) M = M + 1
      IF (N .EQ. TAILS(1)) GO TO 3000
      N = NXSTAK(N)
      GO TO 2800
C
 3000 OUTPTR = (OUTL - M) / 2 + 2
      IF (OUTPTR .LT. 1) OUTPTR = 1
      STKIND = 2
      INHEAD = .TRUE.
      CALL CLEAR
      INHEAD = .FALSE.
      CALL CARRIJ(65)
      RETURN
C
C     BEGIN TITLE
 3600 CURTYP = 2
      HS = HEADS(2)
      TS = TAILS(2)
      CS = CURRS(2)
      STKIND = 4
```

```
749               M = HS
750               IF (NOELTS(M) .EQ. 0) RETURN
751         C     CLEAR OUT PREVIOUS TITLE
752               NOELTS(M) = 0
753          3800 IF (HS .EQ. TS) RETURN
754               CALL FRSTAK(PVSTAK,NXSTAK,TS,TAILS)
755               CURRS(2) = HEADS(2)
756               GO TO 3800
757         C
758         C     END TITLE
759          4000 TTLENG = 0
760               N = HEADS(2)
761          4200 TTLENG = TTLENG + NOELTS(N)
762               IF (SPCONT(N) .EQ. 0 .AND. NOELTS(N) .GT. 0) TTLENG = TTLENG + 1
763               IF (N .EQ. TAILS(2)) GO TO 4400
764               N = NXSTAK(N)
765               GO TO 4200
766         C
767          4400 CURTYP = 1
768               HS = HEADS(1)
769               TS = TAILS(1)
770               CS = CURRS(1)
771               STKIND = 2
772               RETURN
773         C
774         C     UNIT OF MEASURE
775          4600 SPCONT(CS) = 1
776         C
777         C     INTERCHANGE LAST TWO ROWS OF CURRENT STACK
778               N = PVSTAK(TS)
779               IF (N .EQ. 0) GO TO 4700
780               M = PVSTAK(N)
781               PVSTAK(TS) = M
782               NXSTAK(TS) = N
783               PVSTAK(N) = TS
784               NXSTAK(N) = 0
785               TS = N
786               TAILS(CURTYP) = N
787               CS = N
788               CURRS(CURTYP) = N
```

```
789               IF (N .NE. HS) GO TO 4650
790               HS = PVSTAK(N)
791               HEADS(CURTYP) = HS
792               GO TO 4700
793          4650 NXSTAK(N) = PVSTAK(N)
794             C
795          4700 IF (BUFFER(1) .EQ. LETS) CALL SHIFT(1)
796               RETURN
797             C
798             C     MARGIN RESET
799          4800 CALL DECODE(2,MARGIN)
800               CALL SHIFT(2)
801               RETURN
802             C
803             C     RESTART
804          5000 DO 5200 N=1,3
805          5200 CDIGIT(N) = 0
806               CDIGIT(4) = 1
807               CALL RESET
808               RETURN
809             C
810          5600 CALL CARRIJ(65)
811               ENDFILE BRDEV
812               ENDFILE PRDEV
813               STOP 'OK'
814          5800 CALL DECODE(1,T)
815               TABTYP(T) = BUFFER(2)
816               CALL SHIFT(2)
817               CALL DECODE(2,TABCLM(T))
818               CALL SHIFT(2)
819               RETURN
820               END


821               SUBROUTINE CARRIJ(OUTSIG)
822             C
823             C     CARRIJ(OUTSIG) = CARRIAGE CONTROL.
824             C
825               IMPLICIT INTEGER(A-Z)
826               COMMON /OUT/ OUTPTR,OUTL,LINES,LPG
```

```
827             COMMON /LC/ LCOUNT
828             COMMON /DUB/ DOUBLE
829             LOGICAL DOUBLE
830           C
831             IF (LCOUNT .GE. LPG) CALL PAGEHD
832             IF (OUTPTR .EQ. 1 .AND. (OUTSIG .EQ. 65 .OR. OUTSIG .EQ. 67))
833            *   GO TO 600
834             CALL OUTOUT(OUTSIG)
835             IF (DOUBLE .AND. LCOUNT .LT. LPG) CALL OUTOUT(OUTSIG)
836          600 IF (OUTSIG .NE. 67) GO TO 800
837             OUTPTR = 3
838             RETURN
839          800 OUTPTR = 1
840             END


841             SUBROUTINE CLEAR
842           C
843           C     CLEAR - CLEAR CURRENT STACK: LINE COMPOSER. ROWS OF CURRENT STACK ARE SENT
844           C        TO OUTPUT BUFFER. PERFORMS TABULATION OR SKIP-LINES WHERE CALLED FOR.
845           C
846             IMPLICIT INTEGER(A-Z)
847             COMMON /STAX/ TS,HS,CS,PS,TAILS(2),HEADS(2),CURRS(2),CURTYP,STKIND
848            *,FFSTAK,ELT(19,30),SPCONT(19),NOELTS(19),PVSTAK(19),NXSTAK(19)
849             COMMON /OUT/ OUTPTR,OUTL,LINES,LPG
850             COMMON /TAB/ TABTYP(9),TABCLM(9),TABULA,LETR,LETD
851             COMMON /INDENT/ MARGIN,INHEAD
852             LOGICAL INHEAD
853           C
854          100 PS = HS
855             NPS = NOELTS(PS)
856          400 IF (SPCONT(PS) .LT. 64) GO TO 3000
857             OUTSIG = SPCONT(PS)
858             EP3 = ELT(PS,3)
859             IF (OUTSIG .NE. 68) GO TO 600
860           C     ONE-TIME TABULATION
861             TABULA = EP3
862             GO TO 4100
863          600 IF (OUTSIG .NE. 71) GO TO 1800
864           C     SKIP LINES
```

```
      CALL CARRIJ(65)
      CALL PCLEAR
      DO 1600 M=1,EP3
      OUTPTR = 0
 1600 CALL CARRIJ(71)
      GO TO 4200
C
C     PRESET TABULATION
 1800 IF (OUTSIG .NE. 72) GO TO 2800
      M = ELT(PS,3)
      XX = NXSTAK(PS)
      IF (TABTYP(M) .EQ. R) GO TO 2000
      IF (TABTYP(M) .EQ. D) GO TO 2200
      TABULA = TABCLM(M)
      GO TO 4100
C
 2000 TABULA = TABCLM(M) - NOELTS(XX) + 1
      GO TO 4100
C
 2200 T = NOELTS(XX)
      DO 2400 XYZ=1,T
 2400 IF (ELT(XX,XYZ) .EQ. 40) GO TO 2600
 2600 TABULA = TABCLM(M) - XYZ + 1
      GO TO 4100
C
 2800 CALL CARRIJ(OUTSIG)
      GO TO 4200
C
 3000 IF (NPS .EQ. 0) GO TO 4200
      IF (OUTPTR + NPS .LT. OUTL + 2) GO TO 3800
      OUTPTR = OUTL + 1
      CALL CARRIJ(0)
      IF (MARGIN .LE. 0) GO TO 3400
      DO 3200 N=1,MARGIN
 3200 CALL MOVEEL(64)
 3400 IF (.NOT. INHEAD) GO TO 3800
      DO 3600 N=1,3
 3600 CALL MOVEEL(64)
 3800 DO 4000 M=1,NPS
 4000 CALL MOVEEL(ELT(PS,M))
```

```
      IF (SPCONT(PS) .NE. 1) CALL MOVEEL(64)
      GO TO 4200
C
C     TABULATION
 4100 TABULA = TABULA - OUTPTR
      IF (TABULA .LE. 0) GO TO 4120
      DO 4110 N=1,TABULA
 4110 CALL MOVEEL(64)
      GO TO 4140
 4120 TABULA = TABULA + OUTPTR - 1
      CALL CARRIJ(65)
      DO 4130 N=1,TABULA
 4130 CALL MOVEEL(64)
 4140 TABULA = 0
C
 4200 IF (HS .EQ. TS) GO TO 4800
      CALL FRSTAK(NXSTAK,PVSTAK,HS,HEADS)
      GO TO 100
 4800 NOELTS(HS) = 0
      SPCONT(HS) = 0
      CURRS(CURTYP) = HS
      CS = HS
      END


      SUBROUTINE FRSTAK(STAK,STAK1,HT,HTS)
C
C     FRSTAK(STAK,STAK1,HT,HTS) - FREE-STACK. SETS STACK POINTERS TO TRANSFER
C        LAST ROW OF CURRENT STACK TO FREE STACK.
C
      IMPLICIT INTEGER(A-Z)
      COMMON /STAX/ TS,HS,CS,PS,TAILS(2),HEADS(2),CURRS(2),CURTYP,STKIND
     *,HFSTAK,ELT(19,30),SPCONT(19),NOELTS(19),PVSTAK(19),NXSTAK(19)
      DIMENSION STAK(19),STAK1(19),HTS(2)
C
      NOELTS(HT) = 0
      SPCONT(HT) = 0
      IF (STAK(HT) .EQ. 0) RETURN
      T = HT
      N = HFSTAK
```

```
      HT = STAK(M)
      HTS(CURTYP) = HT
      HFSTAK = M
      STAK1(HT) = 0
      NXSTAK(M) = N
      END


      SUBROUTINE MOVEEL(X)
C
C     MOVEEL(X) - MOVE ELEMENT.  TRANSFERS THE SIGN WHOSE CODE IS X TO THE
C        OUTPUT BUFFERS.
C
      IMPLICIT INTEGER(A-Z)
      COMMON /STAX/ TS,HS,CS,PS,TAILS(2),HEADS(2),CURRS(2),CURTYP,STKIND
     *,HFSTAK,ELT(19,30),SPCONT(19),NOELTS(19),PVSTAK(19),NXSTAK(19)
      COMMON /OUT/ OUTPTR,OUTL,LINES,LPG
      COMMON /SIGNS/DOTS14(64),DOTS25(64),DOTS36(64),PCHAR(64),MCODE(64)
      COMMON /OWAREA/ BRAILL(3,40),PROOF(40),CODE(40)
      COMMON /OPTS/ EMBOPT,PFOUT
      LOGICAL EMBOPT,PFOUT
C
      CODE(OUTPTR) = X
      IF (.NOT. PFOUT) GO TO 200
      IF (X .EQ. 0) X = 64
      BRAILL(1,OUTPTR) = DOTS14(X)
      BRAILL(2,OUTPTR) = DOTS25(X)
      BRAILL(3,OUTPTR) = DOTS36(X)
      PROOF(OUTPTR) = PCHAR(X)
  200 OUTPTR = OUTPTR + 1
      END


      SUBROUTINE OUTOUT(OUTSIG)
C
C     OUTOUT(OUTSIG) - OUTPUTS CONTENTS OF AND CLEARS OUTPUT BUFFERS.
C
      IMPLICIT INTEGER(A-Z)
      COMMON /OUT/ OUTPTR,OUTL,LINES,LPG
      COMMON /OPTS/ EMBOPT,PFOUT
```

```
      LOGICAL EMBOPT,PFOUT
      COMMON /SIGNS/DOTS14(64),DOTS25(64),DOTS36(64),FCHAR(64),MCODE(64)
      COMMON /OWAREA/ BRAILL(3,40),PROOF(40),CODE(40)
      COMMON /MEMB/ EMBOUT(11),MEMBCR,MEMBPG,MIDLE,MEMBON,MEMBOF
      COMMON /CPR/ CPRINT
      LOGICAL CPRINT
      COMMON /LC/ LCOUNT
      COMMON /FILES/ TBDEV,INDEV,PRDEV,BRDEV
    1 FORMAT(80X)
    2 FORMAT(X,40A3)
    3 FORMAT(40(X,I2))
    4 FORMAT(20A4)
C
      IF (.NOT. PFOUT) GO TO 400
      IF (.NOT. CPRINT) WRITE(PRDEV,1)
      CPRINT = .FALSE.
      DO 200 N=1,3
  200 WRITE(PRDEV,2) (BRAILL(N,J),J=1,OUTL)
      WRITE(PRDEV,2) PROOF
      WRITE(PRDEV,3) (CODE(J),J=1,OUTL)
  400 IF (.NOT. EMBOPT) GO TO 1400
      DO 600 N=1,39
      IF (N .GT. OUTL) GO TO 800
      T = CODE(N)
      IF (T .EQ. 0) T = 64
      CALL STCHAR(MCODE(T),EMBOUT,N+2)
  600 CONTINUE
  800 IF (OUTL .GE. 39) GO TO 1200
      DO 1000 N = N,39
 1000 CALL STCHAR(MIDLE,EMBOUT,N+2)
 1200 IF (OUTL .GE. 38) GO TO 1300
      N = OUTL + 1
      CALL STCHAR(MEMBCR,EMBOUT,N+2)
 1300 CONTINUE
      WRITE(BRDEV,4) EMBOUT
 1400 CALL PCLEAR
      IF (OUTSIG .NE. 69) GO TO 1800
      LCOUNT = LPG
      RETURN
 1800 LCOUNT = LCOUNT + 1
```

```
      END


      SUBROUTINE PAGEHD
C
C     PAGEHD - PAGE HEADING.  IF PAGINATION REQUESTED:
C        COPIES CONTENTS OF OUTPUT BUFFERS TO TEMPORARY STORAGE. SETS UP TITLE
C        IF ANY AND PAGE NUMBER IN OUTPUT BUFFER. CALLS OUTOUT. RESTORES
C        ORIGINAL OUTPUT BUFFERS. INCREMENTS PAGE NUMBER.
C
      IMPLICIT INTEGER(A-Z)
      COMMON /STAX/ TS,HS,CS,PS,TAILS(2),HEADS(2),CURRS(2),CURTYP,STKIND
     *,HFSTAK,ELT(19,30),SPCONT(19),NOELTS(19),PVSTAK(19),NXSTAK(19)
      COMMON /OUT/ OUTPTR,OUTL,LINES,LPG
      COMMON /OWAREA/ BRAILL(3,40),PROOF(40),CODE(40)
      COMMON /TTL/ TTLENG
      COMMON /LC/ LCOUNT
      COMMON /PAG/ CDIGIT(4),DIGIT(10),PAGINA
      COMMON /OPTS/ EMBOPT,PFOUT
      LOGICAL PFOUT,EMBOPT
      COMMON /MEMB/ EMBOUT(11),MEMBCR,MEMBPG,MIDLE,MEMBON,MEMBOF
      COMMON /CPR/ CPRINT
      COMMON /DUB/ DOUBLE
      LOGICAL PAGINA,CPRINT,DOUBLE
      COMMON /FILES/ TBDEV,INDEV,PRDEV,BRDEV
      DIMENSION BTEMP(3,40),CTEMP(40),PTEMP(40)
    1 FORMAT(80X)
    2 FORMAT(11A4)
C
      IF (.NOT. PAGINA) RETURN
      IF (PFOUT) WRITE(PRDEV,1)
      IF (.NOT. EMBOPT) GO TO 100
      CALL STCHAR(MEMBPG,EMBOUT,3)
      DO 50 N=4,41
   50 CALL STCHAR(MIDLE,EMBOUT,N)
      WRITE(BRDEV,2) EMBOUT
  100 LCOUNT = 0
      CPRINT = .FALSE.
      N = (OUTL - TTLENG + 2) / 2 + 1
      IF (N .LT. 1) N = 1
```

```
1057          C       SAVE CONTENTS OF OUTPUT BUFFER
1058                  DO 200 I=1,OUTL
1059                  PTEMP(I) = PROOF(I)
1060                  CTEMP(I) = CODE(I)
1061                  DO 200 J=1,3
1062            200   BTEMP(J,I) = BRAILL(J,I)
1063                  CALL PCLEAR
1064                  OSAVE = OUTPTR
1065                  OUTPTR = N
1066                  N = HEADS(2)
1067            400   IF (NOELTS(N) .EQ. 0 .OR. SPCONT(N) .GE. 64) GO TO 800
1068                  I = NOELTS(N)
1069                  DO 600 XYZ = 1,I
1070            600   CALL MOVEEL(ELT(N,XYZ))
1071                  IF (SPCONT(N) .EQ. 0) CALL MOVEEL(64)
1072            800   IF (N .EQ. TAILS(2)) GO TO 1000
1073                  N = NXSTAK(N)
1074                  IF (OUTPTR + NOELTS(N) .LE. OUTL - 5) GO TO 400
1075           1000   OUTPTR = OUTL - 4
1076                  DO 1200 N=1,4
1077                  IF (CDIGIT(N) .NE. 0) GO TO 1400
1078           1200   OUTPTR = OUTPTR + 1
1079           1400   CALL MOVEEL(60)
1080                  DO 1600 N=1,4
1081                  XYZ = CDIGIT(N) + 1
1082           1600   CALL MOVEEL(DIGIT(XYZ))
1083          C
1084          C       INCREMENT PAGE NUMBER
1085                  DO 1800 N=4,1,-1
1086                  CDIGIT(N) = CDIGIT(N) + 1
1087                  IF (CDIGIT(N) .LT. 10) GO TO 2000
1088           1800   CDIGIT(N) = 0
1089           2000   CALL OUTOUT(0)
1090                  IF (DOUBLE) CALL OUTOUT(0)
1091          C       RESTORE OUTPUT BUFFER
1092                  OUTPTR = OSAVE
1093                  DO 2200 I=1,OUTL
1094                  CODE(I) = CTEMP(I)
1095                  PROOF(I) = PTEMP(I)
1096                  DO 2200 J=1,3
```

```
      220 BRAILL(J,I) = BTEMP(J,I)
          END


          SUBROUTINE STAKTB
      C
      C     STAKTB - SETS STACK POINTERS TO TRANSFER A ROW FROM FREE STACK TO END OF
      C        CURRENT STACK.
      C
          IMPLICIT INTEGER(A-Z)
          COMMON /STAX/ TS,HS,CS,PS,TAILS(2),HEADS(2),CURRS(2),CURTYP,STKIND
         *,HFSTAK,ELT(19,30),SPCONT(19),NOELTS(19),PVSTAK(19),NXSTAK(19)
      C
          IF (HFSTAK .EQ. 0) RETURN
          N = HFSTAK
          HFSTAK = NXSTAK(N)
          NXSTAK(N) = 0
          PVSTAK(N) = TS
          NXSTAK(TS) = N
          TAILS(CURTYP) = N
          TS = N
          CURRS(CURTYP) = N
          CS = N
          IF (STKIND .EQ. 5) STKIND = 2
          END


          SUBROUTINE PCLEAR
      C
      C     PCLEAR - RESETS PROOF OUTPUT BUFFERS TO SPACES.
      C
          IMPLICIT INTEGER(A-Z)
          COMMON /OWAREA/ BRAILL(3,40),PROOF(40),CODE(40)
          COMMON /SPAC/ SPACE
      C
          DO 200 I=1,40
          CODE(I) = 0
          PROOF(I) = SPACE
          DO 200 J=1,3
      200 BRAILL(J,I) = SPACE
```

```
1133            END


1134            SUBROUTINE RESET
1135      C
1136      C     RESET - CALLED WHEN ST APPEARS IN INPUT. RESETS PAGE NUMBER, TITLE, AND
1137      C        LOGICAL VARIABLES FOR START OF NEW DOCUMENT.
1138      C
1139            IMPLICIT INTEGER(A-Z)
1140            COMMON /STV/ STVAR(10),NSV
1141            COMMON /TTL/ TTLENG
1142            COMMON /N/ LETN
1143            COMMON /INDENT/ MARGIN,INHEAD
1144            COMMON /COMP/ COMPUT
1145            COMMON /DUB/ DOUBLE
1146            LOGICAL DOUBLE,INHEAD,COMPUT
1147      C
1148            TTLENG = 0
1149            MARGIN = 0
1150            DOUBLE = .FALSE.
1151            COMPUT = .FALSE.
1152            INHEAD = .FALSE.
1153            DO 100 I=1,NSV
1154         100 STVAR(I) = LETN
1155            END


1156            SUBROUTINE DECODE(N,Y)
1157      C
1158      C     DECODE - DECODES CHARACTER REPRESENTATION OF INTEGER. USED IN SKIP-LINES
1159      C        AND SETTING TABS AND MARGIN.
1160      C
1161            IMPLICIT INTEGER(A-Z)
1162            COMMON /BUFF/ BUFFER(10)
1163      C
1164            Y = 0
1165            DO 200 I=1,N
1166         200 Y = 10 * Y + IAND(15,ICH(BUFFER,4*I-3))
1167            END
```

```
C
C                    RJE TAPE FILE HANDLING PROGRAM
C
C
C  PERFORMS ONE OF THE FOLLOWING FUNCTIONS DEPENDING ON THE PROGRAM NUMBER
C  (INTEGER ON THE FIRST CARD OF THE INPUT)
C  1  TAKES CONTINUOUS INPUT FROM DEVICE F:105 (CARD READER) AND PRODUCES A
C     LINE BY LINE RJE FILE IMAGE ON DEVICE F:119 (DISC FILE) FROM WHICH AN
C     RJE FILE MAY BE CREATED USING !RJBUILD
C  2  LISTS RJE FILES ON DEVICE F:108 (LINEPRINTER)
C  3  CONVERTS RJE FILES INTO A SUITABLE FORMAT FOR INPUT TO DOTSYS III,
C     OUTPUT ON DEVICE F:121 (TAPE OR DISC FILE)
C  4  COPIES RJE FILES TO ARCHIVE TAPE
C  5  RJE CONVERSION FOLLOWED BY DOTSYS (FORTRAN IMPROVED VERSION)
C  6  COPIES RJE FILE "BRAILLE" TO FGDTEMP FOR EMBOSSING
C  7  DOTSYS (FORTRAN IMPROVED VERSION) WITHOUT RJE CONVERSION
C
      IMPLICIT INTEGER(A-Z)
      COMMON /SUB/ SUBNO,VERSNO
      COMMON /INDEX/ INDEX(62)
    1 FORMAT(2I)
    2 FORMAT(' ERROR IN PROGRAM NUMBER')
    4 FORMAT(' TYPE CORRECT NUMBER')
      READ(104,1) SUBNO,VERSNO
      IF (SUBNO .GT. 1 .AND. SUBNO .LT. 7)
     *   CALL BUFFERIN(120,1,INDEX,62,ISTAT)
      IF (SUBNO .LE. 0 .OR. SUBNO .GE. 8) GO TO 200
      IF (SUBNO .EQ. 7) GO TO 160
  150 CALL SEGLOD(1)
      IF (SUBNO .EQ. 1) CALL CARDS
      IF (SUBNO .EQ. 2) CALL LIST
      IF (SUBNO .EQ. 3) CALL RJECON
      IF (SUBNO .EQ. 4) CALL SAVE
      IF (SUBNO .NE. 5) GO TO 170
      CALL RJECON
  160 CALL SEGLOD(2)
      CALL DOTSYS
      GO TO 180
  170 IF (SUBNO .EQ. 6) CALL EMBOSS
  180 CONTINUE
      STOP 'OK'
```

```
      200 WRITE(115,2)
          WRITE(115,4)
     S       CALL 9,9          WAIT
          READ(115,1) SUBN0,VERSN0
          IF (SUBN0 .GE. 1 .AND. SUBN0 .LE. 7) GO TO 150
          STOP 'ERROR'
          END


          SUBROUTINE CARDS
     C    RJE FILE FROM CARDS.  MACRO FACILITY AVOIDS REPETITIOUS TYPING:
     C    MACRO DEFINITION FORMAT:  @@<CHARACTER-PAIR>=<TEXT>@
     C    MACRO CALL FORMAT:  @<CHARACTER-PAIR>
          IMPLICIT INTEGER(A-Z)
          COMMON /SUB/ SUBN0,VERSN0
          COMMON /RDCH/ ENDRED,INTEMP(20),INPUT(20),INPTR,CHAR
         *   ,INLENG,SPACE
          COMMON /EB/ EBCEXT(256)
          DIMENSION EBCDIC(255)
          EQUIVALENCE(EBCDIC,EBCEXT(2))
          DIMENSION MACTAB(1000),MACSIZ(100),MCBASE(100),MCCHAR(100)
          LOGICAL INMAC,ENDRED,WARN
          DIMENSION OUTPUT(80)
          DATA MTSIZ,NMMAX /4000,100/
          DATA MACSYM,EQUALS/2Z7C,2Z7E/
          DATA SPACE,LSPACE,H1,H2/2Z40,8Z40000000,2Z80,2ZC0/
          DATA INLENG,OLENG/80,72/
        1 FORMAT(80A1)
        2 FORMAT(X,80A1)
       10 FORMAT(' SEE LP')
       11 FORMAT(20X,'MACRO REDEFINED',A4)
       12 FORMAT(' TOO MANY MACROS')
       13 FORMAT(' MISSING ''=''  SIGN')
       14 FORMAT(' MACRO TABLE FULL')
       15 FORMAT(20X,'UNDEFINED MACRO IGNORED',A4)
     C    OLENG = MAXIMUM LENGTH OF OUTPUT LINE (< 80 CHARACTERS)
          INMAC = .FALSE.
          ENDRED = .FALSE.
          WARN = .FALSE.
          MTPTR = 0
```

```
      LAST = SPACE
      INPTR = INLENG
      OPTR = 0
      LINEND = OLENG
      DO 100 I=1,256
  100 EBCEXT(I) = I - 1
      IF (VERS .NE. 1) GO TO 140
C
C     VERSION NUMBER 1 INDICATES CODE CONVERSION SPECIFIED
  110 CALL BUFFEDIN(104,1,INTEMP,20,ISTAT)
      DO 120 I=1,20
      IF (ICH(INTEMP,4*I-2) .NE. EQUALS .OR. ICH(INTEMP,4*I) .NE. SPACE)
     *    GO TO 140
  120 EBCDIC(ICH(INTEMP,4*I-3)) = ICH(INTEMP,4*I-1)
      GO TO 110
  140 CALL BUFFIN(105,1,INTEMP,20)
C
  150 IF (ENDRED) GO TO 500
      IF (INMAC) GO TO 860
      CALL RDCHAR
      IF (ENDRED) CHAR = SPACE
      IF (CHAR .EQ. SPACE .AND. LAST .EQ. SPACE) GO TO 150
      IF (CHAR .EQ. MACSYM) GO TO 1000
  270 OPTR = OPTR + 1
      OUTPUT(OPTR) = CHAR
      LAST = CHAR
      IF (CHAR .EQ. SPACE) LINEND = OPTR - 1
      IF (OPTR .GT. OLENG) GO TO 300
      GO TO 150
  300 REST = 80 - LINEND
      DO 350 I=1,LINEND
  350 OUTPUT(I) = ISL(OUTPUT(I),24)
      WRITE(119,1) (OUTPUT(I),I=1,LINEND),(LSPACE,I=1,REST)
      MOVENO = OLENG - LINEND
      IF (MOVENO .EQ. 0) GO TO 450
      DO 400 I=1,MOVENO
  400 OUTPUT(I) = OUTPUT(LINEND+I+1)
  450 OPTR = MOVENO
      GO TO 150
```

```
120           500 REST = 80 - LINEND
121               DO 550 I=1,LINEND
122           550 OUTPUT(I) = ISL(OUTPUT(I),24)
123               WRITE(119,1) (OUTPUT(I),I=1,LINEND),(LSPACE,I=1,REST)
124               ENDFILE 119
125               IF (WARN) WRITE(115,10)
126               STOP 'OK'
127           700 STOP 'MACRO ERROR'
128         C
129           800 COUNT = COUNT + 1
130               IF (COUNT .LE. MCSIZ) GO TO 880
131               INMAC = .FALSE.
132               GO TO 150
133           880 CHAR = ICH(MACTAB,MCBAS+COUNT)
134               GO TO 270
135         C
136          1000 CALL RDCHAR
137               IF (CHAR .EQ. MACSYM) GO TO 2000
138         C
139         C     MACRO CALL
140               T = CHAR
141               CALL RDCHAR
142               CHAR = 256 * T + CHAR
143               DO 1100 I=1,NMACS
144          1100 IF (CHAR .EQ. MCCHAR(I)) GO TO 1200
145               WRITE(108,15) CHAR
146               WARN = .TRUE.
147               GO TO 150
148          1200 MCSIZ = MACSIZ(I)
149               MCBAS = MCBASE(I)
150               COUNT = 0
151               INMAC = .TRUE.
152               GO TO 150
153         C
154         C     MACRO DEFINITION
155          2000 CALL RDCHAR
156               IF (NMACS .EQ. 0) GO TO 2200
157               DO 2100 I=1,NMACS
158               IF (CHAR .NE. MCCHAR(I)) GO TO 2100
```

```
159               WRITE(108,11) MCCHAR(I)
160               MCCHAR(I) = 0
161               WARN = .TRUE.
162               GO TO 2200
163            2100 CONTINUE
164            2200 NMACS = NMACS + 1
165               IF (NMACS .LE. NMMAX) GO TO 2300
166               WRITE(115,12)
167               GO TO 700
168            2300 T = CHAR
169               CALL RDCHAR
170               MCCHAR(NMACS) = 256 * T + CHAR
171               CALL RDCHAR
172               IF (CHAR .EQ. EQUALS) GO TO 2400
173               WRITE(115,13)
174               GO TO 700
175            2400 MCBASE(NMACS) = MTPTR
176            2500 CALL RDCHAR
177               IF (CHAR .EQ. MACSYM) GO TO 2700
178               MTPTR = MTPTR + 1
179               IF (MTPTR .LE. MTSIZ) GO TO 2600
180               WRITE(115,14)
181               GO TO 700
182            2600 CONTINUE
183           C      ICH(MACTAB,MTPTR) = CHAR
184           S         LW,7       MTPTR
185           S         AI,7       -1
186           S         LW,9       CHAR
187           S         STB,9      MACTAB,7
188               GO TO 2500
189            2700 MACSIZ(NMACS) = MTPTR - MCBASE(NMACS)
190               GO TO 150
191               END


192               SUBROUTINE RDCHAR
193           C   READS NEXT CHARACTER FROM INPUT  (CALLED BY CARDS)
194               IMPLICIT INTEGER(A-Z)
195               COMMON /RDCH/ ENDRED,INTEMP(20),INPUT(20),INPTR,CHAR
196              *    ,INLENG,SPACE
```

```
197             COMMON /EB/ EBCEXT(256)
198             DIMENSION EBCDIC(255)
199             EQUIVALENCE(EBCDIC,EBCEXT(2))
200             LOGICAL ENDRED
201             DATA H1,H2/2Z80,2ZC0/
202           1 FORMAT(' ILLEGAL INPUT CHARACTER')
203             INPTR = INPTR + 1
204             IF (INPTR .LE. INLENG) GO TO 200
205            80 IF (ICHECK(105) .EQ. 1) GO TO 80
206             IF (ICHECK(105) .NE. 3) GO TO 100
207             ENDRED = .TRUE.
208             RETURN
209           100 DO 120 I=1,20
210           120 INPUT(I) = INTEMP(I)
211             CALL BUFFIN(105,1,INTEMP,20)
212             INPTR = 1
213           200 CHAR = EBCDIC(ICH(INPUT,INPTR))
214             IF (CHAR.LT.SPACE .OR. CHAR.GE.H1 .AND. CHAR.LE.H2) GO TO 600
215             RETURN
216           600 WRITE(115,1)
217         S         CALL9   9             WAIT
218             END


219             SUBROUTINE LIST
220         C   LISTS ANY NUMBER OF FILES IN THE ORDER IN WHICH THEY OCCUR ON THE TAPE
221             IMPLICIT INTEGER(A-Z)
222             EXTERNAL FLIST
223             CALL RDLOC
224             CALL FCOPY(FLIST)
225             END


226             SUBROUTINE RJECON
227         C   CONVERTS ANY NUMBER OF FILES IN THE ORDER IN WHICH THEY OCCUR ON THE TAPE
228             IMPLICIT INTEGER(A-Z)
229             DIMENSION TABLES(2),ENDLIN(20)
230             COMMON /BUFF/ BUFFER(20)
231             COMMON /NAMES/ NAMTAB(20,2),FPOSN(20),NAME(2),NFILES
232             COMMON /INDEX/ INDEX(62)
```

```
      COMMON /F/ FILNO
      COMMON /C/ NLINES,LOGTAB
      COMMON /SUB/ SUBNO,VERSNO
      EQUIVALENCE (NFOT,INDEX(4))
      EXTERNAL CCOPY
      DATA TABLES/'TABL','ES8('/
      DATA SPACE,SPACES,OLD/2Z40,'    ','OLD '/
      DATA ENDLIN / '$T  ',19*'    ' /
    1 FORMAT(' RJE TAPE FULL')
    2 FORMAT(//X,I,'CARD IMAGES WRITTEN'/)
      NLINES = 0
      FILNO = -1
      OUTPUT 'RJE CONVERSION'
      CALL DATE
      IF (SUBNO .NE. 5 .OR. NFOT .LT. 19) GO TO 500
      WRITE(115,1)
S        CAL1,9    3           ABORT
  500 IF (SUBNO .NE. 5) CALL BUFFERIN(105,1,NEWTAB,1,ISTAT)
      CALL RDLOC
      IF (SUBNO .EQ. 5 .OR. NEWTAB .NE. OLD) GO TO 630
  620 CALL BUFFERIN(121,1,BUFFER,20,ISTAT)
      IF (BUFFER(20) .NE. SPACES) GO TO 620
      GO TO 700
C     LOCATE AND COPY TABLES
  630 CALL LOCATE(TABLES,TABPOS)
      IF (TABPOS .EQ. 0) STOP 'TABLES NOT ON TAPE'
      CALL SKIPFILE(120,TABPOS)
      DO 650 I=1,2
  650 NAME(I) = TABLES(I)
      CALL CCOPY
      IF (SUBNO .EQ. 5) CALL COPBAK(NFOT-TABPOS+1,.TRUE.)
  700 REWIND 120
      FILNO = 0
      CALL FCOPY(CCOPY)
      CALL BUFFEROUT(121,1,ENDLIN,20,ISTAT)
      IF (ISTAT .NE. 2) STOP 'OUTPUT ERROR'
      IF (SUBNO .EQ. 5) CALL COPBAK(1,.FALSE.)
      ENDFILE 121
      WRITE(108,2) NLINES
      END
```

```
      SUBROUTINE COPBAK(SKIPNO,LOGTAB)
C     COPIES DISC FILE BACK TO TAPE  (CALLED BY REJECON)
      IMPLICIT INTEGER(A-Z)
      COMMON /BUFF/ BUFFER(20)
      LOGICAL LOGTAB
      DATA SPACES / '    ' /
      ENDFILE 121
      REWIND 121
      CALL SKIPFILE(120,SKIPNO)
      IF (LOGTAB) GO TO 200
  100 CALL BUFFERIN(120,1,BUFFER,20,ISTAT)
      IF (BUFFER(20) .NE. SPACES) GO TO 100
  200 CALL BUFFERIN(121,1,BUFFER,20,ISTAT)
      IF (ISTAT .EQ. 3) GO TO 400
      CALL BUFFEROUT(120,1,BUFFER,20,ISTAT)
      GO TO 200
  400 ENDFILE 120
      REWIND 121
      IF (.NOT. LOGTAB) CALL SKIPFILE(120,-1)
      END


      SUBROUTINE RDLOC
C     READ FILE NAMES AND LOCATE IN INDEX
      IMPLICIT INTEGER(A-Z)
      COMMON /NAMES/ NAMTAB(20,2),FPOSN(20),NAME(2),NFILES
    1 FORMAT(X,2A4,' NOT ON TAPE')
    4 FORMAT(2A4)
      DO 200 I=1,20
  200 FPOSN(I) = 0
      DO 300 I=1,20
  300 READ(105,4,END=400) (NAMTAB(I,J),J=1,2)
  400 NFILES = I - 1
      IF (NFILES .EQ. 0) RETURN
      DO 600 I=1,NFILES
      DO 500 J=1,2
  500 NAME(J) = NAMTAB(I,J)
      CALL LOCATE(NAME,POSN)
      IF (POSN .NE. 0) GO TO 550
      WRITE(115,1) NAME
```

```
      S          CALL 9   9            WAIT
           GO TO 600
       550 FPOSN(POSN) = I
       600 CONTINUE
           END


           SUBROUTINE FCOPY(COPY)
      C    COPY FILES IN APPROPRIATE FORMAT
      C    LEAVE TAPE POSITIONED BEFORE LAST END OF FILE MARK
           IMPLICIT INTEGER(A-Z)
           COMMON /NAMES/ NAMTAB(20,2),FPOSN(20),NAME(2),NFILES
           COMMON /INDEX/ INDEX(62)
           EQUIVALENCE (INDEX(4),NFOT)
           DO 1000 I=1,NFOT
           CALL SKIPFILE(120,1)
           IF (NFILES .EQ. 0 .OR. FPOSN(I) .EQ. 0) GO TO 1000
           DO 900 K=1,2
       900 NAME(K) = NAMTAB(FPOSN(I),K)
           CALL COPY
      1000 CONTINUE
           END


           SUBROUTINE FLIST
      C    LISTS RJE FILE ON LP
           IMPLICIT INTEGER(A-Z)
           COMMON /NAMES/ NAMTAB(20,2),FPOSN(20),NAME(2),NFILES
           DIMENSION INPUT(90),OUTPUT(26)
           LOGICAL CONTRL
           DATA DQUOT,G,DOLLAR,SQUOT/2Z7F,2ZC7,2Z5B,2Z7D/
           DATA LDQUOT,LSQUOT,LGRAD /' "  ',' $'' ',' $G '/
           DATA SPACES /'    '/
           DATA LSTAR /'*'/
         1 FORMAT('1WARWICK RESEARCH UNIT FOR THE BLIND REMOTE JOB ENTRY TAPE
          *')
         2 FORMAT(2X,I5,5X,80A1)
         3 FORMAT(' LISTING OF FILE NAMED ',2A4/)
         4 FORMAT(/' *****',A4,'OCCURS ',I,'TIMES')
         5 FORMAT(95X,'WARNING: ',I,'CHARACTERS')
```

```
      CONTRL = .FALSE.
      NGRADS = 0
      NSQUOT = 0
      NDQUOT = 0
      DO 100 I=1,3
  100 OUTPUT(I) = SPACES
      INPTR = 1
      WRITE(108,1)
      CALL DATE
      WRITE(108,3) NAME
      CALL BUFFERIN(120,1,INPUT,90,ISTAT)
  230 NBYTES = ICH(INPUT,INPTR)
      IF (NBYTES .EQ. 0) GO TO 300
      DO 235 I=3,26
  235 OUTPUT(I) = SPACES
      IF (NBYTES .LE. 80) GO TO 240
      WRITE(108,5) NBYTES
      IF (NBYTES .GT. 90) STOP 'NEXT LINE TOO LONG'
C
  240 DO 250 I=1,NBYTES
      IF (INPTR + I .LE. 360) GO TO 242
      CALL BUFFERIN(120,1,INPUT,90,ISTAT)
      INPTR = INPTR - 360
  242 CHAR = ICH(INPUT,INPTR+I)
      IF (.NOT. CONTRL) GO TO 245
C     COUNT $G AND QUOTES, OUTPUT * ON LINES WHERE THEY OCCUR
      IF (CHAR .NE. G) GO TO 243
      NGRADS = NGRADS + 1
      OUTPUT(3) = LSTAR
      GO TO 247
  243 IF (CHAR .NE. SQUOT) GO TO 245
      NSQUOT = NSQUOT + 1
      OUTPUT(3) = LSTAR
      GO TO 247
  245 IF (CHAR .NE. DQUOT) GO TO 247
      NDQUOT = NDQUOT + 1
      OUTPUT(3) = LSTAR
  247 CONTRL = CHAR .EQ. DOLLAR
C     ICH(OUTPUT,12+I) = ISL(CHAR,24)
S        LW,7       I
```

```
387         S         AI,7      11
388         S         LW,9      CHAR
389         S         STB,9     OUTPUT,7
390           250 CONTINUE
391         C
392         C       INCREMENT LINE NUMBER
393         S         LI,7      7
394         S256      LB,9      OUTPUT,7
395         S         CI,9      X'40'
396         S         BNE       $+2
397         S         LI,9      X'F0'
398         S         AI,9      1
399         S         CI,9      X'FA'
400         S         BNE       260S
401         S         LI,9      X'F0'
402         S         STB,9     OUTPUT,7
403         S         BDR,7     256S
404         S26C      STB,9     OUTPUT,7
405                   CALL BUFFEROUT(108,1,OUTPUT,26,ISTAT)
406           270 INPTR = NBYTES + INPTR + 1
407                   IF (INPTR .LE. 360) GO TO 230
408                   CALL BUFFERIN(120,1,INPUT,90,ISTAT)
409                   INPTR = INPTR - 360
410                   GO TO 230
411           300 IF (NSQUOT/2*2 .NE. NSQUOT) WRITE(108,4) LSQUOT,NSQUOT
412                   IF (NDQUOT/2*2 .NE. NDQUOT) WRITE(108,4) LDQUOT,NDQUOT
413                   IF (NGRADS/2*2 .NE. NGRADS) WRITE(108,4) LGRAD,NGRADS
414                   END

415                   SUBROUTINE LOCATE(FNAME,FPOS)
416         C         FINDS POSN OF FILE ON TAPE, RETURNS 0 IF ABSENT
417                   IMPLICIT INTEGER(A-Z)
418                   COMMON /INDEX/ INDEX(62)
419                   EQUIVALENCE (NFOT,INDEX(4))
420                   DIMENSION FNAME(2)
421                   DATA SPACE/2Z40/
422                   DATA NEWLIN/1ZD/
423                   DO 200 I=1,NFOT
424                   BASE = 8 * (I + 1)
```

```
425             DO 100 J=1,8
426             CHAR = ICH(INDEX,BASE+J)
427             CHAR1 = ICH(FNAME,J)
428             IF (CHAR .EQ. NEWLIN .AND. CHAR1 .EQ. SPACE) GO TO 300
429           100 IF (CHAR .NE. CHAR1) GO TO 200
430             GO TO 300
431           200 CONTINUE
432             FPOS = 0
433             RETURN
434           300 FPOS = I
435             END

436             SUBROUTINE COPY
437       C     CONVERTS FILE TO CARD IMAGE FORMAT FOR INPUT TO DOTSYS
438       C     LOGTAB IS TRUE FOR TABLES, OTHERWISE FALSE
439             IMPLICIT INTEGER(A-Z)
440             COMMON /LWR/ OUTPUT(20),OPBASE,LENGTH,SPACE
441             COMMON /C/ NLINES,LOGTAB
442             COMMON /F/ FILNO
443             COMMON /NAMES/ NAMTAB(20,2),FPOSN(20),NAME(2),NFILES
444             DIMENSION INPUT(90)
445             LOGICAL LOGTAB,HYPHEN
446             DATA PAGE,STAR/'$PG ',8Z5C000000/
447             DATA SPACE /'    '/
448             DATA RSPACE / 2Z40 /
449             DATA HYPH,SEMIC,OSB,CSB,LT,GT/2Z60,2Z7A,2ZB4,2ZB5,2Z4C,2Z6E/
450           1 FORMAT('   $PG $%$%$%$% $%$%$%$% $SL01 ',I3,X,2A4,'$SL01 $%$%$%$%
451            *$%$%$%$% $T',11X)
452           2 FORMAT(X,I4,4X,2A4,8X,20A4)
453           3 FORMAT(' TABLE ERROR')
454           4 FORMAT(' MISSING * IN ',2A4,3X,'FIRST LINE IS'/X,20A4)
455             CHAR = 0
456             DO 50 I=1,20
457           50 OUTPUT(I) = SPACE
458             CALL BUFFERIN(120,1,INPUT,90,ISTAT)
459             LOGTAB = FILNO .LT. 0
460             FILNO = FILNO + 1
461             IF (LOGTAB) GO TO 120
462             WRITE(121,1) FILNO,NAME
```

```
463                 LENGTH = 80
464                 NBYTES = ICH(INPUT,1)
465                 IF (NBYTES .GT. 80) STOP 'LINE TOO LONG'
466                 DO 100 I=1,NBYTES
467           C     ICH(OUTPUT,I) = ISL(ICH(INPUT,I+1),24)
468           S        LW,7       I
469           S        LB,9       INPUT,7
470           S        AI,7       -1
471           S        STB,9      OUTPUT,7
472             100 CONTINUE
473           S        LB,9       OUTPUT
474           S        CB,9       STAR
475           S        BE         103S
476                 WRITE(115,4) NAME,OUTPUT
477           S          CAL1,9   9              WAIT
478             103 WRITE(108,2) FILNO,NAME,OUTPUT
479                 OUTPUT(1) = PAGE
480                 INPTR = NBYTES + 2
481                 OPBASE = 4
482                 GO TO 150
483             120 LENGTH = 80
484                 OPBASE = 0
485                 INPTR = 1
486             150 NBYTES = ICH(INPUT,INPTR)
487                 IF (NBYTES .EQ. 0) GO TO 500
488                 IF (.NOT. LOGTAB .OR. NBYTES .EQ. 80) GO TO 155
489                 WRITE(115,3)
490           S        CAL1,9     3              ABORT
491             155 HYPHEN = .FALSE.
492           C
493                 DO 300 I=1,NBYTES
494                 IF (INPTR + I .LE. 360) GO TO 160
495                 CALL BUFFERIN(120,1,INPUT,90,ISTAT)
496                 INPTR = INPTR - 360
497             160 OUTPTR = OPBASE + I
498           C     CHAR = ICH(INPUT,INPTR+I)
499           S        LW,7       INPTR
500           S        AW,7       I
501           S        AI,7       -1
502           S        LB,9       INPUT,7
```

```
503         S         STW,9      CHAR
504                 IF (LOGTAB) GO TO 170
505                 IF (CHAR .EQ. OSB) CHAR = LT
506                 IF (CHAR .EQ. CSB) CHAR = GT
507           C     REPLACE LOWER CASE LETTERS WITH UPPER CASE
508                 IF (CHAR .GT. 2Z80 .AND. CHAR .LT. 2ZC0) CHAR = CHAR + 2740
509                 HYPHEN = CHAR .EQ. HYPH .AND. OLAST .NE. SEMIC .AND.
510                *    OLAST .NE. RSPACE
511                 OLAST = CHAR
512             170 CONTINUE
513           C     ICH(OUTPUT,OUTPTR) = ISL(CHAR,24)
514           S        LW,7       OUTPTR
515           S        AI,7       -1
516           S        LW,9       CHAR
517           S        STB,9      OUTPUT,7
518                 IF (I .EQ. NBYTES .AND. HYPHEN) OUTPTR = OUTPTR - 1
519                 IF (OUTPTR .GE. LENGTH) CALL LWRITE
520             300 CONTINUE
521           C
522                 OPBASE = OUTPTR
523                 IF (OPBASE .EQ. LENGTH) OPBASE = 0
524                 IF (HYPHEN .OR. LOGTAB) GO TO 400
525                 OPBASE = OPBASE + 1
526                 IF (OPBASE .GE. LENGTH) CALL LWRITE
527             400 INPTR = INPTR + NBYTES + 1
528                 IF (INPTR .LE. 360) GO TO 150
529                 CALL BUFFERIN(120,1,INPUT,90,ISTAT)
530                 INPTR = INPTR - 360
531                 GO TO 150
532             500 IF (.NOT. LOGTAB) CALL LWRITE
533                 END


534                 SUBROUTINE LWRITE
535           C     OUTPUTS A LINE IN CARD IMAGE FORMAT  (CALLED BY COPY)
536                 IMPLICIT INTEGER(A-Z)
537                 COMMON /LWR/ OUTPUT(20),OPBASE,LENGTH,SPACE
538                 COMMON /C/ NLINES,LOGTAB
539                 LOGICAL LOGTAB
540                1 FORMAT(' TOO MUCH INPUT')
```

```
541               CALL BUFFEROUT(121,1,OUTPUT,2C,ISTAT)
542               IF (ISTAT .EQ. 2) GO TO 200
543               WRITE(115,1)
544          S       CAL1,9    3            ABORT
545            2C0 OPBASE = OPBASE - LENGTH
546          C     DO I=1,20 OUTPUT(I) = SPACE
547          S       LW,9      SPACE
548          S       LI,7      20
549          S       STW,9     OUTPUT-1,7
550          S       BDR,7     $-1
551               IF (.NOT. LOGTAB) NLINES = NLINES + 1
552               END


553               SUBROUTINE EMBOSS
554          C    COPIES RJE FILE NAMED BRAILLE TO FGDTEMP FOR EMBOSSING
555               IMPLICIT INTEGER(A-Z)
556               DIMENSION INPUT(90),OUTPUT(11),FNAME(2)
557               LOGICAL FIRST,ENDRED,INHEAD,ENDWRT
558               DATA FNAME / 'BRAI','LLE ' /
559               DATA OP1,OUTPUT(11),SPACES / 8Z0A124040,8Z0A3C0000,'     ' /
560               DATA NEWPAG,NEWDOC / 8Z0A120C0A,8Z0A127E7E /
561             1 FORMAT(' BRAILLE NOT ON TAPE')
562             2 FORMAT(' BRAILLE ERROR')
563             3 FORMAT(/' DOCUMENT NO.   START   END   #PAGES   #LINES   TIME'/)
564             4 FORMAT(' EOF ON FGDTEMP AT PAGE ',I)
565             5 FORMAT(6X,I2,8X,I4,4X,I3,5X,I3,5X,I4,5X,I3)
566             6 FORMAT(/X,I,'PAGES'/X,I,'LINES'/' ESTIMATED EMBOSSING TIME ',I,'MI
567              *NUTES'/)
568            50 CALL LOCATE(FNAME,FPOS)
569               IF (FPOS .NE. 0) GO TO 100
570               WRITE(115,1)
571           S      CAL1,9    9            WAIT
572               REWIND 120
573               GO TO 50
574           100 CALL SKIPFILE(120,FPOS)
575               ENDRED = .FALSE.
576               ENDWRT = .FALSE.
577               INHEAD = .FALSE.
578               FIRST = .TRUE.
```

```
579             LINES = 0
580             PAGES = 0
581             TOTLIN = 0
582             TOTPAG = 0
583             TOTTIM = 0
584             DOCNO = 0
585             INPTR = 1
586             WRITE(108,3)
587             CALL BUFFERIN(120,1,INPUT,90,ISTAT)
588           230 OUTPUT(1) = OP1
589             DO 232 I=2,10
590           232 OUTPUT(I) = SPACES
591             NBYTES = ICH(INPUT,INPTR)
592             IF (NBYTES .NE. 0) GO TO 234
593             ENDRED = .TRUE.
594             GO TO 400
595           234 IF (NBYTES .LE. 40) GO TO 235
596             WRITE(115,2)
597             STOP 'ERROR'
598           235 DO 250 I=3,NBYTES
599             IF (INPTR + I .LE. 360) GO TO 240
600             CALL BUFFERIN(120,1,INPUT,90,ISTAT)
601             INPTR = INPTR - 360
602           240 CONTINUE
603       C     ICH(OUTPUT,I) = ICH(INPUT,INPTR+I)
604       S       LW,7      INPTR
605       S       AW,7      I
606       S       AI,7      -1
607       S       LB,9      INPUT,7
608       S       SW,7      INPTR
609       S       STB,9     OUTPUT,7
610           250 CONTINUE
611             IF (OUTPUT(1) .EQ. NEWDOC) GO TO 300
612             IF (OUTPUT(1) .EQ. NEWPAG) GO TO 255
613             LINES = LINES + 1
614             GO TO 260
615           255 PAGES = PAGES + 1
616           260 IF (ENDWRT) GO TO 270
617             CALL BUFFEROUT(118,1,OUTPUT,11,ISTAT)
618             IF (ISTAT .EQ. 2) GO TO 270
```

```
      T = TOTPAG + PAGES + DOCNO
      WRITE(115,4) T
      ENDWRT = .TRUE.
  270 INPTR = NBYTES + INPTR + 1
      IF (INPTR .LE. 360) GO TO 230
      CALL BUFFERIN(120,1,INPUT,90,ISTAT)
      INPTR = INPTR - 360
      GO TO 230
C
  300 IF (.NOT. INHEAD) GO TO 350
      INHEAD = .FALSE.
      GO TO 260
  350 IF (.NOT. FIRST) GO TO 400
      FIRST = .FALSE.
      GO TO 500
  400 DOCNO = DOCNO + 1
      START = TOTPAG + DOCNO
      TOTLIN = TOTLIN + LINES
      IF (.NOT. ENDRED) PAGES = PAGES - 1
      TOTPAG = TOTPAG + PAGES
      FINISH = TOTPAG + DOCNO
      TIME = (PAGES + LINES + 7) / 14
      TOTTIM = TOTTIM + TIME
      WRITE(108,5) DOCNO,START,FINISH,PAGES,LINES,TIME
      IF (ENDRED) GO TO 1000
  500 INHEAD = .TRUE.
      LINES = 0
      PAGES = 0
      GO TO 260
 1000 TOTPAG = TOTPAG + DOCNO
      WRITE(108,6) TOTPAG,TOTLIN,TOTTIM
      ENDFILE 118
      STOP 'OK'
      END


      SUBROUTINE DATE
C     PRINTS CURRENT DATE ON LP
      IMPLICIT INTEGER(A-Z)
      DIMENSION DT(4)
```

```
      1 FØRMAT(' DATE:
S        LI,9       DT
S        AW,9       =8Z10000000
S        CAL1,8     9
        DT(1) = ISL(DT(3),16)
        DT(2) = ISL(DT(2),16) + ISL(ICH(DT(3),1),8)
        DT(3) = ISL(DT(4),16)
        IF (ICH(DT(1),1) .EQ. 2ZF0) DT(1) = DT(1) - 8ZB0000000
        WRITE(108,1) (DT(I),I=1,3)
        END


        SUBRØUTINE SAVE
C       CØPIES SPECIFIED FILES FRØM RJE TAPE TØ ARCHIVE TAPE VIA DISC
C       IF NØ FILE NAMES SPECIFIED, ALL FILES WITH * AS FIRST CHAR ARE CØPIED
        IMPLICIT INTEGER(A-Z)
        CØMMØN /MØV/ BUFFER(1800),BUFPTR,BUFSIZ,INPUT(90)
        CØMMØN /INDEX/ INDEX(62)
        CØMMØN /NAMES/ NAMTAB(20,2),FPØSN(20),NAME(2),NFILES
        EQUIVALENCE (NFØT,INDEX(4)),(NRECS,INPUT(4))
        DIMENSIØN SINDEX(4,19),TIME(4)
        DATA BUFSIZ,STAR /1800,2Z5C/
      1 FØRMAT(X,2A4,' NØT ARCHIVED - MISSING ASTERISK IN FIRST LINE')
      2 FØRMAT(X,2A4,' NØT ARCHIVED - FILE ØF SAME NAME ALREADY ØN TAPE')
      3 FØRMAT(X,2A4,' ARCHIVED')
        CALL RDLØC
        IF (NFILES .NE. 0) GØ TØ 100
        DØ 80 I=1,NFØT
     80 FPØSN(I) = 1
    100 NSAVED = 0
C       FØLLØWING CØDE PLACES DATE AND TIME IN TIME
S        LI,9       TIME
S        AW,9       =8Z10000000
S        CAL1,8     9
        DØ 1000 I=1,NFØT
        CALL SKIPFILE(120,1)
        IF (FPØSN(I) .EQ. 0) GØ TØ 1000
S        LI,7       84
S        AW,7       I
S        LH,9       INDEX,7
```

```
695         S         STW,9      NLIN
696         S         AI,7       20
697         S         LH,9       INDEX,7
698         S         STW,9      NREC
699                   IF (NREC .EQ. 0) GO TO 1000
700                   DO 200 J=1,NREC
701                   CALL BUFFERIN(120,1,INPUT,90,ISTAT)
702                   IF (J .NE. 1) GO TO 150
703                   IF (ICH(INPUT,2) .EQ. STAR) GO TO 150
704                   WRITE(108,1) (INDEX(2*I+J),J=3,4)
705                   GO TO 1000
706            150    CALL BUFFEROUT(119,1,INPUT,90,ISTAT)
707            200    IF (ISTAT .EQ. 3) GO TO 1200
708                   NSAVED = NSAVED + 1
709                   SINDEX(1,NSAVED) = INDEX(2*I+3)
710                   SINDEX(2,NSAVED) = INDEX(2*I+4)
711                   SINDEX(3,NSAVED) = NLIN
712                   SINDEX(4,NSAVED) = NREC
713            1000   CONTINUE
714                   GO TO 1400
715            1200   OUTPUT(115) 'EXCEEDED DISC SPACE'
716            1400   REWIND 120
717                   REWIND 119
718                   IF (NSAVED .EQ. 0) RETURN
719                   DO 1500 I=1,NSAVED
720            1500   FPOSN(I) = 1
721                   OUTPUT(115) 'LOAD ARCHIVE TAPE ON TC'
722         S            CAL1,9    9              BEGIN WAIT
723                   CALL SKIPFILE(120,1)
724                   BUFPTR = BUFSIZ
725                   CALL IMOVE(0)
726            1600   CALL IMOVE(4)
727                   IF (INPUT(1) .EQ. 0) GO TO 1780
728                   DO 1700 I=1,NSAVED
729            1700   IF (SINDEX(1,I) .EQ. INPUT(1) .AND. SINDEX(2,I) .EQ. INPUT(2))
730                *     GO TO 1720
731                   GO TO 1750
732            1720   WRITE(108,2) (INPUT(J),J=1,2)
733                   FPOSN(I) = 0
734            1750   DO 1770 I=1,NRECS
```

```
735           1770 CALL IMOVE(0)
736                GO TO 1600
737           1780 BUFPTR = BUFPTR - 90
738                BACKSPACE 120
739                DO 2400 I=1,NSAVED
740                DO 1800 J=1,4
741           1800 INPUT(J) = SINDEX(J,I)
742                IF (FPOSN(I) .EQ. 0) GO TO 2300
743                INPUT(5) = ISL(TIME(3),16)
744                INPUT(6) = ISL(TIME(2),16) + ISL(ICH(TIME(3),1),8)
745                INPUT(7) = ISL(TIME(4),16)
746                DO 2000 J=8,90
747           2000 INPUT(J) = 0
748                CALL OMOVE
749                DO 2200 J=1,NRECS
750                CALL BUFFERIN(119,1,INPUT,90,ISTAT)
751           2200 CALL OMOVE
752                GO TO 2400
753           2300 DO 2350 J=1,NRECS
754           2350 CALL BUFFERIN(119,1,INPUT,90,ISTAT)
755           2400 CONTINUE
756                DO 2600 I=1,90
757           2600 INPUT(I) = 0
758                CALL OMOVE
759                CALL BUFFEROUT(120,1,BUFFER,BUFSIZ,ISTAT)
760                ENDFILE 120
761                ENDFILE 120
762                DO 2800 I=1,NSAVED
763           2800 IF (FPOSN(I) .NE. 0) WRITE(108,3) (SINDEX(J,I),J=1,2)
764                END


765                SUBROUTINE IMOVE(LENG)
766           C    GETS FIRST 4 WORDS OF NEXT LOGICAL RECORD FROM ARCHIVE TAPE
767                IMPLICIT INTEGER(A-Z)
768                COMMON /MOV/ BUFFER(1800),BUFPTR,BUFSIZ,INPUT(90)
769                IF (BUFPTR .LT. BUFSIZ) GO TO 100
770                CALL BUFFERIN(120,1,BUFFER,BUFSIZ,ISTAT)
771                BUFPTR = 0
772             100 IF (LENG .EQ. 0) GO TO 300
```

```
      DO 200 I=1,LENG
  200 INPUT(I) = BUFFER(BUFPTR+I)
  300 BUFPTR = BUFPTR + 90
      END


      SUBROUTINE OMOVE
C     WRITES LOGICAL 90-WORD RECORD TO ARCHIVE TAPE
      IMPLICIT INTEGER(A-Z)
      COMMON /MOV/ BUFFER(1800),BUFPTR,BUFSIZ,INPUT(90)
      IF (BUFPTR .LT. BUFSIZ) GO TO 100
      CALL BUFFEROUT(120,1,BUFFER,BUFSIZ,ISTAT)
      BUFPTR = 0
  100 DO 200 I=1,90
  200 BUFFER(BUFPTR+I) = INPUT(I)
      BUFPTR = BUFPTR + 90
      END


      SUBROUTINE DOTSYS
      IMPLICIT INTEGER(A-Z)
      COMMON /BUFF/ BUFFER(10)
      EQUIVALENCE (BUFFER(1),BUFF1),(BUFFER(2),BUFF2,RBUFF(1))
      DIMENSION RBUFF(9)
      COMMON /CTAB/ CTABLE(3,750),SIGNS(750),BRANCH(750),NCT,TSIGN(4)
      COMMON /ALPH/ SYMBOL(64),CCLASS(64),SINGSG(64),EXTENT(64),
     *   COMPB(64),ISOLET(64),NALPH
      COMMON /LOGIC/ DECIS(15,25),COND(15,10),TRANS(10,25),NONTER(2),
     *   RCCLAS(4,2),NDTC,NIC,NNT
      COMMON /STV/ STVAR(10),NSV
      COMMON /SPAC/ SPACE
      COMMON /S/ LETS
      COMMON /N/ LETN
      COMMON /COMP/ COMPUT
      LOGICAL COMPUT
      COMMON /RCC/ RCC
C     FOLLOWING COMMONS NOT USED IN MAIN PROG BUT NEEDED FOR OVERLAY
      COMMON /STAX/ TS,HS,CS,PS,TAILS(2),HEADS(2),CURRS(2),CURTYP,STKIND
     *,HFSTAK,ELT(19,30),SPCONT(19),NOELTS(19),PVSTAK(19),NXSTAK(19)
      COMMON /IN/ INPTR,PRIORS,SCOUNT
```

```
      COMMON /OUT/ OUTPTR,OUTL,LINES,LPG
      COMMON /TAB/ TABTYP(9),TABCLM(9),TABULA,LETR,LETD
      COMMON /OPTS/ EMBOPT,PFOUT
      COMMON /SIGNS/DOTS14(64),DOTS25(64),DOTS36(64),PCHAR(64),MCODE(64)
      COMMON /MEMB/ EMBOUT(20),MEMBCR,MEMBPG,MIDLE
      DIMENSION BLANKS(9)
      EQUIVALENCE (BLANKS,EMBOUT(12))
      COMMON /CPR/ CPRINT
      COMMON /PAG/ CDIGIT(4),DIGIT(10),PAGINA
      COMMON /LC/ LCOUNT
      COMMON /VAL/ VALUE(64),BVAL(10)
C
      DATA LETA,LETF,LETG,LETI,LETN,LETO,LETR,LETS,LETT,LETY,STAR,
     *   HYPHEN,ITALIC,SPACE,LETSIG,CAPSIG,ACCENT,OPBRAK,CLBRAK/
     *   'A','F','G','I','N','O','R','S','T','Y','*','-','~',' ','=',
     *   '¬','¢','(',')'/
      DATA DIGIT /26,1,3,9,25,17,11,27,19,10/
      DATA RCC / '|' /
      DATA EMBOUT(1),EMBOUT(11) /8ZCA120000,8Z0A3C0000/
      DATA EMBOUT(1),EMBOUT(11) / '    ','    ' /
      DATA TBB/'    '/
      DATA BLANKS / 9*'    ' /
    1 FORMAT(' NEW CHAR ',Z8)
      OUTPUT 'DOTSYS III-F'
C
      CALL SEGLOD(3)
      CALL INIT
      CALL SEGLOD(4)
      CALL SHIFT(10)
      CALL PCLEAR
      CALL RESET
C
C     IDENTIFY LEFTMOST CHARACTER OF BUFFER
  200 LETTER = BVAL(1)
      IF (LETTER .NE. 0) GO TO 600
      IF (BUFF1 .NE. RCC .AND. BUFF1 .NE. STAR) GO TO 9800
      CALL OPSIGN(98)
C     RUN WILL STOP WITH ABOVE CALL
C
  600 IF (ISOLET(LETTER) .NE. 1) GO TO 3200
```

```
      C
      C     INSERT ITALIC SIGN BEFORE SINGLE LETTER IN BRACKETS, QUOTES, ETC.
            IF (BUFF1 .EQ. ITALIC .AND. (BUFF2 .EQ. LETA .OR. BUFF2 .EQ. LETI
           *   .OR. BUFF2 .EQ. LET0 .OR. BUFF2 .EQ. ITALIC)) GO TO 3200
            N = 1
            K = 1
        800 T = RBUFF(N)
            IF (T .EQ. LETSIG) GO TO 3200
            IF (T .EQ. CAPSIG .OR. T .EQ. ACCENT) GO TO 2200
            IF (T .EQ. ITALIC) GO TO 2400
            X = BVAL(N+1)
            IF (X .NE. 0) GO TO 1200
            WRITE(108,1) T
       1200 IF (CCLASS(X) .NE. 1) GO TO 3200
            M = N + 1
            IF (BUFF1 .EQ. ITALIC) GO TO 2600
            IF (SYMBOL(LETTER) .EQ. OPBRAK) GO TO 1400
            IF (SYMBOL(LETTER) .EQ. RBUFF(M)) GO TO 1600
            GO TO 3200
      C
       1400 IF (RBUFF(M) .NE. CLBRAK) GO TO 3200
       1600 DO 1800 I=1,K
            RBUFF(M) = RBUFF(N)
            BVAL(M+1) = BVAL(N+1)
            N = N - 1
       1800 M = M - 1
            CALL SHIFT(M)
            BUFF1 = LETSIG
            BVAL(1) = ICH(VALUE,2Z7E)
            GO TO 200
      C
       2200 K = K + 1
       2400 N = N + 1
            GO TO 800
      C
       2600 X = BVAL(M+1)
            IF (X .EQ. 0) X = ICH(VALUE,2Z5C)
       3000 IF (CCLASS(X) .EQ. 1) GO TO 3200
            IF (N .GT. K) CALL SHIFT(1)
            BUFF1 = LETSIG
```

```
      BVAL(1) = ICH(VALUE,2Z7E)
      GO TO 200
C
C     CONTRACTION TABLE SEARCH
C     INDEX POINTS TO CONTRACTION TABLE ENTRY UNDER CONSIDERATION
 3200 INDEX = EXTENT(LETTER)
      IF (INDEX .GT. NCT) GO TO 9200
      NXTLEV = 1
C
C     TEST FOR MATCH BETWEEN CURRENT TABLE ENTRY AND BUFFER
 3400 I12 = INDEX * 12 - 13
S         LW,1      NXTLEV      POINTR
S3450     LW,7      I12
S         AW,7      1
S         LB,9      CTABLE,7
S         CB,9      RCC
S         BE        3500S
S         LW,7      1
S         MI,7      4
S         CB,9      BUFFER,7
S         BNE       3900S
S         AI,1      1
S         CI,1      9
S         BLE       3450S
S         AI,1      -1
S3500     LW,2      INDEX
S         B         5000S
C
C     REGISTERS  1=POINTR, 2=INDEX, 3=BRI, 4=-BRI
S3800     LW,1      POINTR
S3900     LW,2      INDEX
S         LW,3      BRANCH-1,2
S         LCW,4     3
S         BLZ       4600S
S         CW,1      NXTLEV
S         BNE       4400S
S4000     CW,4      NXTLEV
S         BL        4400S
S         AI,2      1
S4200     LW,3      BRANCH-1,2
```

```
929          S          LCW,4      3
930          S          BGEZ       4000S
931          S          LW,2       3
932          S          B          4200S
933          S44CO      STW,4      NXTLEV
934          S          CI,4       1
935          S          BLE        9200S
936          S          AI,2       1
937          S          STW,2      INDEX
938          S          B          3400S
939          S46CO      CW,1       NXTLEV
940          S          BE         4800S
941          S          LW,7       2
942          S          AI,7       1
943          S          CW,3       7
944          S          BE         $+2
945          S          MTW,1      NXTLEV
946          S          AI,2       1
947          S          STW,2      INDEX
948          S          B          3400S
949          S48CO      STW,3      2
950          S          STW,2      INDEX
951          S          B          3400S
952          S50CO      LW,7       2
953          S          MI,7       12
954          S          AI,7       -1
955          S          LB,9       CTABLE,7
956          S          CB,9       SPACE
957          S          BE         6100S
958          S          STB,9      TBB
959          S          LW,9       BVAL,1
960          S          STW,9      N
961          S          BEZ        3800S
962          S          STW,1      POINTR
963          S          STW,2      INDEX
964          C
965           54CO DO 6000 K=1,NNT
966                IF (TBB .NE. NONTER(K)) GO TO 6000
967                DO 5800 J=1,4
968                IF (CCLASS(N) .EQ. RCCLAS(J,K)) GO TO 6200
```

```
969          5800 IF (RCCLAS(J,K) .EQ. 99) GO TO 3800
970               GO TO 3800
971          6000 CONTINUE
972               GO TO 3800
973         C
974         C     RIGHT CONTEXT IS 0,K. - NOW CHECK INPUT CLASS FOR COMPATIBILITY
975         C     WITH CURRENT VALUE OF STATE VARIABLES
976         S6100    STW,1      POINTR
977         S        STW,2      INDEX
978          6200 CONTINUE
979         S        LW,7       INDEX
980         S        MI,7       12
981         S        AI,7       -2
982         S        LB,9       CTABLE,7
983         S        STW,9      TCLASS
984               DO 7000 K=1,NDTC
985               DKT = DECIS(K,TCLASS)
986               IF (DKT .EQ. HYPHEN) GO TO 7000
987               IF (DKT .EQ. LETG) GO TO 7600
988               IF (DKT .EQ. LETF) GO TO 3800
989               DO 6800 N=1,NSV
990               CKN = COND(K,N)
991          6800 IF (CKN .NE. HYPHEN .AND. CKN .NE. STVAR(N)) GO TO 7400
992               GO TO 7200
993          7000 CONTINUE
994         C
995          7200 IF (DKT .EQ. LETY) GO TO 7600
996               GO TO 3800
997          7400 IF (DKT .EQ. LETY) GO TO 3800
998         C
999         C     CONTRACTION TABLE ENTRY IS ACCEPTED
1000        C     SEND APPROPRIATE SIGNS TO STACKER AND SHIFT BUFFER LEFT BY
1001        C     NUMBER OF CHARACTERS INDICATED IN CONTRACTION TABLE ENTRY
1002         7600 CONTINUE
1003        S        LW,7       INDEX
1004        S        MI,7       12
1005        S        AI,7       -3
1006        S        LB,9       CTABLE,7
1007        S        STW,9      SHIFNO
1008              CALL SHIFT(SHIFNO)
```

```
1009                I = SIGNS(INDEX)
1010                DO 8200 J=0,3
1011          S        LW,7       J
1012          S        LB,9       T,7
1013          S        CI,9       99
1014          S        BE         8400S
1015          S        STW,9      OUTSIG
1016           8200 CALL OPSIGN(OUTSIG)
1017          C
1018          C       ADJUST STATE VARIABLES ACCORDING TO TRANSITION TABLES
1019           8400 I = TCLASS
1020                DO 9000 N=1,NSV
1021                TNI = TRANS(N,I)
1022                IF (TNI .EQ. HYPHEN) GO TO 9000
1023                IF (TNI .NE. LETS .AND. (TNI .NE. LETT .OR. STVAR(N) .NE. LETN))
1024               *    GO TO 8800
1025                STVAR(N) = LETY
1026                GO TO 9000
1027           8800 STVAR(N) = LETN
1028           9000 CONTINUE
1029                GO TO 200
1030          C
1031          C       NO CONTRACTION TABLE ENTRY IS ACCEPTABLE - OUTPUT SINGLE CHARACTER
1032          C       OBTAINED FROM ALPHABET TABLE
1033           9200 IF (.NOT. COMPUT) GO TO 9400
1034                OUTSIG = COMPB(LETTER)
1035                GO TO 9600
1036           9400 OUTSIG = SINGSG(LETTER)
1037           9600 CALL SHIFT(1)
1038                CALL OPSIGN(OUTSIG)
1039                TCLASS = CCLASS(LETTER)
1040                GO TO 8400
1041          C
1042          C       LEFTMOST CHARACTER OF BUFFER DOES NOT OCCUR IN ALPHABET TABLE
1043          C       OUTPUT ERROR MESSAGE
1044           9800 WRITE(108,1) BUFF1
1045                BUFF1 = STAR
1046                BVAL(1) = ICH(VALUE,2Z5C)
1047                GO TO 200
1048                END
```

```
1049              INTEGER FUNCTION CHAR(ARRAY,I,J)
1050              IMPLICIT INTEGER(A-Z)
1051              DATA ADDEND /6Z404040/
1052              CHAR = ISL(ICH(ARRAY,12*(I-1)+J),24) + ADDEND
1053              END


1054              SUBROUTINE INIT
1055        C
1056        C     INITIALISATION SUBROUTINE
1057              IMPLICIT INTEGER(A-Z)
1058              COMMON /STAX/ TS,HS,CS,PS,TAILS(2),HEADS(2),CURRS(2),CURTYP,STKIND
1059             *,HFSTAK,ELT(19,30),SPCONT(19),NOELTS(19),PVSTAK(19),NXSTAK(19)
1060              EQUIVALENCE(HEADS1,HEADS(1)),(TAILS1,TAILS(1)),(CURRS1,CURRS(1))
1061              EQUIVALENCE(HEADS2,HEADS(2)),(TAILS2,TAILS(2)),(CURRS2,CURRS(2))
1062              COMMON /IN/ INPTR,PRIORS,SCOUNT
1063              COMMON /OUT/ OUTPTR,OUTL,LINES,LPG
1064              COMMON /TAB/ TABTYP(9),TABCLM(9),TABULA,LETR,LETD
1065              COMMON /OPTS/ EMBOPT,PFOUT
1066              COMMON /SIGNS/DOTS14(64),DOTS25(64),DOTS36(64),PCHAR(64),MCODE(64)
1067              COMMON /MEMB/ EMBOUT(20),MEMBCR,MEMBPG,MIDLE
1068              COMMON /CPR/ CPRINT
1069              COMMON /PAG/ CDIGIT(4),DIGIT(10),PAGINA
1070              COMMON /SPAC/ SPACE
1071              COMMON /CHEK/ CARDNO,PREVNO,ECHO,SEQERR
1072              COMMON /CTAB/ CTABLE(3,750),SIGNS(750),BRANCH(750),NCT,TSIGN(4)
1073              COMMON /ALPH/ SYMBOL(64),CCLASS(64),SINGSG(64),EXTENT(64),
1074             *   COMPB(64),ISOLET(64),NALPH
1075              COMMON /LC/ LCOUNT
1076              COMMON /LOGIC/ DECIS(15,25),COND(15,10),TRANS(10,25),NONTER(2),
1077             *   RCCLAS(4,2),NDTC,NIC,NNT
1078              COMMON /STV/ STVAR(10),NSV
1079              COMMON /RCC/ RCC
1080              COMMON /VAL/ VALUE(64),BVAL(10)
1081              LOGICAL EMBOPT,PFOUT,SEQERR,ECHO,PAGINA,CPRINT
1082              DIMENSION LARRAY(10)
1083              DATA LETE,LETL,LETM,LETP / 'E','L','M','P' /
1084              DATA CTMAX /1000/
1085        C
1086              1 FORMAT(80X)
```

```
1087              2 FORMAT(' ** TABLE SEQUENCE ERROR')
1088              3 FORMAT(A1,71X,I8)
1089              4 FORMAT(A1,16X,3(I2,X),3X,I2,41X,I8)
1090              5 FORMAT(A1,3A4,4X,2(I2,X),4I2,41X,I8)
1091              6 FORMAT(17X,I2,53X,I8)
1092              7 FORMAT(12X,A1,10X,4I2,41X,I8)
1093              8 FORMAT(25A1,47X,I8)
1094              9 FORMAT(NA1,NA1,I)
1095             10 FORMAT(3A2,A3,A1,62X,I8)
1096             11 FORMAT(A1,14X,5A1,52X,I8)
1097             12 FORMAT(A1,30X,4I1,37X,I8)
1098             13 FORMAT(' TOO MANY CONTRACTION TABLE ENTRIES')
1099          C
1100                WRITE(108,1)
1101                INPTR = 81
1102                SEQERR = .FALSE.
1103                PREVNO = 0
1104                ECHO = .TRUE.
1105                READ(131,3) T,CARDNO
1106                ECHO = T .EQ. LETE
1107                CALL SEQCHK
1108                DO 100 N=1,64
1109              100 VALUE(N) = 0
1110          C
1111          C     SET UP STACKS
1112                DO 200 N=1,19
1113                NOELTS(N) = 0
1114                SPCONT(N) = 0
1115                PVSTAK(N) = N - 1
1116              200 NXSTAK(N) = N + 1
1117                NXSTAK(19) = 0
1118                NXSTAK(1) = 0
1119                PVSTAK(2) = 0
1120                HFSTAK = 3
1121                CURTYP = 1
1122                HEADS1 = 1
1123                TAILS1 = 1
1124                CURRS1 = 1
1125                HEADS2 = 2
1126                TAILS2 = 2
```

```
1127                CLRPS2 = 2
1128                HS = 1
1129                TS = 1
1130                CS = 1
1131                PVSTAK(3) = 0
1132                NXSTAK(2) = 0
1133          C
1134                DO 600 N = 1,9
1135                TABTYP(N) = LETL
1136              600 TABCLM(N) = 2 * N + 1
1137                LCOUNT = 0
1138                OUTPTR = 1
1139                TABULA = 0
1140                SCOUNT = 0
1141                TEMPL = 0
1142                STKIND = 2
1143                CPRINT = .FALSE.
1144                PRICRS = 0
1145          C
1146          C     READ ALPHABET TABLE
1147                DO 800 N=1,65
1148                EXTENT(N) = 999
1149                READ(131,4) SYMBOL(N),CCLASS(N),COMPB(N),SINGSG(N),ISOLET(N),
1150               *    CARDNO
1151                CALL SEQCHK
1152                IF (CCLASS(N) .EQ. 99) GO TO 1000
1153                T = SYMBOL(N)
1154          C     ICH(VALUE,ICH(SYMBOL(N),1)) = N
1155          S        LB,7       T
1156          S        AI,7       -1
1157          S        LW,9       N
1158          S        STB,9      VALUE,7
1159              800 CONTINUE
1160             1000 NALPH = N
1161                I = 1
1162          C
1163          C     FILL CONTRACTION TABLE
1164                DO 1800 N=1,CTMAX
1165                READ(131,5) LET1,(CTABLE(J,N),J=1,3),ICLASS,SHIFNO,
1166               *    (TSIGN(J),J=1,4),CARDNO
```

```
1167          C      PUT ICLASS IN BYTE 11 OF CTABLE ENTRY
1168          S         LW,7       N
1169          S         MI,7       12
1170          S         AI,7       -2
1171          S         LW,9       ICLASS
1172          S         STB,9      CTABLE,7
1173          S         AI,7       -1
1174          S         LW,9       SHIFNO
1175          S         STB,9      CTABLE,7
1176                 CALL SEQCHK
1177          C      PACK SIGNS INTO SIGNS(N)
1178                 DO 1100 J=0,3
1179          S         LW,7       J
1180          S         LW,9       TSIGN,7
1181          S1100     STB,9      T,7
1182                 SIGNS(N) = T
1183                 IF (LET1 .EQ. TEMPL) GO TO 1400
1184                 TEMPL = LET1
1185                 EXTENT(I) = N
1186                 IF (SYMBOL(I) .EQ. TEMPL .OR. ICLASS .EQ. 99) GO TO 1200
1187                 ECHO = .TRUE.
1188                 SEQERR = .TRUE.
1189                 WRITE(108,2)
1190           1200  I = I + 1
1191           1400  CONTINUE
1192           1800  IF (ICLASS .EQ. 99) GO TO 2000
1193                 WRITE(108,13)
1194                 STOP 'ERROR'
1195          C
1196           2000  NCT = N - 1
1197                 J = I - 1
1198                 EXTENT(I) = EXTENT(J) + 1
1199                 MAXEXT = J
1200          C
1201          C      CONTRACTION TABLE SEARCH SET-UP ALGORITHM
1202                 T = NCT + 1
1203                 DO 2200 I=1,T
1204           2200  BRANCH(I) = 0
1205                 DO 4000 I=1,MAXEXT
1206                 J = 1
```

```
1207                LOWNO = EXTENT(I)
1208                J = I + 1
1209                HIGHNO = EXTENT(J) - 1
1210                VHIGH = HIGHNO
1211             2600 II = LOWNO + 1
1212                IF (II .GT. HIGHNO) GO TO 3600
1213                IF (CHAR(CTABLE,LOWNO,N) .EQ. RCC) GO TO 3000
1214             2800 IF (II .GT. HIGHNO) GO TO 3600
1215                IF (CHAR(CTABLE,LOWNO,N) .NE. CHAR(CTABLE,II,N)) GO TO 3000
1216                II = II + 1
1217                GO TO 2800
1218             3000 LARRAY(N) = II
1219                BRANCH(LOWNO) = II
1220                BRANCH(II) = N
1221                N = N + 1
1222                LOWNO = LOWNO + 1
1223                HIGHNO = II - 1
1224             3200 IF (LOWNO .LE. HIGHNO) GO TO 2600
1225                N = N - 1
1226             3400 IF (N .EQ. 0) GO TO 4000
1227                IF (N .LE. 1) GO TO 3500
1228                HIGHNO = LARRAY(N-1) - 1
1229                GO TO 3200
1230             3500 HIGHNO = VHIGH
1231                GO TO 3200
1232             3600 IF (II .NE. LOWNO + 1) GO TO 3800
1233                BRANCH(LOWNO) = - BRANCH(II)
1234                LOWNO = LOWNO + 1
1235                BRANCH(II) = N
1236                GO TO 3400
1237             3800 LARRAY(N) = II
1238                N = N + 1
1239                BRANCH(LOWNO) = -N
1240                LOWNO = LOWNO + 1
1241                GO TO 2600
1242             4000 CONTINUE
1243           C
1244           C     READ RIGHT-CONTEXT TABLE
1245                READ(131,6) NNT,CARDNO
1246                CALL SEQCHK
```

```
1247              DO 4200 N=1,NNT
1248              READ(131,7) NUNTER(N),(RCCLAS(I,N),I=1,4),CARDNO
1249           4200 CALL SEQCHK
1250          C
1251          C       READ STATE TABLES
1252              READ(131,6) NSV,CARDNO
1253              CALL SEQCHK
1254              READ(131,6) NIC,CARDNO
1255              CALL SEQCHK
1256              DO 4400 I=1,NSV
1257              READ(131,8) (TRANS(I,J),J=1,25),CARDNO
1258           4400 CALL SEQCHK
1259              READ(131,6) NDTC,CARDNO
1260              CALL SEQCHK
1261              DO 4600 N =1,NDTC
1262              READ(131,9) NIC,(DECIS(N,I),I=1,NIC),NSV,
1263             *    (CONO(N,I),I=1,NSV),CARDNO
1264           4600 CALL SEQCHK
1265          C
1266          C       READ SIGN TABLE
1267              DO 5000 N = 1,64
1268              READ(131,10) DOTS14(N),DOTS25(N),DOTS36(N),PCHAR(N),MCODE(N),
1269             *    CARDNO
1270           5000 CALL SEQCHK
1271          C
1272          C       READ CONTROL CARDS
1273              READ(131,3) T,CARDNO
1274              CALL SEQCHK
1275              PFOUT = T .EQ. LETP
1276              READ(131,11) T,MEMBON,MEMBOF,MIDLE,MEMBCR,MEMBPG,CARDNO
1277              CALL SEQCHK
1278              EMBCPT = T .EQ. LETM
1279              READ(131,6) OUTL,CARDNO
1280              CALL SEQCHK
1281              READ(131,6) LPG,CARDNO
1282              CALL SEQCHK
1283              READ(131,12) T,(CDIGIT(I),I=1,4),CARDNO
1284              CALL SEQCHK
1285              PAGINA = T .EQ. LETP
1286              IF (SEQERR) STOP 'SEQUENCE ERROR'
```

```
1287                  END


1288                  SUBROUTINE SEQCHK
1289            C     SEQUENCE CHECK FOR TABLES INPUT
1290                  IMPLICIT INTEGER(A-Z)
1291                  COMMON /CHEK/ CARDNO,PREVNO,ECHO,SEQERR
1292                  LOGICAL SEQERR,ECHO
1293                1 FORMAT(' **FOLLOWING CARDS OUT OF SEQUENCE ',I8)
1294            C
1295                  IF (CARDNO .LE. PREVNO) GO TO 100
1296                  PREVNO = CARDNO
1297                  IF (.NOT. ECHO) RETURN
1298                  GO TO 200
1299              100 SEQERR = .TRUE.
1300                  PREVNO = 0
1301              200 WRITE(108,1) CARDNO
1302                  END


1303                  SUBROUTINE TABDIS
1304                  IMPLICIT INTEGER(A-Z)
1305                  COMMON /STV/ STVAR(10),NSV
1306                  COMMON /LOGIC/ DECIS(15,25),COND(15,10),TRANS(10,25),NONTER(2),
1307                 *   RCCLAS(4,2),NDTC,NIC,NNT
1308                 10 FORMAT(///16X,'DECISION TABLE'//'-COLUMN')
1309                 11 FORMAT(///16X,'RIGHT CONTEXT TABLE'//' NON-TERMINAL   INPUT CLASSE
1310                *S'//)
1311                 12 FORMAT(6X,A1,8X,4(I2,2X))
1312                 13 FORMAT(19X,20(2X,I2))
1313                 14 FORMAT(/'-INPUT CLASS')
1314                 15 FORMAT(17X,I2,3X,20(A1,3X))
1315                 16 FORMAT(/'-STATE VARIABLE')
1316                 17 FORMAT(///16X,'TRANSITION TABLE'//'-STATE VARIABLE')
1317                 18 FORMAT(//)
1318                  WRITE(108,11)
1319                  DO 6200 K=1,NNT
1320              6200 WRITE(108,12) NONTER(K),(RCCLAS(I,K),I=1,4)
1321                  WRITE(108,10)
1322                  WRITE(108,13) (K,K=1,NDTC)
```

```
1323                WRITE(108,14)
1324                DO 6400 N=1,NIC
1325           6400 WRITE(108,15) N,(DECIS(K,N),K=1,NDTC)
1326                WRITE(108,16)
1327                DO 6600 N=1,NSV
1328           6600 WRITE(108,15) N,(COND(K,N),K=1,NDTC)
1329                WRITE(108,17)
1330                WRITE(108,13) (K,K=1,NSV)
1331                WRITE(108,14)
1332                DO 6800 N=1,NIC
1333           6800 WRITE(108,15) N,(TRANS(K,N),K=1,NSV)
1334                WRITE(108,18)
1335                END


1336                SUBROUTINE SHIFT(NCHARS)
1337           C    SHIFT BUFFER LEFT BY NCHARS
1338                IMPLICIT INTEGER(A-Z)
1339                COMMON /IN/ INPTR,PRIORS,SCOUNT
1340                COMMON /OUT/ OUTPTR,OUTL,LINES,LPG
1341                COMMON /BUFF/ BUFFER(10)
1342                COMMON /OPTS/ EMBOPT,PFOUT
1343                LOGICAL EMBOPT,PFOUT
1344                COMMON /SPAC/ SPACE
1345                COMMON /CPR/ CPRINT
1346                LOGICAL CPRINT
1347                COMMON /RCC/ RCC
1348                DIMENSION TEXT(20)
1349                COMMON /VAL/ VALUE(64),BVAL(10)
1350                DATA NXTCHR /'    '/
1351              2 FORMAT(X,20A4)
1352              3 FORMAT(' INPUT RECORD LESS THAN 20 WORDS')
1353           C
1354                IF (NCHARS .EQ. 0) RETURN
1355                DO 1400 I=1,NCHARS
1356           S       LI,7       -9
1357           S50     LW,9       BUFFER+10,7
1358           S       STW,9      BUFFER+9,7
1359           S       LW,9       BVAL+10,7
1360           S       STW,9      BVAL+9,7
```

```
1361          S         BIR,7     50S
1362            100 IF (INPTR .LE. 80) GO TO 200
1363                IF (SCOUNT .GT. 0) GO TO 800
1364                CALL BUFFERIN(131,1,TEXT,20,ISTAT,N)
1365                IF (ISTAT .EQ. 3) GO TO 800
1366                IF (N .NE. 20) WRITE(108,3)
1367                INPTR = 1
1368                IF (.NOT. PFOUT) GO TO 200
1369                WRITE(108,2) TEXT
1370                CPRINT = .TRUE.
1371            200 CONTINUE
1372          S         LW,7      INPTR
1373          S         AI,7      -1
1374          S         LB,9      TEXT,7
1375          S         STB,9     NXTCHR
1376          S         MTW,1     INPTR
1377          S         CB,9      SPACE
1378          S         BNE       400S
1379                IF (PRIORS .EQ. 0) GO TO 100
1380                PRIORS = PRIORS - 1
1381                GO TO 600
1382            400 PRIORS = 1
1383            600 IF (NXTCHR .EQ. RCC) GO TO 100
1384                GO TO 1200
1385            800 SCOUNT = SCOUNT + 1
1386                IF (SCOUNT .EQ. 1 .AND. PRIORS .GT. 0) GO TO 1000
1387                NXTCHR = RCC
1388                GO TO 1200
1389           1000 NXTCHR = SPACE
1390           1200 BUFFER(10) = NXTCHR
1391          S         LB,7      NXTCHR
1392          S         AI,7      -1
1393          S         LB,9      VALUE,7
1394          S         STW,9     BVAL+9
1395           1400 CONTINUE
1396                END
```

```
1397              SUBROUTINE OPSIGN(OUTSIG)
1398        C     STORE SIGN ON CURRENT STACK
1399              IMPLICIT INTEGER(A-Z)
1400              COMMON /STAX/ TS,HS,CS,PS,TAILS(2),HEADS(2),CURRS(2),CURTYP,STKIND
1401             *,FFSTAK,ELT(19,30),SPCONT(19),NOELTS(19),PVSTAK(19),NXSTAK(19)
1402              EQUIVALENCE(HEADS1,HEADS(1)),(TAILS1,TAILS(1)),(CURRS1,CURRS(1))
1403              EQUIVALENCE(HEADS2,HEADS(2)),(TAILS2,TAILS(2)),(CURRS2,CURRS(2))
1404              COMMON /OUT/ OUTPTR,OUTL,LINES,LPG
1405              COMMON /PAG/ CDIGIT(4),DIGIT(10),PAGINA
1406              LOGICAL PAGINA
1407              COMMON /BUFF/ BUFFER(10)
1408              COMMON /OPTS/ EMBOPT,PFOUT
1409              LOGICAL EMBOPT,PFOUT
1410              COMMON /S/ LETS
1411              COMMON /TTL/ TTLENG
1412              COMMON /COMP/ COMPUT
1413              COMMON /INDENT/ MARGIN,INHEAD
1414              LOGICAL INHEAD,COMPUT
1415              COMMON /TAB/ TABTYP(9),TABCLM(9),TABULA,LETR,LETD
1416              COMMON /DUB/ DOUBLE
1417              LOGICAL DOUBLE
1418              DATA CCHIGH,SCHIGH/80,89/
1419        C
1420        C     CHARACTER TEST
1421              IF (OUTSIG - 64) 800,400,200
1422          200 IF (OUTSIG .LE. CCHIGH) GO TO 1400
1423              IF (OUTSIG .LE. SCHIGH) GO TO 1800
1424              GO TO 2000
1425        C
1426        C     OUTPUT SPACE
1427          400 IF (STKIND .NE. 2) GO TO 600
1428              CALL CLEAR
1429              RETURN
1430          600 CALL STAKTB
1431              RETURN
1432        C
1433        C     OUTPUT ORDINARY CHARACTER
1434          800 IF (NOELTS(CS) .NE. 30) GO TO 1000
1435              CALL STAKTB
1436              NOELTS(CS) = 1
```

```
1437                GO TO 1200
1438          1000 NOELTS(CS) = NOELTS(CS) + 1
1439          1200 ELT(CS,NOELTS(CS)) = OUTSIG
1440                RETURN
1441               C
1442               C     CARRIAGE CONTROL
1443          1400 IF (NOELTS(CS) .NE. 0) CALL STAKTB
1444                NOELTS(CS) = 30
1445                SPCNT(CS) = OUTSIG
1446                IF (OUTSIG .NE. 72) GO TO 1600
1447                CALL DECODE(1,ELT(CS,3))
1448                CALL SHIFT(1)
1449          1500 IF (STKIND .EQ. 2) STKIND = 5
1450                RETURN
1451          1600 IF (OUTSIG .NE. 68 .AND. OUTSIG .NE. 71) RETURN
1452          1700 CALL DECODE(2,ELT(CS,3))
1453          1750 CALL SHIFT(2)
1454                RETURN
1455               C
1456               C     STACK CONTROL
1457          1800 T = OUTSIG - 80
1458                GO TO (2400,2600,5000,1500,1900,4600,1900,3600,4000),T
1459          1900 RETURN
1460               C
1461               C     MODE CONTROL
1462          2000 IF (OUTSIG .EQ. 93) GO TO 5800
1463                IF (OUTSIG .EQ. 95) GO TO 4800
1464                IF (OUTSIG .EQ. 98) GO TO 5600
1465                IF (OUTSIG .EQ. 97) COMPUT = .NOT. COMPUT
1466                IF (OUTSIG .EQ. 90) DOUBLE = .NOT. DOUBLE
1467                RETURN
1468               C
1469               C     BEGIN HEADING
1470          2400 IF (NOELTS(CS) .NE. 0) CALL CLEAR
1471                CALL CARRIJ(65)
1472                STKIND = 4
1473                RETURN
1474               C
1475               C     END HEADING
1476          2600 H = HEADS1
```

```
1477                M = 0
1478             2800 M = M + NOELTS(N)
1479                IF (SPCONT(N) .EQ. 0 .AND. NOELTS(N) .GT. 0) M = M + 1
1480                IF (N .EQ. TAILS1) GO TO 3000
1481                N = NXSTAK(N)
1482                GO TO 2800
1483             C
1484             3000 OUTPTR = (OUTL - M) / 2 + 2
1485                IF (OUTPTR .LT. 1) OUTPTR = 1
1486                STKIND = 2
1487                INHEAD = .TRUE.
1488                CALL CLEAR
1489                INHEAD = .FALSE.
1490                CALL CARRIJ(65)
1491                RETURN
1492             C
1493             C     BEGIN TITLE
1494             3600 CURTYP = 2
1495                HS = HEADS2
1496                TS = TAILS2
1497                CS = CURRS2
1498                STKIND = 4
1499                M = HS
1500                IF (NOELTS(M) .EQ. 0) RETURN
1501             C     CLEAR OUT PREVIOUS TITLE
1502                NOELTS(M) = 0
1503             3800 IF (HS .EQ. TS) RETURN
1504                CALL FRSTAK(PVSTAK,NXSTAK,TS,TAILS)
1505                CURRS2 = HEADS2
1506                GO TO 3800
1507             C
1508             C     END TITLE
1509             4000 TTLENG = 0
1510                N = HEADS2
1511             4200 TTLENG = TTLENG + NOELTS(N)
1512                IF (SPCONT(N) .EQ. 0 .AND. NOELTS(N) .GT. 0) TTLENG = TTLENG + 1
1513                IF (N .EQ. TAILS2) GO TO 4400
1514                N = NXSTAK(N)
1515                GO TO 4200
1516             C
```

```
1517          4400 CURTYP = 1
1518               HS = HEADS1
1519               TS = TAILS1
1520               CS = CURRS1
1521               STKIND = 2
1522               RETURN
1523          C
1524          C    UNIT OF MEASURE
1525           4600 SPCENT(CS) = 1
1526          C
1527          C    INTERCHANGE LAST TWO ROWS OF CURRENT STACK
1528               N = PVSTAK(TS)
1529               IF (N .EQ. 0) GO TO 4700
1530               M = PVSTAK(N)
1531               PVSTAK(TS) = M
1532               NXSTAK(TS) = N
1533               PVSTAK(N) = TS
1534               NXSTAK(N) = 0
1535               TS = N
1536               TAILS(CURTYP) = N
1537               CS = N
1538               CURRS(CURTYP) = N
1539               IF (N .NE. HS) GO TO 4650
1540               HS = PVSTAK(N)
1541               HEADS(CURTYP) = HS
1542               GO TO 4700
1543           4650 NXSTAK(M) = PVSTAK(N)
1544          C
1545           4700 IF (BUFFER(1) .EQ. LETS) CALL SHIFT(1)
1546               RETURN
1547          C
1548          C    MARGIN RESET
1549           4800 CALL DECODE(2,MARGIN)
1550               GO TO 1750
1551          C
1552          C    RESTART
1553           5000 DO 5200 N=1,3
1554           5200 CDIGIT(N) = 0
1555               CDIGIT(4) = 1
1556               CALL RESET
```

```
1557              RETURN
1558          5500 CALL CARRI_(5)
1559              ENDFILE 121
1560              STOP 'OK'
1561          5800 CALL DECODE(1,T)
1562              TABTYP(T) = BUFFER(2)
1563              CALL SHIFT(2)
1564              CALL DECODE(2,TABCLM(T))
1565              CALL SHIFT(2)
1566              RETURN
1567              END


1568              SUBROUTINE CARRIJ(OUTSIG)
1569              IMPLICIT INTEGER(A-Z)
1570              COMMON /OUT/ OUTPTR,OUTL,LINES,LPG
1571              COMMON /LC/ LCOUNT
1572              COMMON /DUB/ DOUBLE
1573              LOGICAL DOUBLE
1574          C
1575              IF (LCOUNT .GE. LPG) CALL PAGEHD
1576              IF (OUTPTR .EQ. 1 .AND. (OUTSIG .EQ. 65 .OR. OUTSIG .EQ. 67))
1577             *    GO TO 600
1578              CALL OUTOUT(OUTSIG)
1579              IF (DOUBLE .AND. LCOUNT .LT. LPG) CALL OUTOUT(OUTSIG)
1580            600 IF (OUTSIG .NE. 67) GO TO 800
1581              OUTPTR = 3
1582              RETURN
1583            800 OUTPTR = 1
1584              END


1585              SUBROUTINE CLEAR
1586          C       CLEAR CURRENT STACK
1587              IMPLICIT INTEGER(A-Z)
1588              COMMON /STAX/ TS,HS,CS,PS,TAILS(2),HEADS(2),CURRS(2),CURTYP,STKIND
1589             *,HFSTAK,ELT(19,30),SPCONT(19),NOELTS(19),PVSTAK(19),NXSTAK(19)
1590              COMMON /OUT/ OUTPTR,OUTL,LINES,LPG
1591              COMMON /TAB/ TABTYP(9),TABCLM(9),TABULA,LETR,LETD
1592              COMMON /INDENT/ MARGIN,INHEAD
```

```
1593            LOGICAL INHEAD
1594          C
1595           100 PS = HS
1596               NPS = NOELTS(PS)
1597           400 IF (SPCONT(PS) .LT. 64) GO TO 3000
1598               OUTSIG = SPCONT(PS)
1599               EP3 = ELT(PS,3)
1600               IF (OUTSIG .NE. 68) GO TO 600
1601          C    ONE-TIME TABULATION
1602               TABULA = EP3
1603               GO TO 4100
1604           600 IF (OUTSIG .NE. 71) GO TO 1800
1605          C    SKIP LINES
1606               CALL CARRIJ(65)
1607               CALL PCLEAR
1608               DO 1600 M=1,EP3
1609               OUTPTR = 0
1610          1600 CALL CARRIJ(71)
1611               GO TO 4200
1612          C
1613          C    PRESET TABULATION
1614          1800 IF (OUTSIG .NE. 72) GO TO 2800
1615               M = ELT(PS,3)
1616               XX = NXSTAK(PS)
1617               IF (TABTYP(M) .EQ. R) GO TO 2000
1618               IF (TABTYP(M) .EQ. D) GO TO 2200
1619               TABULA = TABCLM(M)
1620               GO TO 4100
1621          C
1622          2000 TABULA = TABCLM(M) + NOELTS(XX) + 1
1623               GO TO 4100
1624          C
1625          2200 T = NOELTS(XX)
1626               DO 2400 XYZ=1,T
1627          2400 IF (ELT(XX,XYZ) .EQ. 40) GO TO 2600
1628          2600 TABULA = TABCLM(M) + XYZ + 1
1629               GO TO 4100
1630          C
1631          2800 CALL CARRIJ(OUTSIG)
1632               GO TO 4200
```

```
1633          C
1634           3000 IF (NPS .EQ. 0) GO TO 4200
1635                IF (OUTPTR + NPS .LT. OUTL + 2) GO TO 3800
1636                OUTPTR = OUTL + 1
1637                CALL CARRIJ(0)
1638                IF (MARGIN .LE. 0) GO TO 3400
1639                DO 3200 N=1,MARGIN
1640           3200 CALL MOVEEL(64)
1641           3400 IF (.NOT. INHEAD) GO TO 3800
1642                DO 3600 N=1,3
1643           3600 CALL MOVEEL(64)
1644           3800 DO 4000 M=1,NPS
1645           4000 CALL MOVEEL(ELT(PS,M))
1646                IF (SPCONT(PS) .NE. 1) CALL MOVEEL(64)
1647                GO TO 4200
1648          C
1649          C     TABULATION
1650           4100 TABULA = TABULA - OUTPTR
1651                IF (TABULA .LE. 0) GO TO 4120
1652                DO 4110 N=1,TABULA
1653           4110 CALL MOVEEL(64)
1654                GO TO 4140
1655           4120 TABULA = TABULA + OUTPTR - 1
1656                CALL CARRIJ(65)
1657                DO 4130 N=1,TABULA
1658           4130 CALL MOVEEL(64)
1659           4140 TABLLA = 0
1660          C
1661           4200 IF (HS .EQ. TS) GO TO 4800
1662                CALL FRSTAK(NXSTAK,PVSTAK,HS,HEADS)
1663                GO TO 100
1664           4800 NOELTS(HS) = 0
1665                SPCONT(HS) = 0
1666                CURRS(CURTYP) = HS
1667                CS = HS
1668                END
```

```
1669              SUBROUTINE FRSTAK(STAK,STAK1,HT,HTS)
1670              IMPLICIT INTEGER(A-Z)
1671              COMMON /STAX/ TS,HS,CS,PS,TAILS(2),HEADS(2),CURRS(2),CURTYP,STKIND
1672             *,HFSTAK,ELT(19,30),SPCONT(19),NOELTS(19),PVSTAK(19),NXSTAK(19)
1673              DIMENSION STAK(19),STAK1(19),HTS(2)
1674          C
1675              NOELTS(HT) = 0
1676              SPCONT(HT) = 0
1677              IF (STAK(HT) .EQ. 0) RETURN
1678              M = HT
1679              N = HFSTAK
1680              HT = STAK(M)
1681              HTS(CURTYP) = HT
1682              HFSTAK = M
1683              STAK1(HT) = 0
1684              NXSTAK(M) = N
1685              END


1686              SUBROUTINE MOVEEL(X)
1687          C   MOVE SIGN TO OUTPUT BUFFER
1688              IMPLICIT INTEGER(A-Z)
1689              COMMON /STAX/ TS,HS,CS,PS,TAILS(2),HEADS(2),CURRS(2),CURTYP,STKIND
1690             *,HFSTAK,ELT(19,30),SPCONT(19),NOELTS(19),PVSTAK(19),NXSTAK(19)
1691              COMMON /OUT/ OUTPTR,OUTL,LINES,LPG
1692              COMMON /SIGNS/DOTS14(64),DOTS25(64),DOTS36(64),PCHAR(64),MCODE(64)
1693              COMMON /OWAREA/ BRAILL(3,40),PROOF(40),CODE(40)
1694              COMMON /OPTS/ EMBOPT,PFOUT
1695              LOGICAL EMBOPT,PFOUT
1696          C
1697              CODE(OUTPTR) = X
1698              IF (.NOT. PFOUT) GO TO 200
1699              IF (X .EQ. 0) X = 64
1700              BRAILL(1,OUTPTR) = DOTS14(X)
1701              BRAILL(2,OUTPTR) = DOTS25(X)
1702              BRAILL(3,OUTPTR) = DOTS36(X)
1703              PROOF(OUTPTR) = PCHAR(X)
1704          200 OUTPTR = OUTPTR + 1
1705              END
```

```
1706                SUBROUTINE OUTOUT(OUTSIG)
1707          C     WRITE OUTPUT BUFFER
1708                IMPLICIT INTEGER(A-Z)
1709                COMMON /OUT/ OUTPTR,OUTL,LINES,LPG
1710                COMMON /OPTS/ EMBOPT,PFOUT
1711                LOGICAL EMBOPT,PFOUT
1712                COMMON /SIGNS/DOTS14(64),DOTS25(64),DOTS36(64),PCHAR(64),MCODE(64)
1713                COMMON /OWAREA/ BRAILL(3,40),PROOF(40),CODE(40)
1714                COMMON /MEMB/ EMBOUT(20),MEMBCR,MEMBPG,MIDLE
1715                COMMON /CPR/ CPRINT
1716                LOGICAL CPRINT
1717                COMMON /LC/ LCOUNT
1718              1 FORMAT(80X)
1719              2 FORMAT(X,40A3)
1720              3 FORMAT(40(X,I2))
1721              4 FORMAT(20A4)
1722              5 FORMAT(' TOO MUCH INPUT - TRANSLATION INCOMPLETE')
1723          C
1724                IF (.NOT. PFOUT) GO TO 400
1725                IF (.NOT. CPRINT) WRITE(108,1)
1726                CPRINT = .FALSE.
1727                DO 200 N=1,3
1728            200 WRITE(108,2) (BRAILL(N,J),J=1,OUTL)
1729                WRITE(108,2) PROOF
1730                WRITE(108,3) (CODE(J),J=1,OUTL)
1731            400 IF (.NOT. EMBOPT) GO TO 1400
1732                DO 600 N=1,39
1733                IF (N .GT. OUTL) GO TO 1300
1734                T = CODE(N)
1735                IF (T .EQ. 0) T = 64
1736          C     ICH(EMBOUT,N+2) = ISL(MCODE(T),-24)
1737          S        LW,7       T
1738          S        AI,7       -1
1739          S        MI,7       4
1740          S        LB,9       MCODE,7
1741          S        LW,7       N
1742          S        AI,7       1
1743          S        STB,9      EMBOUT,7
1744            600 CONTINUE
1745           1300 CALL BUFFEROUT(132,1,EMBOUT,20,ISTAT)
```

```
1746                IF (ISTAT .EQ. 2) GO TO 1400
1747                WRITE(108,5)
1748          S        CAL1,9    3            ABORT
1749           1400 CALL PCLEAR
1750                IF (OUTSIG .NE. 69) GO TO 1800
1751                LCOUNT = LPG
1752                RETURN
1753           1800 LCOUNT = LCOUNT + 1
1754                END


1755                SUBROUTINE PAGEHD
1756                IMPLICIT INTEGER(A-Z)
1757                COMMON /STAX/ TS,HS,CS,PS,TAILS(2),HEADS(2),CURRS(2),CURTYP,STKIND
1758               *,HFSTAK,ELT(19,30),SPCONT(19),NOELTS(19),PVSTAK(19),NXSTAK(19)
1759                COMMON /OUT/ OUTPTR,OUTL,LINES,LPG
1760                COMMON /OWAREA/ BRAILL(3,40),PREOF(40),CODE(40)
1761                COMMON /TTL/ TTLENG
1762                COMMON /LC/ LCOUNT
1763                COMMON /PAG/ CDIGIT(4),DIGIT(10),PAGINA
1764                LOGICAL PAGINA
1765                COMMON /OPTS/ EMBOPT,PFOUT
1766                LOGICAL PFOUT,EMBOPT
1767                COMMON /MEMB/ EMBOUT(20),MEMBCR,MEMBPG,MIDLE
1768                COMMON /CPR/ CPRINT
1769                LOGICAL CPRINT
1770                COMMON /DUB/ DOUBLE
1771                LOGICAL DOUBLE
1772                DIMENSION BTEMP(3,40),CTEMP(40),PTEMP(40)
1773              1 FORMAT(80X)
1774              2 FORMAT(11A4)
1775          C
1776                IF (.NOT. PAGINA) RETURN
1777                IF (PFOUT) WRITE(108,1)
1778                IF (.NOT. EMBOPT) GO TO 100
1779          C     ICH(EMBOUT,3) = ISL(MEMBPG,-24)
1780          S        LB,9      MEMBPG
1781          S        LI,7      2
1782          S        STB,9     EMBOUT,7
1783                DO 50 N=4,40
```

```
1784          C      ICH(EMBOUT,N) = ISL(MIDLE,-24)
1785          S         LB,9       MIDLE
1786          S         LW,7       N
1787          S         AI,7       -1
1788          S         STB,9      EMBOUT,7
1789             50 CONTINUE
1790                CALL BUFFEROUT(132,1,EMBOUT,20,ISTAT)
1791            100 LCOUNT = 0
1792                CPRINT = .FALSE.
1793                N = (OUTL - TTLENG + 2) / 2 + 1
1794                IF (N .LT. 1) N = 1
1795          C     SAVE CONTENTS OF OUTPUT BUFFER
1796                DO 200 I=1,OUTL
1797                PTEMP(I) = PROOF(I)
1798                CTEMP(I) = CODE(I)
1799                DO 200 J=1,3
1800            200 BTEMP(J,I) = BRAILL(J,I)
1801                CALL PCLEAR
1802                OSAVE = OUTPTR
1803                OUTPTR = N
1804                N = HEADS(2)
1805            400 IF (NOELTS(N) .EQ. 0 .OR. SPCONT(N) .GE. 64) GO TO 800
1806                T = NOELTS(N)
1807                DO 600 XYZ = 1,T
1808            600 CALL MOVEEL(ELT(N,XYZ))
1809                IF (SPCONT(N) .EQ. 0) CALL MOVEEL(64)
1810            800 IF (N .EQ. TAILS(2)) GO TO 1000
1811                N = NXSTAK(N)
1812                IF (OUTPTR + NOELTS(N) .LE. OUTL - 5) GO TO 400
1813           1000 OUTPTR = OUTL - 4
1814                DO 1200 N=1,4
1815                IF (CDIGIT(N) .NE. 0) GO TO 1400
1816           1200 OUTPTR = OUTPTR + 1
1817           1400 CALL MOVEEL(60)
1818                DO 1600 N=N,4
1819                XYZ = CDIGIT(N) + 1
1820           1600 CALL MOVEEL(DIGIT(XYZ))
1821          C
1822          C     INCREMENT PAGE NUMBER
1823                DO 1800 N=4,1,-1
```

```
1824              CDIGIT(N) = CDIGIT(N) + 1
1825              IF (CDIGIT(N) .LT. 10) GO TO 2000
1826           1800 CDIGIT(N) = 0
1827           2000 CALL OUTOUT(0)
1828              IF (DOUBLE) CALL OUTOUT(0)
1829           C  RESTORE OUTPUT BUFFER
1830              OUTPTR = OSAVE
1831              DO 2200 I=1,OUTL
1832              CODE(I) = CTEMP(I)
1833              PROOF(I) = PTEMP(I)
1834              DO 2200 J=1,3
1835           2200 BRAILL(J,I) = BTEMP(J,I)
1836              END


1837              SUBROUTINE STAKTB
1838           C  SET POINTERS TO INTRODUCE A NEW ROW IN THE CURRENT STACK
1839              IMPLICIT INTEGER(A-Z)
1840              COMMON /STAX/ TS,HS,CS,PS,TAILS(2),HEADS(2),CURRS(2),CURTYP,STKIND
1841             *,HFSTAK,ELT(19,30),SPCONT(19),NOELTS(19),PVSTAK(19),NXSTAK(19)
1842           C
1843              IF (HFSTAK .EQ. 0) RETURN
1844              N = HFSTAK
1845              HFSTAK = NXSTAK(N)
1846              NXSTAK(N) = 0
1847              PVSTAK(N) = TS
1848              NXSTAK(TS) = N
1849              TAILS(CURTYP) = N
1850              TS = N
1851              CURRS(CURTYP) = N
1852              CS = N
1853              IF (STKIND .EQ. 5) STKIND = 2
1854              END


1855              SUBROUTINE PCLEAR
1856           C  RESET PROOF OUTPUT BUFFER TO BLANKS
1857              IMPLICIT INTEGER(A-Z)
1858              COMMON /OWAREA/ BRAILL(3,40),PROOF(40),CODE(40)
1859              COMMON /SPAC/ SPACE
```

```
1860          C
1861                DO 200 I=1,40
1862                CODE(I) = 0
1863                PROOF(I) = SPACE
1864                DO 200 J=1,3
1865            200 BRAILL(J,I) = SPACE
1866                END


1867                SUBROUTINE RESET
1868                IMPLICIT INTEGER(A-Z)
1869                COMMON /STV/ STVAR(10),NSV
1870                COMMON /TTL/ TTLENG
1871                COMMON /N/ LETN
1872                COMMON /INDENT/ MARGIN,INHEAD
1873                COMMON /COMP/ COMPUT
1874                COMMON /DUB/ DOUBLE
1875                LOGICAL DOUBLE
1876                LOGICAL INHEAD,COMPUT
1877          C
1878                TTLENG = 0
1879                MARGIN = 0
1880                DOUBLE = .FALSE.
1881                COMPUT = .FALSE.
1882                INHEAD = .FALSE.
1883                DO 100 I=1,NSV
1884            100 STVAR(I) = LETN
1885                END


1886                SUBROUTINE DECODE(N,Y)
1887          C     DECODES FIRST N CHARACTERS OF BUFFER TO GIVE INTEGER IN Y
1888                IMPLICIT INTEGER(A-Z)
1889                COMMON /BUFF/ BUFFER(10)
1890          C
1891                Y = 0
1892                DO 200 I=1,N
1893            200 Y = 10 * Y + IAND(15,ICH(BUFFER,4*I-3))
1894                END
```

APPENDIX 2

EVALUATION

In order to evaluate the system a small-scale pilot service was operated for producing single copies in braille of short documents for blind subjects. The subjects were chosen from those who approached Warwick Research Unit for the Blind asking for documents to be transcribed into braille. In the first few months of operation the system was undergoing almost continual modification based on informal feedback from the blind subjects which made it impractical to start any formal evaluation programme.

Examples of experimental material produced includes:

Domestic

Washing machine instructions
Blender/Grinder - instructions and recipes
Refrigerator instructions
Automatic dishwasher instructions
Recipes
Diabetic recipes
Electric cooker instructions
Instructions for record player
Instructions for Dritz bound button hole maker
Russell Hobbs coffee percolater
Sanyo Stereo Set
Teletron Cassette Tape Recorder/Radio
Habitat Chicken Brick
Sinclair 5-function calculator
Information on freezing food.

Leisure

Record cover - classical
Times Literary Article

Knitting patterns

List of holiday bungalows

Poetry/songs

"Coffee Pot" activities

Royal Northern College of Music - Programme

Arts Centre, University of Warwick - Programme

Leaflet on hotels

Record sleeves

List of monthly record releases

Religious

Christmas Carols for vicar

Methodist Church Services

Catholic Mass Sheet

Methodist Church - Aylesbury circuit plan

Education

Song sheets

Sociology "O" level handouts

English handouts

Lesson notes - primary school

Examination papers (old)

Bibliographies

City & Guilds Certificate in Further Education - Information

TUC Training College handouts

Employment

Non-accidental injury pamphlet - social worker

Open University material

ATTI leaflet

University library - information

Agendas

Minutes

APEX handout / Musician's Union handout

Letters

Dialling Codes

Reports

Bibliographies

NASWB

International Dialling Codes

Miscellaneous

Minutes and agendas of voluntary organisations
Bus Timetables
Train Timetables
Mortgage Statement
BBC Radio Birmingham
Disabled Parking Stickers - Information
Children's stories
Call Charges
Local dialling codes
Newspaper articles
Birth pill instructions

The proportion of the experimental material in each group is:

Subject	%
Domestic	10.4
Leisure	15.4
Religious	2.5
Education	25.4
Employment	29.1
Miscellaneous	17.2

Errors can be caused by typing errors in the text, incorrect control characters, incorrect choice of contraction by the computer program, malfunction of the embosser or damage to the braille in transit. Little is gained by just measuring the number of miscontractions caused by the translation program unless one also measures the error rates for the various systems using manual transcribers. Once the error rate is relatively low, it becomes a very expensive operation to attempt to measure it accurately.

A practical measure of the error rate is the number of users who find the number and types of errors unacceptable for their application. For instance any error in a list of telephone numbers is serious, but a miscontraction may often go unnoticed.

A braille questionnaire was circulated to subjects who had used material generated by the most recent system. The results are (N=23):

1. Mean age = 40.3 years (σ = 12.6 years)

2. Occupations

Professional	9
Non-professional white-collar	7
Students	2
Manual workers	1
Not employed	4

"The following 4 questions use a 1 to 5 scale for answering. For example 1 is very poor, 2 poor, 3 average, 4 good and 5 very good. You should just write down the number which best describes your own opinion. In these questions I have just given you the end points on the scale although you can answer with any number between 1 and 5".

	mean	σ
3. For your applications, the turnround time is: (1 is so slow as to make the service useless and 5 is perfectly acceptable).	3.8	0.9
4. Are the miscontractions caused by the computer program: (1 is so bad as to stop you using the service and 5 is perfectly acceptable).	4.6	0.6
5. Is the physical quality of the braille: (1 is very poor and 5 is good).	4.3	0.9
6. Is single-sided, as compared with double-sided, embossing: (1 is so severe a disadvantage as to stop you using the service and 5 is perfectly acceptable).	4.9	0.3

7. If there was a central document transcription service, or a number of regional ones, how much would you use it on average (braille pages per month):

1. less than 10
2. 10 to 50
3. 50 to 100
4. 100 to 1000
5. more than 1000

mean = 2.5

σ = 0.9.

8. Would you want the document service to transcribe specialist braille codes such as music, mathematics and computing? If so, which ones?

Code	Number
Music	6
Mathematics	1
Computing	1

9. How many embossed diagrams would you want per month (on average):

1. none
2. less than 1
3. 1 to 100
4. 10 to 100
5. more than 100

Mean = 1.4

σ = 0.6.

APPENDIX 3

FURTHER RELATED RESEARCH PROJECTS

This appendix lists some of the areas in which further substantial research may be expected to yield significant results of value to the ultimate user. Certain items are already being investigated, and reference should be made to the International Register of Research on Blindness and Visual Impairment (Gill, 1975) for details. This list should not be taken as implying that the short document service should be suspended or that the prime object of this study has been achieved. Rather it should be regarded as an indication of the continuing program envisaged as a result of discussions, and comments and experience from some four years of work in this area and from the particular information gained whilst undertaking the project described in the main body of this report.

1. Motivation in braille learning

Time has not permitted an investigation on whether the provision of material of particular interest to an individual can help motivate him to learn braille. This study would necessitate particular assistance from other experts, notably in the field of education.

2. Formal definition of braille

A formal definition of contracted braille would facilitate the writing of translation programs. A considerable literature exists on the formal definition of natural languages and has assisted the machine translation of one language to another (e.g. Russian to English). No such complex analysis has been undertaken for translation into braille, where similar advantages may result. It may be that successful completion of this formal definition would ease the production of translation of many other natural languages to braille.

3. Mathematics

The input processor and DOTSYS should be developed in order to handle the English mathematics braille code.

4. Choral Music

Design of an interactive system for the input and translation of simple choral music by DOTSYS. Unlike existing systems and systems under development, computer based editing is envisaged as a vital element of this proposal.

5. Tables

Design of an interactive system to assist the operator in the layout of tables when using DOTSYS.

6. New page control

The operator should have an extra instruction for DOTSYS which checks the amount of space left on the page to ensure that a table is not unnecessarily split between two pages.

7. Fill-in character

For tables it would be sometimes useful to replace blanks by a fill-in character such as a dash. DOTSYS could be modified to do this automatically.

8. Alternate page inversion

DOTSYS should be modified to include a post-processor to handle alternate page inversion which could make significant financial savings, by minimising the amount of manual binding work required.

9. Financial information

With the increasing use of digital systems by financial organisations, more financial information could be produced automatically in braille, e.g. bank statements. This does not usually require a pre-processor to a braille translation program such as DOTSYS since the quantity of text is often minimal, and much of it is drawn from a limited vocabulary of standard words.

10. Telephone directories

Many organisations store the data for their internal telephone directories in digital form. So it would be possible to automatically generate braille editions either alphabetically or by department.

11. Computer-based composing systems

Some printers are using computer-based systems for the generation and correction of text. These systems often incorporate features which are very useful for braille translation, e.g. different symbols for apostrophe and single quotation mark. With a suitable pre-processor, these tapes appear to be ideal for braille production requiring only that the operator specifies the layout for tables, which is best done interactively.

12. Design of data structures

Sometimes the braille producer can specify desirable features when a data structure is being formulated. For instance magazines such as New Beacon, are produced in inkprint as well as braille, so the use of one database for both editions could offer significant savings. It should be possible in many cases to completely eliminate the currently duplicated data preparation and proof reading.

13. Automatic indexing

A modification to DOTSYS that would be useful for reference books is a facility to allow the typist to flag terms required in the index and for DOTSYS to automatically generate the index.

14. Double indexing

When computer-generated compositor's tapes are available for books, some users would find it advantageous to have a double indexing system giving both inkprint and braille page numbers; this could be done automatically with the development of suitable software.

15. Current awareness services

Pre-processors need to be developed in order to selectively produce braille listings from the standard databases such as INSPEC. It should be possible to distribute the individual braille listings no later than the publication of the inkprint abstract journal.

16. Retrospective abstract services

It would also be technically possible to offer a retrospective service using these same databases. However the choice of index terms becomes more critical so an information broker may be necessary.

17. Small-configuration DOTSYS

It would significantly affect the economics of computerised braille production in both the developed and developing countries if DOTSYS were modified to run on a minicomputer or a microprocessor.

18. Compiler-compiler

It has been proposed that a braille translation program, based on the compiler-compiler principle, could be most beneficial particularly for facilitating a language-independent translator.

19. Design of tables

Simple tables can be translated into braille automatically but this is not true for complex tabular presentations. For instance it has been found preferable to use an inverted file structure when translating a scan-column index into braille. The design of tables is a large area deserving of research effort.

20. Design of embossed diagrams

So little research has been done on the optimum design of embossed diagrams that it is impossible even to list all that still has to be done.

21. Double-sided embosser

The development of a digitally-controlled high-speed double-sided braille embosser will soon become a matter of urgency.

22. Stereotype machine

There will also be an increasing demand for a high-speed digitally-controlled braille stereotype machine. Since the potential market is small and the development costs high, it will have to be undertaken as an international project with guaranteed sales.

APPENDIX 4

"SOME DEVELOPMENTS IN COMPUTER-AIDED INFORMATION SERVICES FOR THE BLIND".

by Professor J.L. Douce

Chairman's address to the Control and Automation Division of the Institution of Electrical Engineers on 7th October 1975.

To be published in the Proceedings of the Institution of Electrical Engineers.

SOME DEVELOPMENTS IN COMPUTER-AIDED INFORMATION SERVICES FOR THE BLIND

John L. Douce, DSc FIEE

Abstract

This paper reviews the needs of the blind community for printed material, and surveys current developments in the applications of automation to increase the availability of braille. The problems of automatic translation of English text to braille are considered.

Some possible systems for giving a greatly improved access to braille are surveyed and an automated system for map and diagram production is reviewed.

The future provision of information services for the blind, using special systems or existing systems with special terminals, is discussed. The technical problems are not trivial but collaborative ventures which, with international cooperation, could offer solutions in a relatively short time-scale.

* - * - * - *

Professor Douce is Professor of Electrical Science at the University of Warwick and Director of the Inter-University Institute of Engineering Control.

INTRODUCTION

One of the most significant handicaps resulting from blindness is the severely restricted access to information of many kinds particularly to that conveyed to sighted people by the printed text and diagram and the medium of television. At the present time, significant developments are occurring in the application of automation to augment the supply and accessibility of this type of information needed by the blind for the purposes of employment and leisure activities. This paper reviews some of the current problems with regard to the provision of textual and diagramatic material, and examines possible future systems and their potential applications.

1. THE BLIND POPULATION

A person may be registered as blind in the U.K. if he is unable to read at a distance of three feet an optical test card which can be read with normal vision at a distance of sixty feet. It is perhaps more meaningful for the purposes of this discussion, to consider the term 'blind' as applying to that section of the community who find conventional ink-print unacceptably difficult to read, and for whom braille or similar embossed material offers the only useful printed reading material. Perhaps the impact of the tape recorder has been overestimated as an information storage device, due to the lack of indexing capabilities and the problems of handling numerical information.

Statistics concerning the visually handicapped section of the population cannot be presented with the desirable accuracy, since the definitions of 'blind' and 'partially sighted' are not particularly closely related to the group involved, and it is believed that many people who are eligible for registration do not register for a variety of reasons. However, appreciation of the size and age distribution of the population may be obtained from Table 1 and Figure 1, which show the age distribution of the 106,000 blind persons registered in England and Wales in 1974.

From this table, and noting also that over 60% of the newly registered blind are aged 75 or over, age alone is likely to result in the majority of the group not learning braille. Furthermore, almost 50% of the children born blind suffer additional handicaps. The overall result is that only about 10% of the registered blind read braille regularly and competently. Thus it must be appreciated that even a non-specialised aid such as braille cannot be expected to aid the majority of the blind.

Table 1. Total blind registrations on 31st March 1974 in England and Wales.

Age Group	Male	Female
under 21 yrs	1783	1419
21 - 39	3177	2252
40 - 59	7378	6071
60 - 74	11986	15543
over 75 yrs	15981	39884

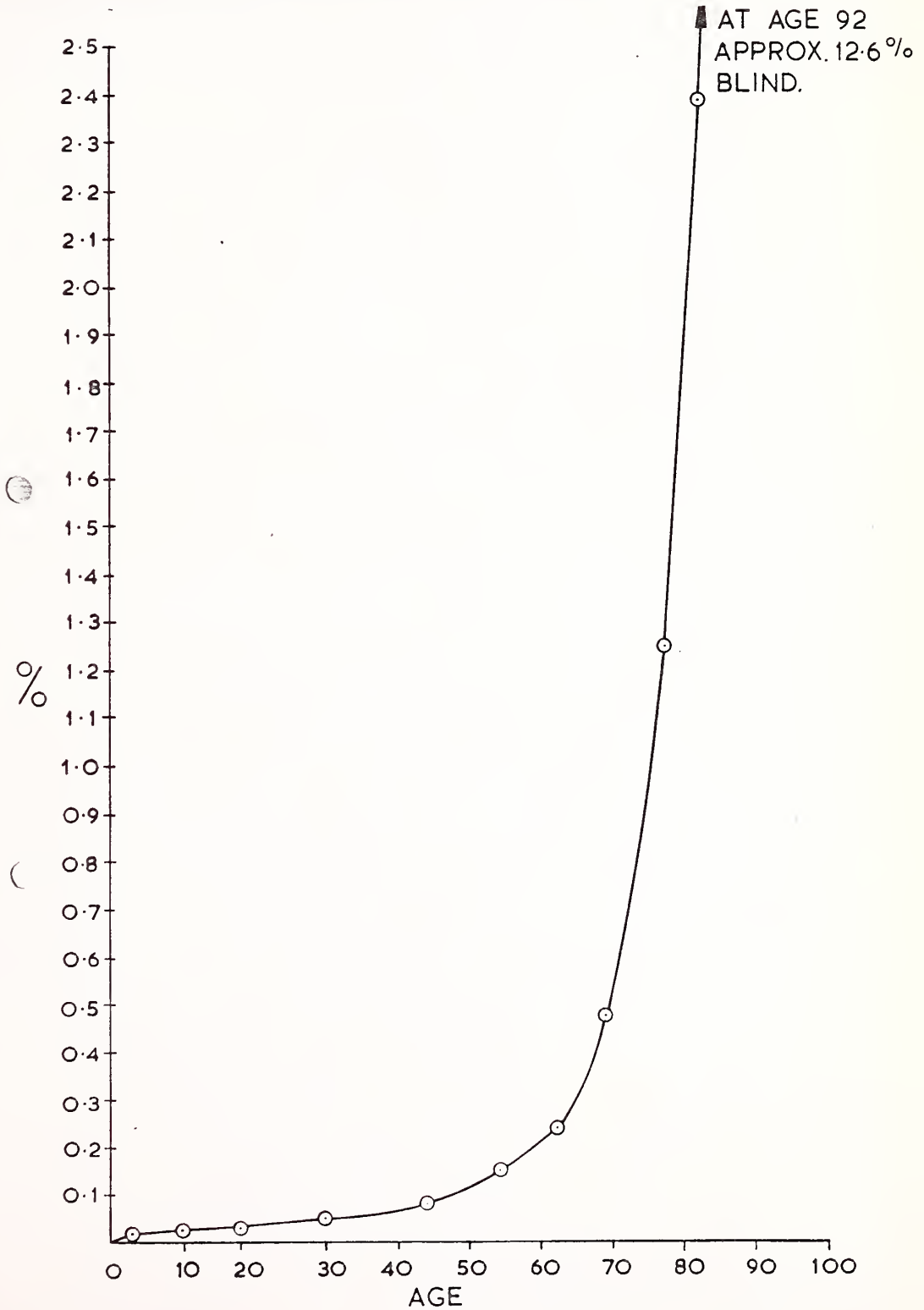


Fig 1. % of each age group (England and Wales), for 1974 of registered blind

When the blind population is examined by professional employment, we find only a tiny fraction of the population in this category, and even the most accessible occupations, such as physiotherapy and teaching have only of the order of one hundred people in each group. The traditional approach has been to find jobs which do not require vision, rather than developing aids to enable a blind person to do the job of his choice.

Thus it may be concluded that the demand for specialised aids confined to one age group or one professional occupation is small. The most useful devices and systems are likely to be versatile and able to assist relatively large numbers of users at an acceptable cost.

This is not to minimise the usefulness of many aids currently available, often developed by talented workers with much voluntary effort and expense. It is rather to attempt to devote effort where the highest cost-effectiveness may be anticipated.

At present, limited information access seems to be the common theme of many frustrations and disadvantages of blindness. With the increasing use of digital data bases and distributed computer terminals, the situation could easily deteriorate in the near future in the absence of a directed program of research and development.

It seems inevitable that braille will be used in increasing amounts in the future, in spite of its deficiencies, and it is appropriate to consider the achievements and possible developments automation offers in this area in some detail.

2. BRAILLE

The most widely used medium for printed material, including books of text, mathematics, music and personal notes and letters is based on the braille cell, developed one hundred and fifty years ago by the Frenchman Louis Braille. It is remarkable how little the fundamental information element has changed since its origination, and the basic design seems close to achieving maximum information density.

The braille cell consists of six dot positions, Figure 2, spaced 2.5mm apart, with adjacent cells spaced 4mm apart.

Figure 2.

Information is conveyed by raising certain dot positions, so that in principle, 64 different patterns are available. These cells are packed at about 1.6 per sq. cm., which should be compared with ink-print text (for which a greater character set is available) with a packing density of the order of 22 characters per square cm. The ratio of these two figures, in conjunction with the thick paper necessary for braille text explains the weight, volume and to some extent the cost of the library of a blind professional person. For example the bible, which contains 773,746 words, takes up 72 braille volumes but NCR have a sighted version on a 50 mm square microform. At present, there is no braille equivalent of high density storage media although the need is obvious.

Grade I braille uses one cell pattern for each letter of the alphabet, with some of the spare patterns for simple punctuation signs and abbreviations of a few very common words. Thus it is a trivial matter to automate the translation of text to Grade I braille since a small look-up table is all that is required.

The automatic translation of text to English (or American) Grade II braille, rather than Grade I, presents formidable difficulties if perfection is desired. To appreicate the nature of the difficulties, some illustrative aspects are worth remarking. Following standard practice, groups of letters represented by a group of one or more braille signs are printed in capital letters, with an oblique stroke inserted where necessary for clarity to separate adjacent contractions written with no separating space.

Grade II braille makes extensive use of contractions to reduce the length of text, with a resulting reduction in the number of pages of braille and, for skilled users, in the reading (and writing) time.

One simple and obvious step is to use the letters of the alphabet to represent whole words, and 23 letters are thus employed. Since the alphabet requires only 26 of the 63 available braille signs there are 37 signs available for common words (e.g. AND/FOR/OF/THE/WITH), punctuation, and for contractions of common groups of letters (e.g. CH/WH/COM/DIS/ING).

Many of these rules are simple to formulate and hence to program, but difficulties of various magnitudes soon become apparent.

For example, some contractions may only be used at the beginning of a word or at the beginning of a braille line. This means that the format of the output print must be considered during the translation phase. More difficult, in automatic translation, some contractions can only be used when they represent a complete syllable - for example DIS may be used in DISTURB but not in diSHes. Other contractions may only be used where the pronunciation is unchanged. Thus UNDER may be used in blUNDER but not in laundER. On the other hand ONE may be used as a contraction when all the three letters it represents are pronounced as a single syllable - ST/ONEs but not anemone. Some, but not all, braille abbreviations may only be written within longer words where the sense of the contracted part is

unchanged. Thus the abbreviation for 'after' may not be used in 'rafter' nor that for 'across' used in 'lacrosse'. Even the general rule for dividing words at the end of a line, 'divide between syllables', is fraught with difficulty for the mechanised translator.

In spite of these difficulties, the savings achieved with Grade II braille, resulting from the reduction of about 40% in the number of cells required, has encouraged many groups to program the conversion. Perhaps surprisingly, at the current time, cost saving, projected or actual, is not the prime motivation for automation. The main reason is at present the lack of skilled braillists to meet a greatly increasing demand.

The possibility of automatically translating text to contracted braille has attracted considerable attention in the last decade. The developers of these systems foresee many potential advantages, such as consistent results achievable by automatic translation and fast turnround, but most of these are only now being investigated at the pilot study stage.

Two types of computer program have been developed, the dictionary-based conversion in which of the order of 80,000 common words are stored, and smaller programs which try to translate by systematic application of the ordered rules and exceptions, and exceptions to the exceptions, with a relatively modest dictionary. Most programs are in high-level languages such as FORTRAN, COBOL or PL1, and modest efforts at international collaboration have been undertaken. It should be noted, however, that Grade II braille is not an international standard - even the conversion of an American to English grade II translation program is not trivial. On a medium-sized machine available at Warwick, a translation speed of about 5000 words per minute is currently being achieved, using 13k words of store.

There has been pressure from diverse groups for changes in the rules or semantics of grade II braille, and the increased interest in automated

translation is giving added impetus to this minor revolution. Other countries have initiated this change, apparently successfully. The main motivation for changing some of the rules of grade II braille is to make it easier to learn by blind readers and sighted transcribers.

For example it may be greatly regretted that the very valuable single letter k is devoted to the relatively infrequent work KNOWLEDGE, and a simplification of the use of contractions could well aid both the user and the translator. The revision of braille however is a surprisingly emotional subject and has led in the past to bitter argument and to the permanent rupture of close friendships. It must also be borne in mind that a great amount of braille is still provided by a voluntary labour force built up over many years, and the introduction of a new system with scientifically chosen and evaluated contractions would introduce major transitional problems over a long period.

Two working systems for text translation are illustrated diagramatically in Figure 3. The first is used at the R.N.I.B. for book production, the final output of the system being embossed zinc plates for use in a printing press for producing two sided braille sheets, and considerable effort is devoted to ensuring that the end product is as accurate as possible. The contrasting system at Warwick provides a document service aimed at rapid and cheap production of short documents in small production runs, the error rate being relatively higher but hopefully tolerable, and an acceptable alternative to increased cost. Reduced human intervention coupled with good operator-machine dialogue seems to be the only way in which translation costs can be substantially reduced, and the amount of braille increased, perhaps tenfold, in the near future.

The provision of braille text is only one part of the problem of information provision for the blind. Much more difficult is the presentation of tabular information, e.g. timetables, mathematics and music scores.

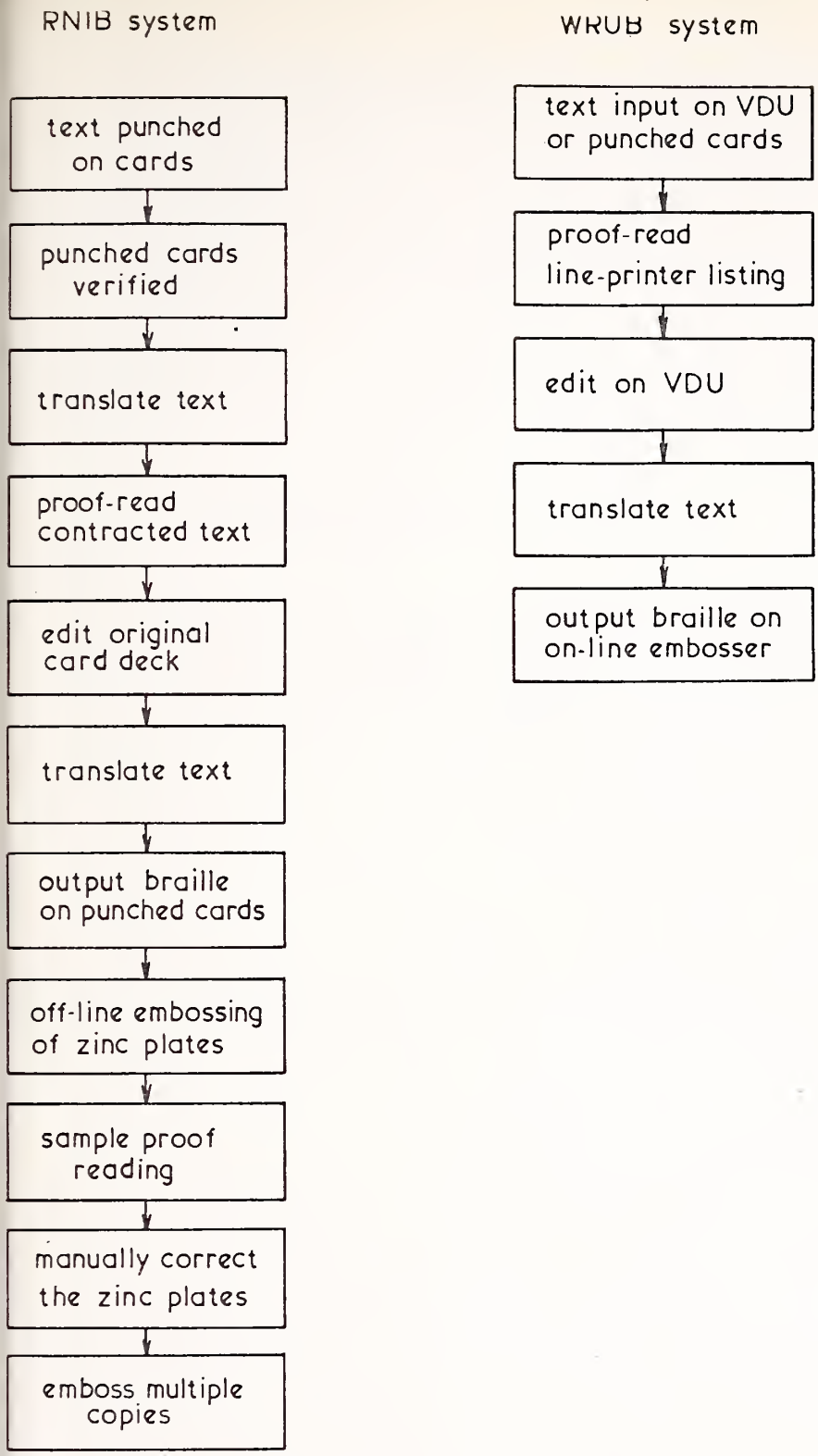


Fig. 3. The main steps in the automated production of contracted braille.

The American Printing House for the Blind is making a serious attack on this latter problem, with a music typewriter connected to a card punch for initial data input (Figure 4). The translation program for single stave music will be at least four times larger than that for text translation.

3. MAPS AND DIAGRAMS

The sighted scientific community regards graphical presentation of information as a vital ingredient of information storage and communication. A well designed diagram can be scanned rapidly for extraction of the salient features, and a small area selected almost instantaneously for detailed examination. The information density on a complex figure, with multicolour printing can be extremely high.

Graphical information for the blind is usually in the form of embossed sheets, read by touch. The limited resolution of the fingertips, the general lack of properly designed symbols, and the difficulty in scanning the diagram, severely restrict the amount of information presented.

In an attempt to assist the proper design of tactual figures, research effort has been directed to the design of symbols for points, lines and areas, the objective being sets of symbols which are discriminable by the majority of readers. The one advantage of embossed symbols over ink-print is the added flexibility available in that height can be used as an additional coding dimension. This has led, in particular, to the use of a line with height varying in a saw-tooth fashion along its length. This line is extremely acceptable for conveying directional information, to show information flow patterns on block diagrams or the sense of traffic

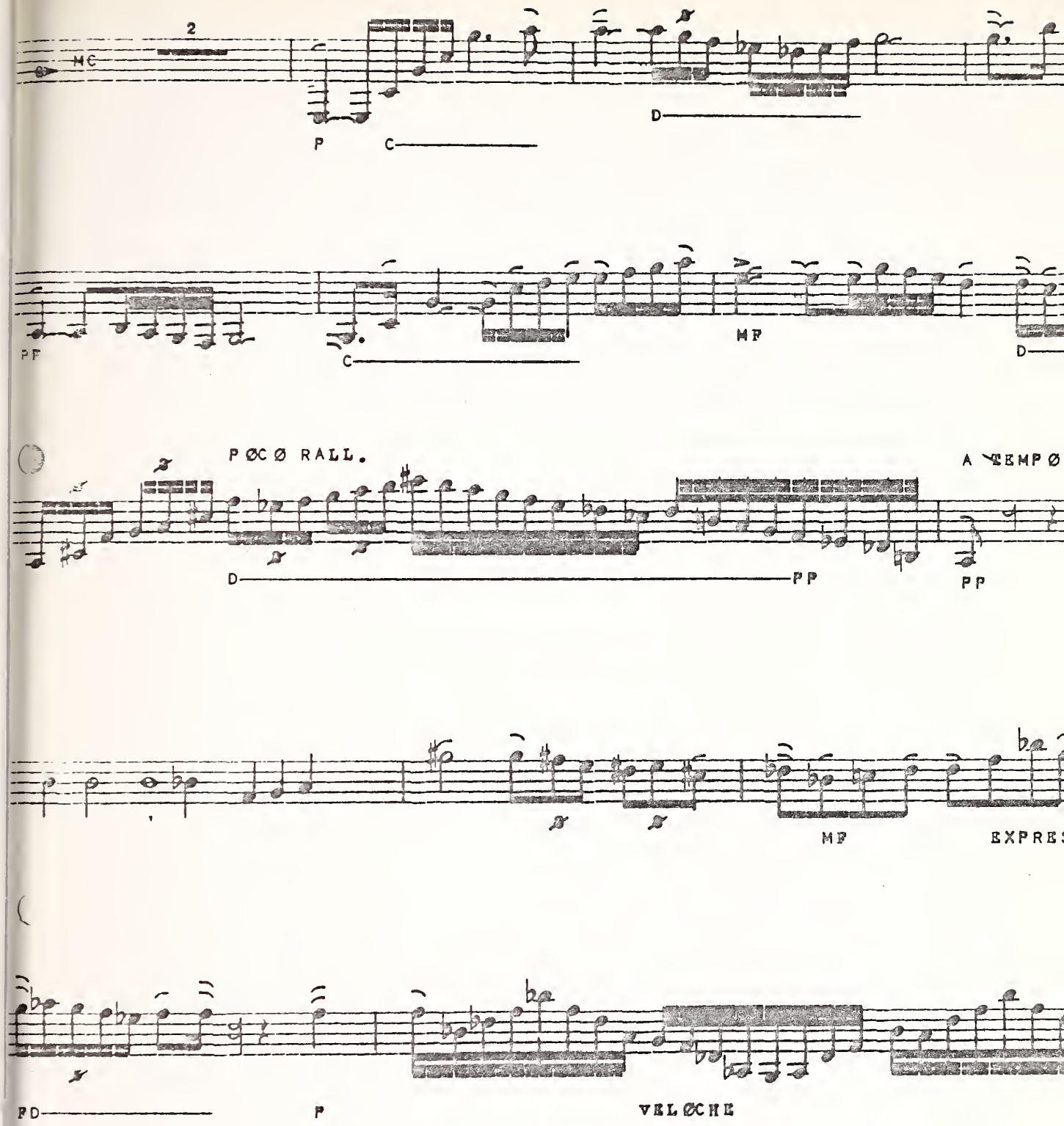


Fig.4 Printout of a digitised music score from the system under development at the American Printing House for the Blind.

flow in a one-way street, and tests prove this representation to be superior to the embossed version of the traditional arrowhead marking.

The initial research on symbols was a joint project involving the Engineering Department at Warwick and the Blind Mobility Research Unit at Nottingham University. One result of the studies has been the production of a kit of symbols for amateur map makers, useful both to provide proper symbols to other constructors and also hopefully to help standardise on a selected notation.

The Warwick production facilities can now make maps of a high quality at relatively low cost, and several dozen orientation maps have been provided for local and national facilities. As with automated braille production, the main cost is the input of the information to the system, which requires considerable design skill.

The system implemented at Warwick University is a fairly integrated computer-based production facility, offering good interactive facilities to the designer with the aim of providing relatively cheap high quality maps and diagrams in small production runs. The designer must have an annotated master diagram, using his skill to select the essential features without introducing an unacceptable quantity of information.

Data from the visual copy is supplied to the system via either a graphics terminal with joystick or through a simple coordinate table. Control commands specify the type of line - solid, dotted, dashed, sawtooth, etc. and the height of the line. Special symbols to specify landmarks or hazards can be inserted, and these symbols can be chosen from the standard set or be user-designed.

Versatile editing facilities enable the graphical data to be modified by rescaling the diagram, changing line parameter, deletion and substitution of data and by squaring up of lines which are 'sufficiently' close to being perpendicular. Text can be positioned where desired, and automatic translation to grade I braille is provided. The final data can be stored on paper tape for reference and later modification if necessary. The final output of the system is an engraving in laminated plastic produced by a numerically-controlled machine tool, This is driven by the computer used in the design process or by a smaller computer using paper tape control data. A male epoxy resin master is cast from this female engraving, and used to manufacture the required number of copies by conventional vacuum forming.

Several hundred maps of Euston Station, reproduced in Figure 5, with the alphabet replacing the corresponding braille symbols, have been distributed. This particular location has, in the past, proven particularly difficult to transverse, with no distinguishing floor textures nor acoustic cues to aid the location of central islands of importance like entrances to the taxi stand or to the underground. One finding from the questionnaire, filled in by the users of this map, was that they all found that the map greatly increased their knowledge of the environment and facilities at Euston Station.

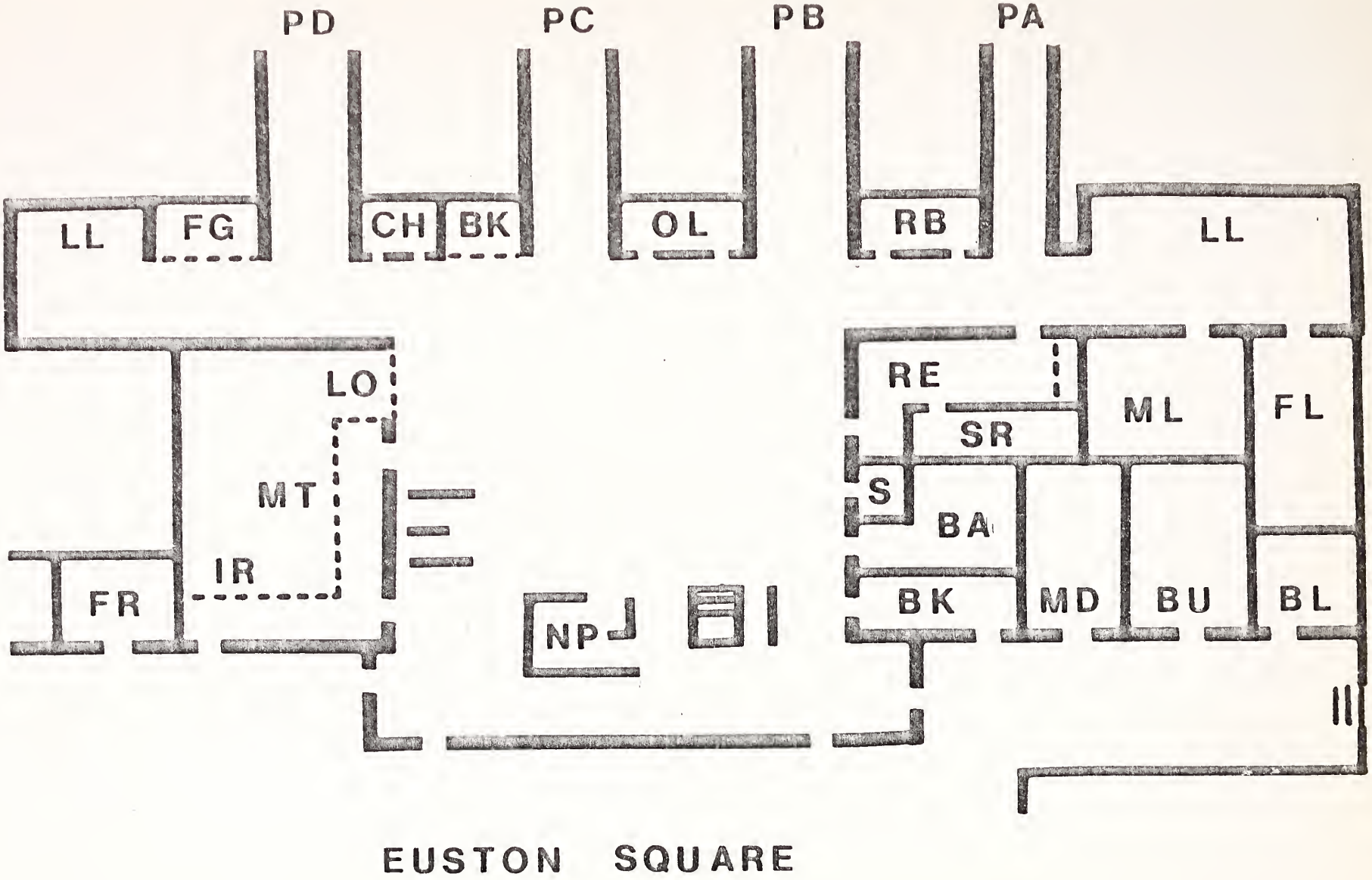


Fig.5 Sighted Version of braille map of Euston Station.

4. SOME CURRENT DEVELOPMENTS

The most immediate advance in the provision of information due to automation will probably be the computer-aided production of braille books and periodicals. It is not difficult to envisage a ten-fold increase in the output of braille, using existing techniques, and gradually reducing the amount of manual checking and editing, with better output facilities by large-scale introduction of on-line braille embossers.

One device of great potential is the braille writer under development by Dr. A. Grunwald in Chicago. This displays a line of braille on an embossed plastic loop or belt, the information being erased after presentation prior to re-embossing with subsequent information. Text is stored in digital form on magnetic tape, with some indexing capability, this medium considerably reducing the volume and cost of a braille library. The speed of the plastic belt is controlled by the user, and this feature, together with the continuous rolling display of braille, appears to provide a useful increase in the rate at which braille can be read.

It is also hoped that more integrated information services will become of proven reliability and more widespread in the future. A possible system, the ARTS (Audio-Response Time-Sharing) Service has already been demonstrated, in which many users interrogate a central data bank with typewriter input and synthetic speech feedback and optionally printed braille output, either locally, on-line, or by central print-out. Some combination of these two previous systems could well offer a braille computer terminal as a direct alternative to a visual display device, with considerable possibilities for increasing the employment opportunities for the blind.

More mundane perhaps, but by no means trivial, is the increasing use of digital compositors tapes for the source material for braille. Although

this appears, at first consideration, to offer an error-free input, the practical problems are considerable, since there is a daunting multiplicity of codes amongst printers and final proof correction may never in actual useage produce a clean tape, and control characters for ink print and braille version present a challenging problem.

However, more promising are the computer-based systems currently being introduced by some printers, where the text is corrected on a visual display unit and the error-free data can be output on magnetic tape. Some of these systems have the advantage that they include flags indicating headings, page numbers, italics and footnotes, and they also differentiate between apostrophe and single quotation mark which are represented by different symbols in braille.

With the increasing availability of data in machine-readable form, many possibilities exist for completely automatic braille production, with the consequent reduction in routine human intervention and hence in cost. For example at Warwick a pilot scheme for the translation of bank statements to braille is now being undertaken on a routine basis, with a twenty-four hour turnround from receipt of the magnetic tape containing the data to print out both inkprint in line printer output and braille. These statements are complete, apart from deletion of the account holders identification. Internal telephone directories are fairly re adily produced in braille from digital data in standard format, with the incidental benefit that alphabetic lists, as well as the perhaps more usual listing by section or department, can be produced for switchboard operators if required.

Professional blind people have a particularly serious problem in keeping up to date in their fields. For instance it is particularly difficult for a blind person to scan the journals in a library. One attempt to alleviate this problem is being made by the Warwick Research Unit for the Blind in cooperation with the Institution. The pilot scheme produces

selective listings of the INSPEC Computer and Control Abstracts in braille. The feedback from the blind programmers participating in this experiment has been sufficiently encouraging that the possibility of extending the pilot scheme to other subjects is being investigated.

If these pilot schemes are successful, then it is hoped they will be taken over, in due course, by the established service organisations. At the present time these activities serve the important purpose of keeping research in close contact with the ultimate users, though financing such useful developments is incredibly difficult.

It is hoped that the provision of more braille will provide to increase the number of braille readers, by making more teaching material available and by a larger library giving more motivation for the better and wider use of braille.

Some of the possibilities for automation in the provision of information to the blind have been outlined. It is an area where interdisciplinary effort is absolutely essential, and international cooperation can offer unusual dividends. There is a long way to go before the blind are able to fully utilise their talents in employment and in their leisure activities.

Although this survey has concentrated on the aids which automation and relatively complex systems may offer, it is important to appreciate the equally important apparently trivial aids which are lacking for the blind. It is particularly regrettable that gadgets are frequently developed and never produced on a large scale and marketed, whilst other devices, like an embosser of braille onto adhesive plastic strip are suddently withdrawn. For instance, of all the gadgets developed in British universities in the last years, none are currently available, as standard items, to the British blind population.

A major advance could well result if the cost-effectiveness of complete systems could be properly measured. This assessment should include effects of increase and higher grade employment, reduced welfare costs and the value of a better awareness of the present day environment which greater information availability can offer to the blind.

ACKNOWLEDGEMENTS

I am indebted to Dr. John Gill who has been the principal worker at the University of Warwick in this area for the past five years. The late Dr. J.A. Leonard and his successor Dr. J.D. Armstrong of the Blind Mobility Research Unit have provided much useful collaboration. The work would have been impossible without the facilities provided by the Science Research Council, and the assistance of the Royal National Institute for the Blind, the American Foundation of the Blind, and the Medical Research Council for the early work. Many individuals have given considerable time and effort to guide the orientation of the research, in particular Dr. Fred Reid, of the National Federation of the Blind and Dr. M.J. Tobin of the Research Centre for the Education of the Visually Handicapped.

REFERENCES

"Braille mathematics notation". Royal National Institute for the Blind, London, 1973.

Coleman P.W.F. "Computer terminals for the blind". Electronics and Power, Vol. 18, March 1972; pp 84 - 86.

Douce J.L. & Gill J.M. "Computer drawn maps for the blind". Electronics and Power, Vol. 19, No. 14, August 1973, pp 331 - 332.

Gerhart W.R., Millen J.K. & Sullivan J.E. "DOTSYS III: a portable program for grade 2 braille translation". MITRE Corporation, U.S.A., 1971, 139 pp.

Gill J.M. "Orientation maps for the visually handicapped". British Cartographic Journal, Dec. 1974, pp 94 - 98.

Gill J.M. "Methods of increasing the accessibility of reading material by the blind". The Louis Braille Conference, Cambridge, England, Jan. 1975, pp 155 - 162.

Gill J.M. & Reid F. "Visual handicap: aids and support". Open University, The Handicapped Person in the Community, Unit 6, 1975.

Gill J.M. "Non-visual computer peripherals". American Foundation for the Blind, New York, 1975.

Gill J.M. "The international register of research on blindness and visual impairment". Warwick Research Unit for the Blind, University of Warwick, England, 1975.

Goldish L.H. "Braille in the United States: its production, distribution and use". I.R.I.S., American Foundation for the Blind, New York.

Grunwald A.P. "A braille reading machine". Research Bulletin of the American Foundation for the Blind, No. 16, pp 74 - 78.

Ingham K.R. "Audio-response time-shared service bureau for the blind". MIT Report, 1971.

Irwin R.B. "The war of the dots". American Foundation for the Blind, New York, 56 pp.

Lawes W.F. "The feasibility study into the usage of computers by and for the blind". Royal National Institute for the Blind, London, Jan. 1975.

Proceedings of the workshop "Towards the communality of algorithms among braille transcription systems for multilingual usage". University of Munster, Germany, March 1973.

Proceedings of the workshop "Computerised braille production". Danish Association of the Blind, Copenhagen, Sept. 1974, to be published.

"A restatement of the lay-out definitions and rules of the standard English braille system". Royal National Institute for the Blind, London, 1969 & 1971.

Schack A. & J. "Studies in the automation of braille mathematics and music". Final Report to the Department of Health, Education, and Welfare, American Printing House for the Blind, June 1969, pp 8 - 78.

Schiff W., Kaufer L. & Mosak S. "Informative tactile stimuli in the perception of direction". Perceptual and Motor Skills, Vol. 23 (Monogr. Suppl. 7), 1966.

APPENDIX 5

"METHODS OF INCREASING THE ACCESSIBILITY OF READING MATERIALS BY THE BLIND".

by Dr. J.M. Gill

Paper presented at the Louis Braille Conference, Cambridge, England, January 1975.

Methods of increasing the accessibility of reading materials by the blind

J.M. Gill

A totally blind person has three main methods for accessing written information.

1. Use of a reading aid with the information in ordinary sighted form. The reading aid can be of either direct or indirect access type; the former involves automatic character recognition whereas the latter involves the user in skilled interpretation of the output. An example of a direct access type reading aid is the Textobrail and an example of an indirect access type is the Optacon.

2. The information can be recorded verbally in analogue form on magnetic tape or disc. This system has the advantage that it is compatible with the sighted community. The main problems are those of indexing and speed of data transfer.

3. The information can be in a tactile form. The most usual coding systems are braille or Moon, which are often embossed directly on paper.

This paper is limited to considering the problems concerning braille.

The previous speaker has described a system for increasing braille production by increasing the output of a skilled braille transcriber. Another approach is to use relatively unskilled typists and to use a computer to translate the text to an approximation to grade II braille. Since the demand for braille is greatly in excess of supply, it is essential to use all systems available to increase braille production.

The computer-based systems involve:

1. Input of text
2. Proof reading
3. Editing
4. Translation
5. Output

A basic system might be:

1. Input on punched cards or paper tape.
2. Listing on a line-printer for proof reading.
3. Editing on a visual display unit or even by correcting the punched cards.
4. Translation by a program such as DOTSYS III.
5. Output to an on-line embosser.

The advantages of such systems are that they are fast and do not require the use of highly skilled braille transcribers. Such systems are usually cheaper than employing manual transcribers.

There are applications where output in grade I braille is required. For instance a braille learner might want a list of football clubs playing the coming Saturday or the diary of events for the local darts club; however if this motivates the person to learn braille then it is worthwhile.

Input

Format information has to be included at this or the editing stage. Control characters are needed for paragraphs, forced new lines, running titles, centred headings and non-standard characters such as Greek letters. It may also be desirable to flag italics, foreign words and phrases, inflected letters and single letters other A,I or O. If footnotes are to be included in the main text, it will be necessary to flag the beginning and end of this text.

Tables

The major problem is material whose layout is an intrinsic part of the content, such as tables. The problem of tabular presentations is of the same order of magnitude as that of producing meaningful embossed diagrams. It is impossible to handle tables automatically in any but the simplest cases. However the problem of layout of tables and diagrams can be alleviated by using an interactive method on a visual display unit.

An example of this problem with tables is the O'Connor scan-column index; it is impossible to transcribe this into braille such that the user

can do a multi-column search.

Special codes

Further problems arise when inputting mathematics or music from an ordinary keyboard although feedback can be provided. For instance music can be coded using alphanumerics which can be directly displayed at the top of a visual display unit screen; underneath the music can be displayed in staff notation for checking against the original score. The data can be edited and then the braille can be displayed across the bottom of the screen.

Other modes of input

One method to reduce the cost of data preparation is to use prisoners. The text is input on a standard compositors keyboard with the output on punched paper tape. The trade unions should not object to such an arrangement since it is not in competition with open industry. The prison authorities might view the system favourably since the prisoners would be partly trained for employment in the printing industry on their release from prison.

Computer-compatible compositors tapes are possible input media but printers often make corrections on the type itself. Another problem is the variety of codes used by printers in this country but this is not insuperable since the pre-processor can be table driven.

However many organisations are storing digitally information which requires frequent updating such as internal telephone directories and bibliographies. Access to data bases such as ERIC, MEDLARS and INSPEC is very important to some professional blind people.

Optical character recognition is another mode of input but the accuracy and the range of acceptable typefaces are a function of bandwidth and therefore cost. At present OCR is not economically viable for this application.

Indexing

A sighted person with an inkprint textbook may use a number of different ways for locating a specific item:

1. The subject index at the back of the book.
2. Chapter titles, then serially through the chapter.
3. Subject headings which are repeated at the top of each page.
4. The reader may remember that it was "near the front of the book and mid-way down the page".
5. The reader may look for a diagram which he remembers is physically near the item he requires.

There are obviously other methods but the important aspect is that the system, or combination of systems, used is chosen by the individual reader although the choice will depend on what has been provided by the publisher.

The provision of indexing facilities for non-visual information systems deserves attention since it is an area which has tended to be neglected in the design of transcription systems.

Translation

A variety of computer programs have been written to produce an approximation to grade II braille. These programs will translate up to 25,000 words per minute although speeds of 1000 to 4000 words per minute are more common.

If there was a rigorous mathematically unique definition for grade II braille, the program could work directly from this definition. It is possible to consider translation to a grade II, which is not context dependent, being done on a microprocessor. This may be particularly relevant to developing countries since the cost of a microprocessor and the necessary peripherals may be less than the cost of training and using manual transcribers.

Outputs

If a large number of copies are required, the output can be off-line to machines which emboss the conventional zinc plates. Very often less than three copies are required so it is economical to emboss directly onto paper.

It is usual to interface the translation program and the output device with a post-processor which handles tasks such as page numbering. It can also handle alternate page inversion which involves embossing upside-down alternate pages of fan-fold paper in order to simplify the task of collating and binding.

Because of the considerable bulk of embossed braille and the increasing cost of paper, many alternative modes of storage have been proposed and sometimes developed to prototype stage. The storage media has usually been digital cassette, floppy disc, printed page or microfiche. All these systems involve the user in having a special reading machine. However the main application for such systems is for white-collar workers such as computer programmers where the text for manuals is sometimes available in digital form.

For example braille can be stored on microfiche. A device for the computer output of microfiche can produce 1 sheet per minute for a materials cost of 10 pence per sheet. Copies cost 2 pence. Each sheet can contain up to 100,000 braille characters.

A simple reading machine could consist of a line of photocells directly connected to a line of braille display. For all these digital storage systems it is possible to build more sophisticated reading machines which incorporate automatic search functions.

These methods for storing braille will not replace embossing on paper but may serve as an additional aid for some of the visually

handicapped in professional employment. Both digital cassettes and floppy discs can be used for writing as well as reading; a pair of storage devices is required for editing.

The main problem is in the design of the display. Although many displays have been built, little is known concerning the fundamental parameters of the man-machine interface.

Accuracy of computer transcription systems

Errors come from:

1. Incorrect input data which is not noticed at the proof-reading stage.
2. Incorrect or non-optimum format commands.
3. Incorrect choice of contraction by the translation program.
4. Malfunction of the output device.

The standards for accuracy should not be absolute but should be relative to the accuracy of other transcription systems in common use. Also the severity of the error should be taken into account; for example an incorrect choice of contraction may not significantly reduce the reading speed. The reading speed may not be impaired at all once the reader is used to a particular dialect of grade II braille.

Future research

In conclusion, there are eight areas which seem most deserving of attention by research workers.

1. Since data preparation is time-consuming and therefore expensive it is necessary to make best use of compositors tapes, other texts already held in digital form and other sources of inexpensive data preparation such as prisons.

2. The professional blind person is likely to find it increasingly important to have selective access to large data bases such as INSPEC. Some of the users may require on-line access through a soft-copy terminal in order to work in an interactive mode.

3. The development of interactive programs to assist in the design of tables and diagrams. This will require some psychological research on how tables and diagrams are 'read'.

4. The development of computer programs to handle specialist codes such as mathematics and music.

5. Fundamental studies on indexing systems.

6. Formal definition of grade II braille.

7. Design of inexpensive production systems for the developing countries.

8. Fundamental studies on the design of non-visual displays.

JMG/lcw

8.1.75

uments
er commands
a paragraph.
and any
tion, on a

otherwise

irst colum

y do occur
gle spaces.

$HDE after. This
entres one-line headings.
d for headings up to 36
ere a heading consists
o characters the heading
as a paragraph ($P).

ginning.

full stops in, e.g. R.N.I.B.,

sign) immediately before accented
e.g. café = caf¢e or für = f¢ur.

eft and) for right.

left and > for right.

full stop. Denomination changes
sign, i.e., 9.25 a.m. = 9#25 a.m.

Equals	$= must be used to differentiate from the letter sign.
Interlining	$DOUBLE before and after a piece of text gives automatic double line spacing. It is automatically cancelled at the end of a file. This is used in young children's books (primary age) and in articles for braille learners.
Italics	Underscore _ before each word for 1, 2 or 3 words. For 4 of more words type two underscores __ before the first word and one underscore _ before the last, e.g., __For 4 or more _words. TYPED UNSPACED FROM WORD.
Letter sign	=, e.g., from =A to =Z; postal code =CV4 7AL. TYPED UNSPACED FROM LETTER. Used to distinguish letters not forming words, but not required if immediately after a digit.
New Line	$L
New Page	$PG
Poetry	Begin with $P $PY. Start each subsequent verse with $P. At the end of each line, except the last line of the poem, type $PS WITHOUT ANY PRECEDING SPACE. Applies to songs and hymns.
Pound sterling	Use L for pound (£) sign, e.g., £500 is L500.
Quotes	" for both left and right. Inner quotes (quotes within a quote) type: Left quote: $' leaving a SPACE before commencing word. Right quote: $'R UNSPACED FROM preceding word. e.g. $' The rules for inner quote$'R. An asterisk in the left hand margin of the line printer listing denotes lines where quotes are used. Check that quotes are correctly typed. If the quotes are odd in number, assuming you have missed one out, this will be stated at the end of the line printer listing.

Roman numerals	= sign, e.g., =I =VIII
Skip line(s)	$SLnn nn represents number of lines to be skipped, i.e., $SL01 = skip 1 line. $SL15 = skip 15 lines. The zero must be included if the number is less than 10.
Tabs	$#n where n is the number of the tab. There are 9 tabs numbered 1 to 9 which are initially set every two columns starting with column 3. e.g. $#2 would cause the following text to start in column 5. If it is necessary to change the position of a tab, $STBnLmm should be used, where n is the number of the tab and mm is the new column which the tab is to refer to. e.g. after $STB2L04, any occurrence of $#2 would cause the following text to start in column 4.
Uncontracted braille	$G before and after the appropriate words. Foreign words are an example where uncontracted words are used, e.g., $G Aberich glaube, er ist gut $G. If $G are odd in number, assuming you have missed one out, this will be stated at the end of the line printer listing.

QUICK REFERENCE:

Two most common commands:

Heading	$HDS _____ $HDE
Paragraph	$P

Alphabetical

Accent:	¢
Brackets	()
Brackets (square)	<>
Equals	$=
Italics	Underscore (see full list)
Letter sign	=

New Line	$L
New Page	$PG
Poetry	(see full list)
Pound sterling	L
Quotes	"
Quotes, inner	$'space/no space$'R
Roman numerals	=
Skip Line	$SLnn
Tabs	$#n (to change position) $STBnLmm
Uncontracted braille	$G _____ $G

FORMAT

Format on the whole remains as the printed copy, for the sake of clarity sometimes modifications are necessary. Underlining and ornamental type are disregarded. Capital letters are also rarely used in English braille.

Subjects within a document need some form of division, i.e., address at the top of a letter, ex. 2; subjects under side-headings, ex. 3. Imagine print with no block capitals and no underlining to catch the eye for side headings, etc. Likewise, in braille, the finger needs something clear to pick-up if reference documents are to be effective. General paragraphing in many documents is clear enough but, when side headings incorporate more than one paragraph a blank line should be left between subjects ($SL01). Another form of division, when more than one article appears in a document, is the centring of three spaced asterisks, i.e., $HDS * * * $HDE.

Some examples of format are given below:

EXAMPLE 1 - General:

LADDIE TO THE RESCUE

This is a true story. There are two boys in the story - David and Christopher - but the hero of the

Text input:

```
$HDS LADDIE TO THE RESCUE $HDE $P THIS IS A TRUE
STORY. THERE ARE TWO BOYS IN THE STORY - DAVID
AND CHRISTOPHER - BUT THE HERO OF THE
```

EXAMPLE 2 - Letters: (for quick reference the name is shown before the address).

1 Ash Road,
Birmingham, B3 4TS

16th Sept., 1975.

Dear Mr. Smith,

Thank you for your letter and invite. I am pleased to accept and look forward to meeting you.

I remain,

Yours sincerely,

Joe Brown.

Text input:

```
MR JOE BROWN, $L 1 ASH ROAD, $L BIRMINGHAM =B3 4TS.
$L 16TH SEPT., 1975. $SL01 DEAR MR. SMITH, $P THANK
YOU FOR YOUR LETTER AND INVITE. I AM PLEASED TO
ACCEPT AND LOOK FORWARD TO MEETING YOU. $P I REMAIN,
YOURS SINCERELY.
```

EXAMPLE 3 - Minutes: (for clarity in braille a blank line is left between sections)

Committee for the Physically Handicapped
J. Briggs Chairman

Minutes for the meeting of the Committee for the Physically Handicapped. Held Tues 12 July, 1975. 7.30 p.m.

Present:

Mr. J. Briggs	-	Chairman
Mr. L. Smith	-	Vice Chairman
Mr. F. Brown	-	Social Services Dept.

29/75 MINUTES OF THE MEETING HELD 21 MAY 75

The minutes above corrected and approved.

MATTERS ARISING:

Car Parking for disabled. Mr. Brown reported

Text input:

$P COMMITTEE FOR THE PHYSICALLY HANDICAPPED. J. BRIGGS: CHAIRMAN $P MINUTES FOR THE MEETING OF THE COMMITTEE FOR THE PHYSICALLY HANDICAPPED. HELD TUES 12 JULY, 1975. 7 30 P.M. $P PRESENT $P MR. J. BRIGGS - CHAIRMAN $P MR. L. SMITH - VICE CHAIRMAN $P MR. F. BROWN - SOCIAL SERVICES DEPT. $SL01 $P 29/75 MINUTES OF THE MEETING HELD 21 MAY 75. $P THE MINUTES ABOVE CORRECTED AND APPROVED. $SL01 $P MATTERS ARISING $P CAR PARKING FOR DISABLED. MR. BROWN REPORTED

EXAMPLE 4 - Statistics:

In braille it is impracticable to show statistics in tabular form. The headings are set out as a paragraph, each heading separated by a semi colon. The leader to the paragraph reading: "In the following table the column headings are:"

Age	Nature of Accident	Loss of Work	Compensation
39	Spine fracture	1 year	£1,500
56	Broken foot	18 weeks	Nil

Text input:

```
$P IN THE FOLLOWING TABLE THE FOUR COLUMN HEADINGS ARE:
AGE; NATURE OF ACCIDENT; LOSS OF WORK; COMPENSATION.
$P 39; SPINE FRACTURE; 1 YEAR; L1,500. $P 56; BROKEN
FOOT; 18 WEEKS; NIL.
```

EXAMPLE 5 - Tables

BUS TIMETABLE

Monday-Friday				
Hanley (Bus station)	0630	0655	0715	0830
Smallthorne	0639	0704	0724	0804

Text input:

```
$HDS BUS TIMETABLE $HDE $P MONDAY-FRIDAY $P HANLEY
(BUS STATION) 0630 ; 0655; 0715; 0830. $P SMALLTHORNE
0639; 0704; 0724; 0804.
```

EXAMPLE 6 - Poetry

The Dewdrops quiver on the cobweb tents,
Birds leave their love and sit in meek suspence.

A disk of fire aeons old cuts through
The rocks of earth and rolls up into view.

Text input:

```
$P $PY THE DEWDROPS QUIVER ON THE COBWEB TENTS,$PS
BIRDS LEAVE THEIR LOVE AND SIT IN MEET SUSPENSE.$PS
$P A DISK OF FIRE AEONS OLD CUTS THROUGH$PS THE ROCKS
OF EARTH AND ROLLS UP INTO VIEW.
```

HV1703 D460 c.1
DESIGN AND EVALUATION OF A SYSTEM FOR THE PRODUCTION OF SHORT DOCUMENTS IN CONTRACTED BRAILLE.
(1975)

Date Due

HV1703 D460 c.1
DESIGN AND EVALUATION OF A SYSTEM FOR THE PRODUCTION OF SHORT DOCUMENTS IN CONTRACTED BRAILLE. (1975)

DATE	ISSUED TO
	Reference Copy

AMERICAN FOUNDATION FOR THE BLIND
15 WEST 16th STREET
NEW YORK, N. Y. 10011

Printed in U.S.A.