1976

TEXT EDITOR IMPLEMENTATION FOR THE THIRD NORMAL FORM SYNTHESIS SYSTEM

BY

T. J. STEVENS

B.S., UNIVERSITY OF NEBRASKA, OMAHA, 1972

A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

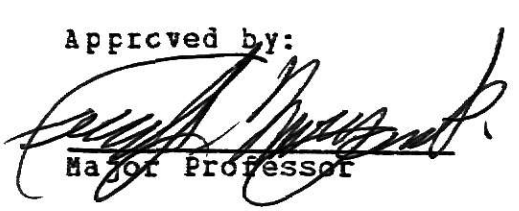
MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1978

Approved by:



Major Professor

ACKNOWLEDGEMENTS

For their guidance and assistance in the preparation of this report, I would like to thank Doctor Fred Maryanski, Doctor Paul Fisher, and Ed Basham.

TJS

ILLEGIBLE DOCUMENT

THE FOLLOWING DOCUMENT(S) IS OF POOR LEGIBILITY IN THE ORIGINAL

THIS IS THE BEST COPY AVAILABLE

TABLE OF CONTENTS

ILLUSTRATIONS

CHAPTER 1

INTRODUCTION

1.1 Defining the Project Problem

This report deals with the implementation of a Text Editor for the THIRD NORMAL FORM SYNTHESIS SYSTEM. The goal of this report is to describe in a top down structured format how the Text Editor was conceived and developed.

The need of a Text Editor was created in the initial implementation of a Data Base Management System (DBMS) based upon a relational model. To process data in a relational environment requires the attributes of a relationship to be consistent and non-duplicative[2]. To obtain this state, one must maintain proper representation of the data in a Third Normal Form (3NF). Several methods for reducing data to 3NF have been developed. One of these is Bernstein's algorithm which automatically synthesizes 3NF relations from functional dependencies of attributes. A decision was made to use Berstein's algorithm and to establish a basic system called the THIRD NORMAL FORM SYNTHESIS SYSTEM.

As more detailed study went into the development of this system, it was decided to set up four distinct sections as delineated by function. These are as follows:

1. User Interface
2. Functional Dependency Generator
3. Functional Dependency Analyzer
4. Text Editor

1-2 Defining The Text Editor Problem

The Text Editor function was defined as to have the ability to alter input statements from the User Interface phase for reprocessing of that input. The development of the problem statement led to defining the function of a Text Editor and specifying its capabilities for the 3NF Synthesis System.

A Text Editor is an interactive software tool that uses pattern matching techniques in allowing the on-line user to create and modify symbolic text for any purpose[6]. User capabilities should include inserting, deleting and changing lines of texts, and a symbolic search feature with a program listing function[1].

With this definition in mind, and the problem statement requirement, the Text Editor for the 3NF Synthesis was evaluated. The use of the system editor of the NCR 8250 System was considered because of its availability. One major argument against using the system editor was to make the 3NF Synthesis System portable for commercialization. Additionally, the system editor was slow and required more than twice the storage space of the proposed editor. So a decision was made to construct a completely new Text Editor in NCR interactive COBOL language[4].

CHAPTER 2

BACKGROUND

2-1 General Overview of 3NF Synthesis System

To fully understand why the 3NF Synthesis System was develcped, requires a basic understanding of Data Base Management Systems (DBMS). Currently, there are three recognized data models in DBMS that are used and they are as follows:

1. Network
2. Hierarchical
3. Relational

The most common commercial systems used are the Network and Hierarchical data models. The relaticnal mcdel is not yet commonly used because of its extra memory and processor requirements. However, the Relaticnal data model is the only data model that has a sound workable mathematical basis.

Therefore to develop a workable Relational Data Base Model requires the initial set up of data and its relationships into third normal form (3NF). By representing data in 3NF allows the use of the mathematical tools of Relaticnal Algebra and Calculus[5] in constructing and using data in a relational environment. The representing of data in 3NF is not a trivial task. Therefore, the need for a 3NF synthesis algorithm arises. To use Berstein's algorithm requires the input to be in a valid form of functional dependencies. To take basic data off a business report of a commerical user and synthesized functional dependencies requires considerable programming effort.

2-2. Structure of the System

The 3NF Synthesis system accepts a description of the user data in a form that can easily be realized from a typical report. This input is collected interactivly by the User Interface program which produces a hierarchical representation of the data. This output is known as the Data Description Source File. The Functional Dependencies Generator program (FDGEN) reads the hierarchical data representation and interacts with the user to produce a set of functional dependencies. Then the Functional Dependencies Analyzer (FDA) which implements the Bernstein's algorithm generates relations in 3NF.

It is possible that the user might make a mistake during the User Interface program or that an error may be detected during the Functional Dependencies Generator program. In order to prevent the user from having to reenter all the data from the Data Description Source File, the Text Editor was developed. The editor operates on the output of the User Interface Program and permits the user to change any erroneous information. The editor then produces a file which is used as input to a special version of the User Interface program which checks for additional errors and reconstructs the hierarchical data representation. The flow of the system is depicted in Figure 1. Additional information on the 3NF Synthesis system can be found in reference 2.

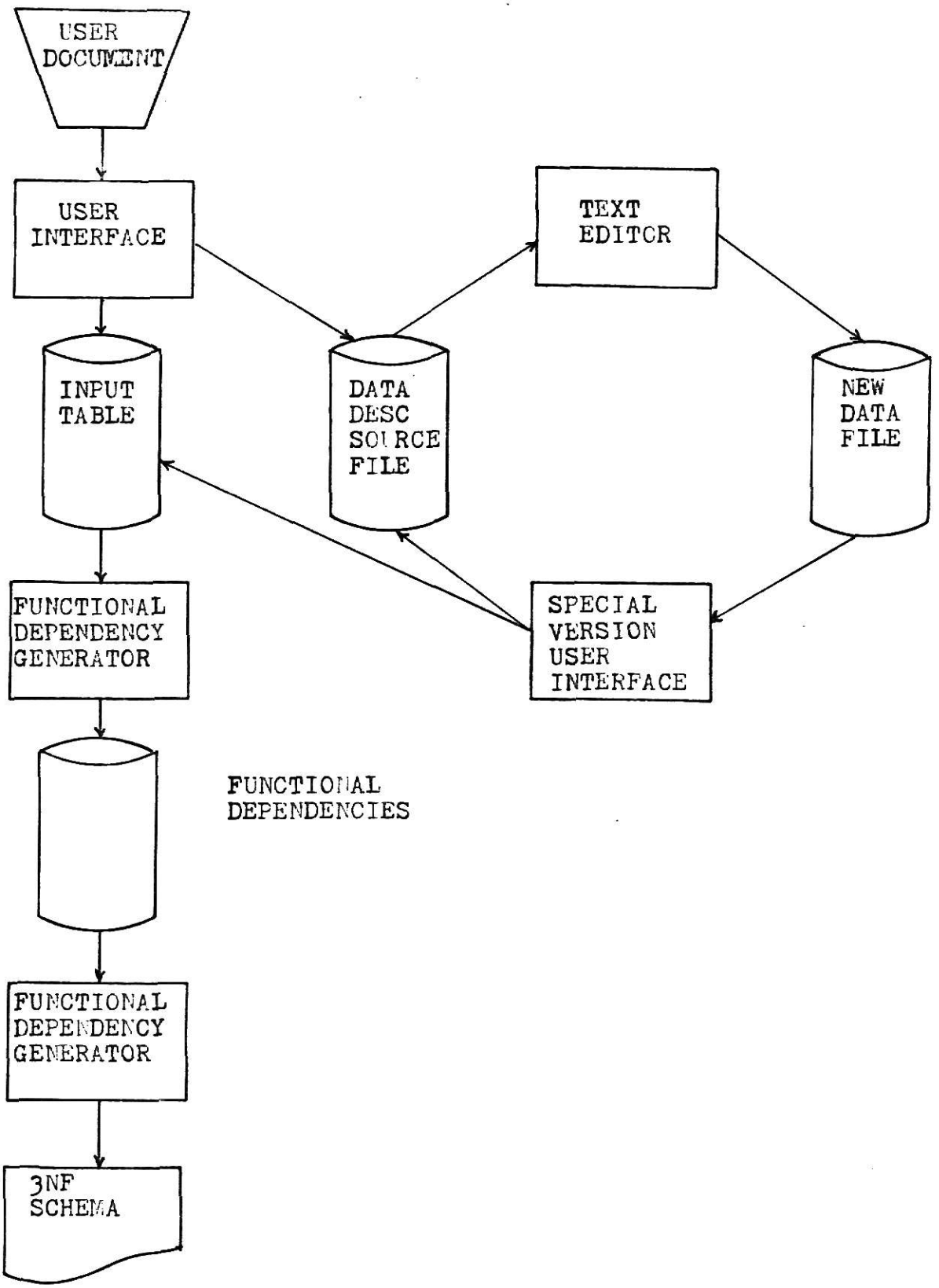


Figure 1: DATA FLOW OF THE THIRD NORMAL FORM SYNTHESIS SYSTEM.

2-3 General Features of Other Editors

All interactive computing systems have some form of editing facility but it is often primitive. Basically, the editor when initiated, copies a file into an internal buffer. The text is modified in the buffer and eventually written back to some external file. The text is never modified except by explicit command. If a file is ruined in the editor routine, it can be cancelled out and a fresh copy read into the buffer and editing start anew. Most editors are "line oriented" in that most editing commands operate on one or more lines. This is a natural organization since most text intrinsically comes in lines. Most editors are not specific about the structure on lines. It is usually presumed that the user of an editor program knows what he is doing and does not require detailed information. The common features are:

Add lines

Delete lines

Print lines

Change or modify characters

Find character(s)

Concatenate strings

Merge lines

Sort lines by keys

One of the major consideration in designing an editor is to structurally engineer the interactive commands to be concise and clear. Error recovery is the second major influence in

the design. The editor maintains precious files and so must be cautious in that when a user enters erroneous commands, it must recovery gracefully and not lose the files[1].

CHAPTER 3

TEXT EDITOR DESCRIPTION

3-1 Design Requirements

The Text Editor designed for the Third Normal Form (3NF) Synthesis System had to satisfy the initial following conditions.

1. Operate upon the Data Description Source File produced by the User Interface program.

2. Provide basic deletion, insertion and resequencing capabilities.

3. Output a modified file using the same data structure used by the input file. This modified file must be sequentially processed through a modified User Interface program which checks for additional errors and reconstructs the hierachial data representation.

4. Implement the editor on the NCR 8250 System using the NCR Interactive COBOL language.

3-2 Design Characteristics

In addition to the basic requirements the following design characterstics were included to allow for a more complete and useful Text Editor. These characteristics are as follows:

1. Make the editor a powerful software tool in creating and modifying the Data Description Source File.

2. Stress error recovery in assisting the user.

3. Human engineer the interactive CRT display of questions and commands for conciseness and simplicity for the unsophisticated user.

4. Design the editor in a top down fashion.

5. Modularize the different function of the editor.

3-3 Overview of Lifecycle

From the basic requirements and design considerations the overall functions and routines of the Text Editor evolved. The editor was implemented using the NCR version II Interactive COBOL language. The editor was structured so that each function would be modular and independent. This allowed the modules to be in different stages of construction and facilitated debugging and testing of the editor program. The editor was envisioned to perform the following operation on the Data Description Source File:

1. Display the file.
2. Delete lines of the file.
3. Add lines to the file.
4. Resequence the lines of the file.
5. Find a character string in the file.
6. Change a character string in the file.
7. Provide a tutorial on the use of the editor.
8. Be able to exit the program gracefully and save the modified file or return the original Data Description Source File.

The first four functions, Display, Add, Delete, and Resequence, are line oriented and were envisioned to operate on a single line, groups of contiguous lines, or all the lines in the file. The two functions, Find and Change, were

character oriented thus permitting the user to locate and change the first occurrence of a specific character string in each line of the file. The HELP function provides a tutorial on the syntax and semantics of the other functions.

3-4 Design Philosophy

In accordance with top down structuring techniques [3] and from the definition of the editor requirement, data structures (Appendix A) and data access graphs (Figure 2) were completed. These software tools gave an overview of the editor program and indicated possible trouble areas. A high level algorithm (Appendix A) was constructed. At the completion of the algorithm, a walk through was conducted for each module to see if the program would fulfill the basic and design requirements. Error recovery was stressed and test cases were developed. As each module was coded, testing and debugging took place. Approximately sixty hours of machine time was used for the complete program. Documentation of each module was kept in a historical file to preclude the same error being made in another module.

3-5 Features

The finialized form of the Text Editor has the following features that provide editing capabilities of the

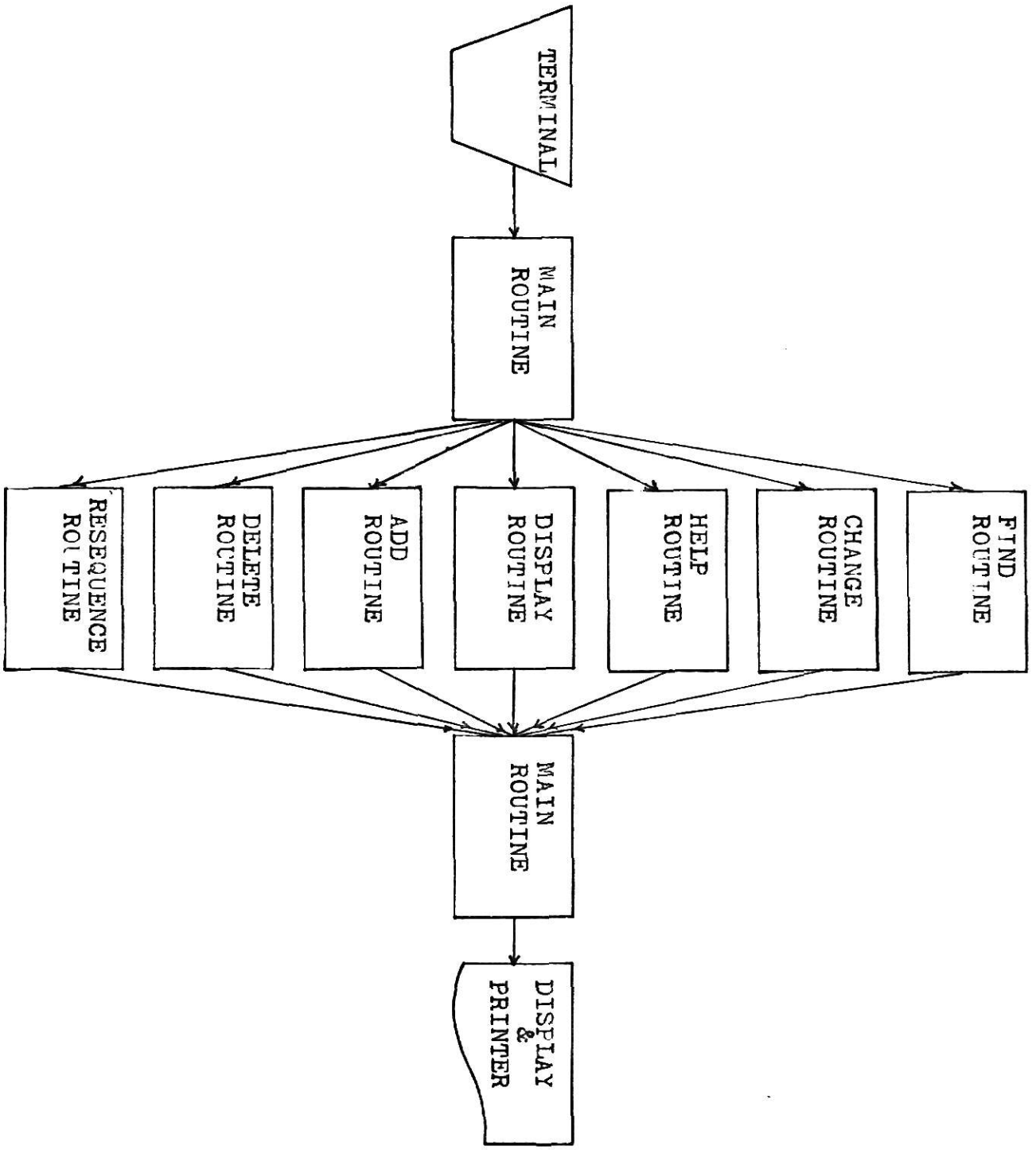


Figure 2: TEXT EDITOR DATA ACCESS GRAPH

Data Description Source File that is created during the User Interface program:

DISPLAY ON CRT

ADD

DELETE

RESEQUENCE

FIND

CHANGE

HELP

DISPLAY FUNCTION

The DISPLAY function displays on the CRT the Data Description Source File. The complete file, selected portions of the file or a single line can be displayed. The format is "D aaaa ,bbbb ", where aaaa is the beginning line number and bbbb is the ending line number to display. To display the complete file "D ALL" is the format. General format:

$$\text{DISPLAY or D} \left[\begin{array}{l} \text{,aaaa} \\ \text{,aaaa,bbbb} \\ \text{ALL or A} \\ \text{LAST or L} \end{array}\right]$$

Blanks, stars and commas are used as delimiters in the command. In "DISPLAY ALL", the screen displays 22 records then request new/line to be typed in to roll the screen.

DELETE FUNCTION

The DELETE function deletes one to n lines of existing records from the Data Description Source File. Format is

"DEL aaaa ,bbbb ", where aaaa is the beginning line number and bbbb is the ending line number. Presence of aaaa indicates deletion of a single line. General format:

```
DELETE [,aaaa        ]
or   D [aaaa,bbbb    ]
```

Blanks, stars and commas are used as delimiters in the command. If the complete file is deleted, no records can be added from the editor routine.

ADD ROUTINE

The ADD function inserts one or more lines to the existing Data Description Source File. The format is "ADD AFTR nnnn", where nnnn is a valid line number of an existing record. One or more lines can be added. Additionally a previous record can be recopyed by typing in the line number. To stop the ADD function, a blackslash "\" must be typed in position 1 of the last line of input. General format:

```
   ADD [AFTR nnnn          ]
or  A  [,AFTR,nnnn         ]
    \  [(to stop function) ]
```

Blanks, stars and commas are used as delimiters in the command. No more than 100 lines are allowed in the Data Description Source File.

RESEQUENCE FUNCTION

The RESEQUENCE function allows the line numbers of the Data Description Source File to be changed to new sequential

line numbers. Format is "RESEQ cccc ,dddd", where RESEQ will resequence the complete file starting at line number 0000, by default, by 10. The use of cccc indicates the starting line number and dddd indicates the amount the lines are tc be resequence. General format:

```
RESEQ  [                ]
or RE  [ ,cccc          ]
       [  cccc,dddd     ]
```

Blanks, stars and commas are used as delimiters for the command. Although the lines can be resequenced to any amcunt while in the editor routine, they are resequenced by 10 when the editor is terminated. This resequence is necessary as a result of the structure of the User Interface program[2].

FIND FUNCTION

The FIND function allcws for a string eeee to be found in the Data Description Source File. The search area can be a single line, a group cf contiguous lines or the complete file. Format is "FIND/eeee/ALL", where the maximum length of eeee is 40 characters. General format:

```
FIND[/eeee/  ALL or A   ]
or  [/eeee/  aaaa       ]
  FI[/eeee/  aaaa,bbbb  ]
```

String eeee must be enclosed by slashes. Commas and stars are delimiters. If a string eeee is not found in the search area then a message to indicate string not found is displayed.

CHANGE FUNCTION

The CHANGE function allows a string eeee to be changed to string ffff in the first occurrence of a single line, a group of contiguous lines, or the complete file. Format is "CH/eeee/ffff/ALL,V" where "V" means the old and modified lines are displayed for user's verification. General format:

```
CHANGE [/eeee/ffff/ ALL or A                ]  [,V]
 or CH [/eeee/ffff/ aaaa                    ]
                    aaaa,bbbb               ]
```

Slashes are used to enclose strings eeee and ffff. Blanks and commas are used as delimiters.

HELP FUNCTION

The HELP function provides a visual display on the CRT of the general formats of the six other functions of the editor routine. Two pages are used in the visual display. The first page contains the formats of DISPLAY, DELETE and ADD. When the viewer finishes with the page new/line is typed in which displays the second page. The formats of FIND, CHANGE and RESEQUENCE are displayed. The format is "HELP".

CONCLUSION

To terminate the Text Editor, the format "QUIT" or "QU" is used. Once this is entered, the question whether the changed file (Edit File) is to be saved is displayed. If the answer is "yes", the Edit file replaces the old Data

Description Source File. If the answer is "no", the old Data Description Source File is returned instead of the modified file.

3-6 OVERVIEW OF THE TEXT EDITOR

In order to use the Text Editor, job control language statements are required to assign the Data Description Source File to the editor's Old-Data-File. This file is read into a working storage table called WS-Table. The format of the input is shown in Figure 3. The size of the WS-Table is 101 lines with each line containing a maximum of 77 characters. Line 101 of the WS-Table is used for the Find and Change routines. As the file is read into the table the lines are doubly linked by pointers WS-NEXT and WS-LAST. If the file being read does not take all 100 lines, then the remainder are doubly linked in a free list that is used in the Add and Delete routines (Fig 4). Once the file is read in, a question on the use of the line printer is asked. If the answer is yes to the question, then the line printer is set up to produce a printed output when the editor routine is terminated.

Next a statement appears on the CRT asking the user to enter the appropriate command for the needed edit function. To facilitate the discussion of the editor functions and user interactions, each command will be discussed in the order displayed on the CRT.

```
POS  1 - 4 5 6 7                                                    76
     XXXX  C b XX----------------------------------------------------XX
```

User Data

COLUMN 1-4 : SEQUENCE NUMBER
COLUMN 5 : STATUS
COLUMN 6 : LEVEL
COLUMN 7-76 : USER DATA

Figure 3: INPUT DATA STRUCTURE FROM USER'S DATA DESCRIPTION SOURCE FILE

WS-TABLE

WS-LAST 1 - 3	LINE NO	SEQ NO 1 4567	USER DATA	77	WS-NEXT 1 - 3
0	1	0000	REPORT ORGANIZATION		2
1	2	0010	DEPT DBUDGET MANAGER	Y	3
2	3	0020	EMP PROJ OFFICE PHONE	Y	4
3	4	0030	PROJ PBUDGET	S	5
4	5	0040	OFFICE AREA	S	6
5	6	0050	END		0
0	7				8
7	8				9
•					
•					
•					
97	98				99
98	99				100
99	100				0
	101				

FREE LIST POINTER = 7

Figure 4: DATA STRUCTURE OF WS-TABLE

If the user requests the DISPLAY function, then a single line, multiple lines, or the complete file can be displayed. The DISPLAY function is line oriented on the sequence number of each line. If "DISPLAY ALL" is the command typed in, then the routine sets pointers at the start of the file (WS-START) and searches until the end of the file and sets the ending pointer (WS-END). The Write submodule is called and displays line by line of the doubly linked table until the end pointer is reached. If more than 22 lines are to be displayed, a roll feature allows the first 22 lines to be displayed. The screen is rolled when new/line is typed in which allows the rest of the lines to be displayed. If the command is to "DISPLAY nnnn" for a single line then the table is searched until nnnn is found. A pointer WS-START is set and the write submodule is called and the line nnnn is displayed. If the command is "DISPLAY nnnn,mmmm" then the table is searched for nnnn. Once found the pointer WS-START is set and the the table is searched for mmmm. Once found the pointer WS-END is set. The write submodule is called and all lines that are linked from WS-START to WS-END are displayed. The command "DISPLAY LAST" is handled like a single line. Error routines include the standard sequence not found, sequence out of order and syntax error in formatting.

The DELETE routine deletes lines from the WS-Table and adds the deleted lines to the free list. The command "DELETE nnnn" will delete only one line. When the command

is entered the format is checked and the line nnnn is searched for in the table. Once found the pointer WS-START is set and the write submodule is called to display the line nnnn. The statement that the following lines are to be deleted is displayed with a request for verification. This prevents the erroneous deletion of pertinent information. If the deletion is verified then the pointer WS-START is moved to the free list while the index pointers WS-LAST and WS-Next are modified for the WS-Table and free list. This is the same as deleting a node from a doubly linked list and adding the deleted node to space available list. See Figure 5 for an example. If more than one line is to be deleted then the pointer WS-END is set up to point to the last line to be deleted. No change is made in the deletion of the node submodule. The standard error routine for formatting and sequencing are the same.

WS-TABLE

WS-LAST	LINE NO	SEQ NO	USER DATA		WS-NEXT
1 - 3		1 4567		77	1 - 3
0	0	0000	REPORT ORGANIZATION		2
1	2	0010	DEPT DBUDGET MANAGER	Y	3
2	3	0020	EMP PROJ OFFICE PHONE	Y	4
3	4	0030	PRCJ PBUDGET	S	5
4	5	0040	OFFICE AREA	S	6
5	6	0050	END		0
0	7				8
7	8				9
•					
•					

FREE LIST POINTER = 7

BEFORE DELETION OF LINES 10-20

AFTER DELETION OF LINES 10-20

WS-LAST	LINE NC	SEQ NO	USER DATA		WS-NEXT
1 - 3		1 4567		77	1 - 3
0	1	0000	REPCRT ORGANIZATICN		4
0	2				3
2	3				7
1	4	0030	PROJ PBUDGET	S	5
4	5	0040	CFFICE AREA	S	6
5	6	0050	END		0
3	7				8
7	8				9
•					
•					

FREE LIST POINTER = 2

Figure 5: DELETION OF LINES FROM WS-TABLE

The ADD routine inserts lines by modifying the existing WS-LAST and WS-NEXT pointers in the free list of the WS-Table. This gives the appearance of the lines being added even though the lines are already in the WS-Table. When the command is of the form "ADD AFTR nnnn", the format is checked and then the WS-Table is searched for nnnn and the pointer WS-START is set. "ADD AFTER THIS LINE" and line nnnn are displayed on the CRT. The routine then waits for the data to be entered by the user. For each line entered, a question concerning the level of the data is asked [2]. When the user gives the correct response, the line pointed to by the WS-FREE-LIST pointer is inserted after line number nnnn. This is accomplished by the same technique of inserting a node in a doubly linked list. Figure 6 illustrates the use of the ADD function. Pointers WS-LAST and WS-NEXT are modified in the WS-TABLE. If the data entered is numeric, a line previously entered is to be repeated and the line number is searched for in the table. If not found a statement of invalid line number is given and the user is told to reenter the data for that line again. If the line is a beginning or ending of a report, the question of level is not asked. After the data has been entered the new line is displayed on the CRT. The add routine is terminated when the backslash "\" is typed in as data. The standard error routine for formatting and sequencing are the same. The ADD routine will allow no more than 100 lines of data to be in the WS-Table. If this

amount is exceeded an error message is displayed indicating maximum lines in the table.

The RESEQUENCE function reorders the line numbers by a requested amount from a starting line number until the end of file is reached. The command "RESEQ" resequences the complete file from the starting line to the end of the file. The amount that is resequenced defaults to 10. The command is unstrung into TEXT1, TEXT2 and TEXT3. If TEXT3 is blank then the amount is 10. If the command is "RE nnnn", which indicates that TEXT2 is numeric and indicates a starting point for the resequencing. A statement is displayed on the CRT indicating that resequencing has taken place from line nnnn by 10. The command "RE nnnn,mmmm" indicates that from starting point nnnn, the resequence will be by amount mmmm. A statement is displayed that indicates all lines are sequenced by mmmm from nnnn. If data is entered that makes TEXT3 not numeric or a not positive number, then an error message indicates that all lines are resequenced from nnnn by 10 and that the requested amount, mmmm, is not possible. When the routine is finished, control is returned to the main module. See Figure 6 for an example of resequencing.

Addition of 2 New Lines Added After Line 0030

```
ADD AFTER 0030

    CUSTOMER                                          N
    EMP  PROJ  OFFICE  PHONE                          Y

                              WS-TABLE

WS-LAST LINE NO SEQ NO          USER DATA                  WS-NEXT
 1 - 3          1   4567                              77   1 - 3
   0       1    0000   REPORT ORGANIZATION                   4
   4       2           CUSTOMER                        M     3
   2       3           EMP  PROJ  OFFICE  PHONE        Y     5
   1       4    0030   PRCJ  PBUDGET                  S     2
   3       5    0040   OFFICE  AREA                    S     6
   5       6    0050   END                                   0
   0       7                                                 8
   7       8                                                 9
   .
   .

FREE LIST POINTER = 7

              AFTER RESEQ OF WS-TABLE FROM 0000 BY 10

WS-LAST LINE NO SEQ NO          USER DATA                  WS-NEXT
 1 - 3          1   4567                              77   1 - 3
   0       1    0000   REPORT ORGANIZATION                   4
   4       2    0020   CUSTOMER                        N     3
   2       3    0030   EMP  PROJ  OFFICE  PHONE        Y     5
   1       4    0010   PRCJ  PBUDGET                  S     2
   3       5    0040   OFFICE  AREA                    S     6
   5       6    0050   END                                   0
   0       7                                                 8
   7       8                                                 9
   .
   .

FREE LIST POINTER = 7
```

Figure 6: ADDITION OF LINES AND RESEQUENCING THE WS-TABLE

The FIND routine is character oriented and searches the WS-Table for string eeee. The maximum length of string eeee is 40 characters. From the main routine the FIND function is called with the command being accepted as TEXTS and unstrung into TEXT1, TEXT3 and TEXT2. Slashes, "/", act as delimiters. The syntax and format of the command is first checked then the length of the string is determined with pointer WS-INDEX pointing to the beginning of the string eeee. If TEXT3 is "ALL" or "A", then the complete file is searched. If TEXT3 is numeric and TEXT2 is blank then the file is searched only at the line indicated by TEXT3. If both TEXT3 and TEXT2 are numeric then the file is searched from TEXT3 to TEXT2. Beginning and ending pointers "WS-START and CNT-AMT are set by the search routine. The first line as pointed to by WS-START is multipled by 77 giving the end of the line (WS-END), then 70 is subtracted giving the start of the search area of that line. The first six characters contain the line numbers which are searched by a line search routine. Pointer WS-START is reset to the first search character position. This line is searched until the first occurrence of the first character is found or until end of line. Pointers WS-START and WS-INDEX are incremented and the next character is checked for a match. If no match then WS-INDEX is reset to the original value and the checking of a match on the first character continues until the end of the line. The checking of subsequent matches continues until the string is completely matched then the

line is displayed on the CRT. If no match is found in the first line then WS-START and WS-END pointers are set on the second line until the pointer CNT-AMT is found. This continues until pointer CNT-AMT is encountered. If no matches are found in the specified search area then a message stating string not found is displayed. The routine has the same standard error routines of formatting and sequencing. See Figure 7 for an example.

The next routine, the CHANGE routine is similar to the search routine except twice as many pointers are used. The routine searches the WS-Table for string eeee and once found modifies it to string ffff unless the verification option is requested or the modified line is truncated because of excessive length. The maximum length of string eeee and ffff is 35 characters. If these lengths are exceeded an error message is generated. From the main routine the CHANGE function is called with command being accepted as TEXTS and unstrung into TEXT1, ANS, TEXT3, TEXT2 and REM. The slashes in the command act as delimiters and help determine the length of string eeee and string ffff. Pointers are established at the beginning of string eeee and ffff in TEXTS. If TEXT3 is "ALL" or "A" then the complete file is searched and modified if any strings are found. If TEXT3 is numeric and TEXT2 is blank then only the line TEXT3 is searched and modified. If TEXT3 and TEXT2 are both numeric then the file is searched from line TEXT3 to line TEXT2 and modified if string eeee is found. If "V" appears

in TEXT2 or REM then both the modified and unmodified lines are displayed for verification before any modification is completed. Basically the search operation is similar to the FIND function. Pointers are set up by the values in TEXT3 and TEXT2, WS-START at the starting point and CNT-AMT at the ending point. The line pointed to by WS-START is searched for string eeee until the end of line, or until a match is found. Once a string match is found, a set of pointers are established for the beginning and ending position of string eeee. All characters in the old string are moved to WS-Table line 101 up the beginning of string eeee. Line 101 is used as a working area for modifying the strings. The modified string ffff is then moved into line 101 with a pointer indicating the ending position of string ffff. Then the rest of the old line is moved into line 101. If verification is requested in the command, the old line and modified line are displayed. If the user answers "yes" to the verification query then line 101 is moved to the position of the old line in the WS-Table. If a change is requested that causes the modified line to exceed 77 characters then an error message is generated telling the user to reenter the change request. If string eeee is not found in the search area as prescribed by the command, then a statement is displayed that the string is not found. Once the CHANGE function is completed control is returned back to the main module. See Figure 7 for an example.

Listed below is the diagram of the WS-Table and the individual pointers in the finding and changing of string eeee to ffff.

CH/OFF/XXX/30

Pointer WS-INDEX point to "0"
Pointer NAME-LENGTH points to the second "F"
Ponter CH-LENGTH points to the third "X"

WS-TABLE

WS-LAST	LINE NO	SEQ NO	USER DATA		WS-NEXT
1 - 3		1 4567		77	1 - 3
0	1	0000	REPORT ORGANIZATION		4
4	2	0020	CUSTOMER	N	3
2	3	0030	EMP PROJ OFFICE PHONE	Y	5
1	4	0010	PROJ PBUDGET	S	2
3	5	0040	OFFICE AREA	S	6
5	6	0050	END		0
0	7				8
.					
.					
99	100				0
	101	0030	EMP PROJ XXXICE PHONE	Y	

CNT pointer points to Line No 4.

Figure 7: FINDING AND CHANGING OF STRINGS

The Help function is a tutorial on the use of the seven basic commands in the TEXT Editor. When help is requested page one is displayed on the CRT with commands of DISPLAY, ADD and DELETE. Once the user finishes viewing the commands, new/line is typed in allowing the user to see three other commands of RESEQUENCE, FIND, and CHANGE. When the user finishes viewing the second page, new/line is typed in which returns control back to the main module. See Figure 8 for the display of the HELP routine.

When the editor routine is no longer needed, the command "QUIT" is typed in from the keyboard. When this command is received the question is asked if the Edit file is to replace the Old Data Description Source File. If the answer is yes then the Edit file is run through a special version of the User Interface Program. This file is automatically resequenced from line number 0000 by 10. If the answer is no then the Old Data Description Source File is returned.

There are two outputs when the editor routine is terminated. One output goes to the line printer if requested at the start of the editor. The output data structure is the same as that is displayed upon the CRT. The second output is the output that goes to the special version of User Interface program. This data structure is identical with the data structure that was originally sent in by the Old Data Descritpion Source File.

The following are the commands displayed when the HELP routine is called

Page 1

```
THIS INPUT EDITOR HAS 7 COMMANDS AND IS A MODULE WHICH PROVIDES
BASIC EDIT CAPABILITIES IN ORDER TO ALTER THE INPUT CREATED
DURING USER INTERFACE.

COMMANDS      FORMAT              DESCRIPTION

DISPLAY       DISPLAY,ALL         DISPLAYS COMPLETE FILE
              D,0100              DISPLAYS ONE LINE
              D,0100,0200         DISPLAYS MULTIPLE LINES
              D,LAST or 1         DISPLAYS LAST LINE
NOTE:  DISPLAY OR D, ARE BOTH CORRECT FORMATS.  COMMAS OR BLANKS
       CAN BE USED.

ADD           ADD AFTR NNNN       THE ADD FUNCTION ADDS 1 OR MORE
              A,AFTR NNNN         LINES AFTER LINE NNNN(4 DIGIT NO).
              STOP ROUTINE \      AFTER EACH LINE IS ADDED A QUESTION
                                  OF LEVEL IS ASKED.  TO TERMINATE
                                  ROUTINE TYPE IN \.

DELETE        DELETE AAAA,BBBB    DELETES FROM LINES AAAA TO BBBB
              DEL,AAAA            DELETES LINE AAAA ONLY
NOTE:  DELETE OR DEL ARE CORRECT FORMATS.  BLANKS OR COMMAS ARE
       DELIMITERS.
NOTE:  AAAA & BBBB ARE 4 DIGIT LINE NUMBERS

TO CONTINUE HELP, TYPE IN NEW LINE.
```

Figure 8A: DISPLAY PAGE 1 OF HELP ROUTINE

PAGE 2

COMMAND	FORMAT	DESCRIPTION
RESEQUENCE	RESEQ OR RE	RESEQ THE COMPLETE FILE BY 10.
	RE,AAAA,CCC	RESEQ FROM LINE AAAA BY AMT CCCC.

NOTE: AAAA & CCCC MUST BE 4 DIGIT NUMBERS, RE 0100,0002

FIND	FIND/EEEE/ALL OR A EEEE.	SEARCHES THE COMPLETE FILE FOR
	F/EEEE/100	SEARCHES LINE 100 FOR EEEE
	F/EEEE/10,50	SEARCHES FROM LINE 0010 TO 0050 FOR STRING EEEE.

NOTE: STRING EEEE CAN BE 40 CHARACTERS LONG.
NO: DELIMITERS ARE , OR *, OR /.
NOTE: THE STRING EEEE MUST BE ENCLOSED BY SLASHES, /.

CHANGE	CH/EEEE/FFFF/ALL	CHANGES ALL STRING EEEE TO FFFF.
	CH/EEEE/FFFF/100,200	CHANGES LINE 0100 TO 0200 EEEE TO FFFF.
	CH/G/H/10,V	CHANGES G TO H WITH DISPLAY FOR VERIFICATION.

NOTE: V OR VERIFICATION OF TEXT IS DONE BEFORE THE CHANGES ARE MADE.
NOTE: DELIMITERS ARE , OR *, OR /.
NOTE: STRING EEEE AND FFFF MUST BE ENCLOSED BY /.
NOTE: MAX LENGTH OF EEEE AND FFFF IS 35 CHARACTERS

THIS IS END OF HELP ROUTINE
TO CONTINUE EDITOR ROUTINE, TYPE IN NEW LINE.

Figure 8B: DISPLAY PAGE 2 OF HELP ROUTINE

CHAPTER 4

TEXT EDITCR IMPLEMENTATION

4-1 System Considerations

The software the Text Editor is implemented on is the NCR 8250 System using the NCR interactive COBOL language. This System runs under the NCR Interface Multiprogramming Operating System (IMOS), issue II. The IMCS includes the operating system, COBOL complier, run time interpreter, text editcr, and interactive user debug modules. The hardware of the NCR 8250 System consists of the fcllcwing:

1	6850 Processor
128K	Bytes MOS Memory
1	Cassette Drive
2	Hawk 656 Disc Drives
2	796-101 CRTS
1	349-300 Line Printer
1	368 Card Reader
2	Asynchronous Adapter (2400 baud)
1	Battery back-up,

Using the NCR System required special knowledge of the interactive COBOL language and its difference with standard ANSI CCBOL.

4-2 Text Editor Detailed Description

When the user decides to use the Text Editor, the Old Data Descriptien Source File is assigned to the editor's Old-Data-File. The submodule, 0010-CPENS, opens this input file and the output file New-Data-File. Next, the submodule 0020-READ-IN reads one record at a time from the Old-Data-

File into a 77 character holding variable called WS-IN-Hold. A counter (WS-COUNTER) that counts the total records in the file is increased by one. Then individual parts of the record in WS-IN-HOLD, line number and data, are transferred into the correspondencing position of the appropriate line in the WS-Table. This is done to duplicate the display and print out of the User Interface program. The WS-Table is set up as a doubly linked list by pointers WS-LAST and WS-NEXT. When the end of file is determined, control is passed to submodule 0100-CONTROL-SECTION. This submodule calls ancther submodule 0200-FREE-LIST that sets a pointer (WS-FREE-LIST) to the start of the free list. This module calls ancther mcdule which doubly links the free list by pointers WS-NEXT and WS-LAST until the 100 line is reached in the WS-Table. The free list is used with the addition and deletion of lines. Control is returned back tc 0100-CONTROL-SECTION. A pointer (WS-CURRENT-PTR) is set to the first line in the WS-Table. A question is asked, "FOR EDITOR ROUTINE DO YCU WANT TO USE THE PRINTER TYPE Y OR N."? If the answer is "Y" then the Cut-Print-file is opened and the statement "PRINTER OPTION CN" is displayed on the CRT. Next the submoule 1000-INPUT-TEXT is called which determines which routine is to be executed. When the submodule is entered a "ENTER COMMAND FOR EDIT(DISPLAY,DEL,ADD,RESEQ,FIND,CHANGE,HELP,QUIT)" is displayed. The answer is broken down into TEXT1, TEXT2, or TEXT3 by delimiters of blanks, commas, stars, and slashes.

Depending upon the contents in TEXT1, the appropriate routine is called. If one of the eight commands in TEXT1 is not in the proper form, then a statement "DO YOU NEED HELP" is displayed. If the answer is "Y", then the HELP function is called. If the answer is "N" then the initial statement for entering commands is displayed.

If TEXT1 equals "DISPLAY" OR "D" then the module 2000-DISPLAY is called. The first thing that happens in this module is that the screen is erased and TEXT2 contents are checked. If the word "ALL" or "A" appears, submodule 2100-ALL-DSPLY is called. This submodule displays the complete file in groups of 22 lines per page. This is accomplished by dividing the total lines (WS-END-INDEX) by 22 giving the total number of pages (WS-SEARCH-CNT) and the remainder that is left over (REM). Submodule 2150-ALL-DISPLY is performed SEARCH-CNT times. This submodule sets up a pointer (WS-START) at the start of the page that is to be displayed. The pointer (CNT) is set to WS-START. The actual displaying is done by other submodules 2300-WRITE and 2500-WRITE. Submodule 2300-WRITE is called 22 times. In this submodule the current line pointer (CNT) is passed to the next submodule, 2500-WRITE, for that line to be displayed. Control is returned to 2300-WRITE and the next line pointer WS-NEXT is moved to the current line to be displayed pointer (CNT). This continues until the complete file or 22 lines of the file is displayed. If more than 22 are lines in the file, then after each page is displayed a

statement "TO CONTINUE THE DISPLAY, TYPE IN NEW/LINE" is portayed on the screen. After all the pages are displayed, control is returned to 2100-ALL-DSPLY. This submodule calls 2300-WRITE to display the remainder of the lines of a page, REM times. When the lines are displayed, control is returned to 1000-INPUT-TEXT.

If TEXT2 equal "LAST" or "L" then the last record of the file is to be displayed. Submodule 2200-LAST-DSPLY is called which establishes the initial search pointer by moving WS-CURRENT-PTR to CNT. The submodule 10100-SEARCH is called. This submodule searches the pointer file WS-NEXT until zero value is found. This value indicates the end of the file with the pointer CNT pointing to the last record. Control is returned 2200-LAST-DSPLY which displays the statement "LAST RECORD OF THE FILE". Submodule 2500-WRITE is called to display the record.

If TEXT2 is numeric, which indicates that this is a possible line number of a record, the contents is checked by calling 10300-NUM-CHECK-TEST2. This submodule determines if the contents is a one, two, three, or four digit number. If it is not a four digit number, zeros are added to make it a complete four digit number which is required when the submodule 101000-SEARCH is called. This submodule determines if the line number is in the file. If not, a flag (ANS-FLAG) is set to two and control is returned to the 2000-DISPLAY which prints out an error message "INVALID DISPLAY OPTION". If the ANS-FLAG equals four, then the line

number has been found and the pointer CNT points to the line. If CNT equal zero, then the line number was not found and an error message "ERROR, SEQ NCT FOUND" is printed. If the line number is found then pointers, WS-START and WS-END, are set to CNT. If TEXT3 is blank then a single line is displayed by calling 2250-SMALL-DSPLY. If TEXT3 is numeric then submodule 10200-NUM-CHECK-TEXT3 is called. This submodule does the same thing as 10300-NUM-CHECK-TEXT2 and the same error messages applies. If the line number is found, then the pointer CNT is moved to pointer WS-END. Submodule 2250-SMALL-DISPLY is called. This submodule calls 2300-WRITE to display the lines until the pointer WS-NEXT equals WS-END. Before this submodule is called, TEST2 is checked to insure it is greater than TEXT3. If not an error message "ERROR, SEQ # CUT OF ORDER" is displayed. Once the DISPLAY routine is finished, control is returned to 1000-INPUT-TEXT.

If TEXT1 equals "DELETE" or "DEL" then the module 4000-DELETE is called. The screen is erased and TEXT2 contents are checked by calling 10300-NUM-CHECK-TEXT2 for the first line number to be deleted. This module returns a flag, ANS-FLAG, value two, if the line number is found. If the line number is not found the standard error message is returned and control is returned to 1000-INPUT-TEXT. If the value returned is two, then pointer WS-START is set to the pointer CNT that is pointing to the line number found. Next, the program must determine if the user wants more than

one line to be deleted. This is determined by checking TEXT3. If the contents are blanks, then only one line is to be deleted. The ending pointer is then set to the beginnig pointer WS-START. If TEXT3 is numeric and the line number is found, the pointer CNT is moved to the ending pointer WS-END. TEXT2 is checked to insure it is greater than TEXT3. If not the error message of "ERROR, SEQ # OUT OF ORDER" is displayed. If TEXT2 is greater than TEXT3 then submodule 4100-START-DELETING is called. The statement "THE FOLLOWING LINES ARE TO BE DELETED" is displayed. Submodule 2300-WRITE is called and the lines from WS-START to WS-END are displayed. Additionally, the pointer DEL-CNT counts the total lines that are to be deleted. After the lines are displayed, control is returned to 4100-START-DELETING. The statement to verify the correct deletion of the appropriate lines "IS THIS CORRECT, Y OR N" is displayed. If the user answers "N", then control is returned back to 1000-INPUT-TEXT and the statement to enter command for edit routine is displayed. If the answer is "Y" then the total number of lines that are to be deleted (DEL-CNT) are subtracted from the total line counter (WS-COUNTER). Next pointers are set to the line before the starting line (OLDL) and to the line after the ending line (OLDN). Then the WS-NEXT pointer, of the line before the starting line, is changed to point to the line after the ending line. Conversely, the pointer WS-LAST, of the line after the ending line, is set to point to the line before the starting line. This action eliminates

the lines that were to be deleted from the current file and reestablishes the doubly linked list. Next the submodule 4300-DEL-ADD-TO-FREE-LIST is called. This submodule adds the deleted lines to the free list. This is accomplished by moving the pointer that is pointing to the top of the free list to the deleted line pointer WS-NEXT. Then the starting line pointer (WS-START) is moved to the WS-FREE-LIST pointer which points to the start of the free list. Control is then returned back to 1000-INPUT-TEXT.

If TEXT1 equals "ADD" or "A" the module 3000-ADD is called. The screen is erased and TEXT2 is checked to see if it equals "AFTR". If not a statement "INVALID ADD OPTION, REENTER:" is displayed. If TEXT2 contents are correct then TEXT3 is checked to see if a valid line number is present as a starting point for the addition of lines. Submodule 10200-NUM-CHECK-TEXT3 is called. Detailed explanation is in the DISPLAY routine. This submodule basically checks out the text and determines if it is a valid line number. If not an error message is given. If it is a valid line number, a pointer WS-START is set to the pointer of the line number (CNT). The statement "ADD AFTER THIS LINE" and the complete line is displayed. Submodule 3100-ADD-LINES is called and the free list (WS-FREE-LIST) pointer is checked to see if any free lines are available. If not the statement "ERROR MAX LINES IN THE FILE" is displayed and control is returned to 1000-INPUT-TEXT. If lines are available, then data is accepted.

The maximum length of data allowed is 70 characters. The data is accepted in the working area called TEXTS. TEXTS is unstrung into TEXT2 so it can be checked out. If TEXT2 equals the backslash "\", then the ADD routine is terminated and control is returned to 1000-INPUT-TEXTS. If not a backslash, then the contents of TEXT2 are checked out by calling 10300-NUM-CHECK-TEXT2 submodule. This is done to test if the data entered is a line number that has been previously entered in the Data Description Source File. If not, the statement "INVALID LINE NUMBER, REENTER" is displayed. If the line number is valid, then the pointer CNT is set to the line and the line data, characters 7 to 76, is moved to the line pointer by the pointer WS-FREE-LIST. The next number of line in the FREE LIST is moved to the pointer WS-FREE-LIST. This eliminates the line that has been used and puts the next available line on top. The next submodule 3300-ADD-LN-CK is called and will be performed until a flag's (ANS-FIAG) value has been changed. In this submodule, the line before the added line is checked to see if it is the end of a report. If the end of a report is reached, or if the data entered equals "END", then the question of level is not asked.

The question "IS THIS LINE A SUBLIST TO THE PREVIOUS LINE" is displayed for all other lines entered. The reason for the question is that the information is used in modified User Interface Program in setting the data in a hierarchical form for the Functional Dependencies Generator [2]. After

the response is given, it is checked to insure that it is of the proper form. If not the statement "ANS MUST BE Y, C, S OR N" is displayed. This statement is displayed until the proper level is entered. Next, submodule 3400-ADD-IN-WRITE is called. This submodule moves the level response into column 77 of the added line and relinks the pointers WS-NEXT and WS-LAST to line nnnn and line nnnn + 1. The total line counter WS-END-INDEX is incremented by one. The added line is displayed when submodule 2500-WRITE is called. Control is returned to 3100-ADD-LINES. This round robin portion of the routine continues until the ADD routine is terminated by the backslash "\". When this happens control is returned 1000-INPUT-TEXT.

If TEXT1 equals "RESEQ" or "RE" then the module 5000-RESEQUENCE is called. The screen is erased and TEXT2 is checked by calling 10300-NUM-CHECK-TEXT2. This submodule determines if a valid line number has been entered. If not then the statement "ALL LINES RESEQ BY 10" is displayed. This resequence is accomplished by finding the starting line by moving WS-CURRENT-PTR to the pointer CNT. Submodule 5100-RESEQUENCE is called. This submodule does the actual resequencing by multiplying the line number times ten and moves this value in the line number pointed to by CNT. As each line is completed the next line in the doubly linked list is moved to CNT. When the pointer WS-NEXT equals zero, control is returned to 1000-INPUT-TEXT. If TEXT2 is numeric and a valid line number, then submodule 5200-RESEQ-SHORT is

called. In this submodule the line number (WS-IN-LINE) is pointed to by CNT and moved to the starting pointer (WS-START) for the starting location and value for resequencing. TEXT3 is checked to see if a valid resequence amount is requested by calling submodule 10200-NUM-CHECK-TEXT3. If not then the following statement "ALL LINES RESEQUENCED BY 10 FROM dddd" is displayed where dddd is a valid line number. If TEXT3 is not equal to blanks, then the statement "ERROR, RESEQ BY REQUESTED AMT IMPOSSIBLE". If a valid amount is requested, then submodule 5100-RESEQUENCE is called with starting location pointed to by TEXT3 and the resequenced amount by TEXT2. Each line number pointed to by CNT is then multiplied by amount TEXT2 and added to the starting level TEXT3. This value is moved into WS-IN-COUNT. The statement "ALL LINES RESEQUENCED FROM cccc BY dddd" is displayed. Control is returned back to 1000-INPUT-TEXT.

If TEXT1 equals "FIND" then module 7000-FIND is called and the screen erased. TEXTS is checked in order to determine the length of the string eeee. This is accomplished by finding the first slash "/". If the first slash is not found in the first five characters then an error message "SYNTAX ERROR, CHECK FORMAT, REENTER" is displayed. If the slash is found, one is added to the pointer I and moved to WS-INDEX which indicates the starting point of string eeee. The ending string location is determined by the second slash. If the slash is not found in the first 45 characters then the same error message as

above is displayed. If the slash is found the string ending pointer I is moved to NAME-LENGTH. A void string eeee is check by determining if pointer WS-INDEX and NAME-LENGTH are equal. If they are equal the statement "VOID STRING, REENTER" is displayed. If not equal, then TEXTS is unstrung into TEXT1, ANS, TEXT3 and TEXT2. This unstringing of TEXTS is to insure that the maxiumum string length will be detected. The roles of TEXT2 and TEXT3 are reversed then what they were for the first four routines. Submodule 7200-SEARCH-CHAR is called and TEXT3 is checked. If TEXT3 equals "ALL" or "A" then the complete file is checked for string eeee. Submodule 7300-CHAR-SEARCH is called. This submodule will be discussed later. If TEXT3 is not equal to "ALL", then submodule 10200-NUM-CHECK-TEXT3 is called. If TEXT3 is not a valid line number then statement "ERROR SEQ # NOT FOUND" is displayed. If TEXT3 is a valid line number than CNT, the pointer to the line, is moved to the pointer WS-START as the starting line to find string eeee. TEXT2 is checked by calling submodule 10300-NUM-CHECK-TEXT2. If TEXT2 is not a valid line number, then the same error message as above is displayed. If TEXT2 is blank, then a single line is checked for string eeee. If TEXT2 is a valid line number then TEXT3 is checked to insure it is less than TEXT2. If not then the error message, "ERROR SEQ # OUT OF ORDER" is displayed. If TEXT3 is less than TEXT2, then the ending pointer for the ending line of the file to be search is set by moving the pointer CNT to CNT-AMT. If the format

is wrong then the error message "INVALID FIND OPTION, SYNTAX" is displayed. After any error message, control is returned back to 1000-INPUT-TEXT. If no errors, then submodule 7300-CHAR-SEARCH is called. This submodule is called until end of file is reached for "ALL", for a single time when TEXT2 is blank and for multiple lines when the current line being exam equal the ending pointer CNT-AMT. This submodule sets up the beginning and ending pointers for the character search for string eeee for each line. WS-START is moved to pointer CNT which is multiplied by 77 giving an ending pointer WS-END. Then 7C is subtracted from WS-END giving the starting pointer 1. Then submodule 7400-SRCH-THE-LN is called. This submodule searches each character until the first character match is made between string eeee and the line that is being searched. If string eeee is only one character long, then when the first character is matched, the submodule 7600-DISPLAY-STRING is called. This submodule displays the complete line and sets beginning and ending pointers equal. Control is returned back to 7300-CHAR-SEARCH. If string eeee is more than one character, then when the first match occurs, submodule 7500-CHAR-MATCH is called. This submodule increments the pointers to the string and also to the next character being searched in the line. If the second character matches, then subsequent characters are checked until the end of string eeee is reached. If a match occurs, then submodule 7600-DISPLAY-STRING is called to display the line. Control

is returned back to 7300-CHAR-SEARCH. If the complete file or multiple lines are to be searched, then the next line to be searched is obtained from the pointer WS-NEXT. If more than 20 lines are displayed, a statement "TO CONTINUE, TYPE IN NEW LINE" is displayed. This is to prevent the rolling of the screen and eliminating the first group of lines displayed. When all the lines are searched, control is returned back to 7200-SEARCH CHAR. If string eeee was not found in the search, then the statement "ERROR STRING NOT FOUND" is displayed.

If TEXT1 equals "CHANGE" or "CH" then module 6000-CHANGE is called and the screen erased. This module searches for and modifies character strings up to a maximum length of 35 characters in the first occurrence in each line that is searched. The original TEXTS that was accepted is used to establish the length of string eeee and the modified string ffff. This is accomplished by keying off the slashes that are used to separate the strings. This first slash is searched for by the routine and if found the pointer I is moved to the pointer WS-INDEX. The other slashes are searched and the pointer NAME-LENGTH is set to the second slash with the pointer CH-LENGTH set to the third slash. If any of the slashes are not found, then the statement "ERROR, SEQ # NOT FOUND" is displayed and control is returned back to 1000-INPUT-TEXT. A void string eeee is checked by seeing if the pointers NAME-LENGTH and WS-INDEX are equal. If they are then the statement "ERROR VOID STRING, REENTER" is

displayed. If the format is correct then one is subtracted from the pointers NAME-LENGTH and CH-LENGTH to point to the end of strings eeee and ffff respectively. Also the pointer to the start of the first line in WS-Table is set by moving WS-CURRENT-PTR TO CNT. This is to initialize the search location. A statement "CHANGING ROUTINE" with TEXTS is displayed.

TEXTS is then unstrung into TEXT1, REM, OLDN, TEXT3, TEXT2 and ANS. This delimiting is required to put the search location in TEXT3, TEXT2 and using ANS to hold the verification request. Submodule 6200-CHANGE-STRING is called. This submodule sets up the search parameters and checks the numeric contents of TEXT3 and TEXT2. A flag (ANS-FLAG) is used for verification of modification to the string by setting it to one if "Y" has been requested. This flag will be checked in submodule 6300-CHANGE-LINE at a later time. TEXT3 is checked to see if its contents equal "ALL" or "A". If so then submodule 6300-CHANGE-LINE is called until the pointer CNT equals zero. If not, then TEXT3 is checked for numeric content by calling submodule 10200-NUM-CHECK-TEXT3. If a non-numeric content is found, then the statement "INVALID CHANGE OPTION" is displayed. If a valid numeric content is found the pointer CNT is checked to see if it equals zero. If so, then a valid line number was not found and the statement "ERROR, SEQ NOT FOUND" is displayed. If a valid line number is found then the pointer WS-START is set to CNT. Then TEXT2 is checked by calling

submodule 10300-NUM-CHECK-TEXT2. The same error messages applies as above. If TEXT2 contents are blanks then only one line is to be searched and modified. Submodule 6300-CHANGE-LINE is called. If TEXT2 is a valid line number then the value is checked to insure it is greater than TEXT3. If it is less than TEXT3, then the statement "ERROR, SEQ OUT OF ORDER" is displayed. If it is greater than TEXT3, then the ending pointer (CNT-AMT) for the search area is set.

Submodule 6300-CHANGE-LINE is called until the pointer CNT equals to the line pointed to by CNT-AMT. Submodule 6300-CHANGE-LINE sets the pointers for the current line to be searched as pointed to by WS-START. The ending pointer of the line, WS-END is set by multiplying the line pointer CNT by 77. The starting of the line is set up by subtracting 70 from WS-END giving WS-START. This is done to keep the search and modify routine for data items in column 7 to 77. Columns 1 thru 4 contain the line numbers. This area is already searched by the submodule 10000-SEARCH. Columns 5 and 6 are blank so it is necessary to search only columns 7 to 77. Since each character of the line is searched, pointer I will be set to WS-START. Submodule 6500-CH-THE-STRING sets up the search pointers by moving the pointer WS-INDEX, that points to the beginning of string eeee in the working area TEXTS, to the pointer DEL-CNT. The WS-Table is broken down into 101 lines and into 7777 characters. This allows the search to be by line and by

character. Additionally the submodule checks for first character match. If string eeee is more one character, than submodule 6600-STRING-MATCH is called until the string is matched completely or the flag NC-MATCH equals one.

Submodule 6600-STRING-MATCH checks for second and subsequent character matches between the string eeee and the current line under investigation. This accomplished by moving the pointers I and DEL-CNT. If the string character under investigation does not match the line character, then 1 is moved to NC-MATCH and control is returned back to 6500-STRING-CHANGE. This submodule sets the pointer J to the beginning of string ffff and resets the pointer I at the beginning of the string found in the line. The length of string ffff is determine by subtracting the beginning pointer CH-LENGTH from J giving the amount REM. Then submodule 6800-MAKE-CHANGE is called to move the line and the modified string ffff to line 101 of the WS-TAble. Pointer OLDL and OLDN is set to point to the start of the data in the line 101. REM is added to the amount OLDN to see if the data will exceed the 70 character limit. Submodule 6850-MOVE-STRING is called. This submodule moves the characters of the line up to the beginning of string eeee as pointed to by pointer I. Control is returned to 6800-MAKE-CHANGE which immediately calls another submodule 6900-MODIFY-STRING. This submodule moves the modified string ffff from TEXTS to line 101 and returns control back to 6800-MAKE-CHANGE. OLDN is checked to see if it exceeds

the 7C character limit. If so then the statement "ERROR, LINE EAS EEEN TRUNCATED, REENTER" is displayed. Pointers I and WS-END are then reset in the line that is being modified to the end of string eeee. At this time the length of string eeee and ffff are checked tc see if they are equal. If they are not, then the status level in column 77 is moved to line 101. This is accomplished because if the strings lengths were different, the status level might be transferred to the wrong column. Next the submodule 6950-MOVE-REST-STNG is called. This submodule moves the rest of the line starting after string eeee until the end of the line is reached. Control is returned to 6800-MAKE-CHANGE which determines if the end of the line 101 is reached. If not then submodule 6975-FINISH-MOVE is called. This submodule moves blanks into line 101 until the end of that line is reached. Control is returned to 6800-MAKE-CHANGE which returns control back to 6300-CHANGE-LINE. If "V" or verification was requested then submodule 6400-VERIFY-CHANGE is called which displays the old line and line 101. The statement "TO FINALIZE CHANGE, TYPE IN Y, TO CANCEL, N" is displayed. If the answer is "N" then no changes are made. If the answer is "Y" then the line 101 is moved to the line pointed to by CNT. If "V" was not requested then the move is made without any display. When this line is finished, the CNT pointer is changed to pointing to the next line that is to be searched. Control is returned to the submodule 6200-CHANGE-STRING when the

searched area is completed. If no strings FFFF were found then the statement "ERROR STRING NOT FOUND" is displayed. Control is then returned back to 1000-INPUT-TEXT.

When TEXTS equals "HELP" then module 8000-HELP is called. This module is a tutorial display for the use of the TEXT Editor. Two pages of commands are displayed. Page one displays the DISPLAY, DELETE, and FIND commands. When the user has finished viewing this page he must type in "NEW/LINE" for the second page. The second page displays the RESEQUENCE, FIND and CHANGE commands. When the user has finished viewing this page, he must type in "NEW/LINE". This returns control back to 1000-INPUT-TEXT. See Chapter 3 for detailed display of the HELP command.

When TEXTS equals "QUIT" or "QU" the editor routine is terminated. Control is returned back to the main module 0100-CONTROL-SECTION. Once back in this module the complete modified file is resequenced from zero by ten. The pointer to the first line of the file, WS-CURRENT-PTR, is moved to CNT in preparation of the output. Module 9000-OUTPUT is called with the statement "DO YOU WANT THE EDIT FILE SAVED? TYPE IN YES OR NO. IF NO THE OLD DATA DESCRIPTION SOURCE FILE WILL BE RETURNED." If the answer is "YES" then submodule 9100-OUTPUT which immediately calls submodule 9200-OUTPUT which immediately calls submodule 9200-OUTPUT. This submodule prints the record from the WS-TABLE that is pointed to by the pointer CNT. Then the status level is moved from position 77 which it is printed out on the

printer to position 5 which is needed for the input to the special version of the User Interface Program[2]. Next the line pointed to by CNT is moved to the New Record location in New Data File which is the input to the User program. Control is returned back to 9100-OUTPUT which increments the pointer to the next line and moves it to CNT. This continues until end of file is found by CNT equaling zero. If the Edit file is not saved, "NO" is typed in, then submodule 9300-OUTPUT-OLD FILE thru 9500-FINISH-OUTPUT is called. This submodule closes the Old Data File and immediately reopens it. Submodule 9400-READ-WRITE-OLD-FILE reads in the Old Data Description Source File and outputs to the New Data File which is the input to the Functional Dependency Generator[2]. When the complete file has been written, control is returned back to 0100-CONTROL-SECTION. The program finishes running and depending upon the job control language, control is returned to the User Interface Program or the Functional Dependency Generator program.

CHAPTER 5

TESTING AND RESULTS

5-1 Testing Philosphy

The testing philosophy was an approach that was consistent with the Top Down Structuring technique[3]. This is a natural extension of this philosophy called top down testing. At the end of the high level coding, a test walk through was conducted. Several logic and design errors were found in the modules. These errors were corrected and the walk through was conducted again. At that time, the extensive test program for individual modules was finalized. The basic guidelines for the test program were as follows:

Check format of Commands

Check syntax of Commands

Check minimum and maximum limit conditions

Check human engineering of error and messages code

Run error checks for all the above guidelines

In addition to the guidelines, individual students attending KSU were requested to use the editor program. Any misunderstanding on the use of the program by way of instructions or any malfunctions or errors that occurred were immediately reviewed. If the problems required changing, then the changes were put through the same stringent testing procedures. This testing philosophy resulted in additional commands being added and the total restructuring of some commands to make it easier for the user to understand. Additional dividends were reaped from the use of this philosophy which allowed each module to be built and tested independently. Despite a number of stupid

mistakes, the overall program was written and debugged in short order.

5-2 Sample Operations

The following are sample operations of the interactive display in the use of the Text Editor. The first will be the initial set up after the editor routine has been requested. Sentences with all capital letters indicate the response from the editor program, while sentences that are underlined indicate the user response.

```
DO YOU WANT TO USE THE LINE PRINTER FOR EDIT ROUTINE
TYPE IN Y(ES) OR N(O)
Y
PRINTER OPTION ON
ENTER COMMAND FOR EDIT(DISPLAY,DEL,ADD,RESEQ,FIND,CHANGE,HELP,QUIT)
```

The following display will be seen when the editor is terminated.

```
ENTER COMMAND FOR EDIT(DISPLAY,DEL,ADD,RESEQ,FIND,CHANGE,HELP,QUIT)
QUIT
DO YOU WANT THE EDIT FILE SAVED? TYPE IN YES OR NO.
IF NO THE OLD DATA DESCRIPTION SOURCE FILE WILL BE RETURNED.
```

The following will be from the DISPLAY Routine.

```
ENTER COMMAND FOR EDIT(DISPLAY,DEL,ADD,RESEQ,FIND,CHANGE,HELP,QUIT)
DISPLAY ALL OR D A
(screen erased)

0000    REPORT ORGANIZATION
0010    DEPT   DBUDGET   MANAGER                    Y
0020    EMP    PROJ      OFFICE    PHONE           Y
0030    PROJ   PBUDGET                             S
0040    OFFICE   AREA                              S
0050    END
```

```
ENTER COMMAND FOR EDIT(DISPLAY,DEL,ADD,RESEQ,FIND,CHANGE,HELP,QUIT)
D L OR DISPLAY LAST
(screen erased)
LAST RECORD OF FILE
0050 END

ENTER COMMAND FOR EDIT(DISPLAY,DEL,ADD,RESEQ,FIND,CHANGE,HELP,QUIT)
Display Function continued:
DISPLAY 10,20
(screen erased)
0010   DEPT   PBUDGET   MANAGER                          Y
0020   EMP    PROJ      OFFICE     PHONE                 Y
ENTER COMMAND FOR EDIT(DISPLAY,DEL,ADD,RESEQ,FIND,CHANGE,HELP,QUIT)
The following are some of the error messages in the use of the
DISPLAY Function.

D 100
(screen erased)
ERROR SEQ  NOT FOUND:  100
ENTER COMMAND FOR EDIT(DISPLAY,DEL,ADD,RESEQ,FIND,CHANGE,HELP,QUIT)
D 20,10
(screen erased)
ERROR SEQ   OUT OF ORDER:  D 20,10
ENTER COMMAND FOR EDIT(DISPLAY,DEL,ADD,RESEQ,FIND,CHANGE,HELP,QUIT)
D D 30
(screen erased)
SYNTAX ERROR, CHECK FORMAT, REENTER:  D D 30
ENTER COMMAND FOR EDIT(DISPLAY,DEL,ADD,RESEQ,FIND,CHANGE,HELP,QUIT)
DISPLAY FIRST
```

```
(screen erased)

INVALID DISPLAY OPTION:  DISPLAY FIRST

ENTER COMMAND FOR EDIT(DISPLAY,DEL,ADD,RESEQ,FIND,CHANGE,HELP,QUIT)

D 0,200

(screen erased)

ERROR SEQ   NOT FOUND:  200
```

The following are some of the interactive display for the ADD Function:

```
ENTER COMMAND FOR EDIT(DISPLAY,DEL,ADD,RESEQ,FIND,CHANGE,HELP,QUIT)

ADD AFTR 30

(screen erased)

ADD AFTER THIS LINE
0030    PROJ    PBUDGET                                  S

CUSTOMER
IS THE LINE A SUBLIST TO THE PREVIOUS LINE
L
ANS MUST BE Y, C, S, OR N
N
        CUSTOMER                                         N
20
IS THIS LINE A SUBLIST TO THE PREVICUS LINE
Y
        EMP    PROJ    OFFICE    PHONE                   Y
100
INVALID LINE NUMBER, REENTER
\
ENTER COMMAND FOR EDIT(DISPLAY,DEL,ADD,RESEQ,FIND,CHANGE,HELP,QUIT)

D A

(screen erased)

0000    REPORT ORGANIZATION                              Y
0010    DEPT    PBUDGET    MANAGER                       Y
0020    EMP    PROJ    OFFICD    PHONE                   S
0030    PROJ    PBUDGET                                  N
        CUSTOMER                                         Y
        EMP    PROJ    OFFICE    PHONE                   S
0040    OFFICE    AREA                                   S
0050    END
```

```
ENTER COMMAND FOR EDIT(DISPLAY,DEL,ADD,RESEQ,FIND,CHANGE,HELP,QUIT)
A AFTR 100
(screen erased)
ERROR, SEQ   NOT FOUND:  100
ENTER COMMAND FOR EDIT(DISPLAY,DEL,ADD,RESEQ,FIND,CHANGE,HELP,QUIT)
```

The following are some of the interactive displays for the DELETE Function:

```
ENTER COMMAND FOR EDIT(DISPLAY,DEL,ADD,RESEQ,FIND,CHANGE,HELP,QUIT)
DELETE 10,20
(screen erased)
THE FOLLOWING LINES ARE TO BE DELETED
0010    DEPT    PBUDGET    MANAGER                          Y
0020    EMP    PROJ    OFFICE    PHONE                      Y

IS THIS CORRECT, Y OR N
Y
ENTER COMMAND FOR EDIT(DISPLAY,DEL,ADD,RESEQ,FIND,CHANGE,HELP,QUIT)
DEL C
(screen erased)
THE FOLLOWING LINES ARE TO BE DELETED
0000   REPORT ORGANIZATION

IS THIS CORRECT, Y OR N
N
ENTER COMMAND FOR EDIT(DISPLAY,DEL,ADD,RESEQ,FIND,CHANGE,HELP,QUIT)
D A
(screen erased)
0000    REPORT ORGANIZATION
0030    PROJ    PBUDGET                                     S
        CUSTOMER                                            N
        EMP    PROJ    OFFICE    PHONE                      Y
0040    OFFICE    AREA                                      S
0050   END
ENTER COMMAND FOR EDIT(DISPLAY,DEL,ADD,RESEQ,FIND,CHANGE,HELP,QUIT)
```

The following are some error messages in the use of DELETE Function:

```
DEL 40,0

(screen erased)

ERROR SEQ   OUT OF ORDER:  DEL 40,0

ENTER COMMAND FOR EDIT(DISPLAY,DEL,ADD,RESEQ,FIND,CHANGE,HELP,QUIT)

DEL 20

(screen erased)

ERROR SEQ   NOT FOUND:  20
```

The following are some of the interactive display for the RESEQUENCE Function:

```
ENTER COMMAND FOR EDIT(DISPLAY,DEL,ADD,RESEQ,FIND,CHANGE,HELP,QUIT)

RESEQ or RE

(screen erased)

ALL LINES RESEQ BY 10

ENTER COMMAND FOR EDIT(DISPLAY,DEL,ADD,RESEQ,FIND,CHANGE,HELP,QUIT)

D A

(screen erased)

0000   REPORT ORGANIZATION
0010   PROJ  PEUDGET                                    S
0020   CUSTOMER                                         N
0030   EMP   PROJ   OFFICE   PHONE                      Y
0040   OFFICE   AREA                                    S
0050 END

ENTER COMMAND FOR EDIT(DISPLAY,DEL,ADD,RESEQ,FIND,CHANGE,HELP,QUIT)

RE 10,1

(screen erased)

ALL LINES RESEQUENCED FROM 10 BY 1

ENTER COMMAND FOR EDIT(DISPLAY,DEL,ADD,RESEQ,FIND,CHANGE,HELP,QUIT)

D A
```

```
(screen erased)

0000   REPORT CRGANIZATICN
0010   PROJ   PBUDGET                                    S
0011   CUSTOMER                                          N
0012   EMP   PROJ   OFFICE   PHONE                       Y
0013   OFFICE   AREA                                     S
0014  END

ENTER CCMMAND FOR EDIT(DISPLAY,DEL,ADD,RESEQ,FIND,CHANGE,HELP,QUIT)

RE 11

(screen erased)

ALL LINES RESEQUENCE BY 10 FROM 11

ENTER COMMAND FOR EDIT(DISPLAY,DEL,ADD,RESEQ,FIND,CHANGE,HELP,QUIT)

D A

(screen erased)

0000   REPORT ORGANIZATION
0010   PRCJ   PBUDGET                                    S
0011   CUSTOMER                                          N
0021   EMP   PROJ   OFFICE   PHONE                       Y
0031   OFFICE   AREA                                     S
0041  END

ENTER COMMAND FOR EDIT(DISPLAY,DEL,ADD,RESEQ,FIND,CHANGE,HELP,QUIT)

RE 50

(screen erased)

ERROR, SEQ   NCT FOUND:  50

ENTER COMMAND FOR EDIT(DISPLAY,DEL,ADD,RESEQ,FIND,CHANGE,HELP,QUIT)

RE 0, -5

(screen erased)

ERROR RESEQ BE REQUESTED AMT NOT POSSIBLE:  -5
ALL LINES RESEQ BY 10 FRCM 0

ENTER COMMAND FOR EDIT(DISPLAY,DEL,ADD,RESEQ,FIND,CHANGE,HELP,QUIT)

D A

(screen erased)

0000   REPORT ORGANIZATION
```

```
0010    PROJ    PBUDGET                                         S
0020    CUSTOMER                                                N
0030    EMP    PROJ    OFFICE    PHONE                          Y
0040    OFFICE    AREA                                          S
0050  END

ENTER COMMAND FOR EDIT(DISPLAY,DEL,ADD,RESEQ,FIND,CHANGE,HELP,QUIT)

RESEP ALL

(screen erased)

DO YOU NEED HELP.  TYPE IN Y(ES) OR N(O).

N

ENTER COMMAND FOR EDIT(DISPLAY,DEL,ADD,RESEQ,FIND,CHANGE,HELP,QUIT)
```

The following are some of the interactive displays for the FIND Function:

```
ENTER COMMAND FOR EDIT(DISPLAY,DEL,ADD,RESEQ,FIND,CHANGE,HELP,QUIT)

FIND/OFF/ALL

(screen erased)

SEARCHING FOR /OFF/ALL
0030    EMP    PROJ    OFFICE    PHONE                          Y
0040    OFFICE    AREA                                          S

ENTER COMMAND FOR EDIT(DISPLAY,DEL,ADD,RESEQ,FIND,CHANGE,HELP,QUIT)

F/OFF/0,10

(screen erased)

SEARCHING FOR /OFF/0,10
ERROR, STRING NOT FOUND

ENTER COMMAND FOR EDIT(DISPLAY,DEL,ADD,RESEQ,FIND,CHANGE,HELP,QUIT)

F/OFF/100,200

(screen erased)

SEARCHING FOR /OFF/100,200
ERROR, SEQ   NOT FOUND:  100

ENTER COMMAND FOR EDIT(DISPLAY,DEL,ADD,RESEQ,FIND,CHANGE,HELP,QUIT)
```

```
FIND/CFF/20,0

(screen erased)

SEARCHING FOR /OFF/20,0
ERROR,SEQ   OUT OF ORDER:
ENTER COMMANDS FOR EDIT(DISPLAY,DEL,ADD,RESEQ,FIND,CHANGE,HELP,QUIT)

FIND/ALL

(screen erased)

SEARCHING FOR //ALL
ERROR, VOID STRING

ENTER COMMAND FOR EDIT(DISPLAY,DEL,ADD,RESEQ,FIND,CHANGE,HELP,QUIT)
```

The following are some of the interactive displays for the CHANGE Function:

```
ENTER COMMAND FOR EDIT(DISPLAY,DEL,ADD,RESEQ,FIND,CHANGE,HELP,QUIT)

CHANGE/OFF/XXX/ALL,V

(screen erased)

CHANGING ROUTINE  CHANGE/CFF/XXX/ALL,V
0030    EMP    PROJ    OFFICE    PHONE                    Y
0030    EMP    PROJ    XXXICE    PHONE                    Y

TO FINALIZE CHANGE, TYPE IN Y. TO CANCEL, N.
Y
0040    OFFICE    AREA                                     S
0040    XXXICE    AREA                                     S

TO FINALIZE CHANGE, TYPE IN Y.  TO CANCEL, N.
N

ENTER COMMAND FOR EDIT(DISPLAY,DEL,ADD,RESEQ,FIND,CHANGE,HELP,QUIT)

D A

(screen erased)

0000    REPORT ORGANIZATTICN
0010    PROJ    PBUDGET                                  S
0020    CUSTOMER                                         N
0030    EMP    PROJ    XXXICE    PHONE                   Y
0040    OFFICE    AREA                                   S
```

```
0050    END

ENTER COMMAND FOR EDIT(DISPLAY,DEL,ADD,RESEQ,FIND,CHANGE,HELP,QUIT)

CH/PPPP/10,V

(screen erased)

CHANGING ROUTINE  CH/PPPP/10,V
ERROR, STRING NOT FOUND

ENTER COMMAND FOR EDIT(DISPLAY,DEL,ADD,RESEQ,FIND,CHANGE,HELP,QUIT)

CH//XXX/0,10

(screen erased)

CHANGING ROUTINE CH//XXX/0,10
ERROR VOID STRING

ENTER COMMAND FOR EDIT(DISPLAY,DEL,ADD,RESEQ,FIND,CHANGE,HELP,QUIT)
```

CHAPTER 6

CONCLUSION

6-1 Summary

The interactive Text Editor for the Third Normal Form (3NF) Synthesis System was engineered tc make the editor a powerful software tool. The overall design was to make each function module completely independent so that each module could be built, debugged and tested without changing the overall program. The global variables in the working storage section were designed so that they could be used in each module thereby eliminating additional storage requirements. Total storage requirement is 18092 bytes with 962 tctal lines. Approximately 180 hours of time was used in top down designing and implementaticn.

The Text Editor was implemented on the NCR 8250 System using the NCR interactive COBOL language. Approximately sixty hours of machine time was used to implement the editor.

6-2 Enhancements

The Text Editor can be enhanced by adding additional routines of Merge, Move, Concat and Sort. The Merge routine would merge section of files together. The Move routine would move lines or files from one positicn to another. The Concat routine would concatenate strings together. The Sort routine would sort the file in acccrdance to some specific key. All these enhancements could be added to the editor, but would require additional storage and manpower requirements which would far exceed the actual productivity in usage for the 3NF Synthesis System.

The editor could be modified with very little change to make it into a universal COBOL Text Editor. The main change would be to increase the size of the storage table in lines and characters per line. This change would be in direct proportion to which machine and what role the editor would be used. Additionally the NON-ANSI COBOL would need to be converted to ANSI COBOL. This basic change would only pertain to the CRT displays and would depend upon the system on which the editor is to be implemented.

In optimizing the Text Editor one could consolidate the FIND and CHANGE routine thereby eliminating approximately 100 lines of code and 1800 bytes of storage requirement. Also one could add to the Display routine the code to find the first record in the file. This would require only two lines of code. Similar error and syntax checking routines could be combined thereby eliminating additional lines of code and storage requirements.

Overall, any enhancements or optimization of the editor should be planned in detail thereby eliminating any damage to other routines in the Text Editor. The above listed enhancements and optimizing suggestions could round out the editor and make it an exceedingly powerful software tool.

Appendix A

DATA STRUCTURES AND HIGH LEVEL ALGORITHM

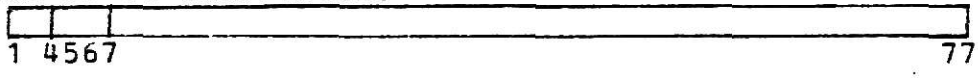
TEXT EDITOR

The Text Editor is an interactive program which accepts as input a file generated by the user interface program and provides six basic edit functions. Upon initial input, a copy of the input records are saved as USRINTER to avoid having to re-enter all data for subsequent processing. The Editor provides the basic following functions:

DISPLAY ON CRT
ADD LINES
DELETE LINES
RESEQUENCE LINES
FIND STRINGS (up to 40 characters long)
CHANGE STRINGS (up to 35 characters long)
HELP (Give format instructions of the above functions on the CRT).

DATA STRUCTURES

The input and output file is a table of records with a maximum of 76 characters.



Characters 1-4: Line Number
5: Status
6: Level Number
7-76: Data-name-items (separated by one or more blanks).

Note 1: Output in column 6 is blank. Level numbers must be reassigned by USER Interface Program.
Note 2: Display on CRT will show STATUS in column 77.
Note 3: Output can be the original file or the edited file that was changed. If file is original, level number will be transmitted in column 6.
Note 4: Output produced for the printer is identical to output displayed on the CRT with STATUS in column 77.
Note 5: Commas cannot be used as delimiters in the data columns 7-76.

ALGORITHM

The Text Editor is of modular construction and the modules are independent of each other. One module can be completely deleted without harming the other modules or changing their code. The global variables in the Working-Storage areas are all used in most every module. Standard declarations for the COBOL Program are assumed.

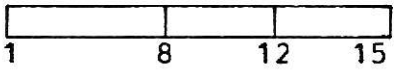
```
     Main Module:  Text Editor

BEGIN:
     DO until all records are read
     Read a record into WS-Table (Max. 100 records)
     Double link records to LAST and NEXT POINTERS.
     END
     Set up double link free list until records=100.
     Ask question of TYPE of EDIT
     Loop
     CASE STATEMENT
          :TEXTS = DISPLAY:  CALL DISPLAY function
          :TEXTS = ADD:  CALL ADD function
          :TEXTS = DELETE:  CALL DELETE function
          :TEXTS = RESEQ:  CALL RESEQUENCE
                                function
          :TEXTS = CHANGE:  CALL CHANGE function
          :TEXTS = FIND:  CALL FIND function
          :TEXTS = HELP:  CALL HELP function
          :TEXTS = QUIT:  Terminate Editor mode
          :ELSE:  ERROR
     END CASE
     IF ANS = SAVE EDIT FILE then
          PRINT OUTPRINT FILE,
          OUTPUT EDIT FILE,
          Else OUTPUT ORIGINAL FILE.
END TEXT Editor
```

DISPLAY FUNCTION

The Display function will display on the CRT all or part of the USER Interface file.

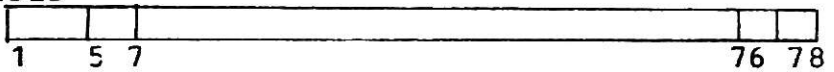
DATA STRUCTURES

TEXTS



Columns 1-7: TEXT1 - The word DISPLAY or D
8-11: TEXT2 - The word ALL, A, LAST, L or 4-digit numeric
12-15: TEXT3 - 4-digit numeric or blanks.

WS-TABLE



		Data inputed by
Columns 1-4:	Line number	Data Description
7-76:	Data	Source file or
77:	Status	DATA EDITOR

WS-LAST and WS-NEXT (max. pointers 100)

Columns 1-4: WS-LAST - pointer to record before
WS-NEXT - pointer to the record next in line

ALGORITHM DISPLAY

```
If TEXT1 - "DISPLAY" or "D" then check TEXT2 and
           TEXT3
     CASE
          :TEXT2 = ALL: DISPLAY complete file
          :TEXT2 = NUMERIC:  DISPLAY one line
          :TEXT2 and TEXT3 = NUMERIC:  DISPLAY
                    lines from TEXT2 and TEXT3
     END CASE
END IF
END DISPLAY
```

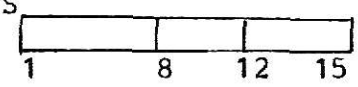
Note: Numeric is assumed to be a valid line number.

ADD FUNCTION

The ADD Function will add one or more lines to the existing Data Description Source file.

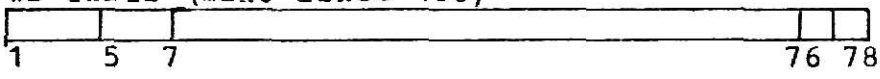
DATA STRUCTURES

TEXTS



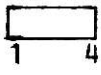
Columns 1-7: TEXT1 - The word "ADD"
8-11: TEXT2 - The word "AFTR"
12-15: TEXT3 - 4-digit numeric

WS-TABLE (max. lines 100)



COLUMNS 1-4: BLANK - This will become a line number after resequencing.
7-76: Data Entered
77: If beginning of a report or end report blank else question of sublist is asked.

WS-LAST and WS-NEXT



Columns 1-4: WS-LAST - pointer to record before
WS-NEXT - pointer to record next in line

ALGORITHM ADD

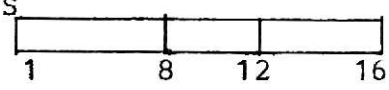
```
If TEXT1 = "ADD" or "A" then check TEXT2 and TEXT3.
     CASE
          :TEXT2 = AFTR:  Check TEXT3, ELSE ERROR
          :TEXT3 = NUMERIC:
               Get a line from free list and set pointers
               Accept texts and enter it in Columns 7-76
               Check if beginning or ending of report
                 If not then ask sublist question
               Accept texts until = " " then exit
          :ELSE:  ERROR
     END CASE
END IF
END ADD
```

RESEQUENCE FUNCTION

The Resequence function will resequence the Data Description Source file line numbers from 0 to 9999. When the Editor program is terminated, the line numbers are automatically resequenced by 10 from zero.

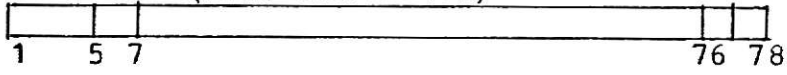
DATA STRUCTURES

TEXTS



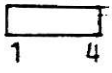
COLUMNS 1-7: TEXT1 - The word "RESEQ" or "RE"
8-12: TEXT2 - Blanks or 4-digit numeric
12-15: TEXT3 - Blanks or 4-digit numeric

WS-TABLE (max. lines 100)



Coluns 1-4: Line number
7-76: Data
77: Status

WS-LAST and WS-NEXT



Columns 1-3: WS-LAST - pointer to record before
WS-NEXT - pointer to next record in line

ALGORITHM RESEQUENCE

```
If TEXT1 = "RESEQ" or "RE"
     CASE
          :TEXT2 = blanks: resequence by 10
          :TEXT2 = numeric: resequence by 10 from TEXT2
          :TEXT2 and TEXT3 = numeric: resequence by TEXT3
                             from TEXT2
          :ELSE:  ERROR
     END CASE
END IF
END RESEQUENCE
```

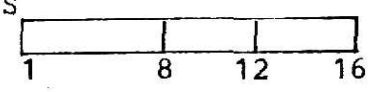
Note: TEXT2 must be a valid line number

DELETE FUNCTION

The Delete function will delete one or more lines from the existing Data Description Source file.

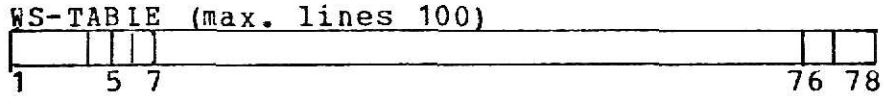
DATA STRUCTURES

TEXTS



Columns 1-7: TEXT1 - The word "DELETE" or "DEL"
8-11: TEXT2 - 4-digit numeric
12-15: TEXT3 - 4-digit numeric or blanks.

WS-TABLE (max. lines 100)



Columns 1-4: Line number 4-digit line number
7-76: Data
77: Status

WS-LAST and WS-NEXT (max. 100 pointers each)

Columns 1-3: WS-LAST - pointer to record before
WS-NEXT - pointer to next record in line

ALGORITHM DELETE

```
If TEXT1 = "DELETE" or "DEL" check TEXT2 and TEXT3
     CASE
          :TEXT2 = numeric: delete one line
          :TEXT2 and TEXT3 = numeric:
               Delete from line (TEXT2) to line (TEXT3)
               Reset pointers of the active file
               ADD deleted lines to free list
          :ELSE:  ERROR
     END CASE
END IF
END DELETE
```

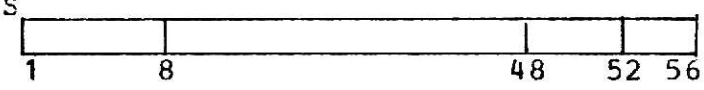
Note: Numeric must be a valid line number.

FIND FUNCTION

The Find function searches the data description source file for a string EEEE up to 40 characters long.

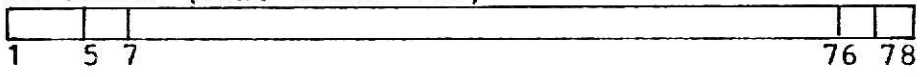
DATA STRUCTURES

TEXTS



Columns 1-7: TEXT1 - Word "FIND" or "F"
8-47: ANS - Data String EEEE
48-51: TEXT3 - Word "ALL" or 4-digit numeric
52-55: TEXT2 - Blank, or 4-digit numeric

WS-TABLE (max. lines 100)



Columns 1-4: Line number
7-76: Data
77: Status

WS-NEXT and WS-LAST (max. 100 pointers each)
1 4
Columns 1-4: WS-NEXT - pointer to next record in line
WS-LAST - pointer to record before

ALGORITHM FIND

```
If TEXT = "FIND" or "F" check TEXT3 and TEXT2
     CASE
          :TEXT3 = "ALL" or "A": Do search until end
                   of file
          :TEXT3 = numeric: Do search of line (TEXT3)
                   only
          :TEXT2 and TEXT3 numeric:  Do search from
                   TEXT3 to TEXT2
     END CASE
END IF
END FIND
```

Submodule

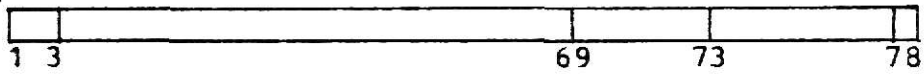
```
Search
     Start with appropriate line and set up
        start and ending pointers
     Search the line until first character
        matches or reached end pointer.
          Search the line until subsequent
             characters match
          Display line on CRT
          Else display string not found
END Search
```

CHANGE FUNCTION

The Change function searches the Data Description Source file for a string EEEE and changes it to string FFFF. The function will change strings up tc 35 characters long.

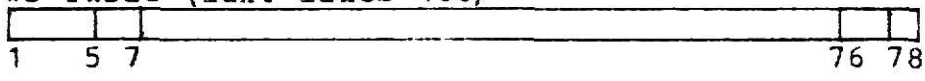
DATA STRUCTURES

TEXTS



Columns 1-2: Word "CH"
3-68: String EEEE and FFFF
69-72: TEXT3 - 4-digit numeric or "ALL" or "A"
73-76: TEXT2 - 4-digit numeric, "V" or blanks.
77: ANS - word "V" or blank

WS-TABLE (max. lines 100)



Columns 1-4: Line number
Columns 1-4: Line number
7-76: Data
77: Status

WS-LAST and WS-NEXT

Columns 1-3: WS-LAST - pointer to record before
WS-NEXT - pcinter to next record

ALGORITHM CHANGE

If TEXTS = "CHANGE" or "CH" check TEXT3, TEXT and ANS.
CASE
:TEXT3 = "ALL" or "A": Do search and change until end of file
:TEXT3 = numeric: do search and change of line number (TEXT3 only)
:TEXT2 and TEXT3 = numeric: Do search and change from line number (TEXT3) to

```
                    line number (TEXT2)
               :TEXT2 or ANS = "V":  Display
                    changes for verification
               :ELSE:  ERROR
          END CASE
     END IF
END CHANGE
```

Submodule

```
     Search and Change
          Start with appropriate line and set up start
             and ending pointers
          Search the line until first character matches
             or reaching end pointer.
               Search the line until subsequent character
                  matches
               Set pointer for start and end or string
                  EEEE
               Move characters inline up to start pointer
                  EEE to WS-TABLE(101)
               Move string FFFF in place of string EEEE
                  into WS-TABLE(101)
               Move rest of string until reach line
                  pointer
               Check for line truncation
               Display on CRT the modified and original
                  line if "V" was requested
                    ELSE continue search.
     END Search and Change
```

APPENDIX B

DETAILED CODE

```
000000 IDENTIFICATION DIVISION.
000010 PROGRAM-ID. TEXT-EDITOR.
000020*
000030*  THIS IS A TEXT EDITOR THAT ACCEPTS INPUT FROM THE USER
000040* INTERFACE PROGRAM AND MAKES DATA ERROR CORRECTIONS. THE
000050* OUTPUT IS A CORRECTED VERSION OF THE USER INTERFACE
000060* PROGRAM OR THE ORIGINAL PROGRAM ITSELF.
000070*
000080 ENVIRONMENT DIVISION.
000090 CONFIGURATION SECTION.
000100 SOURCE-COMPUTER. NCR-CENTURY-8200.
000110 OBJECT-COMPUTER. NCR-CENTURY-8200.
000120 INPUT-OUTPUT SECTION.
000130 FILE-CONTROL.
000140     SELECT OLD-DATA-FILE ASSIGN TO DISC.
000150     SELECT OUT-PRINT-FILE ASSIGN TO PRINTER.
000160     SELECT NEW-DATA-FILE ASSIGN TO DISC.
000170 DATA DIVISION.
000180 FILE SECTION.
000190 FD  OLD-DATA-FILE BLOCK CONTAINS 6 RECORDS.
000200         RECORD CONTAINS 76 CHARACTERS.
000210         LABEL RECORDS ARE STANDARD.
000220 01  IN-REC.
000230     02  IN-INDEX                    PIC XXXX.
000240     02  IN-STAT                     PIC X.
000250     02  IN-LOL                      PIC X.
000260     02  IN-TEXT                     PIC X(70).
000270 FD  OUT-PRINT-FILE
000280         LABEL RECORDS OMITTED.
000290 01  OUT-PRINT                       PIC X(132).
000300 FD  NEW-DATA-FILE
000310         BLOCK CONTAINS 6 RECORDS.
000320         RECORD CONTAINS 76 CHARACTERS
000330         LABEL RECORDS ARE STANDARD.
000340 01  NEW-DATA-REC.
000350     02  INDEX-OUT                   PIC XXXX.
000360     02  FILLER                      PIC X(72).
```

```
*       /
*000370 WORKING-STORAGE SECTION.
*000380 01  WS-IN-HOLD.
*000390     02  WS-IND                          PIC XXXX.
*000400     02  WS-ST                           PIC X.
*000410     02  WS-LVEL                         PIC X.
*000420     02  WS-TXT                          PIC X(70).
*000440 01  WS-INPUT-TABLE.
*000450     02  TEXT-TABLE.
*000460         03 WS-TABLE OCCURS 101 TIMES.
*000470             05  WS-IN-COUNT.
*000480                 06  WS-IN-LINE PIC  X(4).
*000490                 06  WS-STATS   PIC  X(2).
*000500             05  WS-TEXT        PIC X(70).
*000510             05  WS-STAT        PIC X.
*000520     02  CHAR-TABLE REDEFINES TEXT-TABLE.
*000530         03  CHAR-INDEX OCCURS 7777 TIMES.
*000540             05  CHAR-ENTRY     PIC X.
*000550 01  INDEX-POINTER.
*000560     02  PNT-IND OCCURS 100 TIMES.
*000570         03  WS-NEXT              PIC  999.
*000580         03  WS-LAST              PIC  999.
*000590 01  FD-TEXT.
*000600     02  TEXTS.
*000610         03  NAME-INDEX OCCURS 77 TIMES.
*000620             05  NAME-ENTRY  PIC X.
*000630
*000640 01  WS-TEXT3.
*000650     02  TEXT3              PIC X(4).
*000660     02  TEXT3-3NUM REDEFINES TEXT3.
*000670         03  TXT3           PIC X(3).
*000680         03  FILLER         PIC X.
*000690     02  TEXT3-2NUM REDEFINES TEXT3.
*000700         03  TXT2           PIC XX.
*000710         03  FILLER         PIC XX.
```

```
000720     02  TEXT3-1NUM REDEFINES TEXT3.
000730         03  TXT1          PIC X.
000740         03  FILLER        PIC XXX.
000750/
000760 01  WS-TEXT2.
000770     02  TEXT2             PIC X(4).
000780     02  TEXT2-3NUM REDEFINES TEXT2.
000790         03  TX3           PIC XXX.
000800         03  FILLER        PIC X.
000810     02  TEXT2-2NUM REDEFINES TEXT2.
000820         03  TX2           PIC XX.
000830         03  FILLER        PIC XX.
000840     02  TEXT2-1NUM REDEFINES TEXT2.
000850         03  TX1           PIC X.
000860         03  FILLER        PIC XXX.
000870 77  WS-END-INDEX          PIC 9999  VALUE 0.
000880 77  WS-INDEX              PIC 9999  VALUE 0.
000890 77  WS-SCREEN-CNT         PIC 999   VALUE 0.
000900 77  WS-COUNTER            PIC 999   VALUE 0.
000910 77  TEXT1                 PIC X(7).
000920 77  WS-START              PIC 9999.
000930 77  ANS                   PIC X     VALUE SPACE.
000940 77  TXTN                  PIC 9999.
000950 77  WS-END                PIC 9999.
000960 77  I                     PIC 9999.
000970 77  REM                   PIC 99.
000980 77  CNT                   PIC 999.
000990 77  ADD-COUNT             PIC 9.
001000 77  CNT-AMT               PIC 999.
001010 77  DEL-CNT               PIC 99    VALUE 0.
001020 77  OLDN                  PIC 9999.
001030 77  OLDL                  PIC 9999.
001040 77  ANS-FLAG              PIC 9.
001050 77  WS-FREE-LIST          PIC 999.
```

```
107*  001060 77  WS-CURRENT-PTR          PIC 999.
108*  001070 77  OUT-PRINT-FLAG          PIC 9       VALUE 0.
109*  001080 77  NO-MATCH                PIC 9.
110*  001090 77  NAME-LENGTH             PIC 99.
111*  001100 77  SEARCH-CNT              PIC 999.
112*  001110 77  CH-LENGTH               PIC 999.
113*  001120 77  J                       PIC 999.
114*  001130 77  ERROR-FLAG              PIC 9.
115*  001140 77  SPA                     PIC X(4)   VALUE SPACES.
116*  001150/
117*  001160 77  ERROR1                  PIC X(36)
118*  001170          VALUE "SYNTAX ERROR, CHECK FORMAT, REENTER:".
119*  001180 77  ERROR2                  PIC X(23)
120*  001190          VALUE "ERROR, SEQ # NOT FOUND:".
121*  001200 77  ERROR3                  PIC X(26)
122*  001210          VALUE "ERROR, SEQ # OUT OF ORDER:".
123*  001220
```

```
001230/
001240 PROCEDURE DIVISION.
001250
001260* THIS SECTIONS OPENS THE FILES.
001270
001280 0010-OPENS.
001290      OPEN INPUT OLD-DATA-FILE
001300      OUTPUT NEw-DATA-FILE.
001310*
001320
001330* THIS SECTIONS INITIALIZES AND READS IN USERS FILE.
001340
001350 0020-READ-IN.
001360      READ OLD-DATA-FILE INTO WS-IN-HOLD
001370          END GO TO 0100-CONTROL-SECTION.
001380      ADD 1 TO WS-COUNTER.
001390      MOVE WS-ST TO WS-STAT (WS-COUNTER).
001400      MOVE WS-TXT TO WS-TEXT (WS-COUNTER).
001410      MOVE WS-IND TO WS-IN-COUNT (WS-COUNTER).
001420      IF WS-COUNTER GREATER THAN 1,
001430          SUBTRACT 1 FROM WS-COUNTER GIVING WS-LAST(WS-COUNTER),
001440          SUBTRACT 1 FROM WS-COUNTER GIVING I,
001450          MOVE WS-COUNTER TO WS-NEXT(I),
001460          MOVE 0 TO WS-NEXT(WS-COUNTER),
001470          ELSE
001480          MOVE 0 TO WS-NEXT(WS-COUNTER),
001490          MOVE 0 TO WS-LAST(WS-COUNTER).
001500      GO TO 0020-READ-IN.
001510*
```

```
001520*
001530* THIS SECTION CONTROLS THE LINKING OF THE FREE LIST AND THE
001540* RUNNING OF THE SEVEN ROUTINES.
001550*
001560 0100-CONTROL-SECTION.
001570     PERFORM 0200-FREE-LIST.
001580     MOVE 1 TO WS-CURRENT-PTR.
001590*
001600     MOVE WS-COUNTER TO WS-END-INDEX.
001610     DISPLAY " ", ERASE.
001620     DISPLAY " FOR EDITOR ROUTINE DO YOU WANT TO USE THE PRINTER T
001630-    "YPE Y OR N.".
001640     ACCEPT ANS.
001650     IF ANS = "Y" OPEN OUTPUT OUT-PRINT-FILE.
001660         DISPLAY "PRINTER OPTION ON".
001670         MOVE 1 TO OUT-PRINT-FLAG.
001680*
001690     PERFORM 1000-INPUT-TEXT UNTIL TEXTS = "QUIT" OR TEXTS = "QU".
001700     MOVE WS-CURRENT-PTR TO CNT.
001710     PERFORM 5100-RESEQUENCE VARYING TXTN FROM 0 BY 10
001720         UNTIL CNT EQUAL 0.
001730     MOVE WS-CURRENT-PTR TO CNT.
001740     PERFORM 9000-OUTPUT.
001750     IF OUT-PRINT-FLAG = 1,
001760         CLOSE OLD-DATA-FILE, OUT-PRINT-FILE,
001770         ELSE CLOSE OLD-DATA-FILE.
001780     STOP RUN.
001790*
```

```
001800* THIS SECTION LINKS THE FREE LIST OF EMPTY LINES.
001810
001820 0200-FREE-LIST.
001830     MOVE WS-COUNTER TO I.
001840     ADD 1 TO I.
001850     MOVE I TO WS-FREE-LIST. WS-START.
001860     MOVE 0 TO WS-LAST (I), WS-NEXT (I).
001870     PERFORM 0300-LINK-FREE-LIST UNTIL I IS EQUAL TO 100.
001880 0300-LINK-FREE-LIST.
001890     MOVE I TO OLDN.
001900     ADD 1 TO I.
001910     MOVE WS-START TO WS-LAST (I).
001920     MOVE 0 TO WS-NEXT(I).
001930     MOVE I TO  WS-NEXT(OLDN).
001940     ADD 1 TO WS-START.
001950
001960
```

```
198*   001970/
199*   001980* THIS SECTION DECIDES WHICH ROUTINE IS RAN.
200*   001990 1000-INPUT-TEXT.
201*   002000     DISPLAY "ENTER COMMAND FOR EDIT (DISPLAY, DEL, ADD, RESEQ, FI
202*   002010-    "ND, CHANGE, HELP, QUIT).".
203*   002020     MOVE SPACES TO TEXTS, WS-TEXT2, WS-TEXT3.
204*   002030     MOVE 0 TO TXTN, ANS-FLAG.
205*   002040     ACCEPT TEXTS.
206*   002050     UNSTRING TEXTS DELIMITED BY " " OR "," OR "*" OR "/"
207*   002060         INTO TEXT1, TEXT2, TEXT3.
208*   002070     IF TEXT1 = "HELP" PERFORM 8000-HELP,
209*   002080     ELSE IF TEXT1 = "DISPLAY" OR TEXT1 = "D"
210*   002090             PERFORM 2000-DISPLAY,
211*   002100             ELSE IF TEXT1 = "ADD" OR TEXT1 = "A"
212*   002110                 PERFORM 3000-ADD,
213*   002120                 ELSE IF TEXT1 = "DELETE" OR TEXT1 = "DEL"
214*   002130                     PERFORM 4000-DELETE,
215*   002140                     ELSE IF TEXT1 = "RESEQ" OR TEXT1 = "RE"
216*   002150                         PERFORM 5000-RESEQUENCE,
217*   002160                         ELSE IF TEXT1 = "CHANGE" OR TEXT1 = "CH"
218*   002170                             PERFORM 6000-CHANGE,
219*   002180                             ELSE IF TEXT1 = "FIND" OR TEXT1 = "F"
220*   002190                                 PERFORM 7000-FIND
221*   002200                                 ELSE IF TEXT1 = "QUIT" OR
222*   002210                                     TEXT1 = "QU" NEXT SENTENCE,
223*   002220                                 ELSE DISPLAY "DO YOU NEED HELP, T
224*   002230-    "YPE IN (Y)ES OR (N)O.",
225*   002240                                     ACCEPT ANS,
226*   002250                                     IF ANS = "Y",
227*   002260                                         PERFORM 8000-HELP.
228*   002270
```

```
229*  002280/
230*  002290*************************DISPLAY FUNCTION*****************************
231*  002300*THE DISPLAY ROUTINE TO LIST THE PGM.
232*  002310 2000-DISPLAY.
233*  002320      DISPLAY " ", ERASE.
234*  002330      IF TEXT2 = "ALL" OR TEXT2 = "A",
235*  002340          PERFORM 2100-ALL-DSPLY,
236*  002350          ELSE IF TEXT2 = "LAST" OR TEXT2 = "L",
237*  002360              PERFORM 2200-LAST-DSPLY,
238*  002370              ELSE PERFORM 10300-NUM-CHECK-TEXT2,
239*  002380              IF ANS-FLAG NOT EQUAL 4,
240*  002390                  IF CNT = 0,
241*  002400                      DISPLAY ERROR2, TEXT2, LINE 2, POSITION 30,
242*  002410                      ELSE MOVE CNT TO WS-START,
243*  002420                      MOVE 0 TO TXTN, ANS-FLAG,
244*  002430                      PERFORM 10200-NUM-CHECK-TEXT3,
245*  002440                      IF ANS-FLAG = 3,
246*  002450                          MOVE WS-START TO WS-END,
247*  002460                          PERFORM 2250-SMALL-DSPLY,
248*  002470                          ELSE IF CNT = 0
249*  002480                              DISPLAY ERROR2, TEXT3, LINE 2
250*  002490                                  POSITION 30,
251*  002500                              ELSE IF TEXT3 LESS THAN TEXT2,
252*  002510                              DISPLAY ERROR3, TEXT5, LINE 2
253*  002520                                  POSITION 30,
254*  002530                              ELSE MOVE CNT TO WS-END,
255*  002540                              MOVE WS-START TO CNT,
256*  002550                              PERFORM 2250-SMALL-DSPLY,
257*  002560                  ELSE DISPLAY "INVALID DISPLAY OPTION:",
258*  002570                      DISPLAY TEXT5, LINE 2, POSITION 25.
259*  002580* THIS SECTION WILL DISPLAY THE COMPLETE FILE.
260*  00
261*  002600 2100-ALL-DSPLY.
262*  002610      MOVE WS-COUNTER TO WS-END-INDEX.
263*  002620      MOVE WS-CURRENT-PTR TO WS-START.
264*  002630      DIVIDE WS-END-INDEX BY 22 GIVING SEARCH-CNT
265*  002640          REMAINDER REM.
```

```
266*  002650      PERFORM 2150-ALL-DISPLY SEARCH-CNT TIMES.
267*  002660      MOVE WS-START TO CNT.
268*  002670      PERFORM 2300-WRITE REM TIMES.
269*  002680/
270*  002
271*  002700 2150-ALL-DISPLY.
272*  002710      MOVE WS-START TO CNT.
273*  002720      PERFORM 2300-WRITE 22 TIMES.
274*  002730      MOVE CNT TO WS-START.
275*  002740      DISPLAY "TO CONTINUE THE DISPLAY, TYPE IN NEW LINE.".
276*  002750          ACCEPT ANS.
277*  002760
278*  002770*  THIS SECTION DISPLAYS THE LAST RECORD OF THE FILE.
279*  002780
280*  002790 2200-LAST-DSPLY.
281*  002800      MOVE WS-CURRENT-PTR TO CNT.
282*  002810      PERFORM 10100-SEARCH UNTIL WS-NEXT(CNT) = 0.
283*  002820      DISPLAY "LAST RECORD OF THE FILE:".
284*  002830      PERFORM 2500-WRITE.
285*  002840*  THIS SECTION WILL DISPLAY ONLY THOSE LINES REQUESTED.
286*  002850 2250-SMALL-DSPLY.
287*  002860      MOVE WS-START TO CNT.
288*  002870      PERFORM 2300-WRITE UNTIL CNT = WS-END.
289*  002880      PERFORM 2300-WRITE.
290*  002890
291*  002900*  THIS SECTION WILL WRITE OUT THE LINES ON THE CONSOLE.
292*  002910 2300-WRITE.
293*  002920      PERFORM 2500-WRITE.
294*  002930      ADD 1 TO DEL-CNT.
295*  002940      MOVE WS-NEXT(CNT) TO CNT.
296*  002950 2500-WRITE.
297*  002960      DISPLAY WS-TABLE(CNT).
298*  002970*
```

```
002980/
002990*************************ADD FUNCTION*********************************
003000*  THIS IS THE ROUTINE FOR ADDING LINES.
003010 3000-ADD.
003020     DISPLAY " ", ERASE.
003030     IF TEXT2 IS NOT EQUAL TO "AFTR",
003040         DISPLAY "INVALID ADD OPTION, REENTER:", TEXT5,
003050             LINE 2, POSITION 32,
003060         ELSE PERFORM 10200-NUM-CHECK-TEXT3,
003070         IF ANS-FLAG = 3,
003080             DISPLAY ERROR2, TEXT3, LINE 2, POSITION 27,
003090                 ELSE IF CNT = 0,
003100                         DISPLAY ERROR2,
003110                         DISPLAY TEXT3, LINE 2, POSITION 30,
003120                         ELSE MOVE CNT TO WS-START,
003130                         DISPLAY "ADD AFTER THIS LINE",
003140                         PERFORM  2500-WRITE,
003150                         MOVE 0 TO TXTN,
003160                             PERFORM 3100-ADD-LINES THRU
003170                                 3200-ADD-LINES-EXIT
003180                                 UNTIL TEXT2 = "\".
003190     MOVE WS-END-INDEX TO WS-COUNTER.
003200
003210*  THIS SECTION IS THE INNER LOOP  FOR ADDING LINES.
003220 3100-ADD-LINES.
003230     IF WS-FREE-LIST = 0,
003240         MOVE "\" TO TEXT2,
003250         DISPLAY "ERROR MAX LINES IN FILE  ",
003260         GO TO 3200-ADD-LINES-EXIT.
003270     ACCEPT TEXT5.
003280     UNSTRING TEXT5 DELIMITED BY " " INTO TEXT2.
003290     IF TEXT2 IS EQUAL TO "\" GO TO  3200-ADD-LINES-EXIT.
003300     MOVE 0 TO TXTN, ANS-FLAG.
003310     PERFORM 10300-NUM-CHECK-TEXT2.
```

```
333*  003320      IF ANS-FLAG NOT EQUAL 4,
334*  003330          IF CNT = 0,
335*  003340              DISPLAY "INVALID LINE NUMBER, REENTER:",
336*  003350              GO TO 3200-ADD-LINES-EXIT.
337*  003360              ELSE MOVE WS-TEXT(CNT) TO WS-TEXT(WS-FREE-LIST),
338*  003370          ELSE MOVE TEXTS TO WS-TEXT (WS-FREE-LIST).
339*  003380/
340*  003390      MOVE WS-FREE-LIST TO WS-COUNTER.
341*  003400      MOVE WS-NEXT(WS-FREE-LIST) TO WS-FREE-LIST.
342*  003410      IF WS-FREE-LIST IS NOT EQUAL 0,
343*  003420          MOVE 0 TO WS-LAST(WS-FREE-LIST).
344*  003430      MOVE 1 TO ANS-FLAG.
345*  003440      PERFORM 3300-ADD-LN-CK UNTIL ANS-FLAG NOT EQUAL TO 1.
346*  003450
347*  003460*  THIS IS USED ONLY FOR GETTING OUT OF THE ADD ROUTINE FOR
348*  003470*  PROGRAMMERS.
349*  003480 3200-ADD-LINES-EXIT.
350*  003490      EXIT.
351*  003500
352*  003510*  THIS SECTION IS USED TO ANS QUESTION ON LINE LEVEL.
353*  003520 3300-ADD-LN-CK.
354*  003530      IF TEXT2 IS NUMERIC AND WS-TEXT(CNT) = "END",
355*  003540          PERFORM 3400-ADD-LN-WRITE,
356*  003550          ELSE IF WS-TEXT(WS-START) = "END" OR TEXTS = "END",
357*  003560           MOVE SPACES TO ANS,
358*  003570           PERFORM 3400-ADD-LN-WRITE,
359*  003580           ELSE DISPLAY "IS THIS LINE A SUBLIST TO THE PREVIOUS LIN
360*  003590-     "E.",
361*  003600                  ACCEPT ANS,
362*  003610                  IF ANS = "Y" OR ANS = "N" OR ANS = "S" OR ANS = "C",
363*  003620                      PERFORM 3400-ADD-LN-WRITE,
364*  003630                          ELSE DISPLAY "ANS MUST BE Y,C,S OR N".
365*  003640
```

```
366*  003650*  THIS SECTION DISPLAY ON THE CONSOLE THE ADDED LINE.
367*  003660 3400-ADD-LN-WRITE.
368*  003670      MOVE 2 TO ANS-FLAG.
369*  003680      MOVE ANS TO WS-STAT(WS-COUNTER).
370*  003690      MOVE WS-NEXT(WS-START) TO WS-NEXT(WS-COUNTER).
371*  003700      MOVE WS-START TO WS-LAST(WS-COUNTER).
372*  003710      ADD 1 TO WS-END-INDEX.
373*  003720      MOVE WS-COUNTER TO WS-NEXT(WS-START), CNT, WS-START.
374*  003730      MOVE SPACES TO WS-IN-COUNT (WS-COUNTER).
375*  003740      PERFORM 2500-WRITE.
376*  003750
```

```
003760/
003770*****************************DELETE FUNCTION*******************************
003780*  THIS IS THE ROUTINE TO DELETE LINES.
003790 4000-DELETE.
003800     DISPLAY " ", ERASE.
003810     MOVE 0 TO WS-END.
003820     PERFORM 10300-NUM-CHECK-TEXT2.
003830     IF ANS-FLAG = 4,
003840         DISPLAY ERROR2, TEXT2, LINE 2, POSITION 30,
003850         ELSE IF CNT = 0,
003860             DISPLAY ERROR2, TEXT2, LINE 2, POSITION 27,
003870     ELSE MOVE CNT TO WS-START,
003880     MOVE 0 TO TXTN, ANS-FLAG,
003890     PERFORM 10200-NUM-CHECK-TEXT3,
003900     IF ANS-FLAG = 3,
003910         MOVE WS-START TO WS-END
003920         PERFORM 4100-START-DELETING,
003930         ELSE IF TEXT3 IS LESS THAN TEXT2,
003940             DISPLAY ERROR3, TEXT3, LINE 2, POSITION 30,
003950             MOVE 0 TO CNT
003960             ELSE IF CNT = 0,
003970                 DISPLAY ERROR2, TEXT3, LINE 2, POSITION 30,
003980                 ELSE MOVE CNT TO WS-END,
003990                     PERFORM 4100-START-DELETING.
004000
004010 4100-START-DELETING.
004020     DISPLAY "THE FOLLOWING LINES ARE TO BE DELETED".
004030     MOVE WS-START TO CNT.
004040     MOVE 0 TO DEL-CNT.
004050     PERFORM 2300-WRITE UNTIL CNT = WS-END.
004060     PERFORM 2300-WRITE.
004070     DISPLAY "IS THIS CORRECT, Y OR N".
004080     ACCEPT ANS.
004090     IF ANS IS NOT EQUAL "Y" GO TO 4200-DELETE-EXIT.
```

```
411*  004100/
412*  004110* IF ANS IS YES, WE DELETE THE LINES BY RESETTING THE DOUBLE LINK
413*  004120*LIST POINTERS IN THE WS-NEXT AND WS-LAST FILE.
414*  004130      SUBTRACT DEL-CNT FROM WS-END-INDEX.
415*  004140      MOVE WS-END-INDEX TO WS-COUNTER.
416*  004150      MOVE WS-NEXT(WS-END) TO OLDN.
417*  004160      MOVE WS-LAST(WS-START) TO OLDL.
418*  004170      IF OLDL IS NOT EQUAL 0 MOVE OLDN TO WS-NEXT(OLDL).
419*  004180           ELSE MOVE OLDN TO WS-CURRENT-PTR.
420*  004190      IF OLDN IS NOT EQUAL 0 MOVE OLDL TO WS-LAST(OLDN).
421*  004200
422*  004210      PERFORM 4300-DEL-ADD-TO-FREE-LIST.
423*  00422
424*  004230*  THIS SECTION IS USED TO GET OUT OF THE ROUTINE FOR PROGRAMMERS
425*  004240*  ONLY.
426*  004250 4200-DELETE-EXIT.
427*  004260      EXIT.
428*  004270
429*  004280*  THIS SECTION ADDS THE DELETED LINES TO TOP OF THE FREE LIST.
430*  004290 4300-DEL-ADD-TO-FREE-LIST.
431*  004300      MOVE WS-FREE-LIST TO WS-SCREEN-CNT.
432*  004310      MOVE WS-START TO WS-FREE-LIST.
433*  004320      MOVE WS-SCREEN-CNT TO WS-NEXT(WS-END).
434*  004330      IF WS-SCREEN-CNT IS NOT EQUAL TO 0,
435*  004340          MOVE WS-END TO WS-LAST(WS-SCREEN-CNT).
436*  004350      MOVE 0 TO WS-LAST(WS-START).
437*  004360
438*  004370
```

```
439*  004380/
440*  004390*****************************RESEQUENCE FUNCTION*************************
441*  004400*  THIS ROUTINE RESEQUENCE THE FILE FROM 0 TO 9999.
442*  004410 5000-RESEQUENCE.
443*  004420      DISPLAY " ", ERASE.
444*  004430      MOVE 0 TO ANS-FLAG.
445*  004440      PERFORM 10300-NUM-CHECK-TEXT2.
446*  004450      IF ANS-FLAG = 4,
447*  004460          DISPLAY " ALL LINES RESEQ BY 10",
448*  004470          MOVE WS-CURRENT-PTR TO CNT.
449*  004480          PERFORM 5100-RESEQUENCE VARYING TXTN FROM 0 BY 10
450*  004490              UNTIL CNT = 0,
451*  004500          ELSE IF CNT = 0,
452*  004510              DISPLAY ERROR2, TEXT2, LINE 2, POSITION 30,
453*  004520                      ELSE PERFORM 5200-RESEQ-SHORT.
454*  004530
455*  004540*  THIS SECTION WILL RESEQ FROM A STARTING POINT IN THE PGM TO
456*  004550*  THE END.
457*  004560 5200-RESEQ-SHORT.
458*  004570      MOVE WS-IN-LINE(CNT) TO WS-START.
459*  004580      MOVE 0 TO TXTN, ANS-FLAG.
460*  004590      MOVE CNT TO CNT-AMT.
461*  004600      PERFORM 10200-NUM-CHECK-TEXT3.
462*  004610      MOVE CNT-AMT TO CNT.
463*  004620      IF ANS-FLAG = 3
464*  004630          DISPLAY "ALL LINES RESEQUENCED BY 10 FROM ", LINE 2,
465*  004640          DISPLAY WS-START, LINE 2, POSITION 34,
466*  004650          PERFORM 5100-RESEQUENCE VARYING TXTN FROM
467*  004660              WS-START BY 10 UNTIL CNT = 0,
468*  004670          IF TEXT3 NOT EQUAL " ",
```

```
469*  004680          DISPLAY "ERROR. RESEQ BY REQUESTED AMT IMPOSSIBLE:",
470*  004690          DISPLAY TEXT3, LINE 3, POSITION 47,
471*  004700          ELSE MOVE CNT-AMT TO CNT,
472*  004710          ELSE MOVE TEXT3 TO I,
473*  004720              MOVE TEXT2 TO TXTN,
474*  004730              PERFORM 5100-RESEQUENCE VARYING TXTN
475*  004740                  FROM WS-START BY I UNTIL CNT = 0,
476*  004750              DISPLAY "ALL LINES RESEQUENCED FROM", WS-START,
477*  004760              LINE 2, POSITION 27,
478*  004770              DISPLAY " BY", LINE 2, POSITION 32,
479*  004780              DISPLAY I, LINE 2, POSITION 36.
480*  004790
481*  004800*  THIS SECTION DOES THE ACTUAL RESEQ OF THE NUMBER OF THE LINES.
482*  004810 5100-RESEQUENCE.
483*  004820      MOVE TXTN TO WS-IN-COUNT (CNT).
484*  004830      MOVE WS-NEXT (CNT) TO CNT.
```

```
004840*/
004850*************************CHANGE FUNCTION*****************************
004860*   THIS ROUTINE FINDS AND CHANGES STRING UP TO 35 CHAR LONG.
004870 6000-CHANGE.
004880     DISPLAY " ", ERASE.
004890     MOVE SPA TO TEXT2, TEXT3.
004900     MOVE 1 TO I.
004910     PERFORM 6100-CH-LGTH UNTIL NAME-ENTRY(I) = "/"
004920         OR I = 8.
004930     IF NAME-ENTRY(I) NOT EQUAL "/",
004940         DISPLAY ERROR1, TEXT5, LINE 2, POSITION 37,
004950         ELSE ADD 1 TO I,
004960             MOVE I TO WS-INDEX,
004970             PERFORM 6100-CH-LGTH UNTIL NAME-ENTRY(I) = "/"
004980                 OR I = 42,
004990             IF NAME-ENTRY(I) NOT EQUAL "/"
005000                 DISPLAY ERROR1, TEXT5, LINE 2, POSITION 37,
005010                 ELSE MOVE I TO NAME-LENGTH,
005020                 ADD 1 TO I,
005030                 IF NAME-LENGTH = WS-INDEX,
005040                     DISPLAY "ERROR, VOID STRING.", TEXT5,
005050                         LINE 2, POSITION 22,
005060                     ELSE PERFORM 6100-CH-LGTH UNTIL
005070                         NAME-ENTRY(I) = "/" OR I = 77,
005080                     IF NAME-ENTRY(I) NOT EQUAL "/"
005090                         DISPLAY ERROR1, TEXT5, LINE 2,
005100                             POSITION 37,
005110                         ELSE MOVE I TO CH-LENGTH,
005120                         SUBTRACT 1 FROM CH-LENGTH,
005130                         SUBTRACT 1 FROM NAME-LENGTH,
005140                         MOVE WS-CURRENT-PTR TO CNT,
005150                         DISPLAY "CHANGING ROUTINE", TEXT5,
005160                             LINE 2, POSITION 19,
005170                         UNSTRING TEXT5 DELIMITED BY "/" OR ","
005180                                 OR "*",
005190                          INTO TEXT1, REM, OLDN, TEXT3, TEXT2, ANS
005200                         PERFORM 6200-CHANGE-STRING.
```

```
522*  005210/
523*  005220*  THIS IS USED TO FIND NAME AND CHANGE STRING LENGTHS.
524*  005230 6100-CH-LGTH.
525*  005240      ADD 1 TO I.
526*  005250
527*  005260*  THIS SECTION DECIDES IF ALL THE FILE OR CERTAIN LINES ARE
528*  005270*     CHANGED.
529*  005280
530*  005290 6200-CHANGE-STRING.
531*  005300      MOVE 0 TO ADD-COUNT.
532*  005310      MOVE 0 TO ERROR-FLAG.
533*  005320      IF TEXT2 = "V" OR ANS = "V",
534*  005330          MOVE 1 TO ADD-COUNT.
535*  005340
536*  005350      IF TEXT3 = "ALL" OR TEXT3 = "A",
537*  005360          PERFORM 6300-CHANGE-LINE UNTIL CNT = 0,
538*  005370          ELSE PERFORM 10200-NUM-CHECK-TEXT3,
539*  005380          IF ANS-FLAG NOT EQUAL 3,
540*  005390              IF CNT = 0,
541*  005400                  DISPLAY ERROR2, LINE 3,
542*  005410                  DISPLAY TEXT3, LINE 3, POSITION 30,
543*  005420                  ELSE MOVE CNT TO WS-START,
544*  005430                  MOVE 0 TO TXTN, ANS-FLAG,
545*  005440                  PERFORM 10300-NUM-CHECK-TEXT2,
546*  005450                  IF ANS-FLAG = 4,
547*  005460                      PERFORM 6300-CHANGE-LINE,
548*  005470                      ELSE IF CNT = 0,
549*  005480                          DISPLAY ERROR2, LINE 3,
550*  005490                          DISPLAY TEXT2, LINE 3, POSITION 30,
551*  005500                          ELSE IF TEXT2 IS LESS THAN TEXT3,
552*  005510                              DISPLAY ERROR3, LINE 3,
553*  005520                              ELSE MOVE CNT TO CNT-AMT,
554*  005530                              MOVE WS-START TO CNT,
555*  005540                              PERFORM 6300-CHANGE-LINE UNTIL
556*  005550                                  CNT = WS-NEXT(CNT-AMT),
557*  005560                   ELSE DISPLAY "INVALID CHANGE OPTION, SYNTAX",
558*  005570                        LINE 3.
```

```
559*  005580      IF ERROR-FLAG NOT EQUAL 2,
560*  005590          DISPLAY "ERROR STRING NOT FOUND".
561*  005600/
562*  005610*  THIS SECTION SEARCHES A LINE AND CHANGES THE STRING IF FOUND.
563*  005620
564*  005630 6300-CHANGE-LINE.
565*  005640      MOVE 0 TO ANS-FLAG.
566*  005650      MULTIPLY CNT BY 77 GIVING WS-END.
567*  005660      SUBTRACT 70 FROM WS-END GIVING WS-START.
568*  005670      MOVE WS-START TO I.
569*  005680      PERFORM 6500-CH-THE-STRING UNTIL I IS GREATER THAN WS-END.
570*  005690      IF ADD-COUNT = 1 AND ANS-FLAG = 2,
571*  005700          PERFORM 6400-VERIFY-CHANGE,
572*  005710          ELSE IF ANS-FLAG = 2,
573*  005720              MOVE WS-TABLE(101) TO WS-TABLE(CNT).
574*  005730
575*  005740      MOVE WS-NEXT(CNT) TO CNT.
576*  005750
577*  005760*  THIS SECTION DISPLAYS ON THE CRT THE OLD AND MODIFIED LINE
578*  005770*AND MOVES THE MODIFIED LINE IN THE WS-TABLE IF CORRECT.
579*  005780
580*  005790 6400-VERIFY-CHANGE.
581*  005800      DISPLAY WS-TABLE(CNT).
582*  005810      DISPLAY WS-TABLE(101).
583*  005820      DISPLAY "TO FINALIZE CHANGE, TYPE IN Y. TO CANCEL, N".
584*  005830      ACCEPT ANS.
585*  005840      IF ANS = "Y",
586*  005850          MOVE WS-TABLE(101) TO WS-TABLE(CNT).
587*  005860
```

```
588*  005870*  THIS SECTION TRYS TO FIND THE STRING TO CHANGE IT.
589*  005880
590*  005890 6500-CH-THE-STRING.
591*  005900      MOVE 0 TO NO-MATCH.
592*  005910      MOVE WS-INDEX TO DEL-CNT.
593*  005920          IF CHAR-ENTRY(I) = NAME-ENTRY(DEL-CNT)
594*  005930                  AND NAME-LENGTH = WS-INDEX,
595*  005940              PERFORM 6700-STRING-CHANGE,
596*  005950                  ELSE IF CHAR-ENTRY(I) = NAME-ENTRY(DEL-CNT),
597*  005960                      PERFORM 6600-STRING-MATCH UNTIL
598*  005970                          DEL-CNT = NAME-LENGTH OR NO-MATCH = 1,
599*  005980                          ELSE ADD 1 TO I.
600*  005990/
601*  006000*  THIS SECTION MATCHES THE STRING AND CALLS A ROUTINE TO CHANGE
602*  006010* IT TO THE NEW STRING.
603*  006020
604*  006030 6600-STRING-MATCH.
605*  006040      ADD 1 TO I.
606*  006050      ADD 1 TO DEL-CNT.
607*  006060      IF CHAR-ENTRY(I) = NAME-ENTRY(DEL-CNT)
608*  006070           IF DEL-CNT = NAME-LENGTH,
609*  006080               PERFORM 6700-STRING-CHANGE,
610*  006090               ELSE NEXT SENTENCE,
611*  006100           ELSE MOVE 1 TO NO-MATCH.
612*  006110
613*  006120*  THIS SECTION CHANGES THE STRING BY CALLING A ROUTINE.
614*  0061
615*  006140 6700-STRING-CHANGE.
616*  006150      MOVE NAME-LENGTH TO J.
617*  006160      MOVE WS-IN-COUNT(CNT) TO WS-IN-COUNT(101).
618*  006170      ADD 2 TO J.
619*  006180      SUBTRACT J FROM CH-LENGTH GIVING REM.
620*  006190      SUBTRACT WS-INDEX FROM NAME-LENGTH.
621*  006200      SUBTRACT NAME-LENGTH FROM I.
622*  006210      PERFORM 6800-MAKE-CHANGE.
623*  006220      MOVE 2 TO ANS-FLAG.
624*  006230      MOVE 2 TO ERROR-FLAG.
```

```
625*  006240
626*  006250*  THIS SECTION CALL THREE ROUTINES TO MODIFY AND MOVE THE STRING.
627*  006260
628*  006270 6800-MAKE-CHANGE.
629*  006280      SUBTRACT 70 FROM 7777 GIVING OLDL.
630*  006290      PERFORM 6850-MOVE-STRING UNTIL
631*  006300          WS-START = I.
632*  006310      MOVE OLDL TO OLDN.
633*  006320      ADD 2 TO OLDN.
634*  006330      ADD REM TO OLDN.
635*  006340      PERFORM 6900-MODIFY-STRING UNTIL
636*  006350         J IS GREATER THAN CH-LENGTH OR OLDL = 7778.
637*  006360      IF OLDN IS GREATER THAN 7779,
638*  006370          DISPLAY "ERROR, LINE HAS BEEN TRUNCATED, REENTER.",
639*  006380          DISPLAY WS-TABLE(101),
640*  006390          MOVE 0 TO ADD-COUNT.
641*  006400/
642*  006410      ADD 2 TO NAME-LENGTH.
643*  006420      ADD NAME-LENGTH TO WS-START.
644*  006430      ADD NAME-LENGTH TO I.
645*  006440      SUBTRACT 2 FROM NAME-LENGTH.
646*  006450
647*  006460      IF NAME-LENGTH NOT EQUAL REM,
648*  006470          MOVE WS-STAT(CNT) TO ANS,
649*  006480          MOVE SPACE TO WS-STAT(CNT).
650*  006490      SUBTRACT 1 FROM WS-START.
651*  006500      PERFORM 6950-MOVE-REST-STNG UNTIL WS-START IS
652*  006510          GREATER THAN WS-END OR OLDL = 7778.
653*  00
654*  006530      IF OLDL NOT EQUAL TO 7778,
655*  006540          PERFORM 6975-FINISH-MOVE UNTIL OLDL = 7778.
656*  006550
657*  006560      IF NAME-LENGTH NOT EQUAL REM,
658*  006570          MOVE ANS TO WS-STAT(101),
659*  006580          MOVE ANS TO WS-STAT(CNT).
660*  0
```

```
661*  0
662*  006600      ADD WS-INDEX TO NAME-LENGTH.
663*  006610*  THIS SECTION MOVES THE OLD PART OF LINE UP TO WHERE THE
664*  006620* STRING TO BE CHANGED STARTS.
665*  006630
666*  006640 6850-MOVE-STRING.
667*  006650      MOVE CHAR-ENTRY(WS-START) TO CHAR-ENTRY(OLDL).
668*  006660      ADD 1 TO WS-START.
669*  006670      ADD 1 TO OLDL.
670*  006680
671*  006690*  THIS SECTION MODIFIES AND MOVES THE NEW STRING INTO THE OLD
672*  006700* LOCATION.
673*  006710 6900-MODIFY-STRING.
674*  006720      MOVE NAME-ENTRY(J) TO CHAR-ENTRY(OLDL).
675*  006730      ADD 1 TO J.
676*  006740      ADD 1 TO OLDL.
677*  006750
678*  006760/
679*  006770*  THIS SECTION MOVES THE REST OF THE OLD LINE INTO NEW LOCATION.
680*  006780
681*  006790 6950-MOVE-REST-STNG.
682*  006800      MOVE CHAR-ENTRY(WS-START) TO CHAR-ENTRY(OLDL).
683*  006810      ADD 1 TO WS-START.
684*  006820      ADD 1 TO I.
685*  006830      ADD 1 TO OLDL.
686*  006840
687*  006850*  THIS SECTION MOVES SPACES IN THE NEW LINE IF NOT COMPLETE.
688*  006860
689*  006870 6975-FINISH-MOVE.
690*  006880      MOVE SPACE TO CHAR-ENTRY(OLDL).
691*  006890      ADD 1 TO OLDL.
```

```
692*   006900
693*   006910/
694*   006920************************FIND FUNCTION*********************************
695*   006930*  THIS SECTION FINDS A STRING IF IT IS IN THE FILE.
696*   006940
697*   006950 7000-FIND.
698*   006960      DISPLAY " ". ERASE.
699*   006970      MOVE 0 TO SEARCH-CNT.
700*   006980      MOVE SPA TO TEXT2, TEXT3.
701*   006990      MOVE 1 TO I.
702*   007000      PERFORM 7100-NAME-LENGTH UNTIL NAME-ENTRY(I) = "/"
703*   007010          OR I = 6.
704*   007020      IF NAME-ENTRY(I) NOT EQUAL TO "/"
705*   007030          DISPLAY ERROR1, TEXT5, LINE 2, POSITION 37,
706*   007040          ELSE ADD 1 TO I,
707*   007050              MOVE I TO WS-INDEX,
708*   007060               PERFORM 7100-NAME-LENGTH UNTIL NAME-ENTRY(I) = "/"
709*   007070                   OR I = 50,
710*   007080              IF NAME-ENTRY(I) NOT EQUAL TO "/",
711*   007090                  DISPLAY ERROR1, TEXT5, LINE 2, POSITION 37,
712*   007100                  ELSE MOVE I TO NAME-LENGTH
713*   007110                       IF NAME-LENGTH = WS-INDEX,
714*   007120                       DISPLAY "ERROR, VOID STRING, REENTER.",
715*   007130                       ELSE MOVE WS-CURRENT-PTR TO CNT,
716*   007140                            SUBTRACT 1 FROM NAME-LENGTH,
717*   007150                            DISPLAY "SEARCHING FOR ", TEXT5, LINE 2,
718*   007160                                POSITION 20,
719*   007170                            MOVE 0 TO ANS-FLAG,
720*   007180                            UNSTRING TEXT5 DELIMITED BY "/" OR ","
721*   007190                                    OR "*",
722*   007200                             INTO TEXT1, ANS, TEXT3, TEXT2,
723*   007210                            PERFORM 7200-SEARCH-CHAR.
```

```
724*  007220
725*  007230*  THIS SECTION USED AS AN ADDER TO FIND STRING LENGTH.
726*  007240
727*  007250 7100-NAME-LENGTH.
728*  007260      ADD 1 TO I.
729*  007270/
730*  007280*  THIS SECTION SETS UP THE ROUTINE TO FIND ALL OR PART OF THE
731*  007290* LOCATIONS OF THE STRING WITHIN THE FILE.
732*  007300
733*  007310 7200-SEARCH-CHAR.
734*  007320      IF TEXT3 = "ALL" OR TEXT3 = "A",
735*  007330          PERFORM 7300-CHAR-SEARCH UNTIL CNT = 0,
736*  007340          ELSE PERFORM 10200-NUM-CHECK-TEXT3.
737*  007350               IF ANS-FLAG NOT EQUAL 3,
738*  007360                IF CNT = 0,
739*  007370                   DISPLAY ERROR2, LINE 3,
740*  007380                   DISPLAY TEXT3, LINE 3, POSITION 30,
741*  007390                   ELSE MOVE CNT TO WS-START,
742*  007400                   MOVE 0 TO TXTN,
743*  007410                   PERFORM 10300-NUM-CHECK-TEXT2,
744*  007420                   IF ANS-FLAG = 4,
745*  007430                       PERFORM 7300-CHAR-SEARCH,
746*  007440                       ELSE IF CNT = 0,
747*  007450                           DISPLAY ERROR2, LINE 3,
748*  007460                           DISPLAY TEXT2, LINE 3, POSITION 30,
749*  007470                           ELSE IF TEXT2 LESS THAN TEXT3,
750*  007480                               MOVE 2 TO ANS-FLAG,
751*  007490                               DISPLAY ERROR3, LINE 3,
752*  007500                               ELSE MOVE CNT TO CNT-AMT,
753*  007510                                   MOVE WS-START TO CNT,
754*  007520                                   PERFORM 7300-CHAR-SEARCH UNTIL
755*  007530                                       CNT = WS-NEXT(CNT-AMT).
756*  007540                    ELSE DISPLAY "INVALID FIND OPTION, SYNTAX.",
757*  007550                        MOVE 2 TO ANS-FLAG.
758*  007560      IF ANS-FLAG NOT EQUAL 2,
759*  007570          DISPLAY "ERROR, STRING NOT FOUND".
```

```
760*   007580
761*   007590*  THIS SECTION SETS UP EACH LINE FOR SEARCHING FOR THE STRING.
762*   007600
763*   007610 7300-CHAR-SEARCH.
764*   007620      MULTIPLY CNT BY 77 GIVING WS-END.
765*   007630      SUBTRACT 70 FROM WS-END GIVING I.
766*   007640      PERFORM 7400-SRCH-THE-LN UNTIL I IS GREATER THAN WS-END.
767*   007650      MOVE WS-NEXT(CNT) TO CNT.
768*   007660      IF SEARCH-CNT = 20,
769*   007670          DISPLAY "TO CONTINUE TYPE IN NEW LINE.",
770*   007680          ACCEPT ANS.
771*   007690/
772*   007700*  THIS SECTION SEARCHES EACH CHARACTER UNTIL 1ST MATCH THEN CALLS
773*   007710* A ROUTINE TO CHECK SECOND AND SUBSEQUENT MATCHES.
774*   007720
775*   007730 7400-SRCH-THE-LN.
776*   007740      MOVE 0 TO NO-MATCH.
777*   007750      MOVE WS-INDEX TO DEL-CNT.
778*   007760      IF CHAR-ENTRY(I) = NAME-ENTRY(DEL-CNT) AND
779*   007770          NAME-LENGTH = WS-INDEX,
780*   007780          PERFORM 7600-DISPLAY-STRING,
781*   007790          ELSE IF CHAR-ENTRY(I) = NAME-ENTRY(DEL-CNT),
782*   007800          PERFORM 7500-CHAR-MATCH UNTIL DEL-CNT = NAME-LENGTH
783*   007810              OR NO-MATCH = 1
784*   007820          ELSE ADD 1 TO I.
785*   007830
786*   007840* THIS SECTION CHECK THE 2ND AND SUBSEQUENT MATCHES AND CALLS
787*   007850*     DISPLAY.
788*   007860
789*   007870 7500-CHAR-MATCH.
790*   007880       ADD 1 TO I.
791*   007890       ADD 1 TO DEL-CNT.
792*   007900          IF CHAR-ENTRY(I) = NAME-ENTRY(DEL-CNT),
793*   007910             IF DEL-CNT = NAME-LENGTH,
794*   007920              PERFORM 7600-DISPLAY-STRING,
795*   007930              ELSE NEXT SENTENCE,
796*   007940            ELSE MOVE 1 TO NO-MATCH.
```

```
797*   007950
796*   007960*  THIS SECTION DISPLAYS THE LINE AND SETS I = END OF LINE.
799*   007970
800*   007980 7600-DISPLAY-STRING.
801*   007990     PERFORM 2500-WRITE.
802*   008000     MOVE 2 TO ANS-FLAG.
803*   008010     MOVE WS-END TO I.
804*   008020     ADD 1 TO I.
805*   008030     ADD 1 TO SEARCH-CNT.
806*   008040
```

```
807*  008050/
808*  008060**********************HELP FUNCTION**********************************
809*  008070* THIS SECTION SETS UP A DISPLAY OF COMMANDS FOR EACH ROUTINE.
810*  008080
811*  008090 8000-HELP.
812*  008100      DISPLAY " ", ERASE.
813*  008110      DISPLAY "THIS INPUT EDITOR HAS 7 COMMANDS AND IS A MODULE".
814*  008120      DISPLAY "WHICH PROVIDES BASIC EDIT CAPABILITIES IN ORDER".
815*  008130      DISPLAY "TO ALTER THE INPUT CREATED DURING USER INTERFACE." .
816*  008140      DISPLAY "COMMAND    FORMAT          DESCRIPTION".
817*  008150      DISPLAY " ".
818*  008160      DISPLAY " DISPLAY   DISPLAY,ALL     DISPLAYS COMPLETE FILE.".
819*  008170      DISPLAY "           D,0100          DISPLAYS ONE LINE.".
820*  008180      DISPLAY "           D,0100,0200     DISPLAYS MULTIPLE LINES.".
821*  008190      DISPLAY "           D,LAST OR L     DISPLAYS LAST LINE.".
822*  008200      DISPLAY "NOTE: DISPLAY OR D, ARE BOTH CORRECT FORMATS.  COMMA
823*  008210-     "S OR BLANKS CAN BE USED.".
824*  008220      DISPLAY " ".
825*  008230      DISPLAY "ADD        ADD AFTR NNNN   THE ADD FUNCTION ADDS 1 OR
826*  008240-     " MORE LINES AFTER LINE".
827*  008250      DISPLAY "                           NNNN(4 DIGIT #). AFTER EAC
828*  008260-     "H LINE IS ADDED. A".
829*  008270      DISPLAY "           A,AFTR,NNNN     QUESTION OF LEVEL IS ASKED
830*  008280-     ".  TO TERMINATE ROUTINE,".
831*  008290      DISPLAY "           STOP ROUTINE \  TYPE IN \.".
832*  008300      DISPLAY " ".
833*  008310      DISPLAY "DELETE     DELETE AAAA,BBBB  DELETES FROM LINES AAAA
834*  008320-     " TO LINE BBBB.".
835*  008330      DISPLAY "           DEL,AAAA          DELETES LINE AAAA ONLY".
836*  008340      DISPLAY "NOTE: DELETE OR DEL ARE CORRECT FORMATS. BLANKS OR
837*  008350-     "COMMAS ARE DELIMITERS.".
838*  008360      DISPLAY "NOTE: AAAA & BBBB ARE 4 DIGIT LINE NUMBERS.".
839*  008370      DISPLAY " ".
840*  008380      DISPLAY "TO CONTINUE HELP, TYPE IN NEW LINE.".
```

```
841*  008390/
842*  008400      ACCEPT ANS.
843*  008410      DISPLAY " ", ERASE.
844*  008420      DISPLAY "COMMAND     FORMAT          DESCRIPTION".
845*  008430      DISPLAY " ".
846*  008440      DISPLAY "RESEQUENCE  RESEQ OR RE      RESEQS THE COMPLETE PGM
847*  008450-     " BY 10, BY DEFAULT.".
848*  008460      DISPLAY "              RE,AAAA,CCCC    RESEQ FROM LINE AAAA BY
849*  008470-     "THE AMT CCCC.".
850*  008480      DISPLAY "NOTE: AAAA & CCCC MUST BE 4 DIGIT NUMBERS, RE 0100,
851*  008490-     "0002.".
852*  008500
853*  008510      DISPLAY " ".
854*  008520
855*  008530      DISPLAY "FIND        FIND/EEEE/ALL OR A  SEARCHES THE COMPLETE
856*  008540-     " FILE FOR EEEE.".
857*  008550      DISPLAY "            F/EEEE/100          SEARCHES LINE NUMBER
858*  008560-     "0100 FOR EEEE.".
859*  008570      DISPLAY "            F/EEEE/10,50        SEARCHES FROM LINE 00
860*  008580-     "10 TO 0050 FOR STRING EEEE.".
861*  008590      DISPLAY "NOTE: STRING EEEE CAN BE 40 CHARACTERS LONG.".
862*  008600      DISPLAY "NOTE: DELIMITERS ARE , OR *, OR /.".
863*  008610      DISPLAY "NOTE: THE STRING MUST BE ENCLOSED BY SLASHES, /.".
864*  008620
865*  008630      DISPLAY " ".
866*  008640      DISPLAY "CHANGE  CH/EEEE/FFFF/ALL        CHANGES ALL STRING EE
867*  008650-     "EE TO FFFF.".
868*  008660      DISPLAY "        CH/EEEE/FFFF/100,200    CHANGES LINE 0100 TO
869*  008670-     "0200'S EEEE TO FFFF".
870*  008680      DISPLAY "        CH/G/H/10,V             CHANGES G TO H WITH
871*  008690-     " DISPLAY FOR VERFICATION.".
872*  008700      DISPLAY "NOTE: V OR VERIFICATION OF TEXT IS DONE BEFORE THE
873*  008710-     "CHANGES ARE MADE.".
874*  008720      DISPLAY "NOTE: DELIMITERS ARE , OR *, OR /.".
875*  008730      DISPLAY "NOTE: STRING EEEE & FFFF MUST BE ENCLOSED BY /.".
876*  008740      DISPLAY "NOTE: MAX LENGTH OF EEEE OR FFFF IS 35 CHARACTERS.".
```

```
877*  008750      DISPLAY " ".
878*  008760      DISPLAY "THIS IS END OF HELP ROUTINE.".
879*  008770      DISPLAY "TO CONTINUE EDITOR ROUTINE, TYPE IN NEW LINE.".
880*  008780      ACCEPT ANS.
881*  008790/
882*  008800
883*  008810*  THIS SECTION PRINTS OUT AND DELIVERS THE NEW USER FILE TO
884*  008820*  DISK.
885*  008830 9000-OUTPUT.
886*  008840      DISPLAY "DO YOU WANT THE EDIT FILE SAVED? TYPE IN YES OR NO.
887*  008850-     " IF NO, THE OLD DATA".
888*  008860      DISPLAY "DESCRIPTION SOURCE FILE WILL BE RETURNED.".
889*  008870      ACCEPT TEXT3.
890*  008880      IF TEXT3 = "YES",
891*  008890          PERFORM 9100-OUTPUT UNTIL CNT = 0,
892*  008900          ELSE IF TEXT3 = "NO",
893*  008910              PERFORM 9300-OUTPUT-OLD-FILE THRU 9500-FINISH-OUTPUT,
894*  008920                   ELSE DISPLAY "ANS MUST BE YES OR NO.",
895*  008930                       GO TO 9000-OUTPUT.
896*  008940 9100-OUTPUT.
897*  008950      PERFORM 9200-OUTPUT.
898*  008960      MOVE WS-NEXT (CNT) TO CNT.
899*  008970 9200-OUTPUT.
900*  008980      IF OUT-PRINT-FLAG = 1,
901*  008990          WRITE OUT-PRINT FROM WS-TABLE(CNT).
902*  009000      MOVE WS-STAT(CNT) TO WS-STATS(CNT).
903*  009010      WRITE NEW-DATA-REC FROM WS-TABLE(CNT).
904*  009020
905*  009030 9300-OUTPUT-OLD-FILE.
906*  009040      CLOSE OLD-DATA-FILE.
907*  009050      OPEN INPUT OLD-DATA-FILE.
908*  009060 9400-READ-WRITE-OLD-FILE.
909*  009070      READ OLD-DATA-FILE INTO WS-IN-HOLD,
910*  009080          END GO TO 9500-FINISH-OUTPUT.
911*  009090      WRITE NEW-DATA-REC FROM WS-IN-HOLD.
912*  009100      GO TO 9400-READ-WRITE-OLD-FILE.
913*  009110 9500-FINISH-OUTPUT.
```

```
009120*
009130**********************SEARCH FOR LINE NUMBERS**********************
009140*  THIS SECTION SEARCH FOR THE SEQ NUMBER OF THE LINES AND
009150*  RETURNS LOC.
00916
009170 10000-SEARCH.
009180     MOVE 0 TO SEARCH-CNT.
009190     MOVE WS-CURRENT-PTR TO CNT.
009200     PERFORM 10100-SEARCH UNTIL
009210                CNT = 0 OR WS-IN-COUNT (CNT) = TXTN.
009220     IF CNT = 0,
009230         MOVE 2 TO ANS-FLAG.
009240
009250 10100-SEARCH.
009260     MOVE WS-NEXT(CNT) TO CNT.
```

```
930*  009280**************************NUMBER CHECK TEXT 2 AND TEXT 3***************
931*  009290*  THIS SECTION CHECKS IF A 1, 2, 3 OR 4 DIGIT # IN TEXT3.
932*  009300
933*  009310 10200-NUM-CHECK-TEXT3.
934*  009320      IF TEXT3 IS NUMERIC,
935*  009330         MOVE TEXT3 TO TXTN,
936*  009340          ELSE IF TXT3 IS NUMERIC,
937*  009350              MOVE TXT3 TO TXTN,
938*  009360              ELSE IF TXT2 IS NUMERIC,
939*  009370                  MOVE TXT2 TO TXTN,
940*  009380                       ELSE IF TXT1 IS NUMERIC,
941*  009390                           MOVE TXT1 TO TXTN,
942*  009400                           ELSE MOVE 3 TO ANS-FLAG.
943*  009410      IF ANS-FLAG NOT EQUAL 3,
944*  009420          MOVE TXTN TO TEXT3,
945*  009430          PERFORM 10000-SEARCH.
946*  009440
947*  009450*  THIS SECTION CHECKS IF A 1, 2, 3, OR 4 DIGIT # IN TEXT2.
948*  009460
949*  009470 10300-NUM-CHECK-TEXT2.
950*  009480      IF TEXT2 IS NUMERIC,
951*  009490         MOVE TEXT2 TO TXTN,
952*  009500         ELSE IF TX3 IS NUMERIC,
953*  009510             MOVE TX3 TO TXTN,
954*  009520             ELSE IF TX2 IS NUMERIC,
955*  009530               MOVE TX2 TO TXTN,
956*  009540               ELSE IF TX1 IS NUMERIC,
957*  009550                   MOVE TX1 TO TXTN,
958*  009560                        ELSE MOVE 4 TO ANS-FLAG.
959*  009570      IF ANS-FLAG NOT EQUAL 4,
960*  009580          MOVE TXTN TO TEXT2,
961*  009590          PERFORM 10000-SEARCH.
```

BIBLIOGRAPHY

BIBLIOGRAPHY

1. Kernighan, Brian W., Software Tools. Reading Mass: Addison-Wesley Publishing Company, Inc, 1976.

2. Maryanski, F. J., Buser, R. H., Hoflich, M. A., Hunt, W. C., Kusnyer, S. K., Stevens, T. J., Automatic Generation of Third Normal Form, Kansas State University, Department of Computer Science, Manhattan, Kansas, September 1977.

3. McGowan, Clement L., Kelly, John R. Top Down Structured Programming Techniques. New York: Petrocelli Charter Company, 1975.

4. NCR, Inc, Interface Multiprogramming Operating System, NCR Corporation, 1977.

5. Tsichritzis, Dionysios C. and Lochovsky, Frederick H., DATA BASE MANAGEMENT SYSTEMS, New York: Academic Press, 1977.

6. Van Dam, A., "On-Line Text Editing: A Survey." Computing Surveys, September, 1971.

TEXT EDITOR IMPLEMENTATION FOR THE THIRD NORMAL FORM SYNTHESIS SYSTEM

BY

T. J. STEVENS

B. S., UNIVERSITY OF NEBRASKA, OMAHA 1972

--

AN ABSTRACT OF A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1978

ABSTRACT

Implementation of a Data Base Management System (DBMS) based upon a relational model requires data to be represented in Third Normal Form (3NF). This report deals with the implementation of an interactive Text Editor to a User Interface program in developing an automatic synthesis using Bernstein's Algorithm for 3NF relations. In implementing the Bernstein's Algorithm, a system was set up to accept a description of user data in a form that can be realized from a typical report. This input is collected interactively by a User Interface (UI) program which produces a hierarchical representation of the data. As a result of the possibility to make an error in this program, a Text Editor was developed to prevent the user from having to reenter all data. The Text Editor operates upon the output of the User Interface program and permits the user to display, add, delete, resequence, find, and change any erroneous information. The Text Editor then produces a file which is used to run through a special version of the User Interface program which allows the data to be correctly represented in hierarchical form.