A General Description of the Process for Converting Information to Braille

Translating Academic Information to Braille

The following sections are designed to provide the reader an overview of the process of converting text to alternative print. This article is based on procedures and techniques developed by Arizona State University for the production of mandated alternative print accommodations. Specific software and hardware brands are listed in the article. The listing of vendors does not, however, imply an endorsement of one vendor over another. **DRS/ASU** uses certain brands, versions or models and has developed a system around these choices which is effective. This article assumes that the reader is computer literate and has or can learn the basic command structure of the software and hardware used by DRS/ASU.

The following section will only discuss the conversion of printed text materials into **Grade II Braille**. The next section will discuss the process used to convert graphic information into raised line drawings and the final section in this series will discuss the process of converting mathematic and scientific notation into the **Nemeth Code Braille.**

Converting Text to Braille

Text Conversion Process Components of a Scanning System

In the past, the conversion of printed text to Braille was a slow, painstaking process. Trained transcriptionists, using a **Braille Writer,** manually translated each character or group of characters into it's appropriate braille symbol. Using this method, the transcription of textbooks could take literally hundreds of hours and cost an exorbitant amount.

By using personal computers, optical scanners, and specialized software to expedite the translation process --text to Braille is, in principle, a relatively simple process. The procedure uses a **personal computer, either a Macintosh or an IBM compatible; an optical scanner; OCR (Optical Character Recognition) software and Braille translation software.**

The selection of the equipment to convert printed text to Braille is very important. It should be selected with the task of scanning and translating as its primary objective. If an IBM or compatible is chosen, the computer's processor should be, at least, a 486-66 Mhz with 16 Megabytes RAM and should have a hard- drive with at least 500 Megabytes of storage space. If an Apple computer is chosen, it should be at least a PowerMac 7100 or above. Either computer type should have as much RAM as possible, 16 Megabytes is the minimum RAM recommended. **Access to a printer, a braille embosser and a scanne**r is also required. **Duxbury Braille** translation software and others are often used.

The market has been flooded with OCR packages and scanners selling at a variety of prices. There are many varieties of scanners available: flat bed scanners, hand held scanners, slide scanners, drum scanners, scanners that scan black and white and scanners that scan color, scanners that are fax machines. Accordingly, the prices for these scanners also have a price range from $200.00 to over $30,000.

Be careful in the selection of a scanner, it is crucial to the text conversion program. Do not make the assumption that only a black and white scanner is required to scan printed material. **Many textbooks use color as a method of highlighting important concepts.** A black and white scanner may not pick up colored and highlighted characters. These characters would then not appear in a scanned document. A color scanner will accurately scan the widest variety of text formats. A **flat bed scanner is also recommended.** The hand held scanners must combine many scanning "passes" to produce an integrated document. Manually controlling the angle and the speed of a hand held scanner is difficult and time consuming. A flatbed scanner will control these variables.

The scanner should have a **cut sheet feeder**. While a cut sheet feeder is not essential for the scanning process, it will increase the accuracy and speed of the scanning process. The use of a cut sheet feeder will require that bound texts are unbound. Many print or copy centers have the ability to unbind texts for a minimal cost.

OCR software should be chosen by considering the type of materials that are likely to scanned. There may be information that is graphical and which must be made accessible to visually impaired students. It is possible to convert scanned information into raised line drawings. The quality of the scanner and software will determine the amount of extra work that must be done to create an integrated document.

The process for text conversion is relatively simple. The operator enters the OCR software, sets the parameters that correspond to the type and quality of text being scanned. These parameters may include whether the material is dot matrix or solid type, text only or text and graphics, columns and the contrast between the foreground and the background. The operator can choose to scan only portions of the page by defining "scan windows". Many current OCR software packages have an automatic setup option which will attempt to do these activities. Only through experience can the operator decide when to manually setup a page and when to let the computer do it.

One of the considerations that must be answered is if documents are likely to be single or multiple sheets. If the majority of the academic scanning will be multiple sheets, the software should collate pages as it scans. Scanning multi-page documents will require considerable RAM memory. The number of pages that can be scanned at one time is directly related to the size of the computer's RAM memory.

After the scanning parameters are established, **scan the first page of the document then use the OCR feature of the software to convert the page to ASCII text.** Examine this test page and check for accuracy. If the scanning parameters relating the type, brightness and contrast were not set properly, the document will contain many scanning and OCR

errors. Change the settings and try again until scanning settings are optimize for effectiveness.

OCR software must be used to convert the scanned image to ASCII text after it is scanned. The operator may now edit the material for accuracy. It is possible to do this editing in the OCR software or in a word processor. Editing in a word processor is usually more appropriate since most word processing packages now contain a spelling checker with large dictionary and more effective editing capabilities. To use a file in a word processor it must first be saved in the OCR software in a format suitable for access by the word processor. Most OCR packages have built in formatting for popular word processors.

The latest version of **Duxbury will convert a WordPerfect file directly into Braille**. This product does quite well in converting a word processing file. It is easy underestimate the requirement for editing a text file when products like MegaDots and Duxbury have an automatic conversion feature for word processing documents. These software routines do an acceptable job of converting one or two pages of pure text. They were not designed to convert a 25 page report or a 500 page textbook. Regardless of the product chosen to translate text, the text must be edited for errors.

Text Conversion Process

In a post secondary institution, the purpose of converting textbooks is not just to produce Braille or ASCII files. The goal is to produce a computer based representation of the printed text. It is extremely important to convey to those that control the accommodation budget that the conversion process is complicated as opposed to automatic. The process described here is not inexpensive. It requires a consistent budget and dedicated and trained individuals.

Anyone can scan a document, save it as an ASCII file and run it through the Braille conversion process. That is not the goal of this article. To convert a textbook into an alternative format requires an understanding of what a textbook employs to provide meaning for the reader. The table of contents, footnotes, bold or highlighted words, headings for chapters and sections, text inserts, text page numbers, graphs, captions on pictures and pictures, columns of information, scientific or math symbols are just a few of the types of information that will be lost or corrupted when a printed text book is scanned. This information is just as essential to the learning experience of students with disabilities as it is for any other student. It is this type of information, as well as the textual information, that is mandated for students with disabilities.

The first step in editing is to compare the scanned document with the original text for accuracy. This must be done since OCR software, while it has made great strides, is not infallible. When a document is scanned, the OCR software makes many decisions about the correct format of the text. One of the primary goals of this type of software is to produce a text file that resembles the print copy. While the document may appear as an accurate representation of the original, there may be important differences in the OCR document file. These differences will affect the format of the Braille document and must

be checked at this stage in the translation process. Some obvious errors in formatting are easily edited, other are more difficult to see. For example **hard returns (HRt)** often appear at the end of a line in the middle of a paragraph. While these hard returns are not seen in the word processor document, they may be very apparent in the Braille document. Hard returns are one of the most powerful Braille formatting commands. A Braille document is **only 40 characters wide and the Braille translation software shortens and abbreviates combinations of letters.** Unwanted hard returns will begin a new line of text where it is not required and will confuse the reader. Braille documents with unwanted formatting commands will force the Braille reader to guess the correct meaning of the text unless the problems are corrected at this step in the Braille conversion process. This distinction may appear to be trivial, but documents that are haphazard in presentation will not meet the **academic institution's responsibilities for mandated accommodations under section 504 of the Rehabilitation Act or the Americans with Disabilities Act.**

The obvious way to check for these errors is to use the command in the word processor to show the formatting commands and then to go through the document looking for these errors in formatting. Another method is to increase the margins an inch and visually check for breaks in the formatting.

Another advantage word processors have in the editing process is the use of the **macro function** to expedite the formatting of the document for Braille. A macro is a method of programming the word processor to perform a complex set of functions automatically. For example, it is possible to have the word processor search for erroneous Hard Returns, centering commands for text, special fonts and format changes and to delete them as it finds them.

The word processing document will also need Braille translation commands to preserve document's formatting that is essential to the its meaning. The proper indentation of a Braille document paragraph is two braille cells. It is possible to program a macro that is capable of entering these Braille commands as well.

Appendix A has a list of some of the macros that makes the editing process much easier. Some of these macros are nothing more than a quick way to enter multiple keystrokes. Other macros are very involved and perform multiple editing functions. Please keep in mind that spaces and hard returns are very powerful tools in Braille and that a great deal of care must be taken in controlling when and how they appear in a document. Automatic conversions will miss or confuse many sections of a document.

After a document has been check and edited it is ready to be converted into Braille. Some Braille translators use only ASCII files for conversions. The word processing file must therefore be saved as an ASCII file. I**n WordPerfect 5.1 someone can use command keys to save ASCII text files is "Control F5, 3 Save As , 1 Generic." In WordPerfect 6.0 DOS, save as "ASCII Stripped" DRS does not use Windows based applications for editing purposes.**

The document is now ready to be converted to Braille format by the translation software. After the document is converted it should be reviewed as it will appear when it is printed. Duxbury has a "Browse" command that will allow the editor to see the file as it will be appear in Braille. The Braille file should be reviewed to ensure that the proper overall formatting has been preserved. Make sure that page breaks inserted by the Duxbury translator occur in logical places and that the Braille commands inserted during editing have performed as expected. It is also a chance to check for unwanted Hard Returns which will break lines at unwanted locations. If any errors have occurred, correct them in the word processor document and re-translate the document. This insures that the Braille given to the student is of the highest possible quality.

The file is now ready to be sent to the Braille embosser to be printed out as a Braille document. Print out Braille on Braille paper with a perforated margin. It is an easy process to use a plastic binding to organize the embossed material.

A final step is crucial to the success of academic accommodations. A review of the Braille document by an individual with knowledge of Braille. Several crucial functions are fulfilled by this review. This final review allows feedback on the preferences of Braille users. It permits the blind community on campus the opportunity to feel involved in their accommodations. It gives an individual with a disability a chance to work and be paid for it. Finally, no matter how meticulous the student staff is in the conversion of text, bad habits will creep into the production of Braille. It is essential that the work be reviewed by individuals who were not involved in the production of the Braille and are critical of its quality.

Components of a scanning system

While there are many more worthwhile configurations which would be effective in an OCR function, the products listed below are products with which the Arizona State University has a working knowledge and which have demonstrated the ability to produce quick and accurate results and they recommend.

Microcomputer
IBM or Compatible Pentium with at least 500MG Hard Drive, 16MG RAM (minimum), 3.5 floppy,

DOS 6.2, and Windows 3.1
Macintosh Power Mac 7100, at least 500 MG Hard Drive, 16 MG RAM (minimum), System 7.5

Scanners
HP Scanjet IIIc or better (IBM), cut sheet feeder, Apple Color Scanner (Mac)

Business is very involved in OCR hardware. If more than ten thousand dollars is budgeted for a scanner, it is possible to find scanners that will scan and convert a duplex page (print on both sides) in less than four seconds.

OCR Software
OmniPage Professional (Mac and IBM)
Caere Corp.
100 Cooper Court
Los Gatos, CA 95030
(408) 395-7000

Wordscan Plus (IBM)
Calera Recognition Systems
2500 Augustine Drive
Santa Calera, CA 95054
(408) 720-8300

Braille Translation Software
Duxbury Systems Inc.(MAC,IBM)
435 King St. P.O.Box 1504
Littleton, MA 01460
508-486-9766

Telesensory Systems Inc.
455 N. Bernardo Ave.
Mountain View, CA 94039-7455
415-960-0920

The other major Braille Translation software is MegaDots by Raised Dot Computing, Inc. 408 S. Baldwin St.R Madison, WI, 53703R 800-347-9594R 608-257-9595

Conversion of Math Symbols into Braille

The following is a practical guide to producing **Nemeth Code Braille** with current technology. The discussion will focus on the use of one example of **Braille translation software—Duxbury.** This, however, is not meant to imply that Duxbury is the only software capable of performing these operations. Whatever software you chose to use, the procedure followed will be very similar to the steps listed below.

The current trend in E-Text, Electronic Text production, is toward a **standardized general markup language (SGML).** The use of SGML will allow rapid translation from one format to another through the use of embedded, s**oftware-read commands or "tags."** To change from one output format to another all that would be required would be to change the appropriate tags to produce the required format. At this time, however, SGML is not fully developed or available for general usage. Software standards for future SGML conversions are still being discussed. Until such time that SGML becomes available, the strategy described below will allow a facility to produce readable Nemeth Code Braille without the cost of a Braille transcription contract.

The procedure is very simple. The implementation of the procedure is complex from the perspective of hiring, training and administration. This is a computer based system, but it is still a manual system. The advantage of this system is that it is not necessary for the computer transcriber to be Braille transcriber. It is only necessary that student transcribers have a through knowledge of math, be independent workers and be paid enough to keep them as employees. Student workers should be trained, evaluated and retrained on a continuing basis. It is also necessary to have the Braille produced by this program evaluated by students or transcribers who are knowledgeable in **Nemeth Code Braille.** This procedure will meet the Braille math requirements of post secondary institutions, but it will only work as long as it is evaluated at every stage for accuracy.

The first step is to **realize that most of the symbols used in higher mathematics are not part of the traditional ASCII character set.** Also the printed page typesetting of mathematical equations does not readily allow translation by a **Scanner/OCR system into a symbol-by-symbol, line-by-line ASCII document**. **It is therefore not possible to scan and translate directly from a typeset mathematical equation to an ASCII equivalent**. Instead, a two step process must be done anytime there are mathematical equations or other text or symbols that cannot be directly converted to ASCII. First, the **text must be scanned and recognized by an OCR package to convert any literal text on the page to ASCII.** Symbols or equations that cannot be directly converted will appear incorrectly or possibly not appear at all in the converted ASCII document. The **next step is to edit the document with a word processing package and note where equations or symbols occur in the text**. At the appropriate places, Braille translation software control codes must be inserted to allow for the Nemeth Code Braille characters to replace the non- ASCII symbols or equations. This is essentially the entire mathematics translation process. There are, however, several considerations that make the process more sophisticated than this simple description.

In order to produce readable mathematical/symbolic braille, the computer operator must be aware of how the software package converts text to Braille and how it produces Nemeth code. The most common mode of Braille conversion is to Grade II. **Grade II Braille uses contractions and concatenations which greatly reduce the size of the finished document.** These contractions are automatically done during the conversion of ASCII text to Grade II Braille. If a mathematical equation is entered in Grade II Braille, there is the possibility that the translation software will contract and concatenate the equation making it unreadable or misleading. Alternatively, the translator may replace a mathematical symbol with the word equivalent. For example the following "1 + 2 = 3" may be translated as "1 plus 2 equals 3." In some cases this may be acceptable, but in others it may not. To prevent this, **it is necessary to tell the Braille translation software when to drop out of Grade II and go to literal Braille,** which is a character for character translation. It is in this mode that production of math and graphics is possible.

In Duxbury the command $cz is used to go into literal Braille. This command is inserted at the beginning of a section of math notation to instruct the translator to used the exact characters that follow it and not to try to insert contractions. The codes inserted in literal Braille are Nemeth Code symbols and will be the same codes used to translate the

equation regardless of the Braille translation software. At the end of the math notation the Duxbury command $tx is used to inform Duxbury that it can now go to Grade II conversions again.

While it is important that they have an understanding of mathematics, they do not need an extensive knowledge of Braille. A few formatting rules concerning spacing and a "Nemeth Database" of math symbols and the corresponding Duxbury codes, Appendix B, that are required to produce the Nemeth code for that symbol is all that is needed to produce readable Nemeth Code translations. Initial training time to learn the "system" takes several hours/days. The person must have a knowledge of the math equations printed in the original document. The student need only enter the appropriate control codes and Nemeth Braille symbols to translate mathematical equations.

The code for the equation stated above, **1 + 2 = 3, would be: $cz #1+2 .k #3$tx**

Where
$cz indicates the start of literal translation
 # indicates the symbol following is numeric
(The # is used after a space or a negative sign.)
.k is the Nemeth code for =
$tx indicates return to Grade II translation

Once the codes are inserted, the entire document can be converted to Braille. The areas that specify literal Braille are converted, as is, while the rest of the document is contracted and concatenated to Grade II Braille. At this juncture, to insure that the translated text maintains its readability and integrity, the computer operator should look at the document in a text editor or a word processor to see exactly how the document format will appear after it is printed. The document will not be readable since the translation has contracted characters, but the overall format is discernable. If an equation is separated by a page break, additional editing must be done to preserve the integrity of the equation.

The document can then be printed on a Braille embosser. This procedure is considerably less expensive and usually more accurate than having a Braille transcription of higher mathematics done outside the university.

Tactile Graphics, An Overview and Resource Guide

John A. Gardner
Science Access Project
Department of Physics
Oregon State University See: **http://dots.physics.orst.edu/tactile/tactile.html**

Converting Text to ASCII

(Computer Based Large Print)

The majority of this paper has discussed conversions to Braille. It could be assumed that the conversion of text to large print would be a simple process that avoided the extensive editing of a Braille conversion. If the textbook to be converted is simple literary text, the process is simple; scan, convert with OCR software and check for accuracy. If the textbook has graphs, pictures, inserts, special fonts, etc…, then the process that has been discussed will not be satisfactory.

The objective in a multi-format textbook is to preserve the structure of each page while converting the text into ASCII. The conversion is possible with products such as **Windows based TextBridge Professional by Xerox** (508) 977-2000 will scan text and preserve the original format. This product will allow you to create a computer based file.

Please be aware that anytime graphical information is saved, it will **take considerable more disk space** than simple text. If you use a produce like TextBridge be sure you have the capability to store large files.

Printed, Large Print Documents

Printed Large Print documents are difficult to create. If the document is just text, the process is similar to regular text, but there is the added objective that the document must be printed in a recognizable form. There are many forms of text which are not recognizable when converted to large print on a regular 8.5 X 11.00 paper. T**ables, pictures, graphs, and some equations are examples of text forms which do not translate their meaning when they are printed in an enlarged form. Textual units are often split on different lines and lose their contextual meaning.**

It is possible to edit equations so that they are understandable when they are enlarged. It is also possible to use a product like **VisAbility by Ai Squared,** (803) 362-3612, to enlarge a graphic then segment the enlarged image on different sheets of paper and print each section. The segments must then be assembled on a table or large surface to be viewed.

Appendix A

Appendix B, NEMETH DATABASE

Appendix A

(The Macros listed in this Appendix use Duxbury Braille Translation software.)

Braille Formatting & Duxbury Dollar Codes

Dollar Codes in Duxbury control all the formatting in the Braille translation. Generally type in the word-processor, so that the translator converts automatically when it sees a command. To present translation in a accurate & professional way the following formats are recommended.

1. Main Headings & Chapter Titles.

All main headings and chapter titles **should be centered and should have a blank line after the heading.** Centering the text is done by including the text between $hds[space] and $hde[space]. This command will automatically put a line after the heading. The most recommended places for usage are in the beginning of the chapter and major section of the book etc.

example:
$hds[space]Mathematics 117[HRt]
Chapter 1$hde[Space]

The concept of headings is quite complicated. There are no set rules to use the various heading levels in Braille. It is left to the judgment of the Transcriber to decide on it. The minor headings are indented 5 spaces. This is the most used heading style inside a document. All the minor headings inside a chapter should be formatted using this style.

example:
[HRt]
[4 spaces]Section 1.1[HRt]

Most of the other headings can be formatted by leaving a blank space before it and leaving the heading left aligned without any spaces.

2. Body Text:

The main body of the document is formatted using the **bodytext style.** Body text starts in cell 3, so you need two spaces in front of each new paragraph in the section. The two space indentation holds true for paragraphs starting with numbers also. If document is a list, it should be left-aligned, hence no spaces before them. But to signify the difference between the list and the main body of the text, necessary blank lines should be inserted.

3. Transcriber's Notes:

The Transcriber's Note should start at cell 7, and if it runs into second line it should be from cell 5. This is done by using the $ind command. The correct usage is shown in the example. There is no Tnote, Tn or any other prefix for the notes. It should start with

[comma][apostrophe] and should end with [comma][apostrophe]. After the Transcriber's note is complete you need to reset the indentation by using $ind0[space] in the beginning of the next paragraph/format/text.

example:
$ind5[space][space][space]$cz[space],'$tx[space]

This tables formatted for Braille.
The columns are ..... $cz[space],'$tx[space][HRt]
$ind0[space]Section 2.2[HRt] etc..

4. Indentation:

The command shown in the previous section $ind[space] has many more uses than just in Transcriber's notes. You can use it in formatting multiple choice questions in exams, actual indented print material etc.

example:
1. What is the city that is known as the Emerald City?
$ind2[space]a. Seattle
b. Salt Lake City
c. Phoenix
d. Las Vegas
$ind0[space]2. What is the capital of Iraq? etc....

5. Page Numbering (Print & Braille):

Use **text-book style** to convert all the material into Braille. The standards set for translation require the print page indicator, print page number and Braille page number. Duxbury does this numbering by following the dollar codes that will tell it where each print page starts. The command is $leaf-[space]45. This will create a line across the page with the page number at the end of the line and put the print-page number also. Use of the command has the necessity that you follow the print page indication very accurately. If you forget to indicate the next print page all the Braille pages will be numbered with the previous print page, like a.45, b.45, c.45, etc until it sees the next $leaf- X command. The other advantage of using this command is that you don't have to worry about the print page indicator breaks, or print page indicator running into two pages. The Duxbury translator will not put the print page indicator if it happens to be the beginning of a new Braille page.

example:
$leaf-[space]45[HRt]

6. Page Breaks:

The information when translated into Braille shouldn't confuse the user. The page breaks should be used to signify the difference in sections and documents. For example, in an exam the instructions for the exam and the actual exam questions should start on separate pages. You cannot continue the exam questions right after the instructions. You don't have to worry about how many blank lines you should insert to go to a new page. As soon as you done with one section, if you want to force a new page you can type the command $pg[space]. This will stop the current page and start the following text on a new page beginning.

7. Tables:

Tables are one of the most difficult aspects of translation. The judgment of the Transcriber is very essential to convey the meaning across in the most accurate way to the Braille user. The table should be separated from the other text by lines. The lines in Braille can be generated by using a line of x's, g's, 7's or l's. The difference between these lines is just the dots that are generated by each letter. Usually the beginning of a box is denoted by a line of 7's and the end of the box is generated by a line of g's. Line of l's is like a thick line in the print, and can be used if needed to signify the difference in the tables.

8. Other miscellaneous notes:

There are many other common mistakes that have to be corrected by the Transcriber before using the Translation. The backslash / can be left as it is, if it is in the middle of a word, like he/she or then/now. But it cannot be left alone if it is all by itself, like - - - / -. When it is all by itself we should go convert it as $cz[space]_/$tx[space]. This is true for any other symbols, like -, = etc. If they are in the middle of the text, we don't have to worry about it.

example:
"This is a compliment to him/her"
"This is supposed $cz _/$tx not supposed to be like this."

Other minor things that shouldn't be ignored are including a text indicator before a roman numeral etc. These things we hav to incorporate by referring to the nemeth code book more often.

The following Macros are WordPerfect 5.1 Macros and are designed to insert Duxbury formatting commands into a text file. This is not meant to imply that Duxbury is the only software package capable of converting text to Braille effectively.

Macros : Please assign a meaningful Macro key for each function as you create these macros.

Inserts Braille page numbers

DISPLAY(Off!)
Type("$leaf- ")

Transcribers Note

DISPLAY(Off!)
Type("$ind5 $cz ,'$tx $cz ,'$tx ")
GETSTRING(Note; "Enter the note here:")
PosWordPrevious PosWordPrevious Type (Note)
PosWordNext PosWordNext
Type("$ind1 ")
HardReturn

Heading Centered

DISPLAY(Off!)
PosScreenLeft
Type("$hds ")
PosLineEnd
HardReturn
Type("$hde "

Remove Hard Returns

SAVESTATE PERSISTALL
AutoCodePlacement(OFF!) WP51CursorMovement(ON!) VARERRCHK
(OFF!)

PosDocTop
ASSIGN(Res; "n")

LABEL(Search)
SearchString("") SearchNext(Regular!)

ASSIGN(
//*** Conversion warning ***
//*** Name modifed to avoid conflict with command name
Next_;
//*** Conversion warning ***
//*** Use ?RightCode for codes; ?RightChar for text
?RightChar)
ASSIGN(Set; CTON(
//*** Conversion warning ***
//*** Name modifed to avoid conflict with command name
Next_) DIV 256)
ASSIGN(Num; CTON(

```
//*** Conversion warning ***
//*** Name modifed to avoid conflict with command name
Next_) %256)

PosCharNext
ASSIGN(Next3;
//*** Conversion warning ***
//*** Use ?RightCode for codes; ?RightChar for text
?RightChar)
ASSIGN(Set3; CTON(Next3) DIV 256)
ASSIGN(Num3; CTON(Next3) %256)
PosCharPrevious

PosWordPrevious
ASSIGN(Next2;
//*** Conversion warning ***
//*** Use ?RightCode for codes; ?RightChar for text
?RightChar)
ASSIGN(Set2; CTON(Next2) DIV 256)
ASSIGN(Num2; CTON(Next2) %256)
PosWordNext

IF(Num=32)
IF(Num3=32)
PosLineEnd PosCharNext PosLineUp BlockOn(CharMode!) Pos CharNext
PosWordNext r CHAR(Res2; "Delete? Y)es N)o M)inus:") ASSIGN(Res2; NTO
C(Res2) ) SWITCH(Res2)
CASEOF "y": CALL(DELETE)
CASEOF "Y": CALL(DELETE)
CASEOF "M": CALL(MINUS)
CASEOF "m": CALL(MINUS)
CASEOF "q": CALL(ENDING)
CASEOF "Q": CALL(ENDING)
CASEOF "n": CALL(NO)
CASEOF "N": CALL(NO)
ENDSWITCH
ENDIF
ENDIF

IF(Num2<>35)
IF(Num>96)
IF(Set=0)

SWITCH(Res)
CASEOF "a": GO(BRANCH)
ENDSWITCH
```

```
Type("[-> ") BlockOn(CharMode!) PosCharPrevious PosCharPrevious
PosCharPrevious PosCharPrevious PosCharPrevious
CHAR(Res; "Replace [Return] with [Spc]? Y)es, N)o, A)ll
Q)uit:") ASSIGN(Res; NTOC(Res) )
BlockOff DeleteCharNext DeleteCharNext DeleteCharNext
```

DeleteCharNext

```
LABEL(BRANCH)
SWITCH(Res)
CASEOF "y": CALL(YES)
CASEOF "Y": CALL(YES)
CASEOF "a": CALL(YES)
CASEOF "A": CALL(YES)
CASEOF "q": CALL(ENDING
//*** Conversion problem ***
//*** Label already referenced with different keystrokes
//
)
CASEOF "Q": CALL(ENDING
//*** Conversion problem ***
//*** Label already referenced with different keystrokes
//
)
ENDSWITCH
ENDIF
ENDIF
ENDIF
```

GO(Search)

```
LABEL(YES)
DeleteCharPrevious Type(" ")
RETURN

LABEL(ENDING
//*** Conversion problem ***
//*** Ambiguous keystrokes leading to label
```

```
)

PosScreenUp
QUIT
```

```
LABEL(DELETE)
BlockOff PosLineEnd PosCharNext PosLineUp DeleteWord
DeleteCharPrevious Type(" ")
RETURN

LABEL(MINUS)
PosCharPrevious
CALL(DELETE)
RETURN
//*** Conversion problem ***
//*** Control transfer with pending token
//{Block}{Right}{Word Right}{Left}

LABEL(NO)
BlockOff
RETURN
```

Search for extra spaces and change to 1 space

```
SAVESTATE PERSISTALL
AutoCodePlacement(OFF!) WP51CursorMovement(ON!) VARERRCHK
(OFF!)

DISPLAY(OFF!) ReplaceConfirm(No!) SearchString(" ")
ReplaceString(" ") ReplaceForward(Regular!)
PosDocTop ReplaceConfirm(No!) SearchString(" ") ReplaceS tring(" ")
ReplaceForward(Regular!) PosDocTop
```