FEASIL IMPLEMENTATION UNDER VAX VMS WITH DESIGN INFORMATION( U) ALABAMA UNIV IN HUNTSVILLE DEPT OF ELECTRICAL AND COMPUTER EN. J D HARR ET AL. NOV 86
TECHNICAL REPORT CR-RD-SS-86-5

FEASIL IMPLEMENTATION UNDER VAX VMS WITH DESIGN INFORMATION

J. D. Marr, B. E. Tucker, T. A. Palmer,
D. C. Pool, and T. N. Long
The University of Alabama in Huntsville
Huntsville, Alabama 35805

NOVEMBER 1986

Prepared for:
System Simulation and Development Directorate
Research, Development, and Engineering Center

Contract No. DAAH01-82-D-A008

U.S. ARMY MISSILE COMMAND
Redstone Arsenal, Alabama 35898-5000

Approved for public release; distribution is unlimited.
# FEASIL IMPLEMENTATION UNDER VAX VMS WITH DESIGN INFORMATION

**Abstract**

Implementation of the FEASIL database management system under a VAX VMS operating system is described. Technical considerations are reviewed, and new features that resulted and code changes that were required are summarized. Improvements in the SORT function are also described and quantified. A significant amount of FEASIL design information is presented in the form of flow charts for the entire body of code.

### COSATI Codes

<table>
<thead>
<tr>
<th>FIELD</th>
<th>GROUP</th>
<th>SUB-GROUP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>FEASIL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Database Management System</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FORTRAN</td>
</tr>
</tbody>
</table>

### Subject Terms

- FEASIL
- Database Management System
- FORTRAN

---

1/11 (blank)
This technical report was prepared by the Research Staff of the Electrical and Computer Engineering Department, School of Engineering, The University of Alabama in Huntsville. The purpose of the report is to provide documentation of the technical work performed and results obtained under delivery order 0043 of MICOM Contract No. DAAH01-82-D-A008; Terry N. Long, Principal Investigator. The report also provides an addendum to the documentation of the technical work performed and results obtained under delivery order 0027; Dr. James D. Marr, Principal Investigator. The project was performed by Dr. James D. Marr, Bruce Tucker, Tim Palmer, David Pool, Terry Long, and several other members of the UAH ECE research staff. Dr. M. M. Hallum III, Chief, System Evaluation Branch, was the technical monitor; and Mr. Mark Horton also from the Systems Evaluation Branch of the Research, Development, and Engineering Center, U. S. Army Missile Command, provided technical coordination.

The technical viewpoints, opinions, and conclusions expressed in this report are those of the authors and do not necessarily express or imply policies or positions of the U. S. Army Missile Command.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0 INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>2.0 VAX IMPLEMENTATION TECHNICAL CONSIDERATIONS</td>
<td>2</td>
</tr>
<tr>
<td>3.0 NEW FEATURES</td>
<td>4</td>
</tr>
<tr>
<td>4.0 CODE CHANGES</td>
<td>5</td>
</tr>
<tr>
<td>5.0 REVISED SORT PROGRAM</td>
<td>10</td>
</tr>
<tr>
<td>5.1 Introduction</td>
<td>10</td>
</tr>
<tr>
<td>5.2 Revised SORT Technique</td>
<td>10</td>
</tr>
<tr>
<td>5.3 SORT Improvement Quantification</td>
<td>11</td>
</tr>
<tr>
<td>6.0 FEASIL Design Summary</td>
<td>12</td>
</tr>
<tr>
<td>7.0 CONCLUSIONS AND RECOMMENDATIONS</td>
<td>13</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>25</td>
</tr>
<tr>
<td>APPENDIX A - MAIN</td>
<td>A-1</td>
</tr>
<tr>
<td>APPENDIX B - RETEVD</td>
<td>B-1</td>
</tr>
<tr>
<td>APPENDIX C - DSPLAY</td>
<td>C-1</td>
</tr>
<tr>
<td>APPENDIX D -RLFUNC</td>
<td>D-1</td>
</tr>
<tr>
<td>APPENDIX E - PRINTREL</td>
<td>E-1</td>
</tr>
<tr>
<td>APPENDIX F - SORT2</td>
<td>F-1</td>
</tr>
<tr>
<td>APPENDIX G - EDTREL</td>
<td>G-1</td>
</tr>
<tr>
<td>APPENDIX H - NEWRELAT</td>
<td>H-1</td>
</tr>
<tr>
<td>APPENDIX I - DELREL</td>
<td>I-1</td>
</tr>
<tr>
<td>APPENDIX J - MODCOL</td>
<td>J-1</td>
</tr>
<tr>
<td>APPENDIX K - MERGRL</td>
<td>K-1</td>
</tr>
<tr>
<td>APPENDIX L - RORGRL</td>
<td>L-1</td>
</tr>
<tr>
<td>APPENDIX M - BKUPRL</td>
<td>M-1</td>
</tr>
<tr>
<td>APPENDIX N - DBSTAT</td>
<td>N-1</td>
</tr>
<tr>
<td>APPENDIX O - HELP</td>
<td>O-1</td>
</tr>
</tbody>
</table>
### TABLE OF CONTENTS (CONT'D)

<table>
<thead>
<tr>
<th>Appendix</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>TPL1B1</td>
<td>P-1</td>
</tr>
<tr>
<td>Q</td>
<td>TPL1B2</td>
<td>Q-1</td>
</tr>
<tr>
<td>R</td>
<td>TPL1B3</td>
<td>R-1</td>
</tr>
<tr>
<td>S</td>
<td>TPFILECR</td>
<td>S-1</td>
</tr>
<tr>
<td>T</td>
<td>F710</td>
<td>T-1</td>
</tr>
<tr>
<td>DISTRIBUTION</td>
<td></td>
<td>Dist-1</td>
</tr>
</tbody>
</table>
1.0 INTRODUCTION

The FEASIL Database Management System (DBMS) is a relational system; that is, it is designed to most effectively support tabular data. It was first developed about 1978 by M. M. Hallum and has evolved several new features since that time. Currently, it can accept new data from keyboard or file, alter existing data, select only certain parts of the data for further manipulation, sort data into ascending or descending order, perform limited mathematical functions, display or plot data, print tables of data, output data for use by other programs, and perform several other functions.

FEASIL was first developed under a Perkin-Elmer OS-32 operating system and has since received many refinements through several revisions. The principal task of the work documented by this report was to transfer FEASIL to a VAX-11/780 computer under a VMS operating system. Section 2.0 describes several of the technical considerations which affected the work. New features which resulted from the transfer are then described in Section 3.0. Code changes in addition to the new features were required for the transfer. They are documented in Section 4.0. Major performance improvements resulted during the transfer with the SORT function receiving the greatest improvements. These improvements are described and quantified in Section 5.0. As part of this and previous tasks, an effort was made to carefully examine the body of FEASIL code through development of flow charts. This design summary effort is described in Section 6.0, and the flow charts are presented in the appendices. And finally, an extensive list of recommendations and conclusions is presented in Section 7.0.
2.0 VAX IMPLEMENTATION TECHNICAL CONSIDERATIONS

The principal goal of this project was to port FEASIL from a Perkin-Elmer (P-E) computer to a VAX computer. Differences between the P-E and VAX that made this conversion nontrivial included file structure differences, different file name limitations, Flex dialect differences, and operating system differences. Within the main goal was the requirement that the transported version be upward compatible and have substantially similar capabilities. A secondary goal was to improve feedback to the user.

From the programmer's viewpoint, the P-E files reside directly on a device, with pathnames and filenames such as

USER:FILE.EXT

typical. On the VAX, the device is subdivided into directories and possibly subdirectories, with a name such as

dual:[userdisk1.humanbeingname.activityname]filename.extension

typical. The device name and directory path for a first level user subdirectory is typically 28 characters. Further, files in the "current default directory" of the VAX can be accessed without either the device name or the directory path name. This forces creation of a large buffer for device name (including path) and the option to have no device name.

Filenames on the P-E are limited to 8 characters with a 3-character extension, as noted above. The VAX filenames are limited to 39 characters with a 39-character extension. The easiest solution is to just continue use of the 8+3 filenames. Since the directory name is also limited by the same 39-character limit, and operator tolerance is the principal limit on the number of levels of subdirectory, the device name buffer and any descendant buffers should be huge. As a practical limit, we assumed 44 characters as an upper bound for name and path.

The most annoying difference between the Flex dialects was that the construct

WHEN(I.LT.0)
  I=0
  GO TO 73
ELSE
  I=1
FIN
.....

converts without error on the P-E, while it produces code on the VAX of the form

IF(.NOT.(I.LT.0))GO TO 99999
I=0
GO TO 73
GO TO 99998
99999 I=1
99998 .......

...
This causes a FORTRAN compile error because the statement "go to 99998" is unreachable. A more significant problem is that some of the more powerful FORTRAN 77 (F77) constructs (such as the "if...then...else...endif") have keyword conflicts with Flex, making them incompatible. However, Flex can understand only capital letters; making all F77 constructs lower case lets them pass the flecs preprocessor unmodified and be compiled.

Operating system differences became most apparent in file operations. The two computers differ in their mechanisms for renaming files, but that problem can be overcome within the FILEREN subroutine itself. As expected, I/O error numbers were incompatible, so a separate routine called VAXERR was created and linked through the original error handler. Tape handling is sufficiently difficult that the backup function was forced into a stand-alone program.
3.0 NEW FEATURES

New features visible to the user fall into two categories, flexibility and feedback. The first group includes menu modifications. The second group consists of messages to inform the user of current activity and of the progress of long-duration activities.

The main menu originally had functions 0 to 11 (function 12 is limited to system programmers). The VAX version has the same choices, but will also accept commands by the first letter(s) of the command name; the user need enter only enough letters for a nonambiguous choice. This is of benefit to casual users and novice typists who can reach letters more easily than digits. Another system program function (13) has been added to perform a brute-force tuple dump to the printer; this is effective for archival dumps and for debugging. Help options are available at the main menu and at several of the secondary menus.

FEASIL now provides feedback approximately every twenty seconds or less. When the original .ADF and .TF are being copied at the start of "retrieve, manipulate, and plot" (8), the program declares that the copy is beginning and reports every hundred records copied. The SORT function now provides a crude advance estimate of the time needed and periodically reports how the bubble sort is progressing. The DELETE function in EDIT (4) now reports every twentieth record for large delete actions.

Additional feedback information was added to several of the error messages and to dangerous activities. When a FORTRAN error occurs, instead of giving machine-dependent error numbers, the system now generally gives a text string defining the error. Since the MOVE (M) command in RETRIEVE, MANIPULATE AND PLOT can delete tuples from the original relation, the user is now warned a second time that he has made a dangerous choice and asked to reconfirm the decision.
4.0 CODE CHANGES FROM PERKIN-ELMER VERSION

Changes to the program were made for two reasons, machine factors and program enhancement. Differences in the file structure were the main reason for the machine-dependent changes, and an attempt was made to confine these to the F710 library. Program enhancement changes included message changes, conversion from Flex to FORTRAN 77, and algorithm changes. Errors found in the flowcharting project were corrected on both the VAX and P-E versions; the errors were predominantly evolutionary in origin, frequently being related to the string-in-TF data compression method. The structure of the .TDF, .ADF, and .TF files was maintained for compatibility with the P-E FEASIL. The rest of this section will provide a brief description of the changes to each program file.

Runtime memory usage was considered first. Currently, the main menu driver and all five libraries are permanently resident in memory, with the ten main-menu options overlaid. A secondary overlay is used for some of the activities under "RETRIEVE, MANIPULATE, AND PLOT" (option 8). Specifically, only one of FUNCTION, PLOT, PRINT, or SORT will be in memory at a given time. Table 1 shows that while many subprograms are called by all ten activities, some are called only by one of the main menu choices. An analysis of this subprogram calling hierarchy indicates that a restructuring of the libraries would be both possible and beneficial. The subprograms and their locations are listed in Table 2, and Table 3 shows the same information from the opposite perspective, location and contents. A file-by-file analysis of changes follows.

P-E files have read/write keys, but these are not available on the VAX; FEASIL generally does not ask for keys on the VAX. A more general rewrite would add a function in the machine-dependent library F710 to request or not request keys.

The file BKUPSTUB (BACKUPRL on the P-E) contains the tape backup routine. Since operating system restrictions on the VAX make the traditional backup method extremely inconvenient, the backup routine, BACKUPRL, has been converted to a "stub" explaining how to call the current version of the stand-alone input program for tapes generated by FEASIL on the P-E. It is suggested that backup on the VAX be performed by standard VAX procedures. Efforts are in progress to write a program to prepare tapes for return to the P-E. One of the smaller compatibility problems is that the VAX and P-E disagree on the byte order for integer numbers on a tape; they agree on text order. Additionally, the VAX insists on specialized system calls for tape data transfer rather than standard FORTRAN READ and WRITE statements.

The file DABA08 contains both the main program and a BLOCK DATA initialization section. The main program is predominantly F77, which more effectively supports the text selection of desired activity. The main menu handler now accepts letters as well as digits; only upper case is supported to remain consistent with the following menus in the original P-E version. All subprogram calls have been modified so that the array length is also passed for any array sent to the subprogram. In the future, it will be possible to dimension all arrays in subroutines to the correct
length activate runtime subscript checking, and eliminate most references to the LCONST COMMON; this will aid in discovering runtime errors that can corrupt data. The COMMON's in the BLOCK DATA section have been adjusted to a minimum length of 19 words to accommodate the possibility of longer file-names on the VAX. The new COMMON named MAXLEN has been added to indicate the longest allowable name, which is set at 13 words at the present. The default device/directory name has been set to 0 characters with text of "Current System Default Directory"; the FILDES routine (elsewhere) was also tampered to be able to echo this default despite its zero length.

The file DABAEXIT contains the status reporting routine DBSTAT. It originally supported the normal status function (11 or S) and the system programmer function (12 or P). It has been modified slightly toward F77, but the principal change was the addition of a second system function TOTAL (13 or T), which does a total dump of the entire relation to the printer. Damaged or defective ADF records are reported, rather than system failure resulting. Suppression of this second system function could be accomplished most easily by modification of the main menu to reject the new commands. The function to be performed is now specified by a character variable passed as the final parameter. The modifications were also retrofitted to the P-E version.

The file DELETREL contains the relation deletion routine DELREL. It is still in Flex, but is only 46 lines long including comments. No changes were necessary.

The file DSPLAY contains the plotting routine DSPLAY, originally written by M. Castellano based on earlier code by J. Marr. It was changed only slightly and still relies heavily on Flex. The printer device number was changed from 13 to 6. The pen plotter function is still a nonfunctioning stub. It could stand some modification for compactness and efficiency since it reimplements some functions available in the libraries.

The file EDTREL7 contains the data entry and editing routine EDTREL. Few changes were made in the conversion process, but several logic errors were discovered during flowcharting and were corrected on both the VAX and P-E versions. The most visible changes were periodic feedback in the add-data-from-a-file function (A), the search/find function (F), and the delete function (D); all now comment on progress about every 20 seconds.

The file F710 is a machine-dependent library. All routines have built-in error checking and messages. Subroutine FILEOPEN uses the standard OPEN function for 'OLD' and 'DIRECT'. Subroutine FILEDEL on the VAX opens a file and then closes it with 'DELETE' option; this may be portable for future use. Subroutine CREAFIL checks for file prior existence (OPEN as 'OLD') before creating the file by using an OPEN as 'NEW' and 'DIRECT' with specified record length and file length. The check for prior existence is because the VAX allows creation of a new version of an existing file without producing an error message. Subroutine FILEREN required extensive rewriting. The name of the file on the input logical unit is determined using an INQUIRE; the logical unit is closed; the file is renamed by a system library function; and the file (under either the new or old name) is reOPENed on the logical unit. Subroutine FILECLOSE consists of only the
FORTRAN CLOSE function. Subroutine MODAP is a stub with no executable statements. Finally, subroutine VAXERR accepts an error number, prints to the screen its meaning (if known), and reports back whether a meaning was printed.

The file FUNCTION contains the mathematical functions routine RLFUNC used in option 8. It was modified to run faster by simple rearrangement of some of the code. A logic error was also detected that caused an infinite loop for partially active data; this was corrected in both versions. It still uses Flex.

The file HELP contains the help routine HELP.

The file MERGERL contains the relation-merging routine MERGRL. It now provides a progress report every 20 lines during the add (A) function.

The file MODCOLUM contains the column-modification routine MODCOL. It was substantially unchanged, and still relies heavily on Flex.

The file NEWRELAT contains the relation-creation routine NEWREL. It was changed only slightly and still relies heavily on Flex.

The file PRINTREL contains the printing routine PRINTREL used in option 8. It was changed to account for text length differences. It still relies heavily on Flex.

The file REORGREL contains the relation-reorganization routine REORGRL.

The file RETRIEV7 contains the retrieve/manipulate/plot function RETEVD, which is option 8. Most modifications were implemented to improve readability of the code or feedback to the user, but some logic errors were also eliminated. A warning and request for verification were added to the delete-tuples-after-move option on the move command (M). It relies heavily on Flex.

The file SORT2 contains five routines in support of the sort function used in option 8. The CSORTF routine builds a file containing the tuple number and the tuple as text data from the designated column; it was not changed and still relies heavily on Flex. The SORTFL routine was substantially rewritten and details appear in the next section. The supporting routines SORTFA, ASCTYP, ASCCHR are new and include only two instances of Flex.

The file TPFILECR is a library of routines related to file creation. Unless noted, changes are for readability and feedback only. Subroutine FILDES inquires directory name (calling it "volume name"), verifies legality, updates the default volume name if appropriate, and builds an extended name of volume and filename and period. FILDES needed substantial modification to handle the changes in names on the VAX. The handling of the "default directory" message was ad hoc, and should be changed in later revisions. SECURE gets the read/write keys, but is not called on the VAX; it could be replaced by a stub. EMIARY just writes zeros into an array; it
now consists of a DO loop. ADFCRE, TDFCRE, and TFCRE add the appropriate extension to the base relation name and create the file via CREAFIL. RELFC just calls these three routines. ADFDEL, TDFDEL, and TFDEL add the appropriate extension to the base relation name and delete the file via FILEDEL. RELFD just calls these three routines. EQUIVI copies N words from one array into another; it now consists of a pretest and a DO loop. OPNADF, OPNTDF, and OPNTF add the appropriate extension to the base relation name and open the file on the specified logical unit. FILERR checks for recognized FORTRAN error numbers, but is somewhat machine dependent. NAME gets a relation name of not more than 42 characters, which is later "hashed" to make the file name. All of these routines were originally Flex, but some have been altered to remove some of the vestiges. The OPNTF routine originally used a FORTRAN COMMON area for building the extended name. However, the RETEVD function, which calls OPNTF, used the same COMMON for another purpose, and the conflict resulted in system malfunctions. As a result, OPNADF, OPNTDF, OPNTF, ADFCRE, TF CCRE, TFDEL, ADFDEL, TDFDEL, and TFDEL were modified to eliminate use of the COMMON. This change to local arrays caused no change in the size of the .obj file and reduced the size of the .exe (executable) file.

The file TPLIB1 is a library of text and number manipulation routines. Few changes were necessary in this file, and most routines rely heavily on Flex. STRLEN reports the length of a string and copies it to FEASIL format. FTIPI converts a floating point (real) number to pseudo-integer by saving it to memory as "real" and reading it back as an integer; an EQUIVALENCE installed in a rewrite about 1984 simplified the code and made it portable. PITFP converts pseudo-integer to real by the same approach. YESNO demands from the user an answer of YES or NO (Y or N); it still uses FEASIL string manipulation techniques. Subroutine GETNUM gets text in alpha-numeric form from the user and converts it via REANUM. REANUM converts text to the corresponding integer value. ELEB eliminates leading edge blanks by shifting the FEASIL format text left. ENFPTA converts real numbers to FEASIL string format. ENINTA does the same for integers. GETFP converts a text number from the terminal into floating point. REAFP does the actual conversion. All of the routines in this library use the GET routine to read characters from the terminal.

The file TPLIB2 is a library of routines for file access. Few changes were necessary; about half of the routines are pure F77. Routines PUTTDF and RTVTDF open the TDF file via OPNTDF, write or read the file, respectively, and close the file via FILECLS. ADDTUP adds a tuple to an existing relation. DELTUP removes a tuple. PUTIFO puts data into the .ADF or .TF as appropriate for the data strategy. PTFRC and PADFRC write data to the .TF and .ADF files, respectively. If the string sent to PADFRC is shorter than 8 characters and an override has not been declared, then PADFRC will return a signal and two words to be stored in the TF. RTFRC and RADFRC read data from the .TF and .ADF files respectively. COLNAM gets the name of a specified column from the .ADF file. HASHIT computes the file name by hashing the relation name. The first four letters of the hash name are the same as for the relation name; the fifth is a length signal; the sixth and seventh are a hash of relation name characters 5 to 42; and the eighth letter is 'A'. It was modified recently (1985) to use an algorithm which
reduces hashing collisions; the P-E version checks to see if a file from the previous version exists and recommends conversion. UTFAR converts the two TF words into a 1 to 7 character FEASIL format string into the ADF buffer, inverting the TF storage mode of PADFRC.

The file TPLIB3 is a library of routines for file maintenance. RDOADF and RDOTF create a larger .ADF/.TF file, copy the old information into it, and delete the old version. CPYREL copies both the .ADF and .TF files into a new name. FSLIDE shifts an array downward one word and is used to convert FEASIL format text into FORTRAN format. Only slight changes have been made from the original Flex text.

Some additional tools have been built as part of the conversion process. NAMES examines an existing relation, reports the long name, and prints a macro to rename the relation to the new hash code from either the original "hash" or the original name. QUICK is a VAX program to convert a tape-transferred relation from P-E tape format (copied to the disk first using MOUNT/FOREIGN and COPY) to a standard relation. The command file which does the copy and calls QUICK is named FEASILTAPE. A relation repair function named PATCH is under development.
5.0 REVISED SORT PROGRAM AND TIMING TESTS

5.1 Introduction

Users have long complained of the slowness of the SORT function, and it has been speculated that this is due to the frequent disk accesses of the file-to-file algorithm. When an attempt was made to run the original algorithm on the VAX, the machine malfunctioned. It was then decided to modify the routine. The original sort program operated in two phases, (1) copying of the tuple number and the data from the designated column to a sort file SF in CSORTF and (2) sorting of the data in SORTFL. These two routines were called in sequence by the RETEVD routine. In order to preserve modularity, all changes were made internal to the SORTFL routine.

5.2 Revised SORT Technique

Three acceleration measures were instituted simultaneously. They were an in-memory routine for small data sets, a different buffering scheme for other sorts, and code tightening. The new SORTFL routine first builds file names and opens the SF to a logical unit. If the strategy is nonstring and the tuple count is less than 1000, it calls SORTFA to do the in-memory sort and then returns. Otherwise, it creates two scratch files 11 and 12. If the text is nonstring, the records of these files are only two words wide. The SF is then copied into 11, and the data is sorted using 11 and 12 as "ping-pong" buffers. When the sort is completed, the data is transferred back to the SF as text, the files are closed (and scratch files deleted), and the program returns.

The sort algorithm now used is a weaving bubble sort. Sorting starts at the bottom, and the largest value is carried to the top, remembering when the last swap occurred. Next, from the point of the last swap, the smallest value is carried to the bottom, again remembering the last swap point. The swap points are the new limits for later passes, and execution can stop when they meet. The sort code reports the number of data values swapped every 100 passes to maintain user confidence that the program is still running.

The fast in-memory sort SORTFA also uses the weaving bubble sort, but does not need scratch files. It was decided, for code compactness, to have only one sort sequence and make the order decision based on strategy. The use of logical variables and logical assignment statements and comparisons also aided compactness.

Two auxiliary functions were added to aid the sort. ASCTYP returns an integer value to indicate whether the character input is (1) a blank, (2) a capital letter, (3) a lower case letter, (4) a digit, or (5) something else. This appeared to be close to the original sequence, but there was some evidence that digits should precede letters instead. If some other sorting sequence is desired in the future, this routine could be modified easily. ASCCHR accepts two character values c1, and c2, and returns a logical value of .TRUE. if c1 is less than or equal to c2. ASCCHR uses ASCTYP to separate nonsimilar characters.
5.3 SORT Improvement Quantification

The new version appears to require less memory. The rewrite reduced the .FLX file from 953 to 884 lines, and the derived .FOR (FORTRAN) file from 648 to 500 lines. The .OBJ file which contains resulting assembly code and linkage information was reduced from 32 to 31 blocks, a negligible difference but still a reduction.

The major difference is in speed. The new algorithm requires about 10 seconds for the CSORTF conversion and un-conversion, and 23 seconds to sort 500 random floating point numbers (263 passes). The old algorithm took 10 + 600 seconds. This is a factor of 20 to 25 speed improvement. For larger data sets, the operating system handles buffering and the nonstring records are packed many to a disk block, instead of singly. This results in fewer disk accesses and greater speed of execution. String data appeared to sort slightly faster, but this is hard to measure on a multiuser system.
6.0 FEASIL DESIGN SUMMARY

A major effort was initiated under a previous task to complete examine the body of FEASIL code through flow charting it. This effort greatly assisted the transfer process through identification of dormant errors in FEASIL code. The flow charts are presented in Appendices A through T. They should prove to be a valuable tool in the analysis of errors still hidden in the VAX implementation. They should also assist in the refinement process that will inevitably follow this effort.
7.0 CONCLUSIONS AND RECOMMENDATIONS

Limited testing reveals no errors, so the system appears ready for production work. As with any large software system, problems will appear from time to time and can be treated as they are revealed. Recommendations for future effort can be divided into two areas, enhancement and rewrite.

Several functions could be added to the FEASIL DBMS to enhance its power. The most powerful function would be a mathematical function to compute a linear or nonlinear function of two columns and store the result in a third column. Small changes that might aid the user would include modifying the HELP function to loop, offering its menu again, until the user selects "quit."

Small invisible code changes which would improve performance include (1) increasing the buffer size in SORTFA and (2) when deleting tuples in the edit function, delete the one with the highest number first. The current tuple delete function removes the lowest numbered one and shifts all tuples above it downward; it then deletes the next one. If a relation is 100 tuples long and tuples 11 to 100 are being deleted, then tuple 11 is removed, all 89 remaining are shifted, the new tuple 11 is removed, all 88 remaining are shifted, .... In the same case with the suggested change, no shifts would occur and thousands of disk accesses would be eliminated.

FEASIL has evolved into a powerful tool. It would be desirable to eventually port FEASIL to the HP 9000, the IBM PC, and several other computers. However, the flowcharting and VAX conversion efforts have revealed some problems that impact on portability.

FEASIL was originally written in Flex on a machine subject to frequent malfunctions. Several compromises were made as a result.

(1) Most file accesses (read or write) are preceded by an open and followed by a close; the overhead and disk accesses of these opens and closes can slow down a program significantly.

(2) The string handling capabilities of Fortran 77 were not available, and several activities were performed in a difficult fashion. For example, a 16-character string can be shifted left one character in F77 by

\[ \text{str} = \text{str}(2:16) \]

or the eighth letter of a file name can be changed from 'A' to 'B' by

\[ \text{file}(8:8) = 'B' \]

Other similar operations are as simple. HASHIT was written using the substring capability shown and is much more compact than the functionally equivalent Flex code would be.
(3) As in any evolving program, array lengths changed with time and arrays were declared as IDENT(1) in subprograms; this forced use of the LCONST COMMON and variables for the constants from 0 to 30 to bypass the automatic subscript checking of FORTRAN.

(4) Ad hoc changes have crept into the code, and nesting has resulted in redundant or contradictory testing for certain conditions. For example, two consecutive blocks of code are conditioned on the same test in DBSTAT rather than merging the blocks and only testing once.

(5) The Flex loop construct appears frequently. The code

\[
\begin{align*}
I &= 0 \\
&\text{REPEAT UNTIL (I.EQ.N-1)} \\
&\quad A(I+1) = 0. \\
&\quad I = I + 1 \\
&\text{FIN}
\end{align*}
\]

results in FORTRAN code similar to

\[
\begin{align*}
I &= 0 \\
&\text{GO TO 99999} \\
99998 &\quad \text{IF}(I\text{.EQ.N-1}) \text{GO to 99997} \\
99999 &\quad A(I+1) = 0. \\
&\quad I = I + 1 \\
&\text{GO TO 99998} \\
99997 &\text{....}
\end{align*}
\]

rather than the F77 code of

\[
\begin{align*}
&\text{DO 10 } I = 1, N \\
&\quad A(I) = 0.
\end{align*}
\]

which will probably compiles to more compact and faster assembly language code.

(6) Some of the routines, such as EDTREL, have evolved to over 1000 lines. Even with the use of Flex macrofunctions, this is a larger amount of code based on the same global variables than the average programmer can grasp. Reduction of program unit size and more use of subprograms would simplify the code.

(7) Several functions implemented as Flex macros occur in multiple top-level routines. For example, both EDTREL and RETEVU (main menu options 4 and 8) ask for a column number. This is just a request for a number within a certain range, and could be implemented trivially as an integer FORTRAN FUNCTION in a library.

(8) Several routines, such as TFDEL, ADFDEL, and TDFDEL differ only in one location. A generic routine would reduce code space by nearly a factor of 3; the routine could even be called by the existing routines, each of which could be reduced from 24 to 5 lines.
Flowcharting revealed that several routines frequently appear in sequence. A new routine implementing the entire function would shorten the calling program and improve readability. For example, a new routine EXTEND could call EQUIVI, CATSTR, and FSLIDE to create an extended name from the specified base name and specified extension. This would cost one 6-line routine but reduce dozens of triple calls to single highly readable calls.

Given one machine, it was not obvious which routines were really machine dependent, and such functions occur in many places. For example, the read/write keys should be requested in a machine-dependent subroutine. Similarly, all questions could be asked through a subroutine QUEST which prints the test without a carriage return (if the machine allows such an option). A logical function QUESYN could call QUEST to provide the text and then call YESNO to demand a yes or no answer.

A complete top-down restructuring of the code is recommended. Guidelines would include (1) pure portable Fortran 77 code consistent with the standard, (2) copious use of comments in the code, (3) routines rarely longer than 2 pages including the imbedded comments (header comments do not count), (4) addition of new functions/routines whenever appropriate, and (5) restructuring of the libraries to minimize runtime size and to place machine-dependent code in as few files as possible. Implementation of guideline 1 alone would result in an estimated code reduction as shown in Table 4; the quantitative benefits of the other guidelines have been estimated.

Two additional recommendations are related to "user-friendliness". They are (6) any potentially dangerous or time-consuming activity should offer a 'punt' option, and (7) new algorithms should be considered for time-consuming activities. The code should also allow graceful exits from accidentally chosen paths, lest the user be stuck in an infinite question-answer loop that can only be exited by damaging the relation or by killing the program and losing all previous work. Similarly, any activity, such as sort, which could take hours to perform should offer a graceful exit after providing a time estimate; the user may prefer an unsorted list now rather than a sorted list in three hours (or even fifteen minutes). The rewrite of SORT was a start toward algorithm improvement, but room exists for more improvement.

Implementation of FEASIL on the VAX has greatly improved the usefulness of the system. Reduced execution time and more reliable data handling has greatly increased its acceptance by users. Continued refinement and elimination of any errors that may remain should guarantee user acceptance in the future.
TABLE 1 SUBPROGRAM CALLS OF FEASIL ROUTINES

<table>
<thead>
<tr>
<th>name</th>
<th>called by (*=directly, digit=indirectly)</th>
</tr>
</thead>
<tbody>
<tr>
<td>newrel</td>
<td>*! main menu routines (one routine per file)</td>
</tr>
<tr>
<td>delrel</td>
<td>.!</td>
</tr>
<tr>
<td>edtrel</td>
<td>...!</td>
</tr>
<tr>
<td>modcol</td>
<td>...!</td>
</tr>
<tr>
<td>merg rl</td>
<td>...!</td>
</tr>
<tr>
<td>rorgrl</td>
<td>*......!</td>
</tr>
<tr>
<td>retevd</td>
<td>*.......!</td>
</tr>
<tr>
<td>bkuprl</td>
<td>*........!</td>
</tr>
<tr>
<td>dbstat</td>
<td>*..........!</td>
</tr>
<tr>
<td>help</td>
<td>*...........!</td>
</tr>
<tr>
<td>rdoadf</td>
<td>1.....**..<em>......</em>!</td>
</tr>
<tr>
<td>rdotf</td>
<td>1.....<em>.**..</em>......*!</td>
</tr>
<tr>
<td>putifo</td>
<td>1.....<em>......</em>!</td>
</tr>
<tr>
<td>cpystr</td>
<td>1.....<em>......</em>!</td>
</tr>
<tr>
<td>rlf unc</td>
<td>1.............*!</td>
</tr>
<tr>
<td>display</td>
<td>1.............*!</td>
</tr>
<tr>
<td>prntrell</td>
<td>1.............*!</td>
</tr>
<tr>
<td>csortfl</td>
<td>1.............*!</td>
</tr>
<tr>
<td>asctyp</td>
<td>1.............*t</td>
</tr>
<tr>
<td>ascchr</td>
<td>1.............*t</td>
</tr>
<tr>
<td>sortfa</td>
<td>1.............*t</td>
</tr>
<tr>
<td>cpyrel</td>
<td>1.............*t</td>
</tr>
<tr>
<td>rlfunc</td>
<td>1.............*t</td>
</tr>
<tr>
<td>display</td>
<td>1.............*t</td>
</tr>
<tr>
<td>printrell</td>
<td>1.............*t</td>
</tr>
<tr>
<td>csortfl</td>
<td>1.............*t</td>
</tr>
<tr>
<td>asctyp</td>
<td>2.............*t</td>
</tr>
<tr>
<td>ascchr</td>
<td>2.............*t</td>
</tr>
<tr>
<td>sortfa</td>
<td>2.............*t</td>
</tr>
<tr>
<td>cpyrel</td>
<td>1.............*t</td>
</tr>
<tr>
<td>rlf unc</td>
<td>1.............*t</td>
</tr>
<tr>
<td>display</td>
<td>1.............*t</td>
</tr>
<tr>
<td>printrell</td>
<td>1.............*t</td>
</tr>
<tr>
<td>csortfl</td>
<td>1.............*t</td>
</tr>
<tr>
<td>asctyp</td>
<td>2.............*t</td>
</tr>
<tr>
<td>ascchr</td>
<td>2.............*t</td>
</tr>
<tr>
<td>sortfa</td>
<td>2.............*t</td>
</tr>
<tr>
<td>cpyrel</td>
<td>1.............*t</td>
</tr>
<tr>
<td>rlf unc</td>
<td>1.............*t</td>
</tr>
<tr>
<td>display</td>
<td>1.............*t</td>
</tr>
<tr>
<td>printrell</td>
<td>1.............*t</td>
</tr>
<tr>
<td>csortfl</td>
<td>1.............*t</td>
</tr>
<tr>
<td>asctyp</td>
<td>2.............*t</td>
</tr>
<tr>
<td>ascchr</td>
<td>2.............*t</td>
</tr>
<tr>
<td>sortfa</td>
<td>2.............*t</td>
</tr>
<tr>
<td>cpyrel</td>
<td>1.............*t</td>
</tr>
<tr>
<td>ft ip</td>
<td>1........<em>.**..</em>......*t format conversion routines</td>
</tr>
<tr>
<td>pit fp</td>
<td>1........<em>.**..</em>......***..*t</td>
</tr>
<tr>
<td>enf pta</td>
<td>1........<em>..</em>......*!</td>
</tr>
<tr>
<td>eninta</td>
<td>1........<em>..</em>......*!</td>
</tr>
<tr>
<td>utfar</td>
<td>1........<em>..</em>......*!</td>
</tr>
<tr>
<td>eleb</td>
<td>2........<em>.</em>......<em>!</em></td>
</tr>
<tr>
<td>name</td>
<td>1***** *****, ......... *! file identification and</td>
</tr>
<tr>
<td>hash it</td>
<td>1***** *****, ......... *! confirmation routines</td>
</tr>
<tr>
<td>fild es</td>
<td>1<em>1</em>** ***, ......... *! (fildes also called by reidf)</td>
</tr>
<tr>
<td>yesno</td>
<td>1<em>1</em>** ***, ......... *!</td>
</tr>
</tbody>
</table>

MAIN rorgrl rlfunc name
ret evd display hash it
newrel bkuprl printrel fild es
delrel dbstat csort fl yesno
edtrel help sort fl
modcol cpy rel
mergrl rdoadf
rdotf enf pta
pu. fo en in ta
cpy str ele b
<table>
<thead>
<tr>
<th>name</th>
<th>called by (*=directly, digit=indirectly)</th>
</tr>
</thead>
<tbody>
<tr>
<td>relfc</td>
<td>1 *... . . . . . . . . . . . . . . ! relation/tuple operations</td>
</tr>
<tr>
<td>relfd</td>
<td>1 . . . . . . . . . . . . . . !</td>
</tr>
<tr>
<td>addtup</td>
<td>1 . . . . !</td>
</tr>
<tr>
<td>deltup</td>
<td>1 . . . *...</td>
</tr>
<tr>
<td>colnam</td>
<td>1 . . ! . . . . . . . . . . . . . . !</td>
</tr>
<tr>
<td>adfcre</td>
<td>1 .11* . . . . . . . . . . . . . . !</td>
</tr>
<tr>
<td>tdfcre</td>
<td>1 . . . . . . . . . . . . . . !</td>
</tr>
<tr>
<td>tfcre</td>
<td>1 . . . . . . . . . . . . . . !</td>
</tr>
<tr>
<td>adfdel</td>
<td>1 *... . . . . !</td>
</tr>
<tr>
<td>tdfdel</td>
<td>1 .1111 .1111 . . . . . . . . . . !</td>
</tr>
<tr>
<td>tfdel</td>
<td>1 *... . . . . !</td>
</tr>
<tr>
<td>radfrc</td>
<td>1 .1* .1 .1111 . . . . . . . . !</td>
</tr>
<tr>
<td>rttvdf</td>
<td>1 . . . . . . . . . . . . . . !</td>
</tr>
<tr>
<td>rtfrc</td>
<td>1 . . . . . . . . . . . . . . !</td>
</tr>
<tr>
<td>padfrc</td>
<td>1 . . . . . . . . . . . . . . !</td>
</tr>
<tr>
<td>puttff</td>
<td>1 . *... . . . . . . . . . . . . . !</td>
</tr>
<tr>
<td>ptfrc</td>
<td>1 . . . . . . . . . . . . . . !</td>
</tr>
<tr>
<td>opnadf</td>
<td>2 1111 1111 . . . . . . . . . . . . . !</td>
</tr>
<tr>
<td>opntdf</td>
<td>2 1111 1111 . . . . . . . . . . . . . !</td>
</tr>
<tr>
<td>opntf</td>
<td>2 .1111 1111 . . . . . . . . . . . . . !</td>
</tr>
<tr>
<td>creafil</td>
<td>1 .22 .1 * .1 .11 . . . . . . . . !</td>
</tr>
<tr>
<td>filecls</td>
<td>1 .1111 .1111 . . . . . . . . . . . !</td>
</tr>
<tr>
<td>fileren</td>
<td>1 . . . . . . . . . . . . . . !</td>
</tr>
<tr>
<td>filerr</td>
<td>1 12113 1113 . . . . . . . . . . . !</td>
</tr>
<tr>
<td>emairy</td>
<td>1 .1* .1 .1111 .1111 . . . . . . . !</td>
</tr>
<tr>
<td>equivi</td>
<td>1 .1111 .1111 . . . . . . . . . . !</td>
</tr>
<tr>
<td>filedel</td>
<td>1 .1 . . . . . . . . . . . . . . !</td>
</tr>
<tr>
<td>fileopen</td>
<td>2 12222 1333 1112 . . . . . . . . . . !</td>
</tr>
</tbody>
</table>

**ADF/TDF/TF Operations**

**File Operations and Misc**

**Main**

```
rgrl     rlfnc     name     -cre     opn-
retriev  display  hashit  -del
newrel   bkuprl   printrel  fildes  radfrc
relrel   dbstat   csorf    yesno   rttvdf
delrel   help     sortfl   rtfrc   filedel
edrel    drmpgrl  cpyrel   relfd   rdotf
modcol   rdoafdf  enfpta   addtup  puttff
cpystr   eleb     colnam   ptfrc
```
### TABLE 1 SUBPROGRAM CALLS OF FEASIL ROUTINES (continued)

<table>
<thead>
<tr>
<th>name</th>
<th>called by (*=directly, digit=indirectly)</th>
</tr>
</thead>
<tbody>
<tr>
<td>getfp</td>
<td>1 * . . * . . . . . . . . . . . . . . . . ! keyboard input</td>
</tr>
<tr>
<td>reafp</td>
<td>1 * . . . . . . . . . . . . . . . . . . . *!</td>
</tr>
<tr>
<td>getnum</td>
<td>1 * . . . . . . . . . . . . . . . . . . . . !</td>
</tr>
</tbody>
</table>
| reanum| 1 . . . . . . . . . . . . . . . . . . . . . | *
| getch | 1 * . . . . . . . . . . . . . . . . . . . . . . ! |
| get  | 1 * . . . . . . . . . . . . . . . . . . . . . . . ! |

**Note 1:** The symbol to the right of each line indicates the type of routine:
- (!) terminal routine (calls no others)
- (m) machine-dependent (only system calls or assembly language)

**Note 2:** Some routines are included in the source, but not called:
- catnum (calls catstr, putnum)
- closef
- filrenb
- hash
- modap
- newno
- opnrel
- put (calls catsub, cpysub, cpysub, putch, putnum)
- putch
- putnum (calls putch)
- secure (calls getnum)(called by fildes on the P-E)

---

MAIN rorgrl rlfuc func name -cre opn- streq
treted dplay hashit -del strlen
newrel bkuprl printrel fildes filedel cpysub
delrel dbstat csorf yesno radfrc catsstr
edrel help sortfl rtvtdf getfp catsub
modcol cpyst relfc rtfrc reafp catsub
mergl rdoadf Psrfl relfd padfrc getnum chtyp
rdof efnta addtup putdf reanum
putio eninta deltup ptfrc getch
cpyst eleb colnam get
<table>
<thead>
<tr>
<th>name</th>
<th>file (location)</th>
<th>name</th>
<th>file (location)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAIN</td>
<td>daba08</td>
<td>mergerl</td>
<td>mergerel</td>
</tr>
<tr>
<td>blockdata</td>
<td>daba08</td>
<td>modap</td>
<td>f7io (unused)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>modcol</td>
<td>modcolu</td>
</tr>
<tr>
<td>addtup</td>
<td>tplib2</td>
<td>name</td>
<td>tpfilecr</td>
</tr>
<tr>
<td>adfcre</td>
<td>tpfilecr</td>
<td>opnadf</td>
<td>tpfilecr</td>
</tr>
<tr>
<td>adfdel</td>
<td>tpfilecr</td>
<td>opntdf</td>
<td>tpfilecr</td>
</tr>
<tr>
<td>ascchr</td>
<td>sort2</td>
<td>opntf</td>
<td>tpfilecr</td>
</tr>
<tr>
<td>ascyp</td>
<td>sort2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>bkuprl</td>
<td>backuprl</td>
<td></td>
<td></td>
</tr>
<tr>
<td>catsr</td>
<td>sc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>catsub</td>
<td>sc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>colnam</td>
<td>tplib2</td>
<td>padfrc</td>
<td>tplib2</td>
</tr>
<tr>
<td>cpyre1</td>
<td>tplib3</td>
<td>printrel</td>
<td>printrel</td>
</tr>
<tr>
<td>cpystr</td>
<td>sc</td>
<td>ptfrc</td>
<td>tplib2</td>
</tr>
<tr>
<td>cpysub</td>
<td>sc</td>
<td>putifo</td>
<td>tplib2</td>
</tr>
<tr>
<td>creafil</td>
<td>f7io</td>
<td>puttdf</td>
<td>tplib2</td>
</tr>
<tr>
<td>csortf</td>
<td>sort2</td>
<td>radfrc</td>
<td>tplib2</td>
</tr>
<tr>
<td>dbstat</td>
<td>dabaexit</td>
<td>rdoadf</td>
<td>tplib3</td>
</tr>
<tr>
<td>delrel</td>
<td>delerel</td>
<td>rdotf</td>
<td>tplib3</td>
</tr>
<tr>
<td>deltup</td>
<td>tplib2</td>
<td>reafp</td>
<td>tplib1</td>
</tr>
<tr>
<td>display</td>
<td>display</td>
<td>reanum</td>
<td>tplib1</td>
</tr>
<tr>
<td>edtre1</td>
<td>editre17</td>
<td>relfc</td>
<td>tpfilecr</td>
</tr>
<tr>
<td>eleb</td>
<td>tplib1</td>
<td>relfd</td>
<td>tpfilecr</td>
</tr>
<tr>
<td>emiary</td>
<td>tpfilecr</td>
<td>retevd</td>
<td>retreiv7</td>
</tr>
<tr>
<td>enfpta</td>
<td>tplib1</td>
<td>rlfunc</td>
<td>function</td>
</tr>
<tr>
<td>eninta</td>
<td>tplib1</td>
<td>rogr1</td>
<td>reorgrel</td>
</tr>
<tr>
<td>equivi</td>
<td>tpfilecr</td>
<td>rtfrc</td>
<td>tplib2</td>
</tr>
<tr>
<td>fildes</td>
<td>tpfilecr</td>
<td>rtvtdf</td>
<td>tplib2</td>
</tr>
<tr>
<td>filecls</td>
<td>f7io</td>
<td>secure</td>
<td>tpfilecr (unused)</td>
</tr>
<tr>
<td>filedel</td>
<td>f7io</td>
<td>sortfa</td>
<td>sort2</td>
</tr>
<tr>
<td>fileopn</td>
<td>f7io</td>
<td>sortfl</td>
<td>sort2</td>
</tr>
<tr>
<td>fileren</td>
<td>f7io</td>
<td>strlen</td>
<td>tplib1</td>
</tr>
<tr>
<td>filerr</td>
<td>tpfilecr</td>
<td>tdfcre</td>
<td>tpfilecr</td>
</tr>
<tr>
<td>fslide</td>
<td>tplib3</td>
<td>tfdel</td>
<td>tpfilecr</td>
</tr>
<tr>
<td>ftipi</td>
<td>tplib1</td>
<td>tfcre</td>
<td>tpfilecr</td>
</tr>
<tr>
<td>getfp</td>
<td>tplib1</td>
<td>tfdel</td>
<td>tpfilecr</td>
</tr>
<tr>
<td>getnum</td>
<td>tplib1</td>
<td>utfar</td>
<td>tplib2</td>
</tr>
<tr>
<td>hash</td>
<td>tplib2</td>
<td>yesno</td>
<td>tplib1</td>
</tr>
<tr>
<td>help</td>
<td>help</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note: Library SC is part of Flex (alternate spelling Flecs); we cannot alter the code for these routines.
<table>
<thead>
<tr>
<th>FILE CONTENTS FOR FEASIL ROUTINES</th>
</tr>
</thead>
<tbody>
<tr>
<td>file</td>
</tr>
<tr>
<td>------</td>
</tr>
<tr>
<td>backuprl(P-E)</td>
</tr>
<tr>
<td>bkupstub(VAX)</td>
</tr>
<tr>
<td>daba08</td>
</tr>
<tr>
<td>dabaexit</td>
</tr>
<tr>
<td>deletrel</td>
</tr>
<tr>
<td>dsplay</td>
</tr>
<tr>
<td>editrel7</td>
</tr>
<tr>
<td>f7io</td>
</tr>
<tr>
<td>function</td>
</tr>
<tr>
<td>help</td>
</tr>
<tr>
<td>mergerel</td>
</tr>
<tr>
<td>modcolumn</td>
</tr>
<tr>
<td>newrel+</td>
</tr>
<tr>
<td>printrel</td>
</tr>
<tr>
<td>retreiv7</td>
</tr>
<tr>
<td>reorgrel</td>
</tr>
<tr>
<td>sc</td>
</tr>
<tr>
<td>sort2</td>
</tr>
</tbody>
</table>
TABLE 3 FILE CONTENTS FOR FEASIL ROUTINES (continued)

<table>
<thead>
<tr>
<th>file</th>
<th>contents (subprograms)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>tpfilercr</td>
<td>adfcrc relfc opnadf emiary fildes</td>
</tr>
<tr>
<td></td>
<td>adfdel relfd opntf equivi name</td>
</tr>
<tr>
<td></td>
<td>tdfcre opntdf</td>
</tr>
<tr>
<td></td>
<td>tdfdel filerr</td>
</tr>
<tr>
<td></td>
<td>tfcre tfdel</td>
</tr>
<tr>
<td></td>
<td>tpfilercr adfcrc relfc opnadf emiary fildes</td>
</tr>
<tr>
<td></td>
<td>adfdel relfd opntf equivi name</td>
</tr>
<tr>
<td></td>
<td>tdfcre opntdf</td>
</tr>
<tr>
<td></td>
<td>tdfdel filerr</td>
</tr>
<tr>
<td></td>
<td>tfcre tfdel</td>
</tr>
<tr>
<td></td>
<td>tpfilercr adfcrc relfc opnadf emiary fildes</td>
</tr>
<tr>
<td></td>
<td>adfdel relfd opntf equivi name</td>
</tr>
<tr>
<td></td>
<td>tdfcre opntdf</td>
</tr>
<tr>
<td></td>
<td>tdfdel filerr</td>
</tr>
<tr>
<td></td>
<td>tfcre tfdel</td>
</tr>
<tr>
<td></td>
<td>tpfilercr adfcrc relfc opnadf emiary fildes</td>
</tr>
<tr>
<td></td>
<td>adfdel relfd opntf equivi name</td>
</tr>
<tr>
<td></td>
<td>tdfcre opntdf</td>
</tr>
<tr>
<td></td>
<td>tdfdel filerr</td>
</tr>
<tr>
<td></td>
<td>tfcre tfdel</td>
</tr>
<tr>
<td></td>
<td>tpfilercr adfcrc relfc opnadf emiary fildes</td>
</tr>
<tr>
<td></td>
<td>adfdel relfd opntf equivi name</td>
</tr>
<tr>
<td></td>
<td>tdfcre opntdf</td>
</tr>
<tr>
<td></td>
<td>tdfdel filerr</td>
</tr>
<tr>
<td></td>
<td>tfcre tfdel</td>
</tr>
<tr>
<td></td>
<td>tpfilercr adfcrc relfc opnadf emiary fildes</td>
</tr>
<tr>
<td></td>
<td>adfdel relfd opntf equivi name</td>
</tr>
<tr>
<td></td>
<td>tdfcre opntdf</td>
</tr>
<tr>
<td></td>
<td>tdfdel filerr</td>
</tr>
<tr>
<td></td>
<td>tfcre tfdel</td>
</tr>
<tr>
<td></td>
<td>tpfilercr adfcrc relfc opnadf emiary fildes</td>
</tr>
<tr>
<td></td>
<td>adfdel relfd opntf equivi name</td>
</tr>
<tr>
<td></td>
<td>tdfcre opntdf</td>
</tr>
<tr>
<td></td>
<td>tdfdel filerr</td>
</tr>
<tr>
<td></td>
<td>tfcre tfdel</td>
</tr>
<tr>
<td></td>
<td>tpfilercr adfcrc relfc opnadf emiary fildes</td>
</tr>
<tr>
<td></td>
<td>adfdel relfd opntf equivi name</td>
</tr>
<tr>
<td></td>
<td>tdfcre opntdf</td>
</tr>
<tr>
<td></td>
<td>tdfdel filerr</td>
</tr>
<tr>
<td></td>
<td>tfcre tfdel</td>
</tr>
<tr>
<td></td>
<td>tpfilercr adfcrc relfc opnadf emiary fildes</td>
</tr>
<tr>
<td></td>
<td>adfdel relfd opntf equivi name</td>
</tr>
<tr>
<td></td>
<td>tdfcre opntdf</td>
</tr>
<tr>
<td></td>
<td>tdfdel filerr</td>
</tr>
<tr>
<td></td>
<td>tfcre tfdel</td>
</tr>
</tbody>
</table>

21
<table>
<thead>
<tr>
<th>file</th>
<th>subprogram</th>
<th>lines now</th>
<th>estimated lines in F77</th>
<th>%change</th>
</tr>
</thead>
<tbody>
<tr>
<td>daba08</td>
<td>MAIN</td>
<td>159 87</td>
<td>84</td>
<td>3-</td>
</tr>
<tr>
<td></td>
<td>block data</td>
<td>96 70</td>
<td>62</td>
<td>11-</td>
</tr>
<tr>
<td>dabaexit</td>
<td>dbstat</td>
<td>330 218</td>
<td>203</td>
<td>6-</td>
</tr>
<tr>
<td>deletrel</td>
<td>delrel</td>
<td>46 27</td>
<td>24</td>
<td>10-</td>
</tr>
<tr>
<td>display</td>
<td>display</td>
<td>685 505</td>
<td>450</td>
<td>10-</td>
</tr>
<tr>
<td>editrel7</td>
<td>edtre1</td>
<td>2213 1298</td>
<td>1147</td>
<td>12-</td>
</tr>
<tr>
<td>f7io</td>
<td>creafil</td>
<td>53 29</td>
<td>28</td>
<td></td>
</tr>
<tr>
<td></td>
<td>filecls</td>
<td>12 4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>filedel</td>
<td>33 15</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td></td>
<td>fileopn</td>
<td>52 20</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td></td>
<td>fileren</td>
<td>43 24</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td></td>
<td>modap</td>
<td>4 3</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>vaxerr</td>
<td>169 161</td>
<td>161</td>
<td></td>
</tr>
<tr>
<td></td>
<td>total</td>
<td>377 253</td>
<td>249</td>
<td>1-</td>
</tr>
<tr>
<td>function</td>
<td>rlfnc</td>
<td>310 197</td>
<td>157</td>
<td>21-</td>
</tr>
<tr>
<td>modcolum</td>
<td>modcol</td>
<td>607 306</td>
<td>234</td>
<td>24-</td>
</tr>
<tr>
<td>newrelat</td>
<td>newrel</td>
<td>469 264</td>
<td>230</td>
<td>13-</td>
</tr>
<tr>
<td>printrrel</td>
<td>printrcl</td>
<td>1048 725</td>
<td>629</td>
<td>13-</td>
</tr>
<tr>
<td>retreiv7</td>
<td>retevd</td>
<td>1780 1007</td>
<td>891</td>
<td>12-</td>
</tr>
<tr>
<td>sort2</td>
<td>asctyp</td>
<td>28 11</td>
<td>10</td>
<td>10-</td>
</tr>
<tr>
<td></td>
<td>ascchr</td>
<td>34 16</td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>csorf</td>
<td>180 106</td>
<td>92</td>
<td>13-</td>
</tr>
<tr>
<td></td>
<td>sortfa</td>
<td>165 100</td>
<td>94</td>
<td>4-</td>
</tr>
<tr>
<td></td>
<td>sortfl</td>
<td>448 260</td>
<td>250</td>
<td>4-</td>
</tr>
<tr>
<td>tpfilecr</td>
<td>adfcre</td>
<td>32 12</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td></td>
<td>adfdel</td>
<td>40 19</td>
<td>22</td>
<td></td>
</tr>
<tr>
<td></td>
<td>emlary</td>
<td>13 5</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>equivi</td>
<td>15 7</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td></td>
<td>fildes</td>
<td>89 53</td>
<td>48</td>
<td></td>
</tr>
<tr>
<td></td>
<td>filerrr</td>
<td>52 47</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td></td>
<td>name</td>
<td>64 31</td>
<td>23</td>
<td></td>
</tr>
<tr>
<td></td>
<td>opnadf</td>
<td>53 20</td>
<td>23</td>
<td></td>
</tr>
<tr>
<td></td>
<td>opntf</td>
<td>53 20</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>opntldf</td>
<td>53 20</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>relfc</td>
<td>62 38</td>
<td>33</td>
<td></td>
</tr>
<tr>
<td></td>
<td>relfd</td>
<td>59 23</td>
<td>22</td>
<td></td>
</tr>
<tr>
<td>file</td>
<td>subprogram</td>
<td>lines now</td>
<td>estimated</td>
<td>Zchange</td>
</tr>
<tr>
<td>-------</td>
<td>------------</td>
<td>-----------</td>
<td>------------</td>
<td>---------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>total</td>
<td>code</td>
<td>lines in F77</td>
</tr>
<tr>
<td>secure</td>
<td></td>
<td>34</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>tdfcre</td>
<td></td>
<td>32</td>
<td>12</td>
<td>5</td>
</tr>
<tr>
<td>tdfdel.</td>
<td></td>
<td>40</td>
<td>19</td>
<td>5</td>
</tr>
<tr>
<td>tfcre</td>
<td></td>
<td>32</td>
<td>12</td>
<td>5</td>
</tr>
<tr>
<td>tfdel</td>
<td></td>
<td>40</td>
<td>19</td>
<td>5</td>
</tr>
<tr>
<td>total</td>
<td></td>
<td>802</td>
<td>389</td>
<td>268</td>
</tr>
</tbody>
</table>

| tplib1 | eleb      | 46        | 28         | 14      |        |
|        | enfpta    | 53        | 32         | 27      |        |
|        | eninta    | 26        | 11         | 10      |        |
|        | ftipi     | 15        | 6          | 6       |        |
|        | getfp     | 35        | 14         | 14      |        |
|        | getnum    | 29        | 8          | 8       |        |
|        | pitfp     | 13        | 6          | 6       |        |
|        | reafp     | 225       | 144        | 20      |        |
|        | reanum    | 156       | 89         | 20      |        |
|        | strlen    | 78        | 37         | 21      |        |
|        | yesno     | 59        | 29         | 24      |        |
| total  |           | 774       | 404        | 170     | 58-     |

| tplib2 | addtrup   | 59        | 20         | 14      |        |
|        | colnam    | 24        | 6          | 6       |        |
|        | deltrup   | 47        | 18         | 12      |        |
|        | hashtie   | 61        | 43         | 43      |        |
|        | padfrc    | 103       | 67         | 66      |        |
|        | ptfrc     | 35        | 11         | 11      |        |
|        | putifo    | 83        | 43         | 37      |        |
|        | puttdf    | 40        | 21         | 17      |        |
|        | radfrc    | 56        | 25         | 22      |        |
|        | rtfrc     | 46        | 17         | 16      |        |
|        | rtvtdf    | 38        | 18         | 14      |        |
|        | utfar     | 47        | 22         | 21      |        |
| total  |           | 668       | 311        | 279     | 10-     |

| tplib3 | cpyrel    | 108       | 68         | 59      |        |
|        | fslide    | 26        | 8          | 8       |        |
|        | rdoaif    | 135       | 87         | 76      |        |
|        | rdotf     | 139       | 83         | 76      |        |
| total  |           | 426       | 246        | 219     | 11-     |

**OVERALL TOTALS**

- Total lines including comments: 10,845 (38% comments)
- Total lines code: 6,800
- Estimated lines if FORTRAN 77: 5,758
- Estimated code reduction: 1,042
- Estimated code reduction percentage: 15%

23/(24 blank)
REFERENCES


APPENDIX A

MAIN
MAIN p.1
Calls the other functions.

Block Data
Constants, parameters, defaults

Set array sizes, init menu loop

IQUIT 1
Mac (Fail)

No

Print menu and prompt

Set default

Read client

IANS

1-1-1

Set qualifying IANS, fail, no

NEWREL
Create new relation

DELREL
Delete a relation

1-2

Prompt and await <CR>

Prompt and await <CR>

A-1
Subroutine: RETRIEVE

This subroutine allows the user to output or manipulate relations. In outputting a relation the user may:
1. Print data in a relation
2. Retrieve data
3. Create a relation from data
4. Plot data to screen, plotter, or printer

In manipulating a relation the user may:
1. Sum columns
2. Average columns
3. Calculate the mean and standard deviation
START

SORT = False
ACTIVE = False
AND = False
OK = False
COMP = False

LUTF = 11
LIADF = 12
LUSAV = 13
LUTEMP = 14
LIACF = 10

"Retrieve, manipulate, ppr, plot relation. Name of relation to be manipulated."

NAME
Hashit
FILES
Get name and process

EMARY, EQUIV.
CATSTR, FSLIDE
Build names for
ACT, SFI, S2

LIVINF
Load TDF data

Were there any errors?
yes
\[ \text{RETURN} \]
no

Is the relation empty?
yes
\[ \text{Relation empty. No manipulation possible.} \]
no

\[ I = 0 \]
\[ \text{RETURN} \]

(page 2)
1 (page 1)

Increment 1

Set Up Tempfile Name

CPYREL Copy TF and ADF to tempfiles

Were there any errors?

Set Up Tempfile Name

MDPDEL Delete temp ADF

Set Up Tempfile Name

TRKDEL Delete temp TF

Attempting recovery...

Set Up Tempfile Name

Are there no errors or have there been too tries?

2 (page 1)

RETEVD
RETEVD

PAGE 3

2 (page 2)

Were there any errors?

yes

'Recovery not possible.'

no

RETURN

Initialize Relation Activity File

"Ready for manipulation"

RETURN

Has the exit flag DONE been set to true?

yes

GET,STRLEN CP/SUM
Get command letter CKACT

RETURN

A (page 4)

B (page 4)

B-4
What is CMCR? (command letter) (conditional)

- I Quit
- List Column Names
- Copy Relation
- Check If Relation Empty
  - OK? Yes: PRINTREL Print
  - OK? No: Sort A Relation
- Check If Relation Empty
  - OK? Yes: Move Active Records To New Relation
  - OK? No: And Or Or To Active File
- Check If Relation Empty
  - OK? Yes: And Or Or To Active File
- Check If Relation Empty
  - OK? Yes: Initialize Relation Activity File
- Check If Relation Empty
  - OK? Yes: RLFUNC Math functions
- DISPLAY Display Functions

? or H: "The legal commands are:"
  (display menu)
other: Idle And
B: Nothing
RETEVDO
SET UP TEMPFILE
NAME
PAGE 5

START

ENDARY:
COPY
Copy the first
7 letters of
the filename

CATSTR
Append "B:"

BACK

CHECK IF RELATION EMPTY

START

Is IDENT(L8) greater than 0?

no

OK = true

Is IDENT(L8) = 0?

yes

OK = false

"Relation empty."

no

BACK

BACK
RETEVD
SORT A RELATION

PAGE 6

START

"Column on which to sort?"

Get Column Number (NUM)

IC=CNUM

is the sort column number = 0?

yes

no

CSORT
Create a new sort file.

Has ISORT been set to true?

no

yes

FILEDEL
Delete any old sort files.

ISORT = false

OK = true

"Ascending or descending sort (A or D)?"

GET, STRIP, DATA
Get the choice (CRACT)

ISORT = true

I

(page 7)

A (page 7)

B (page 7)
What was the action command? (EXACT) (conditional)

YES = false

A

YES = true

otherwise

OK = false

Is OK true?

no

yes

SORTFL
sort the column

"Sort on column complete."

BACK
START

APIDEL,IPIDEL
Delete scratch files.

Has a sort occurred?

yes

FILEDEL
Delete sort file.

no

Has an activity file .ACT been created?

yes

FILEDEL
Delete .ACT

no

DONE = true

BACK

---

RETEVD
I QUIT
PAGE 8

---

START

DONE = false

"Type 'H' for help"

BACK

---

IDLE AND
DO NOTHING

---
RETEVD
COPY RELATION
PAGE 10

START

"Copied relation name:"'

GETNAME, HASHT
Get the relation name and hash it.

CATTIN
Append ',' to the end of the relation name.

CPYREL
Copy the TF and the ADF

Were there any errors?

yes

no

TFDFCRE
Create TDF

Were there any errors?

yes

TFDF copy file creation error

no

PUTDF
Save the TDF

Did both the TF ADF copy and the TDF creation work?

"Relation reproduced."

"Nothing done."

BACK
GET COLUMN NUMBER

START

OK = true

RETNUM
Get the number

Is the requested column number negative or greater than the maximum column number?

yes

"Range error Must be 1-

no

Is OK = true?

yes

OK = false

BACK
START

ACTERR = false

Were there any errors on opening? (ACTERR)

no

yes

"Error in opening activity file"

ACTERR = true

BACK
RETEVD
INITIALIZE
RELATION
ACTIVITY
FILE

START

Is the relation empty?

yes

no

Has an activity file .ACT been created?

yes

no

OK = false

CREAFILE
Create an activity file.

Were there any errors during .ACT creation?

yes

no

OK = true

SKIP = false

Is the error code 314 or 999?

yes

no

"Unknown error during create."

"Activity file present already. Delete it and try again."

SKIP = true

ACTIVE = false

OK = true

YESNO
GET the yes or no answer to OK

Was the answer 'yes'?

yes

no

SKIP = true

OK = true

OK = false

"Nothing done."

(page 16) (page 16) (page 16) (page 16)

A B C I D
RETEVD
INITIALIZE RELATION ACTIVITY FILE
PAGE : 6

A (page 15)  B (page 15)  C (page 15)  D (page 15)

I

OK = false

FILEDEL
Delete the old activity file .ACT

Were there any errors deleting the file?

yes

no

"Error deleting activity file."

ACTIVE = false
SKIP = true
OK = true

Has OK been set to true or have there been no errors?

yes

no

Has a new ACT been created?

yes

ACTIVE = true

no

Is SKIP true?

yes

no

FILEOPN
Open the .ACT file.
Check Activity File Opening

Was the .ACT opened successfully?

yes

no

'Nothing Done.'

NUMACT = 0

Have all records been examined?

yes

no

Increment Tuple count

Activate Record

FILECLS close .ACT

BACK
RETEVD
AND ACCEPTABLE
RECORDS TO
ACTIVITY FILE

Is the strategy real or integer or is the character data equal?

NR = 0
J = IDENT(13)+1
IDI = IDF(4,10)

FILEOPEN
Open the .ACT

Check Activity File Opening

Were there any errors in file open?

"Nothing done."

Have all the records been examined?

FILECLS
Close the .ACT

Read a record into ITUP, EDF

Is this the end of file?

RTFRC
Get the tuple

Check Found Or Not Found
RETEVD AND ACCEPTABLE RECORDS TO ACTIVITY FILE

PAGE 19

Is FOUND = true?

no

INREC -MR

Deactivate Record

Sort On Column IOC

Find String Or Single Character GT LT criteria

Is FOUND = true?

no

FILEDOPN Open .A07

Check Activity File Opening

Are there any file open errors?

yes

no

"Nothing Done"

FILEDOPN Open .SF2

B-20
Page 19:

RETRIEVE

AND ACCEPTABLE RECORDS TO ACTIVITY

FILE

Have all the records been examined?

yes

Read sort record into IREC

Decrement IREC

Read .ACT record into ITUP, EOF

Is this the end of file?

no

no

Yes

Deactivate Record

FILECLS
Close .ACT and .SF2

FILEDEL
Delete the sort file .SF2

BACK
REIEVD
SORT ON
COLUMN ICC

START

CSORTF
Set up second sort file .SF2

Was an ascending sort specified? (GT)

yes

SKIDIR = false

no

SKIDIR = true

SORTFL
Sort the column

BACK
Is the strategy real or integer or is the character data equal?
REJECT OR ACCEPTABLE RECORDS TO ACTIVITY FILE

PAGE 23

Is found = true?

IREC = NR

Activate Record

Sort on column IOC

Find String or Single Character GT LT Criteria

Is found = true?

FILEOPEN
Open the .ACT

Check Activity File Opening

Were there any errors in opening the file?

Yes

No

(page 24) (page 24)
Retrieve or Acceptable Records to Activity File

Page 24

FILEOPN
Open the sort file .SF2

Have all the records been examined?

no

yes

Read INREC in from the sort file(SF2)

Increment to the next record in INREC

Rewind (TOP,EOF) in from the .ACT file

Is this the end of file?

yes

no

Activate Records

FILECLS
Close the sort file .SF2

(p. 23)

(page 25)
RETEVD
OR ACCEPTABLE
RECORDS TO
ACTIVITY FILE
PAGE 25

3 (p. 24)

Is FOUND = true?

no

yes

FILECLS
Close the sort file .SF2

FILEDEL
Delete the sort file .SF2

'Orign on column complete.'

BACK
GET THE CHARACTER AND PUT INTO OK = TRUE

IS ILK LESS THAN OR EQUAL TO IUC114?

4 IReEd a record

Z Tnr.o

mMH'Y,

Is ILK less than or equal to • - / zeto mod eq at rto ITEST ?

no - 

- 

is ILK less than or equal to ?

CRITERIA

Find String

OR SINGLE CHARACTER

CHARACTER

P.26

PAGE 27
HE (page 27)

FIND STRING OR SINGLE CHARACTER GT LT CRITERIA

PAGE 28 (p.27) B

Is there string strategy?

no

yes

IREC = 0

Get String From Sort File

Is YES = true?

no

yes

Is IADFRC(1) greater than or equal to FORMAT(1)?

no

yes

STREQ Compare IADFRC with FORMAT

Are they the same?

no

yes

FOUND = true
ILK = IREC-1

Is IREC = 0?

no

yes

FOUND = false
ILK = 0

Is CKACT the same as FORMAT?

no

yes

FOUND = true
ILK = IREC-1

Is IREC = 0?

no

yes

FOUND = false
ILK = 0

(p.29) B

B-29
Increment INDEX

Is \texttt{FOUND} = \text{true} or \texttt{INDEX} = \text{IDENT(5)}

\texttt{FILECLS}
Close the sort file .SF2

BACK
RETEVD

MATCHED TYPE

START

Is ICK less than ITEST?

yes

no

Move Up

Is ICK greater than ITEST?

no

yes

Move Down

BACK

MIS MATCH

START

Is ICK less than ITEST?

yes

no

Move Down

Is ISK greater than ITEST?

no

yes

Move Up

BACK

B-31
START

ILK = IDENT(8) / 2
ILKUP = IDENT(8)
ILKOM = 0

BACK

START SEARCH
PAGE 31

MOVE UP

ILKOM = ILK
I = ILKUP - ILK
ILK = ILK + 1/2

Is I an odd number?

no

yes

ILK = ILK + 1

Is ILK greater than IDENT(8)?

NO

yes

ILK = IDENT(8)

BACK

B-32
START

ILKUP = ILK
I = ILK - ILKON
ILK = ILK - 1/2

Is I an odd number?

no

yes

ILK = ILK - 1

BACK

Is ILK greater than IDENT(8)?

no

yes

ILK = IDENT(8)

BACK

START

Read record from sort file into ITUP, IAPFNC

NR = IADRRC(1)/6

Read the record into ITUP, IAPFNC

BACK

GET STRING FROM SORT FILE
RETEVD
CHECK FOR SUBSTRING MATCH PAGE 33

START

Is IADFRC(1) = FORMAT(1) yes

no

Is IADFRC(1) greater than FORMAT(1)? no

K = 0

Increment K

CPI/SUB Copy first FORMAT(1) characters from IADFRC to CRACT

STREQ Compare FORMAT and CRACT

Is FORMAT = CRACT? no

yes

FOUND = true

Is FOUND true or K FORMAT(1) greater than IADFRC? no

yes

BACK
Is there string or character strategy?

- Yes: Copy K words of HADDRC into FORMAT
- No: 
  - Is the strategy real and EQ not true?
    - Yes: ITEST=ITFRC(IREC)
    - No: 
      - PUTFP: Convert ITFRC(REC) into XTEST
  - BACK

START

ACTIVATE
RECORD

START

HUMACT=HUMACT+1
EDF=true

Write ITUP.EDF to the record IREC+1

BACK
RETEVD
DEACTIVATE
RECORD

START

NUMACT=NUMACT-1
EOF = False

Write ITUP.EOF to record IREC+1

BACK

PAGE 35
GET SELECTION CRITERIA

OK = false

GET, STRLEN
Get character into ACTION

GT = false
LT = false
EQ = false

What is in the ACTION array?
(conditional)

= GTOK
  GT = true

= LTOK
  LT = true

= NOT
  EQ = true

OK\&GT or LT or EQ
OK is set to true if any of the conditions are true.

Is OK equal to true?

yes
"Value:__"

Get Value Or String

BACK
START
IREC=ITDF(4, ICC)
Location in the TF record

Is the strategy real?

no

yes

GETFP
Get real value X, LEN
characters long

Is the length = 0?

no

FTIPI
Convert X into ITFRC(Irec)

yes

ITFRC(IREC)=
NULL(2)

Is there integer strategy?

no

yes

GETNUM
Get integer value ITFRC()

Were there any errors?

yes

Is LEN = 0?

no

no

yes

ITFRC(IREC)=
NULL(2)

I (p.39)
(p.39) A

B-39
Is there character strategy?

- No
- Yes
  - Is the LEN greater than 1?
    - No
      - K = 2
    - Yes
      - "Single character only"
      - OK = false

Is there string strategy?

- No
- Yes
  - LEN-IADFRC(1)
  - K = LEN/6
  - "Tuple editing error! See programmer"
  - OK = false

Is OK = true?

- No
- Yes

BACK
Is there string strategy ITFRC?

no

Is FORMAT(1) less than 8?

no

UTFAR
Convert ITFRC into IADFRC

no

Is YES = true?

Check For Substring Match

Is IADFRC = FORMAT?

yes

FOUND = true

is there string strategy and ITFRC(ID2) greater than or equal to FORMAT(1) ?

no

K=ITFRC(ID2)/4

Is YES = true?

Check For Substring Match

is there integer strategy and GT?

no

Is IADFRC greater than ITTEST?

yes

FOUND = true

no

p.42
1 (p.41)

Is the strategy real and CT?

- yes: PITPP Convert ITDFRC(IDI) to X
  - no
    - yes: PITPP Convert ITDFRC(IDI) to X
      - no
    - yes: Is X greater than XTEST?
      - no: FOUND = true
      - yes: Is X less than XTEST?
        - yes: FOUND = true
        - no:
          - yes: Is the strategy real and LT?
            - yes: IS the strategy real and LT?
              - yes: Compare CKACT and FORMAT equal?
                - yes: BACK
                - no: FOUND = true
            - no: Is the strategy not string and IDI?
              - yes: Are CKACT and FORMAT equal?
                - yes: BACK
                - no: FOUND = true
              - no:
                - yes: Is there integer strategy and LT?
                  - yes: Is ITFRC(IDI) less than ITEST?
                    - yes: FOUND = true
                    - no:
                      - yes: Is the strategy real and LT?
                        - yes: PITPP Convert ITDFRC(IDI) to X
                          - no
                        - no
                          - yes
                          - no
                      - no
                      - yes
                      - no
                - no
      - yes
      - no:
        - yes
        - no

2 (p.41)

CHECK FOUND
CR NOT FOUND

PAGE 42

B-43
RETLVD
MOVE ACTIVE RECORDS TO NEW RELATION

START

ENTIRE, EQUIV.
Copy RELDES to ORGAN and save.

"Relation to move records to."

NAME, HAMLET
Get the relation name.

Is ACTIVE = true
and NUMACT greater
than 0?

no

K1=longmax(1)/6

yes

INSAVE-INDEX
INSAVE-INDEX

FILEDES
Get volume and protection keys

INMOVE-INDEX
INMOVE-INDEX

ENQUIRE
Empty the IDENT array

Fill IDENT with zeroes

IDENT (3)=IDENT (3)
IDENT (6)=IDENT (6)
IDENT (9)=IDENT (9)

n=12-45
IDENT = IDENT

1 (page 44)

A (page 44)
**MOVE ACTIVE RECORDS TO NEW RELATION**

1. \( n \) to 13
   - \( IDENT_1(n+3) = \text{LONGRAM}(n) \)

   **TFDF**
   - Create new TF and TDF

   **NZADF=NMAXC+IDENT(15)+IDENT(9)**

   **ADDF**
   - Create the ADF

   **IDENT(5)=NZADF**
   - **IDENT(7)=NMAXC**

   **Were there any errors?**
   - \( 'File open error. Records not moved.' \)
   - **no**

   **'Erase records after moving? (yes/ no).**

   **YESNO**
   - Get the yes or no answer

   **Have all the records been examined?**
   - \( \text{yes} \)
   - **no**

   \( (1=1 \text{ to } 5) \)
   - \( \text{IDOF}(1,10C)=\text{IDOF}(1,10C) \)

   **IRZEX=IRSAVE**
   - \( \text{IRF}(\text{IRSAVE} / n=\text{INOF}(3,10C) / 6) \)

   **C** (p.45)
   - **2** (p.45)
   - **E** (p.45)
REVIEW
MOVE ACTIVE RECORDS TO NEW RELATION

C (page 44)

E A

C

PAGE 45 (p.44)

2

(p.44)

COLUM
Get the column name

INKEY = INKEY
INKEY = INKEY
OVR = true

PADFRC
Save the ADF

ITDP(2,IIC) = IDENT(1)
IDENT(1) = IDENT(1) - 1
IDENT(2) = IDENT(2) + 1
ADFRC(1) = 4

FILEDPN
Open the ACT

Is ISORT = true?

FILEDPN
Open the scratch file SPF

RMOVED = 0
NR = 0
LREI = IDENT(13) - 1

Is SORT true?

Read ACT record into ITUP EDF

Read SPF record into IREC
IREC = IREC - 1
Read ACT record into ITUP EDF

(p.46)

(p.46)

(p.46)

(p.46)

B-46
Is E0F true?

**IFMI**

**IWKEY**

**ICC-I**

**Is N-IDENTI(15) or ICC-IDENTI(9)?**

**False**

**True**

**ICC=ICC+1**

**Is there string strategy V?**

**No**

**Yes**

**IKEY=IKEY+1**

**IMKEY=IMKEY+1**

**ITUP=ITUP+1**

**Get tuple ITUP**

**Save the tuple to the new TF**

**NR = NR + 1**

**Is NR-IDENTI(9) or IMMOVE=IMMOVE?**

**No**

**Yes**

**Is ITFRIC(k) greater than 0?**

**No**

**Yes**

**J=ITFRIC(k+1)/6**

**IMKEY = IRSAVE**

**IMKEY = IMSAVE**

**Get the ADF record**

**END**
MOVE ACTIVE RECORDS TO NEW RELATION

FILECLS Close the sort file SF1

FILECLS Close the ACT

Is ISORT true?

IRKEY=INMOVE IRKEY=INMOVE

PUTDF Save the new TDF

IRKEY=IRSAVE IRKEY=IRSAVE

Is YES true?

KL=LONGIHAT(1)/6

"Records transferred. Begin deleted at high end".

FILEIDM Open ACT

Have all the records been examined?

Checking record number ___

Read ACT record into ITUF EDF

B-48
MOVE ACTIVE RECORDS TO NEW RELATION

J (p. 47)

S (p. 47)

K (p. 47)

A (p. 47)

Is EOF true?

yes

no

DELETE
Delete corresponding tuple of original

PUTDF
Save the IDF

CPYREL
Copy relation to TEMPFL

FILECLS
Close ACT

FILEDEL
Delete ACT and SFI

ISORT
ACTIVE=false

NUMACT Deleted
K1=LONGRAM(1)/b

"NUMACT records moved to LONGRAM"

BACK

B-49/(B-50 blank)
APPENDIX C

DISPLAY
DISPLAY

DECLARE VARIABLE TYPES AND ASSIGN DATA

COLUMN NUMBERS FOR X-AXIS DETERMINED

"COLUMN NUMBER FOR X-AXIS VARIABLE"

GET COLUMN
GET X-AXIS COLUMN NUMBER

TEST FOR:
TEST STRATEGY FOR COLUMN

IS ANYTHING WRONG?

FALSE

GET COLUMN
GET NAME FOR COLUMN

COLUMN NUMBERS FOR Y-AXIS DETERMINED

"COLUMN NUMBER FOR X-AXIS VARIABLE"

GET COLUMN
GET Y-AXIS COLUMN NUMBER

A
'THE COLUMN THAT YOU HAVE SELECTED DOES NOT HAVE A NUMERIC STRATEGY. DO YOU WISH TO MAKE ANOTHER SELECTION <YES OR NO>?'
THE LEGAL COMMANDS ARE:
- S (QUIT) TERMINATE PLOTTING PROGRAM
- R (RESET) RESET PLOTTING PROGRAM
- (STATE) PRINT STATUS REPORT
- X (X-AXIS) SET X-AXIS RANGE
- Y (Y-AXIS) SET Y-AXIS RANGE
- P (PLOT) DISPLAY THE PLOT
- L (LIST) LIST THE LEGAL COMMANDS
- I (LABEL) LABEL: THE DATA POINTS WITH LINES
- L (LABEL) CREATE X AND Y AXIS LABELS
FEASIL IMPLEMENTATION UNDER VAX VMX WITH DESIGN INFORMATION (U) ALABAMA UNIV IN HUNTSVILLE DEPT OF ELECTRICAL AND COMPUTER EN. J D MARR ET AL. NOV 86 UNCLASSIFIED UAH-5-31325 AMSHI-CR-RD-55-86-5 F/G 12/5
GET AXIS LABEL

'LABEL = ' 

READ NEW LABEL

WANT TO CHANGE LABEL?

FALSE

WANT TO CHANGE LABEL?

TRUE

QUOTE MARK NOT FOUND AS DELIMITER?

FALSE

IS EVERYTHING OK?

TRUE

SET NEW LABEL

FALSE

IS EVERYTHING OK?

TRUE

FIN
Subroutine: FUNCTION (Rlfunc)

This subroutine provides the user with data analysis functions to allow analysis of sum of column, averaging, mean and standard deviation, etc. This subroutine is constructed that more functions may be easily added as the need arises. The functions available are:

1. Total Column Data

   This function algebraically adds the active records of a relation for the column specified. The strategy must be integer or floating point.

2. Mean and Variance

   This function computes the mean and variance of the active records in the specified column. Output is the mean and standard deviation.
**FUNCTION**

1. Define units
   - `LUTF = 11`
   - `LUACT = 12`

2. Set up activity file name

3. Set up `TEMPFILE.TF` name

4. Set up
   - Total number of functions

5. Offer menu

6. Get function selection number

7. If menu number is 0, return
   - "TOTAL Column Data"

8. If menu number is 1, return
   - "TOTAL Column Total"

9. If menu number is 2, return
   - "TOTAL Column Data" and "Write Mean and Variance"

10. Otherwise, return
    - "Response not understood. Returning to menu"
get function selection number

```
RLFUNC\#1

OK = true

(prompt)RMPT

GETNUM

get answer IAMS, error flag IERR

IAMS<0 OR IAMS>NOFUNC

T

selection outside range

OK = false

F

IERR=0 and OK=true

F

BACK
```

D-3
2. get column number

RLFUNC$2

repeat

OK = true

GETNUM

get column
ICC, error
flag IERR

ICC > 0

OR

ICC = #columns

T

"selection outside range"

F

OK = false

IERR = 0

and

OK = true

F

T

BACK

OK
write column mean and variance

Mean and variance of column
For: Records = 

Is the absolute value of xmean

>10^9 or <10^-9
write the mean using scientific notation format E16.9

≥1.0
write the mean using floating point format F12.2

otherwise
write the mean using floating point format F10.7

Is the absolute value of standard deviation (STDEDEV)

>10^9 or <10^-9
write the standard deviation using scientific notation format E16.9

≥1.0
write the standard deviation using floating point format F12.2

otherwise
write the standard deviation using floating point format F10.7

back
write column total

RLFUNC#4

is there integer column strategy

is the absolute value of total

>10^9 or -10^-9

"Column Totals" written in scientific notation in F16.9 format

<2

"Column Totals" written in floating point F12.2 format

otherwise

"Column Totals" (written in floating point F10.7 format)

"with active records"

back
total column data

"column to use?"

get column number

is the strategy not real or decimal?

"column strategy does not allow addition"

return

XTOTAL = 0
ITOTAL = 0
SS = 0
INO = TF

FILEOPW

open TF

active

T

open ACT

F

ISTOP = number rows

ISTOP = number active

ICT = 1
ISTOP

close TF and ACT

back
(repeat)

active

read OK from ACT

read value

OK

read value

read value

increment ICT

OK or ICT = ISTOP + 1

empty

IANS = 1

IANS = 2

by strategy, add to TOTAL or ITOTAL

x = value

inc TOTAL by x

inc ITOTAL by 1

MEAN = TOTAL / ITOTAL

SS = SS + x*x

ITOTAL > 1

VAR = 0

VAR = (SS - MEAN*MEAN*ITOTAL) / (ITOTAL - 1)

STDOEV = SQRT(VAR)
START
NO STRING TO FORMAT
ARRAY
FILL WITH
SPACE WITH BLANKS
FIN
START

WAIT FOR
MORE DATA

DETERMINE
MORE MD
APPLICATION
PARAMETERS

INPUT MD
AND DATE
FORMATION
TO QUEUE

FIN
APPENDIX F

SORT2
Subroutine: SORT

This subroutine allows the user to organize relational data in ascending or descending order. If the relation to be sorted is small (1000 rows or less) then a "fast" sort is done in core. For larger relations, a general file to file weaving bubble sort is implemented.
SORT

ASCTYP

PAGE 1

START

Determine the ASCII number of the character

Set ASCTYP = 5

Is the character blank?

Yes

ASCTYP = 1

No

Is the character a capital letter?

Yes

ASCTYP = 2

No

Is the character a lower case letter?

Yes

ASCTYP = 3

No

Is the character a digit?

Yes

ASCTYP = 4

No

RETURN

F-2
SORT
ASC CHR

START
Pass in two characters.

ASCCHR
Determine the character type.

Are the characters in ascending order?
ASCHR = true

no
RETURN

Are the characters in ascending order?
ASCHR = false

no
RETURN

ASCCHR = true
(characters are of the same type.)

Are the characters in ascending order?
ASCHR = false

yes
RETURN

F-3
Is the strategy = 6? (character string)

yes

'Should not be in SORTFA-strategy 6''

RETURN

yes

Have all the records been examined?

Read in an ADF record

Decode by strategy into DATI or DAIX arrays.

Sort These

Output in Ascending Or Descending Order

FILENAME Close the sort file (USER)

RETURN
START

Set up to start from the beginning and sort forward to the end of file.

Has the last record been examined?
  yes
  no

Is the strategy 1?
  integer
    Ascend = DAT1(I)
    yes
    less than or equal to DAT1(I+1)
  no

Is the strategy 2?
  decimal
    Ascend = DAT1(I)
    yes
    less than or equal to DAT1(I+1)
  no

Is the strategy 3?
  character
    Use ASCONR function to determine order. Then assign to DATI
  no

Is the data in the right order?
  yes
  no

Sweep the data.

Has all the data been successfully sorted?
  yes
  return
  no
Set up to start from the end and sort backwards to the beginning.

Has the last record been examined?

Has all the data been successfully sorted?

Is the strategy - 1? Ascend = DATI(1) less than or equal to DATI(I+1)

Is the strategy - 2? Ascend = DATX(I) less than or equal to DATX(I+1)

Is the strategy - 3? Use ASCXNR function to determine order, then assign D ATI

Is the data in the right order?

Swap the data.

Has all the data been successfully sorted?

RETURN
SORT
OUTPUT IN
ASCENDING OR
DECLENDING
ORDER

START

Have all the ADF records been examined?

no

Encode data according to strategy into ADF

yes

BACK

Write the sorted data back to USF.
START

TXTLEN(4)-STRMAX
MAXADF-STRMAX/6
MULEN(4)+MAXADF

CATSTR
PSLIDE
Build file names for .SF.11,..12

FILEOPEN
Open data file

Is the strategy = 4 and the number of records less than 100?

yes

no

SORTFA
Do an in memory sort.

RETURN

Estimate and report the time needed to do the sort.

INIDE = 0

Is the strategy not equal to 4?

yes

INIDE = 8

no

CREAFIL
Create files .11 and .12
(INIDE x NUMREC)

Were there any errors during file creation?

yes

"Error in creation of file"

no

RETURN

(page 8)
Is the strategy = 4?

no

open .11

no

Were there any errors in opening the .11 file?

yes

"Error in file open"

no

RETURN

yes

FILEOPEN open .11

no

Is the strategy = 4?

no

open .12

no

Were there any errors in opening the .12 file?

yes

"Error in file open"

no

RETURN

yes

KR = 1 + WLEN(strategy)

is the strategy = 4?

yes

no

A (page 9)

B (page 9)
A (page 8)

Has all the data been processed?

no

Read in data from .10

DECODE by strategy into DATI or DIX

Write data back to .11

FILECLS close .SF

SMITH=true

Switch Units

Is the strategy = 1?

yes

Sort Data As Integer

no

Is the strategy = 2?

yes

Sort Data As Real

no

2 (page 10)

B (page 8)

Has all the data been processed?

no

Read in data from .10

Write data back to .11

F-10
Is the strategy = 3?
  yes
  no
  Is the strategy = 4?
    yes
    no
    FILECLS
    Close .SF, .11, .12
    FILEDEL
    Delete .11, .12
    RETURN
  Sort Data As Character
  Sort Data As String
SORT
SORT DATA
AS INTEGER

START

Read a record into ROMBUF, and IBUF

Have all records been examined?

yes

no

Read a record pair into ROM, DATI

Is DATA greater than IBUF?

yes

Write ROMBUF and IBUF back to file UART

no

Swap order and write ROM, DATI back to file UART

Write out the last record to UART

Was any data flagged?

yes

no

BACK

Switch Units

Is this pass a multiple of 100?

yes

no

"on pass"

""
START

Read a record into ROMBUF, and XBUF

Have all records been examined?
   yes
   no

Read a record pair into ROM, DATX

Is DATX greater than XBUF?
   yes
   no

Write ROMBUF and XBUF back to file UMRT.

Swap order and write ROW, DATX back to file UMRT.

Advance through. ROMBUF=ROM XBUF=DATX

Write out the last record to UMRT

Was any data swapped?
   no
   yes

BACK

Switch Units

" on pass

Is this pass a multiple of 100?
   yes
   no
SORT
SORT DATA
AS
- CHARACTER
PAGE 13

START

Read a record into ROMBUF, and IBUF.

Have all the records been examined?

yes

Read a record pair into ROM DATI. Then, using
the ASCII function determine if they are in the
correct order.

Are they in the right order? (ASCEND)

no

Swap the order and write the data back to UART

Write the data back to UART

Advance position

no

Write out the last record to UART

Was any data switched?

no

BACK

yes

Switch Units

Is this pass a multiple of 100?

yes

no

"on pass"

F-14
SORT DATA AS STRING

START

read a record into ROBRUF ADFBUF

Have all records been examined?

yes

Read a record pair into RCM IADFRC

Set ASCORD to true.

Is the length of IADFRC=0?

yes

no

Set ASCORD to false

Is the length of ADFBUF = 0?

yes

no

Have the first two characters in ADFBUF been read in?

yes

no

GETCH Get the second character of ADFBUF

GETCH Get the second character of IADFRC

Using the ASCORD function, compare the second characters of ADFBUF with IADFRC to check for the correct order.

(page 14) (page 15) (page 15) (page 15)

A B C D

F-15
START

Has SWITCH been set to true?

no

LUREAD = .12
LIMRT = .11
SWITCH = true

yes

LUREAD = .11
LIMRT = .12
SWITCH = false

BACK
START

CATSTR
PSLIDE Build .SF file

IFFF="empty" symbol.
SINMAX = 0

Has ISORT been set to true?

yes

no

J=0

Increment J
New number of records

CREAFIL
Create a sort file

Is there an error that is not code 314?

yes

"SortFile creation error"

RETURN

no

Is there an error code 314?

yes

FILEDEL
Delete the old sort file

no

Was there any error?

yes

no

Have there been 2 tries or a successful creation?

yes

"SortFile creation error"

RETURN

no

no

(page 18)

F-18
FILEOPEN
open .SF

Were there any errors in file open?

yes

Error opening file?

sort file in
CSORT Tân

no

OPM TF
Open the TF

RETURN

Is the strategy = 4?

yes

OPMADF
Open the ADF

no

RR = IDENT(13)-1

Have all the tuples been examined?

Read a record into ITFRAC

FILECLS
close .SF, .TF, .ADF

Is the strategy = 1 ?

yes

RETURN

no

Is the data in the desired column empty?

no

Set the data to zero.

A (page 19)

2 (page 19)

B (page 19)
SORT
CSORTF
PAGE 19

Is the strategy = 2?

no

Is the data in the desired column empty?

Set the data to zero (0.0)

yes

no

Is the strategy = 3?

Is the strategy = 4?

no

Is the string empty?

no

yes

ENCODE the data in the particular column into an integer format (I32)

ENCODE the data in the particular column into a scientific notation (E16.9)

Put the value in the particular column into LADFRC(L2)

F-20
SORT
OUTPUT IN
ASCENDING
OR DECENDING
ORDER
PAGE 21

START
FILEOPEN
Open the .SF
LEN-WLEN(strategy)

Have all the records been examined?
yes
no

NUMIN = the number of records + 1 - counter

Is STRDIR true?
yes
NUMIN = counter
no

Read record NUMIN into ROM and DATI or IADFRC by strategy

ENCODE by strategy data into IADFRC

Write ROM, IADFRC

BACK
**Subroutine: EDITREL**

This subroutine allows the user to change, add, delete, or examine tuple data.

The following edit commands are available to the user:

<table>
<thead>
<tr>
<th>Command</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>?(STATUS)</td>
<td>Gives current values of row and column pointers, relation name, number of records and number of columns</td>
</tr>
<tr>
<td>Q(QUIT)</td>
<td>Terminates editing</td>
</tr>
<tr>
<td>C(COLUMNS)</td>
<td>List names of each column by number in the relation</td>
</tr>
<tr>
<td>+<a href="PLUS"></a></td>
<td>Moves current row pointer forward [X] rows (last given if total exceeded) 1 is understood if [0] or [ ] used.</td>
</tr>
<tr>
<td>-<a href="MINUS"></a></td>
<td>Same as + except in other direction</td>
</tr>
<tr>
<td>F(FIND)</td>
<td>Find the first occurrence of data item in specified column.</td>
</tr>
<tr>
<td>S(SUBSTITUTE)</td>
<td>User substitutes new data into location pointed to by current row and current column pointers.</td>
</tr>
<tr>
<td>E(EXAMINE)</td>
<td>Displays to user contents of current row and column pointer location.</td>
</tr>
<tr>
<td>D<a href="DELETE"></a></td>
<td>Deletes [X] number of tuples starting with current tuple. None deleted if none specified.</td>
</tr>
<tr>
<td>I_INSERT)</td>
<td>Inserts tuple after current tuple pointer. Prompts user for column values.</td>
</tr>
<tr>
<td>B(BOTTOM)</td>
<td>Same as insert accept places tuple at bottom.</td>
</tr>
</tbody>
</table>
P[ ] (PRINT) PRINTS [X] TUPLES OF RELATION STARTING WITH OUTPUT DATA IN A FORMAT THAT WILL ALLOW RE'ADD'ING BY THAT OR ANOTHER RELATION WITH EQUAL STRATEGIES WITH CURRENT TUPLE. 1 ASSUMED IF 0 OR BLANK.

R (RE-START) USER MAY RESTART EDIT PROCESS TO CHANGE DISPLAY FORMAT.

H (HELP) LIST THE POSSIBLE COMMANDS

A (ADD TUPLES) READS DATA TUPLES INTO RELATION IN FREE FORMAT, CARD IMAGE SEPARATED BY USER DELIMITER.
**FEASIL**

**EDITREL**

Subroutines and Procedures

(Listed in the order in which they appear in the code.)

<table>
<thead>
<tr>
<th>Subroutine or Procedure Name</th>
<th>Flow Chart Page Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>I Quit</td>
<td>Page 7</td>
</tr>
<tr>
<td>Display Help</td>
<td>Page 7</td>
</tr>
<tr>
<td>Check If Relation Empty</td>
<td>Page 8</td>
</tr>
<tr>
<td>Present Relation Status</td>
<td>Page 8</td>
</tr>
<tr>
<td>Find Column Display Description</td>
<td>Page 9</td>
</tr>
<tr>
<td>Insert A Tuple</td>
<td>Page 11</td>
</tr>
<tr>
<td>Delete Tuples</td>
<td>Page 12</td>
</tr>
<tr>
<td>Add Tuples At Bottom</td>
<td>Page 13</td>
</tr>
<tr>
<td>Print Data</td>
<td>Page 14</td>
</tr>
<tr>
<td>Print Variable Length Output</td>
<td>Page 15</td>
</tr>
<tr>
<td>Generate Write statement Maybe Column Names</td>
<td>Page 16</td>
</tr>
<tr>
<td>Add Tuple Data To Block</td>
<td>Page 17</td>
</tr>
<tr>
<td>Restart Description</td>
<td>Page 18</td>
</tr>
<tr>
<td>Move Tuple Pointer Forward</td>
<td>Page 19</td>
</tr>
<tr>
<td>Get Next Tuple</td>
<td>Page 20</td>
</tr>
<tr>
<td>Move Tuple Pointer Backwards</td>
<td>Page 20</td>
</tr>
<tr>
<td>Find Value</td>
<td>Page 21</td>
</tr>
<tr>
<td>Substitute New Value</td>
<td>Page 25</td>
</tr>
<tr>
<td>Examine Current Values</td>
<td>Page 26</td>
</tr>
<tr>
<td>Idle and Do Nothing</td>
<td>Page 26</td>
</tr>
<tr>
<td>Present Current Value</td>
<td>Page 27</td>
</tr>
<tr>
<td>Get Column Number</td>
<td>Page 27</td>
</tr>
<tr>
<td>Get Value or String</td>
<td>Page 28</td>
</tr>
<tr>
<td>Insert Tuple Data</td>
<td>Page 32</td>
</tr>
<tr>
<td>Display Selected Tuple Values</td>
<td>Page 34</td>
</tr>
<tr>
<td>Check TF Allocation</td>
<td>Page 34</td>
</tr>
<tr>
<td>Add Data Tuples To Relation</td>
<td>Page 35</td>
</tr>
<tr>
<td>Process All Cards</td>
<td>Page 40</td>
</tr>
<tr>
<td>Input As Non string</td>
<td>Page 43</td>
</tr>
<tr>
<td>Copy String Between Delimiter</td>
<td>Page 46</td>
</tr>
</tbody>
</table>
START

Clear the screen.

NAME

Get the relation name.

EQUIV

HASH

Hash the relation name

FILES

setup for disk I/O

IDENTIFY

Get the IDENT and INDF arrays for modification.

Is there an I/O error?

yes

RETURN

no

Initialize pointers and bring IT into area (EXIT)

Present Relation Status

(page 2)
Is the answer "\(\rightarrow\)" (move cursor forward)?

- Yes: Check if Relation Empty
  - Yes: Is the relation empty?
    - Yes: Move Tuple Pointer Forward
    - No: Move Tuple Pointer Backwards
  - No: Display Selected Tuple Values

- No: Is the answer "\(\rightarrow\)" (move cursor backwards)?
  - Yes: Check if Relation Empty
    - Yes: Is the relation empty?
      - Yes: Move Tuple Pointer Forward
      - No: Move Tuple Pointer Backwards
  - No: Display Selected Tuple Values

- Restart Description
EDITREL

Is the answer 'E' (examine)?
- yes: Check if Relation Empty
- no: Is the relation empty?
  - yes: Examine Current Values
  - no: Is the answer 'I' (insert)?
    - yes: Insert a Tuple
    - no: Is the answer 'D' (delete)?
      - yes: Check if Relation Empty
      - no: Is the relation empty?
        - yes: Delete Tuples
        - no: Is the answer 'S' (append to bottom)?
          - yes: Add Tuple At Bottom
          - no: (Back to start)

G-9
Is the answer 'P' (print)?

Check if Relation Empty

Is the relation empty?

Print Data

Is the answer 'A' (add)?

Add Data Tuple To Relation

Is the answer 'H' (help)?

Display Help

Idle and Do Nothing
EDITREL

I QUIT

START

PUTTDF put new IDENT and ITDF arrays into .TDF file

Is there an I/O Error?

yes

RETURN

no

BACK

---

DISPLAY_HELP

START

"The valid commands are:"
(list of valid commands)

BACK

G-11
EDITREL

CHECK IF RELATION EMPTY

START

Is the relation empty?
Is IDENT (LB) greater than zero?

no

Relate empty

yes

Set the empty relation flag to true.

BACK

PRESENT RELATION STATUS

START

"_ Relation"

"_ number of records, current record _"

"_ number of columns, current column _"

BACK
START

Display column names (yes or no)?

YESNO Get yes/no answer. Set DISCOL to true if yes.

Has the last column been displayed?

yes

no

'Display how many columns?'

Get Column Number

Is the response greater than the actual number of columns?

yes

no

'There are not that many columns.'

Is the response negative?

yes

no

'Value must be zero or positive.'
Are there more than zero columns to be displayed?

- Default order (yes or no)?

YES/NO

Get the yes or no answer.

Is the answer YES?

- Display in column number

Get Column Number

Mark the columns to be displayed.

ACK
EDITHEL
INSEK: A TUPLE

PAGE 11

START

Check TF Allocation

Was the allocation successful?

no

yes

Insert Tuple Data

PUTFDEF Update the tuple descriptor.

BACK

G-15
EDITHEL

DELETE TUPLES

PAGE 12

START

COPYSUB
- Strip the letter 'Y' off the action string.

RENAME
- Get the number of tuples to be deleted from the action string.

Is there an I/O error?

yes

no

Mark zero tuples to be deleted.

Mark the number of tuples to be deleted.

Are there more tuples to be deleted?

no

yes

DELTUP
- Delete a tuple.

Are there any remaining tuples in the relation?

no

yes

Move the pointers to reflect deleted elements.

RTFRC
- Retrieve current TF into CTEZ.

(page 13)
EDITHEL
DELETE TUPLES

PAGE 13

PUTTDF
Update the Tuple Descriptor File (TDF)

Are there tuples remaining in the relation?

no

ye

Display Selected Tuple Values

BACK

ADD TUPLE AT BOTTOM

START

Move the pointer to the bottom

Insert a Tuple

BACK
START

CPMSUB
Strip the 'P' off the action

READIN
Get the number of tuples to be printed

Check to see if there are tuples available for printing

Are there tuples remaining?

no

yes

Generate
Write
Statement
Maybe Column Names?

Print
Variable
Length
Output

Print
Next
Tuple

Is the pointer at the end of the relation?

no

yes

Move Tuple Pointer Backwards

BACK
Adjust the string length until it is as long as the variables.

Write out the FORMAT array.

Back
START

BINARY
Initialize and empty the FORMAT array

Are all the column names and tuple data loaded into FORMAT?

yes

no

BACK

Has DISCOL been set to true in Find Column Display Description?

no

yes

Has the column name been loaded into CORE?

no

yes

COLNAM
Get column names into CORE.

Get the column names from the ADF.

CATSTR
Move IADFRC into FORMAT

CATSTR
Move IMTAR into FORMAT

Add Tuple Data To Block

CATSTR
Move ILINK into FORMAT

CATSTR
Move PMBLK into FORMAT
Find Column Display Description

Align to Longest Displayed Column Name

Are there 1 or more elements in the relation?

no

yes

Display Selected Tuple Values

BACK
EDITREL

START

PUT TC
Put the TP back

READM
Get the number of spaces to move

Move the pointer the specified number of spaces forward

Get Next Tuple

Display Selected Tuple Values

BACK

MOVE TUPLE POINTER FORWARD

PAGE 19
Will the increment to be moved to the end of the relation?

- Yes: Set the increment to the end of the relation.
- No: Move the tuple at the specified increment.

MOVE TUPLE POINTER BACKWARD

- Get the tuple back
- Get the number of spaces to move
- Move the pointer the specified number of spaces backward
- Get the new tuple file values into core
START

FIND VALUE

Has the desired column been found?

yes

no

"Find in which column?"

Get Column Number

"Find what string?"

Get Value or String

Is the response an empty string?

yes

Set IADFRC to empty set identifiers


)return

Do the string to be found into a working array (string)

Has the string been found?

yes

Has the search reached the end of the tuple file?

no

"Searching...

(page 22)

A

B

G-25
Is the data non-string?

Is this the element being searched for?

Set a flag that the element has been found.

Is there string data?

Is the string less than or equal to zero?

UTPAR

Is the ACTION string the same length as the data string?

Is the first element of each string empty?
Clear the screen.

"Current record
Current column"

Display selected tuple values

START
EDITREL

SUBSTITUTE NEW VALUE

PAGE 25

START

"Column record"

Present Current Values

"Enter new string."

Is there a new string?

no

Check ADF Allocation

Was the allocation OK?

no

Get Value or String

PUTFD Substitute the new data

Display Selected Tuple Values

BACK

G-29
EXAMINE CURRENT VALUE.

START

/Examine which column?/

Get Column Number

Present Current Value

BACK

IDLE AND DO NOTHING

START

/Command not understood. Type 'N' for help/

BACK

G-30
Editrel

**Present Current Valuf**

**Page 27**

```
START

Adjust current pointers

MTRC
Get the requested data from tuple file

Output Tuple Data

BACK
```

**Get Column Number**

START

Has the column number been found with no problems?

YES

no

GETM
Get the column number

Is the number larger than the TF?

YES

no

"Input error. Value must be in range."

BACK

```
G-31
```
Is there integer strategy?

"Integer"

GETIN
Get the integer data.

Is the data zero?

yes

no

Make a zero length assignment to IPCR.

Is there floating point strategy?

yes

"Decimal"

GETFP
Get the floating point data.

Is the data zero?

yes

no
Is the strategy a character string or single character?

Has OK been set to true?

Has a valid input been provided?

Is the strategy single character?

GET Get the new character data.
EDI
H
FL
G
I
VALUE
PC 30
determine string length

Is the string of zero length?

no

yes
Set the value to zero.

Adjust length qualifiers and make assignments

CPYSUB
Copy the adjusted data in.

CATSTR

Is there single character strategy?

no

yes

Is the data of length one?

no

yes

CPYSUB
"Single character only."

G-34
Is the strategy not a character string or a single character?

If no, continue with the process. If yes, it may indicate a problem with tuple editing or an error. See the programmer for help.

BACK
START

Check IF Allocation

Was the allocation successful?

no

yes → BACK

yes

no

Has the column name been loaded into core?

no

yes → COLNAME

Get column name into core

Write out the column name

Get Value Or String

IS there string strategy?

no

yes

Is the string length zero?

no

yes → ASSIGN

Assign proper IIFRC element to empty.

PADFOR

Return TFAR

1 (page 33)

A (page 33)

PAGE 32
Is TFAR true?

- No:
  - Make assignments to IDENT

- Yes:
  - Make assignments to ITTRC

  ADDTUP
  Add the tuples where they belong.

  Display Selected Tuple Values

  BACK
EDITHEL
DISPLAY
SELECTED
TUPLE VALUES

START

yes

no

Have all the marked tuples been displayed?

Calculate which tuples are to be displayed and which are remaining.

Mark the remaining tuples (selectively) for display.

Output Tuple Data

BACK

CHECK TF
ALLOCATION

START

Is there enough space allocated in the tuple file?

yes

no

RENTF reallocate a new TF

Was the allocation successful?

no

yes

PUTTF Move the data into the new tuple file.

BACK
ELITREL

ADD DATA TUPLES TO RELATION

PAGE 36

A

GET
Get the device or file name.

B

STLEN
Interpret the answer as a string.

EMIARY
empty the IBLK array.

EQUIV
Copy the device or filename into IBLK.

Write IBLK array into character variable ADDNAME.

Open the file which is to have tuples added.

Were there any errors in opening the file?

yes

no

"Data file not opened. Error."

Has the error limit been reached?

yes

no

"Too many errors. Returning to edit."

E (page 37)

G-40
EDIT REL

ADD DATA TO RELATION

Was the error limit reached?

yes

no

Has a valid answer been received?

yes

no

Enter single character delimiter between data elements.

Display Prompt

GET Get the delimiter

strlen Use the delimiter as a string.

Determine the validity of the delimiter.

EQUIV Copy the delimiter into EBLK

Is the tuple file large enough to contain the new data?

yes

no

PAGE 37

BACK

(page 36)
ADD DATA TUPLES TO RELATION

PAGE 38

C (page 37)

2 (page 37)

REDJDF
Reallocate a new tuple file.

Was the allocation successful?

no

yes

Set the number of tuples to zero.

PUTDF
Put the data into the new tuple file.

Is the ADF large enough to contain the new data?

no

REDJDF
Reallocate a new ADF

Was the allocation successful?

no

yes

Set the number of tuples to zero

PUTDF
Put in the new data

Could either of the reallocations not be successfully accomplished?

yes

"Not enough rows to add specified data. ADF exhausted."

no

3 (page 39)
EDIT RELATION
ADD DATA
TUPLES TO
RELATION

 PAGE 39

Process All Cards
Get the data from
all the cards.

PUT DATA
Close the data
file.

"DATA has
been entered
into relation"

BACK
START

Read the next data card

Have all the tuples to be added been processed?

Yes

BACK

No

Have all the columns been examined?

No

Yes

ADDTUP
Add the tuple data.

PUTIDF
Put the new IDF away.

"Working"

Is there non-string strategy?

Input: as
Non String

Is there string strategy?

No

Yes

Have all the data cards been read?

Yes

No

A (page 41)

B (page 41)

C (page 41)
FUTREL

PROCES
ALL CARDS

PAGE 42

A (page 41)

D (page 41)

2 (page 41)

E (page 41)

Increment through the string

CPYSUB
Copy all the non-blank data

PADFRC
Put the data into the ADF

Update ITPRC and IDENT arrays.

EDITREL
INPUT AS NON STRING
PAGE 43

START

Have all the data cards been read?

yes

no

Is the counter greater than or equal to 80?

yes

Read The Next Data Card

Find The Next Delimiter

Increment the counter

Copy String Between Delimiter

CATSTR
Append the delimiter to the string.

Is the counter greater than 80?

yes

no

CPYSUB

CPYSTIR

(page 44)

G-47
Has a non-blank character been found or has the end of string been found?

yes

no

GETCH
Get another character

Increment through the string

CPYSUB
Copy all the data

Remove trailing edge blanks.

Has a non-blank character been found or has the end of string been found?

yes

no

GETCH
Get another character

Increment through string

CPYSUB
Copy all the data.
Is there integer strategy?

- Yes
  - Is the string blank?
    - No
      - Make empty assignments
    - Yes
      - REAMM

- No
  - IS there floating point strategy?
    - Yes
      - Make empty assignments
      - REAPP
      - TRIPI
    - No
      - Is the string blank?
        - Yes
          - Make empty assignments
          - REAPP
        - No
          - IS there single character strategy?
            - Yes
              - Is the string length greater than one?
                - No
                  - "Single character only. Only first character used."
                - Yes
                  - Copy one character into ITPRC.
            - No
              - BACK
EDITRED

START

Mark the string

CPYSUB
Copy the marked string

BACK

COPY STRING BETWEEN DELIMITER

PAGE 46

---

FIND THE NEXT DELIMITER

START

Has the delimiter or the end of string been found?

yes

BACK

no

CPYSUB
Copy in another character.

---

G-50
START

Read a block of data into FORMAT array.

Were there any I/O errors?

no

yes

Write Data Card Read Error

IOSTATUS CODE

Was an end of file detected?

no

yes

Write Data File End of File

BACK
Is the data empty and not nothing?

no

Make empty assignments

no

CATSTR
Copy these empty values into FORMAT

Is there integer data?

no

yes

Transfer the first 10 characters into FORMAT

CATSTR
Append HEXDATA into FORMAT

Is there floating point data?

no

yes

PRINT

Format
Decimal ASCII
Array by Neg

CATSTR
Append the results into FORMAT

51

1

(page 49)

(page 50)

(page 51)

(page 51)

G-54
Is there single character data?

- No:
  - Loop through ISPAC and copy first 10 characters into FORMAT (CATSTR)
  - CATSTR Append data in HTDXT into FORMAT

- Yes:
  - Is there string data?
    - No:
      - Is the string empty?
        - Yes:
          - CATSTR Make empty assignments to FORMAT
        - No:
          - Is the data less than zero?
            - Yes:
              - UTFAR
            - No:
              - RADFRC

  - Yes:
    - Is the string length less than 75 characters?
      - Yes:
        - Loop through ISPAC and copy characters into FORMAT (CATSIM)
      - No:
        - CATSTR Copy LADFRC into FORMAT

  - Has the string been divided into segments long than 75 characters?
    - Yes:
      - CATSTR Copy the LADFRC into FORMAT
    - No:
      - Loop through ISPAC and copy characters into FORMAT (CATSIM)
START

Is the column name longer than 64 characters?

yes

no

Have all the columns been examined?

yes

no

COLNAM
Get the column name

Load the column name into CNAMES

Set COLNAM to TRUE to indicate that column names are in core.

Write out the column names. "Column Names"

BACK
Are the column names loaded into core?

- Yes: Write out the column names (CORNAMES)
- No: Get the column names into core from ADF

START

BACK

Has the counter (i) reached 52?

- Yes: BACK
- No: Set FORMAT(1) equal to zero.

START

FORMAT

ARRAY
EDITREL
SAVE LIKE
ADD COMMAND
EXPECTS

1 (page 55)

Were there more than 2 attempts to open as output file or device?

Yes

Has the output file or device successfully opened?

Yes

Has a valid number of records been given?

Yes

"Output how many records?"

No

Display Prompt

GETNOM
Get the number of records

Has the answer verified?

"Invalid input. Try again."

No

Has the number of records to be outputted greater than zero?

Yes

Close the file (USAV)

"Nothing done."

2 (page 57)

PAGE 56
"Too many errors. Check output device and try again."

A (page 57)
Blank Format Array

Copy the column data into FORMAT depending on position. (PITP, UTPAR, RADFAC)

Write the FORMAT array out into LUSAV

KTPRC: Put away RELDES

BACK
EDITREL
DISPLAY PROMPT

START

"Edit command."

BACK

FORMAT
DECIMAL
ASCII
ARRAY
BY MAG

START

Alter data formats through the_INFO function.

BACK

G-63/(G-64 blank)
OVERLAY: NEWRELAT.FLC

SUBROUTINE: NEWREL.FLC

SYNTAX:
CALL NEWREL(IDENT,ISAFITF,ITDF,INC,INR,IDEM,IDADF,ITDF,ITFRC,IADFRC)

NEWREL---CREATE A NEW RELATION.

THIS ROUTINE IS USED TO INITIALLY ESTABLISH A RELATION. THE THREE FILES .TF, .TDF AND .ADF ARE ALLOCATED, AND THE COLUMN HEADINGS AND STRATEGIES ARE REQUESTED AND SET UP. ALSO THE INITIAL VALUES FOR THE RELATION (IDENT AND ITDF) ARE ASSIGNED TO THE FILES.

THE SUBROUTINE ARGUMENTS ARE:
1. IDENT - RELATION STATUS: CORE RESIDENT,
2. IDEM - DIMENSION OF IDENT,
3. ITDF - CORE RESIDENT TDF,
4. INC - NUMBER OF COLUMNS IN TDF, AND
5. INR - NUMBER OF RECORDS IN TDF.
OVERLAY: DELETREL.FLC

SUBROUTINE: DELREL.FLC

SYNTAX:
CALL DELREL

THIS ROUTINE DELETES AN UNWANTED RELATION. ALL FILES .TF, .TDF, AND .ADF ARE DELETED. THIS ROUTINE REQUIRES CONFIRMATION THAT THE RELATION NAMED IS THE CORRECT RELATION TO BE DELETED BEFORE IT IS DELETED.
APPENDIX J

MODCOL
Subroutine: MODCOLUM

This subroutine provides for the manipulation of columns in an existing relation. No changes in strategy are allowed, since this would assume a change in data.

The basic functions provided are:

1. Delete a column
2. Add a column
3. Alter a column name
4. List current column names
Clear the screen.

Modify column specifications.

Relation to be modified.

NAME
Get the relation name.

HASHM
Hash the relation name.

FILEMS
Set-up for disk operation.

RVIDF
Get IDENT and ITDF arrays for modification.

Is there an I/O Error?

yes
RETURN

no

Has an exit flag been set by the QUIT option?

yes

PUTDF
Put new IDENT and ITDF arrays away before leaving.

no

MODCOL command.

GET
Get the command answer.

A (page 2)

B (page 2)
MODCOLUMN

PAGE 2

A (page 1)

Interpret the answer as a string.

B (page 1)

Is the answer C? (list column names)

yes → C

no

Is the answer R (rename a column)?

yes → D (page 4)

no

Is the answer D (delete a column)?

yes → E (page 5)

no

Is the answer A (add a column)?

yes → F (page 8)

no

Is the answer Q (quit)?

yes → Set an exit flag.

no

Write the valid commands
MODCOLUMN

Page 3

Are there more column names in this relation?

Yes

COLUMN
Get the column name.

"column", column name

Increment to next column.

No

A (page 1, 2)
MODCOLUM

Has OK been set to true?

Yes

No

'Column to rename.'

GET COLUMN NUMBER
find the column, return OK

'New column name'

GET
Get the column name.

STLEN
Interpret the column name as string

Update ADF byte count and add new bytes to the count.

PADFLC
Put column name into ADF

'Column: is now named'

A (page 1, 2)
MODCOLUM

Is OK true?

Have there been more than 3 tries?

"Column to be deleted."

Is OK true?

GET COLUMN NUMBER
Find the column
Return OK

COLUMN
Get the column name.

"Confirm deletion of column needed."

"Yes or No"

YESNO
Return OK

Increment the number of tries

G (page 6)   H (page 6)   J (page 6)

J-6
Is the number of rows greater than 3?

yes

"User response error. Nothing done."

RETURN

Adjust the number of deleted columns, the number of columns in the relation, and the number of bytes in the file.

Is the strategy an integer, single character, or decimal number?

no

Are there more tuples in this relation?

no

Update the strategy count.

K

(page 7)

(page 6)

H

(page 5)

G

(page 5)

J

(page 5)
Is the strategy string data?

Are there more tuples in this relation?

Update string strategy count

Retrieve the TF inco coma.

Reduce the ADF byte count.

Shift each element in ITPRC by two.

Write the shifted TF back.

Adjust the TF length count.

Increment to next TF record.

Shift the column indicators for those columns past the deleted one.
Check to see if ADF file has enough room to add another column name.

Is there enough room in the ADF for another column name?

READADF Allocates new ADF and move data.

Was the allocation successful?

"Column not added"

Has OK been set to true?

"Column after which to add new column."

GET COLUMN NUMBER
Find the column
return OK

"New column name"

GET
Get the column name

STRLNM
Interpret the name as a string

L (page 9)
Has OK been set to true?

no

Write the list of possible strategies.

GETNAM
Get the answer to strategy.

Determine when to stop moving elements.

Is the strategy an integer number, a decimal number, or a single character?

no

Are there more tuple file records?

yes

no

Adjust strategy count.

KTFRC
Retrieve the TF record into core.

Move all the elements out one slot until a hole is made for the new column data to go in.

Put blanks into the new TF location.

P1FRC
Put the new TF record back.

Update TF length count
Increment column counter

Is the strategy a character string?

Are there more tuple file records?

Adjust TDF and ADF lengths

Retrieve TF record

Move all the elements out until a two slot hole is made where new column data will go

Put blanks into new TF locations of new column

Put TF record back

Update TF length

Increment column counter

J-11
Is the strategy answer 1, 2, or 3?

No

Unknown strategy ask for.

Try again.

Yes

Create a hole in TDF for column name.

Increment column count.

Put new column data into TDF.

Update TDF and ADF lengths.

PADRC

Put column name into ADF.
MODCOLUM
GET COLUMN NUMBER

START

Set OK to true.

GETNUM
Get the
column number

Is the column number valid?

yes

no

"request outside
column range"

Set OK to FALSE.

BACK
APPENDIX K

MERGRL
Subroutine: MERGEREL

This subroutine allows the user to merge portions of two similar relations into a third composite relation.

The options are:

1. Merge two relations into a third relation containing only those records which meet the matching of like column pairs.

2. Add two relations by copying first into a third relation and then appending the second relation. The column headings must be identical to use this option.
MERGEREL

1 (page 1)

GET IDENT and TDF for second relation.

Is there an I/O error? Yes -> RETURN

No -> GET

Has an EXIT flag been set? Yes -> RETURN

No -> GET

Get the character string.

STRLEN

Return the length of the character string.

CPYSUB

Do a merge? Yes -> DO A MERGE

No -> DO AN ADD

Do an Add? Yes -> Perform the add.

No -> Quit? Yes -> Set an EXIT flag.

No ->
Do you need help?

Yes

No

Do you need help?

Yes

No

Write the menu

"The valid commands are:"

START

Yes

Reset?

No

(page 2)
START

Is either relation empty?

no

BHINARY
Get the IDENT and IDSM arrays

Empty the ITDFM array

"Name of Merged Relation."

NAME
Get the name of the relation

HASHIT
Hash the relation name

FILEDS
Set up for disk operation

Find Column Pairs For Merge

Do the columns match?

Yes

"Merge Two Relations"

"X Records resulted from this merge."

No

"No column domain matches to perform merge."

BACK

"One relation empty: Merge cannot be performed."

yes
START

Have all columns been checked?

Yes

No

Has an exit flag been set?

Yes

No

Compare column names.

Are the column names the same?

Yes

Move to next column.

No

Compare the lengths and strategies.

Are the lengths and strategies the same?

Yes

Set an exit flag.

No

COLNAME
Get column name for relation 1.

COLNAME
Get the column name for relation 2.

BACK
START

RIFRC
Get the tuple for Rel. 1

Are there any columns remaining?

yes

Is match true?

no

BACK

RIFRC
Get the tuple for Rel. 2

Are there pairs remaining?

no

Do the number of matches equal the number of pairs?

yes

MATCH = true

Increment counter

? (page ?)

A (page ?)
Does \text{II} \text{INFI}(L1,LCCI) = 4 ?

- Yes
  - Do the elements match?
    - Yes
      - Matches = Matches + 1
    - No
      - \text{II} \text{INFI}(L1,LCCI) = \text{II} \text{INFI}(L1,LCCI) + 1

- No
  - \text{II} \text{INFI}(L1,LCCI) = \text{II} \text{INFI}(L1,LCCI) + 1

Does \text{ITFR} (K1+1) = \text{ITFR} (K2+1) ?

- Yes
  - Are both elements of \text{ITFR} empty?
    - Yes
      - Make them unequal.
    - No
      - Are both elements of \text{ITFR}2 empty?
        - Yes
          - Make them unequal.
        - No
          - \text{II} \text{INFI}(L1,LCCI) = \text{II} \text{INFI}(L1,LCCI) + 1

- No
  - Are both elements of \text{ITFR} empty?
    - Yes
      - Make them unequal.
    - No
      - \text{II} \text{INFI}(L1,LCCI) = \text{II} \text{INFI}(L1,LCCI) + 1

\text{MERGEREL}
\text{COMPARE COLUMN PAIRS}
\text{PAGE 7}
MERGEREL
COMPARE
COLUMN PAIRS
PAGE 8

2 (page 7) (page 7A)

Are both elements of ITFRC and ITFRC2 empty?

yes

no

Do they match?

yes

Matches = Matches + 1

no

Is ITFRC2(K2) less than zero?

yes

no

UTFAR

RADFR

Is ITFRC(K3) less than zero?

yes

no

UTFAR

RADFR

K-9
START

Move Headings and strategies 1 to 3

Increment Column

Move nonzero data from relation 2 to merged relation

Increment Next TP Location

COLUMN
Get column name for relation 2

Put Column Name in New Relation

Is there another column?

Yes

No

BACK
MERGEREL
PUTCOLUMN_NAME
IN_NEW_RELATION

START

PADFRC
Get column name information.

Is the data in a tuple file?

No

Make assignments from IDENT

Make assignments from LA1 and LA2

Yes

BACK

PAGE 10
START

Is IDPM pair = 4?

Yes

IDENTM(L13) = IDENTM(L13 + 2)

No

IDENTM(L13) = IDENTM(L13 + 1)

BACK
START

Is there another column?

Yes

Increment Column Number

Move headings and strategies from Relation 1 to merged relation.

Increment Next TV Location

COLNAME
Get column name for Relation 1

Put column name in New Relation

No

BACK
MOVE TUPLE DATA TO NEW RELATION

1 (page 13) (page 13) A
Is ITDFZ(L1,L2,1st counter < 4)?

Yes
Move integer data

No

Is the 1st string empty?

Yes

Is the 2nd string empty?

No

Are the strings less than zero?

No

Is the first string greater than zero?

Yes

Is the first string empty?

No

Make assignments to ITFRC

Make assignments to IRKEY, DAKEY

Is ITFRC less than zero?

No

RADFRC

UTFAR

2 (page 15)
MOVE TUPLE DATA TO NEW RELATION

MERGEREL

(page 15)

1. Make assignments to INKEY, INKEY
2. PADPRC

Is TPAR true?

Yes

Make assignment to ITFRC

No

Make assignment to ITFRC and IDENTITY

A (page 13)
MERGEREL
MOVE TUPLE DATA FROM FIRST

A (page 16)

1 (page 16)

PADFRC

Is TFAR true?

No

Yes

MAKE ASSIGNMENTS TO ITPRC, IDENT

MAKE ASSIGNMENTS TO ITPRC.
MERGEREL
MOVE TUPLE DATA FROM SECOND

START

Does IDENT2(L, counter) = counter?

Yes → BACK

No → Increment counter

Does ITDF2(L1, counter) = 4?

No →

Yes → Is the first string empty?

No →

Yes → Is the second string empty?

No → Make assignments to IRKEY, IMKEY

Yes →

Is ITFRC less than zero?

No → RADFRC

Yes → UTFAR

Make assignments to IRKEY, IMKEY

I (page 19) → A (page 19)
MERGEREL
MOVE TUPLE DATA FROM SECOND PAGE 19

PADFRC

Is TFAR true?

No

Make assignments to ITFRC, IDENT

Yes

Make assignments to ITFRC

A (page 18)
START

TDFORE
Create a tuple data file.

Put name into IDENT

TDFORE
Create a tuple file.

Is IDENT(19) = counter?

Yes

Increment counter

No

Does TDFORE(I1,counter) = 4?

Yes

Have all the pairs in the column been examined?

No

Is this a comparison column?

No

Yes

Zero the counter.

1 (page 21)
MERGEREL
MERGE TWO RELATIONS
PAGE 21

1 (page 20)

Is IDNT2(L9) = counter?

Yes

No

Increment the counter.

Does ITDF2(L,counter) = 4?

No

Yes

Have all pairs in this column been examined?

Yes

No

Is this a comparison column?

Yes

No

Have all the pairs been examined?

Yes

No

Increment pair counter

Is ITDF1(1,counter) = 4?

No

Yes

Make assignments to MRPSS

Compute number of columns in merged relation.

2 (page 22)
MERGEREL
MERGE TWO RELATIONS
PAGE 22

1. Create an ADF
2. Move column names and strategies
3. Fill the IDENT array
4. Zero the counter
5. If IDENT(L9) = counter?
   - Yes
   - Increment counter
   - If [TOPM L1, counter] = 1
   - If [TOPM L2, counter] = 2
   - If [TOPM L3, counter] = 3

6. If no
   - IDENT(L17) = IDENT(L17 + 1)
   - IDENT(L16) = IDENT(L16 + 1)
   - IDENT(L15) = IDENT(L15 + 1)
MERGEREL
MERGE TWO RELATIONS

Make assignments to INDEX, IKEY

PUTDF
Put new file IDENT4 and ITEM4 away.

Are there columns left to examine?

no

BACK

yes

INCREMENT COUNTER

Compare Column Pairs

Do the column pairs match?

no

yes

Move Tuple Data to NEW Relation

ADDUP

PUTDF
MERGEREL
CHECK IF COLUMNS MATCH
PAGE 25

START
Set OK = true

Does IDENT1(L1, counter) = counter
yes -> BACK
no -> Increment Counter

Does ITDF1(L1, counter) = ITDF2(L1, counter)

yes -> OK = false
no -> Does ITDF1(L3, counter) = ITDF2(L3, counter) ?

yes -> OK = false
no -> COLNAME Get column name for relation

COLNAME Get column name for relation

COLNAME Get column name for Relation 2

Are the names the same?
no -> OK = False
yes -> OK = true

K-26
START
Create the destination relation

TFD
Create a Tuple Data File

Calculate the size of the ADF

ADF
Create the ADF

TFC
Create a tuple file.

Empty the relation.

PUTDE
Put IDENTITY and ITDPM into relation

Move Headings and Strategies 1 to 3

"Headings move to new relation"

BACK
START

Is IDENTIFIED counter?

RTFNC
Read first
TPN

Move Tuple
Data from
First

ADD/TUP
Add tuples
from first

PUT/TF
Put tuples
from 1st rel.
into new rel.

Increment
counter

"Tuples copied
from primary
relation".

Zero
the counter.

page 20
MERGEREL
COPY THE DATA

PAGE 28

(page 27)

IS IDENT2(L4) = counter?

Yes

no

MERGEREL
Read 2nd TPM

Move Tuple
Data From
Second

ADDTP
Add tuples from
second relation

PUT2P
Put tuples from
2nd rel.
into new rel.

Increment
counter.

"Tuples added
from second
relation resulting
in a total of X tuples.

BACK

E-29/(R-30 blank)
APPENDIX L

RORGRL
Subroutine: REORGREL

This subroutine is used to reorganize a relation. The purpose is to remove unwanted "dead space" from the .ADF. The deleted records are removed from the .ADF file then the revised .ADF is written back for storage.
**START**

- "Name of relation to be reorganized."

  **NAME**
  Get the relation name.

  **HASHIT**
  Hash the relation name.

  **FILEDS**
  Set up for disk operation.

  **RIVIDF**
  Bring tuple data into core memory.

- Is there an I/O Error?
  - Yes **RETURN**
  - No

  "Beginning reorganization."

  **ENIARY**
  Empty the IDIF and new tuple descriptor array.

  **FIDIF**
  Fill the IDIF array with zeros.

(ScrollBar)

L-2
Solve things in TDF that do not change.

Transfer the relation name into LDNT array.

RESET TDF
Clear the temporary file.

Open the tuple file, change access, prepare CATSTR RELDESA for rename.

DELETE TDF
IF IT EXISTS

Rename RELDESA TF to RELDESA.

RESET TDF
Clear the temporary file.

Open tuple data, change access, prepare RELDESA for rename.

DELETE TDF
IF IT EXISTS

Rename RELDESA TDF to RELDESA.

L-3
2. Open the ADF, change access, prepare RELDES for rename.
3. Delete TEMPFILE if it exists.
4. Rename RELDES to RELDES for access.
5. Create a tuple file.
6. Create the tuple data file.
7. Calculate the size of the ADF.
8. Create the ADF.

Were there any errors in TFORE: TFORE.
APPENDIX M

BKUPRL
Subroutine: BACKUPREL

This subroutine allows the user to backup or restore relational data. In backing up a user will copy the data from the primary storage media (usually system disk) to a secondary storage media (magnetic tape.) This protects the user's data and allows reconstruction in the event of a system failure.

In restoring a relation, data is transferred from magnetic tape back to the system. This utility is used primarily after a system failure. This subroutine is setup to transfer the .TF, .ADF, and .TDF files from the user storage volume to the nine track tape drive. The user must insure mounting of the proper tape and placing the drive in the proper configuration.

The following information is stored on magnetic tape during a backup:

1. The relation name
2. The date of backup
3. Read/write security keys
4. The Tuple Descriptor File (TDF)
5. The Tuple File (TF)
6. The Alpha Data File (ADF)
Do you wish to save an existing relation?
  TYPE on HD?
  YES/NO
  Get the yes or no answer

Use the answer 'yes'

Yes

Place tape on
drive in on-line
state. Press RETURN to
continue.

Wait for any
key to be
pressed.

Get Relation
Description

DATE
Get today's
date.

RETVEF
Retrieve the
IDF into core.

Were there any I/O Errors?
  yes
  RETURN
  no
  Oper. Tape
  File

A (page 2)  I (page 2)
(page 4)
BACKUP I

1. Get Relation Description
2. Open Tape File
3. Read the relation name from tape.
   Is the relation name read from tape the same as the one requested?
   no -> Write No Relation Data Message
   yes -> Is the read key equal to zero?
   yes -> "Enter read permission combination..."
   no -> "Incorrect read key..."
     Rewind the tape
     Close the file.
4. RETURN
 Were there any errors during TV, TDF, or ADF creation?

Were there any errors in creating the TV?

Were there any errors in creating the TDF?

Were there any errors in creating the ADF?

Was the error code 314?

'A Relation already exists with that name. Do you wish to backup to a new name (yes or no)'

Yes or no answer.

Yes or no answer.
BACKUP:

- Put away the ADF element.
- Rewind the tape.
- FILECLS: Close all tape files.
- Relation has been stopped with data, dated

RETURN
START

Setup using system "EXEC" function.

Has the tape file been successfully opened or have there been 3 tries?

yes

Open the tape file.

Were there any file open errors?

no

Tape file not opened properly.

"Place tape drive on drive in on-line state. Press RETURN to continue."

Wait for any character to be pressed.

no

Were there any errors?

no

yes

"Must be having tape problem. Start over."

RETURN
START

This tape does not contain the relational data you seek.

Rewind the tape.

Close the tape file.

BACK

START

ENVARY

empty the NAMELIST array

NAME

get the relation name

HASHT

Hash the relation name

FILES

Setup for disk I/O

BACK

GET

RELATION

DESCRIPTION
DISPLAY ERROR CODE

FILEERR Display the error code.

START

FILECLS Close the tape file

Rewind the tape

Tape input/output error

START

FILECLS Close the tape file

Back

Display the error code.

Tape is rewound for another try.

FILECLS Close the tape file

Rewind the tape

Tape input/output error
APPENDIX N

DBSTAT
Subroutine: DBSTAT

This routine gives the user the physical status of a relation. This information includes the size of the relation, the number of columns, the number of rows, the dead to active space ratio, as well as the column data, if desired.

DBSTAT allows for a printer copy of the relation status, if desired.
APPENDIX O

HELP
OVERLAY: HELP.FLC

SUBROUTINE: HELP.FLC

SYNTAX:
CALL HELP

THIS IS A SUBROUTINE ADDED TO FEASIL 77 TO AID THE USER IN MANIPULATION OF DATA RELATIONS UNDER FEASIL 77. THE VALID PARAMETERS (PASSED IN THE INTEGER (HOLLERITH) ARRAY "PARAM") ARE:

CREATE       - EXPLAINS THE RELATION CREATION UTILITY.
EDIT          - EXPLAINS EDIT COMMAND STRUCTURES.
DELETE        - EXPLAINS THE DELETE UTILITY.
MERGE         - EXPLAINS THE MERGE RELATIONS PROCEDURE.
MODIFY        - EXPLAINS COLUMN MODIFICATION PROCEDURE.
RETRIEVE      - EXPLAINS RETRIEVE AND MANIPULATION UTILITY.
REORGANIZE    - EXPLAINS NEED FOR REORGANIZATION.
STATUS        - EXPLAINS THE STATUS RELATION COMMAND.
BACKUP        - EXPLAINS THE BACKUP UTILITY.
COLUMN        - EXPLAINS RELATION COLUMN STRUCTURES.
RECORD        - EXPLAINS RELATION RECORD STRUCTURES.
VOLUME        - EXPLAINS THE SYSTEM VOLUME AND IT'S USE.
PROTECTION     - EXPLAINS USE AND NEED FOR PROTECTION KEYS.
DEVICE        - EXPLAINS SYSTEM DEVICE STRUCTURE.
STRATEGY      - EXPLAINS FOUR TYPES OF DATA STRATEGIES.
IF (NOT_FOUNDF

RETURN

(I.EQ.1)  T  * 1001

(I.EQ.2)  T  * 1002

(I.EQ.3)  T  * 1003

(I.EQ.4)  T  * 1004

(I.EQ.5)  T  * 1005

(I.EQ.6)  T  * 1006

(I.EQ.7)  T  * 1007

(I.EQ.8)  T  * 1008

(I.EQ.9)  T  * 1009

(I.EQ.10) T  * 1010

(I.EQ.11) T  * 1011

(I.EQ.12) T  * 1012

(I.EQ.13) T  * 1013

(I.EQ.14) T  * 1014

2 (page 3)

3 (page 3)
HELP PAGE 3

Diagram:

1. (page 2) (I.EQ.15) T (page 2) (ェIC15

2. 'NO HELP AVAILABLE YET FOR THAT FUNCTION'

3. RETURN

4. END

0-4
THE VALID HELP PARAMETERS ARE:
BACKUP           CREATE
DELETE           EDIT
MERGE            MODIFY
REORGANIZE       RETRIEVE
STATUS           COLUMN
RECORD           VOLUME
PROTECTION       DEVICE
STRATEGY

"CREATE" IS USED TO CREATE THE FILES ASSOCIATED WITH EVERY FEASIL RELATION. TO CREATE A FILE, YOU MUST KNOW HOW MANY COLUMNS THE RELATION REQUIRES, THE DESIRED PROTECTION KEYS ASSOCIATED WITH THE RELATION (IF DESIRED), AND THE COLUMN STRATEGIES FOR EACH OF THE COLUMNS CREATED. THE USER MUST ALSO KNOW THE NAME OF THE VOLUME ON WHICH THE RELATION IS TO BE CREATED. SEE HELP ON: PROTECTION, STRATEGY, COLUMN, VOLUME.

"EDIT" IS USED TO ENTER DATA INITIALLY OR TO MODIFY ALREADY EXISTING DATA. TO USE THE "EDIT" FEATURE, THE USER MUST KNOW THE NAME OF THE DESIRED RELATION, THE PROTECTION KEYS ASSOCIATED WITH THE RELATION, AND THE NAME OF THE VOLUME ON WHICH THE RELATION EXISTS. SEE HELP ON: PROTECTION, VOLUME.

"DELETE" IS USED TO PERMANENTLY REMOVE A RELATION FROM THE SPECIFIED VOLUME. TO USE THE "DELETE" FUNCTION, THE USER MUST KNOW THE RELATION NAME, THE PROTECTION KEYS, AND THE VOLUME NAME ON WHICH THE RELATION EXISTS. SEE HELP ON: PROTECTION, VOLUME.

"MODIFY" IS USED TO PERMANENTLY CHANGE THE:
A> COLUMN NAME, OR
B> NUMBER OF COLUMNS (DELETE OR ADD ONE).
THE USER MAY ALSO LIST THE COLUMN NAMES USING THIS FEATURE.
SEE HELP ON: COLUMN.

"REORGANIZE" IS USED TO INCREASE THE EFFICIENCY OF A FEASIL RELATION. IT IS PRIMARILY USED AFTER COLUMN SPECIFICATIONS HAVE BEEN MODIFIED (i.e., USING MODIFY) SEE HELP ON: COLUMN, MODIFY.

0-5
"RETRIEVE AND MANIPULATE" IS USED TO OPERATE ON AN EXISTING RELATION BY THE RELATIONAL DATA BASE OPERATORS (i. e., "AND", OR "OR"). IN ADDITION, R & M IS USED TO SORT RELATIONAL DATA BY COLUMNS, PRINT RELATIONAL DATA TO A DEVICE OR FILE, REPRODUCE A RELATION UNDER A DIFFERENT NAME, OR TO MOVE A SUBSET OF THE PRIMARY RELATION TO A NEW RELATION.
SEE HELP ON: COLUMN, DEVICE.

"BACKUP" IS USED TO SAVE A BACKUP (COPY) OF THE RELATION TO (FROM) MAGNETIC TAPE. IN ADDITION TO STORING THE RELATIONAL DATA TO THE TAPE, THE DATE OF BACKUP AS WELL AS AN OPTIONAL PROTECTION KEY SEQUENCE ARE ALSO STORED.

"STATUS" IS USED TO INQUIRE ABOUT THE SIZE OF A RELATION AS WELL AS TO INQUIRE ABOUT THE "DEAD" SPACE USED BY THE FILE. THE "DEAD" SPACE MAY BE MINIMIZED, IF DESIRED FOR EFFICIENCY USING THE REORGANIZE FEATURE.
SEE HELP ON : REORGANIZE.

"MERGE" SERVES TWO PURPOSES. IT'S FIRST PURPOSE IS TO PRODUCE A THIRD RELATION FROM TWO INPUT RELATIONS BY PERFORMING A "MERGE". THE MERGE FUNCTION IS A "UNION" OPERATOR. THEREFORE, THE OUTPUT RELATION CONSISTS OF THE TOTAL NUMBER OF UNIQUE COLUMNS IN THE TWO INPUT RELATIONS. THE ONLY RECORDS WHICH ARE MAINTAINED THROUGH THE "MERGE" ARE THE ONES WHICH INTERSECT ONE-TO ONE WITHIN THOSE COLUMNS WHOSE NAMES INTERSECT ONE-TO ONE.
SEE HELP ON: COLUMN, RECORD, STRATEGY.

"COLUMN" IS THE NAME GIVEN TO EACH CATEGORY OF RELATION. EACH COLUMN HAS A UNIQUE COLUMN NAME AND MAY BE REFERENCED ALONE OR WITH OTHER COLUMNS IN THE SAME RELATION (IF THERE ARE ANY). AN EXAMPLE OF COLUMNS IN A RELATION CALLED "MYCLASS" MAY BE:

1 - NAME
2 - AGE
3 - SEX
4 - SCORE
EACH RECORD OF THE RELATION HAS AN ENTRY FOR EACH COLUMN IN THE RELATION. THE RELATION, THUS, FORMS A DATABASE OF ENTRIES UNDER THE COLUMN HEADINGS. SEE HELP ON: RECORD.

"RECORD" IS A COLLECTION OF COLUMNS WHICH TOGETHER DESCRIBE THE DESIRED ASPECTS OF THE DATABASE. EACH RECORD IN A RELATION CONTAINS DATA FOR ALL OF THE COLUMNS OF THE RELATION. RECORDS MAY BE ACCESSED INDEPENDANTLY OR WITH OTHER RECORDS USING "RETRIEVE AND MANIPULATE". SEE HELP ON: COLUMN, RETRIEVE.

"VOLUME" IS THE NAME GIVEN TO BULK MAGNETIC MATERIAL USED IN COMPUTER INFORMATION STORAGE. FOR EASY LOCATION OF DATA, THESE VOLUMES ARE UNIQUE NAMES. TYPICAL VOLUME NAMES ARE:
- MT32
- MT6A
- PRO1
- TEMP

THE USER SHOULD INQUIRE ABOUT THE PROPER VOLUME TO USE FOR EACH APPLICATION VIA THE SYSTEM OPERATOR.

"PROTECTION". FEASIL -77 POSSESSES THE ABILITY TO OFFER A CERTAIN DEGREE OF PROTECTION AGAINST THE ACCIDENTAL DELETION OF RELATIONAL FILES. THOS SYSTEM OF PROTECTION INVOLVES THE USAGE OF PROTECTION "KEYS" ASSOCIATED WITH EACH RELATION. ON THE PERKIN-ELMER OS/32 SYSTEMS, THE READ/WRITE PROTECTION KEYS ARE CODED AS FOLLOWS:

<table>
<thead>
<tr>
<th>CODE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>UNPROTECTED. ANY KEYS WILL WORK WHEN ACCESSING THIS RELATION.</td>
</tr>
<tr>
<td>10-254</td>
<td>CONDITIONALLY PROTECTED. THE USER MUST KNOW THE CORRECT CODES TO ACCESS THE RELATION.</td>
</tr>
<tr>
<td>255</td>
<td>UNCONDITIONALLY PROTECTED. NO KEYS WILL ALLOW ACCESS TO THE RELATION.</td>
</tr>
</tbody>
</table>
"DEVICE" is the term used to identify a non-file peripheral attached to the computer to/from which data may be sent/received. Example devices are:

PR: (SYSTEM PRINTER)
CON: (USER CONSOLE)
PLOT: (SYSTEM PLOTTER)
MAG0: (9 TRACK TAPE DRIVE)

Contact the system operator for more information about the system's peripheral devices.

"STRATEGY" is the term used to refer to the type of data that is contained in a particular column. The valid strategies are:

"1" - INTEGER NUMBER,
"2" - DECIMAL (REAL) NUMBER,
"3" - SINGLE CHARACTER, AND
"4" - CHARACTER STRING.

See help on: COLUMN.
This library is a collection of subroutines used throughout FEASIL. A list of these subroutines and a brief description of their purpose is provided below.

**STRLLEN** - Returns the length of a string.

**FTIPI** - Converts floating point data to integer format.

**PITFP** - Converts integer format data back to floating point.

**YESNO** - Logical. Gets a "yes" or "no" answer from the console and returns true or false.

**GETNUM** - Get integer data.

**REANUM** - Converts a string into integer data.

**ELEB** - Eliminates leading edge blanks.

**ENFPTA** - Converts floating point data to ASCII format.

**ENINTA** - Converts integer data into ASCII format.

**GETFP** - Get floating point data.

**REAFP** - Converts string data into floating point.
STRLLEN

TRUNCATE
TRAILING BLANKS
FROM STRING

CALCULATE
TOTAL LENGTH
OF STRING

PLACE LENGTH
OF STRING INTO
ARRAY ELEMENT
#1 OF OUTPUT ARRAY

PUT CHARACTERS
4 PER ELEMENT INTO
ARRAY ELEMENT 2
THROUGH N OF
OUTPUT ARRAY

RETURN
GETNUM

GET

ACCEPT INPUT

STRLEN

RESTRUCTURE LINE

READNUM

CONVERT TO INTEGER WITH ERROR CHECK INT

RETURN
REANUM

SCAN ARRAY FOR '-' SIGN

DO CONVERSION FROM ASCII TO DECIMAL

ERRORS?

Y

ERROR MESSAGE
ERROR-AND-RETURN

N

RETURN EQUIVALENCE
DECIMAL OF ASCII
AS A SIGNED INTEGER

RETURN
ADD AiE NET COUNT EV:

fig 1

OE LE7E FIASt r

M/omn 51AIA'6

P-6
ENINTA

ENCODE INTEGER INTO I 12 FORMAT

ELEB

STRIP LEADING BLANKS

RETURN

ENFPTA

ENCODE DATA INTO ASCII STRING OF CHARACTERS, USES DIFFERENT FORMATS BASED ON MAGNITUDE OF DATA

ELEB

STRIP LEADING BLANKS

RETURN
GETFP

GET
ALLOW INPUT

STRLEN
RECONSTRUCT INPUT

REAFP
CONVERT FROM ASCII TO DECIMAL

RETURN
SCAN INPUT FOR A '—' SIGN

LOOK FOR A (*)

'.' FOUND?

TREAT REMAINDER AS A FRACTION

OUTPUT IS A WHOLE NUMBER PLUS FRACTION

RETURN

TAKE OUTPUT AS WHOLE NUMBER

P-9/(P-10 blank)
APPENDIX Q

TPLIB2
Library: TPLIB2

This library is a collection of subroutines used throughout FEASIL. A list of each of these subroutines and a brief description of their purposes is provided below:

- **PUTTDF** - Puts the IDENT and ITDF arrays on file.
- **RTVTDF** - Retrieves the IDENT and ITDF arrays from disk.
- **ADDTUP** - Adds tuple data to a relation.
- **DELTUP** - Deletes tuple data from a relation.
- **PUTIFO** - Put info into a relation.
- **PTFRC** - Puts data record into the tuple file.
- **PADFRC** - Puts data record into the Alpha Data File.
- **RTFRC** - Retrieves a record from the tuple file.
- **RADFRC** - Retrieves a record from the ADF.
- **PUTIFO** - Puts data into TF and ADF.
- **COLNAM** - Get the column names from the ADF.
- **HASHIT** - Hashes a file name into 8 characters.
- **UTFAR** - Retrieves TF alpha record.
PUT TDF

OPNTDF

OPEN TDF FILE

ERROR?

WRITE IDENT AND ITDF ARRAYS

FILECLS
CLOSE TDF FILE

RETURN
RTVTDF

QPNTDF

OPEN TF FILE

ERROR?

Y

READ IDENT AND ITDF ARRAYS

N

FILECLS
CLOSE TF FILE

RETURN

Q-3
ADD\textsc{TUP} \rightarrow \begin{array}{c}
\text{MOVE COMPLETE?} \\
\text{NO} \\
\text{RTFRC}
\end{array} \\
\rightarrow \begin{array}{c}
\text{GET NEXT RECORD}
\end{array} \\
\rightarrow \begin{array}{c}
\text{PTFRC}
\end{array} \\
\rightarrow \begin{array}{c}
\text{PUT INTO NEXT HIGHER RECORD}
\end{array} \\
\rightarrow \begin{array}{c}
\text{PTFRC}
\end{array} \\
\rightarrow \begin{array}{c}
\text{PUT NEW TUPLE IN THE NEWLY CREATED SPACE}
\end{array} \\
\rightarrow \begin{array}{c}
\text{UPDATE THE IDENT ARRAY}
\end{array} \\
\rightarrow \begin{array}{c}
\text{RETURN}
\end{array}
DELTUP

DONE?

Y

REDUCE THE "Number of Tuples" COUNT IN IDENT ARRAY

RETURN

RTFRC
READ THE NEXT TUPLE FILE (I+1)

RENUMBER AS I

PTFRC
WRITE THE TUPLE BACK OUT AS THE TUPLE I
UTIFO

TUPLE AS IT ON THE DISK

READ IN THE TUPLE AS IT ON THE DISK

IS NEW DATA A STRING?

Y

IS NEW STRING EMPTY?

Y

DOES STRING HAVE > 7 CHARACTERS?

Y

ADD STRING TO END OF ADF FILE

PADFRC

UPDATE IDENT

REPLACE TARGET ELEMENT WITH NEW DATA

INDICATE IN TF

PUT DATA IN TF (PACKED)

ERRORS?

Y

REPLACE UPDATED TUPLE

RETURN

Q-6
PTFRC

OPNTF
OPEN THE TUPLE FILE (TF)

ERRORS?
Y

WRITE DATA TO APPROPRIATE TF RECORD #

FILECLS
CLOSE TF FILE

RETURN

N
PADFRC

LENGTH OF STRING > ?

N

TFAR OVERRIDE ?

N

OPEN ADF FILE

ERROR ?

Y

LENGTH > MAX ?

Y

LIMIT TO MAX LENGTH

WRITE TO ADF FILE

RETURN

SET TFAR = .TRUE.

REGROUP THE CHARACTERS IN FORMAT TO BE PLACED IN TF

Q-8
RTFRC

UPNTF
OPEN THE TUPLE FILE

ERROR?

Y

N

READ TUPLE INFORMATION INTO ITFRC ARRAY

FILECLS
CLOSE TF FILE

RETURN
RADFRC

OPNADF
OPEN ADF FILE

ERROR?

READ RECORD
FROM ADF

FILECLS
CLOSE ADF

RETURN
MAHIT

PREPARE-LONG-FILENAME
MAKE ALL SPACES AND DASHES INTO A'S.
MAKE DIGITS 0-9 INTO A-J

PREPARE-HASHED-
EMPTY THE ARRAY
FILENAME

FORM A NEW CHAR.
FROM EVERY GROUP
OF 6 AS FOLLOWS
CHAR* (CHAR + CHAR ... CHAR)

DONE ?

RETURN

COLNAM

RADFRC
READ ADF RECORD
WHICH CONTAINS
COLUMN NAME FOR
COLUMN 'ICC'
REC = ETDF(2, ICC)

RETURN

Q-11
UTFAR

NEGATE THE 1st ELEMENT TO RETURN THE DATA TO ITS PROPER FORMAT

EXTRACT THE STRING LENGTH FROM THE UPPER BYTE OF THE FIRST WORD

UNPACK THE REST OF THE STRING. BYTES 2, 3, 4 of word one and bytes 2, 3, 4 of word two

PLACE DATA INTO STRING ARRAY

RETURN
This is a collection of subroutines used for file reallocation and testing throughout FEASIL. A list of each subroutine and a brief description of their purpose is provided below.

- **RDOADF** - Reallocates the size of the ADF.
- **RDOTF** - Reallocates the size of the TF.
- **CPYREL** - Copies the TF and ADF files.
- **FSLIDE** - Slides the first six elements of the argument one slot forward.
RDOADF

FILERE

RENAME ADF FILE TO A TEMPORARY NAME

CREATE A LARGER ADF FILE UNDER ORIGINAL RELATION NAME

CREATE ERROR?

1. T

OPNADF

OPEN THE TEMPORARY ADF

OPNADF

OPEN THE NEW ADF

T

COPY COMPLETE?

READ NEXT RECORD FROM TEMP. ADF

WRITE NEXT RECORD TO NEW ADF

FILECLS

CLOSE NEW AND TEMPORARY ADF

FILEDEL

DELETE TEMP. ADF

RETURN
FILEREN
RENAME TEMPORARY
FILE BACK TO
ORIGINAL RELATION
NAME

FILERR
DIAGNOSE ERROR

ISSUE WARNING
to the
USER

R-3
RDOTF

FILEREN
RENAME TF FILE TO A TEMPORARY NAME

CREATE LARGER TF FILE UNDER ORIGINAL RELATION NAME

CREATE ERROR?

1. T

OPNTF
OPEN THE TEMPORARY TF

OPEN THE NEW TF

COPY COMPLETE?

READ NEXT RECORD FROM TEMP. TF

WRITE NEXT RECORD TO NEW TF

FILECLS
CLOSE NEW AND TEMPORARY TF

FILEDEL
DELETE TEMP. TF

RETURN
1. RENAME TEMPORARY FILE BACK TO ORIGINAL RELATION NAME

2. DIAGNOSE ERROR
   ISSUE WARNING TO THE USER
COPYREL

TFCRE
CREATE NEW TF FILE FOR NEW RELATION

CREATE ERROR?

Y

ADFCRE
CREATE NEW ADF FILE FOR NEW RELATION

CREATE ERROR?

Y

COPY RECORD FOR RECORD, THE ORIGINAL TF FILE TO THE NEW TF FILE

COPY RECORD FOR RECORD, THE ORIGINAL ADF FILE TO THE NEW ADF FILE

RETURN

DIAGNOSE ERROR

WARN USER OF ERROR
FSLIDE

Move (EL+1) to (EL)

DONE 6 ELEMENTS?

EL = EL + 1

RETURN
The following is a complete list of subroutines included in the overlay TPFILECR.FLC with appropriate syntax for their use and a brief description of each routine. These routines will be described in detail on the following pages along with their flowcharts.

1. **FILDES(TOTFD,NAMFIL)**---creates file descriptor.
2. **SECURE(IWKEY,IRKEY)**---assigns security keys.
3. **EMIARY(IARRAY,N)**---empties the array "IARRAY" of size N.
4. **TFCRE(TOTFD,IWKEY,IRKEY,NSZTR,ISTTF)**---creates .TF file.
5. **ADFCRE(TOTFD,IWKEY,IRKEY,NSZADF,ISTADF)**---creates .ADF file.
6. **TDFCRE(TOTFD,IWKEY,IRKEY,NSZTDF,ISTTDF)**---creates .TDF file.
7. **RELFC(TOTFD,NSZTF,NSZTDF,NSZADF,IFATAL)**---creates relation.
8. **TFDEL(TOTFD,IWKEY,IRKEY,ISTTF)**---deletes .TF file.
9. **TDFDEL(TOTFD,IWKEY,IRKEY,ISTTDF)**---deletes .TDF file.
10. **ADFDEL(TOTFD,IWKEY,IRKEY,ISTADF)**---deletes .ADF file.
11. **RELFD(IDELST,NAMFIL)**---deletes relation
12. **EQUIVI(ARRAY1,ARRAY2,N)**---equates first N parts of 2 arrays.

**FILES BY LOGICAL UNITS.**

13. **OPNTF(TOTFD,IWKEY,IRKEY,ISOTF,LUTF)**---opens .TF file.
15. **OPNTDF(TOTFD,IWKEY,IRKEY,ISOTDF,LUTDF)**---opens .TDF file.
16. **FILERR(IS1ATUS,NOMATCH)**---Diagnoses file I/O error codes.
17. **NAME(I)**---gets name "I".
SUBROUTINE: FILDES.FLC

SYNTAX: CALL FILDES(TOTFD,NAMFIL)

THIS ROUTINE INTERACTIVELY COMPLETES THE CONSTRUCTION OF THE FILE DESCRIPTOR, READ AND WRITE KEYS, AND STORAGE VOLUME. NAMFIL IS AN INPUT NAME ARRAY DEFINED IN SUBROUTINE "NAME" AND TOTFD IS THE TOTAL FILE DESCRIPTOR FOR USE WITH STRING ROUTINES SUCH THAT: TOTFD = <VOLUME NAME>:<FILE NAME>.<EXT>.
SUBROUTINE: SECURE.FLC

SYNTAX: CALL SECURE(IWKEY, IRKEY)

THIS ROUTINE OBTAINS THE SECURITY READ/WRITE KEYS FOR THE RELATION, IRKEY AND IWKEY, RESPECTFULLY.
SECURE

DEFINE VARIABLES AND COMMON BLOCKS. SET LOGICALS TRUE.

ENTER READ PROTECTION KEY (DEFAULT = 0).

GETNUM (IRKEY, IERR, LEN)

IF (LEN .EQ. 0) THEN
  SET READ KEY (IRKEY) TO ZERO.
  SET READ ERROR (IERR) TO FALSE.
ELSE IF (IRKEY .LT. 256) OR (IRKEY .GE. 1)
  SET READ ERROR (IERR) TO TRUE.
ENDIF

IF (IERR) THEN
  NUMBER OUT-OF-RANGE (0-255)
ELSE
  UNTIL (NOT IERR)
END IF

ENTER WRITE PROTECTION KEY (DEFAULT = 0).

GETNUM (IWRITE, IERR, LEN)

IF (LEN .EQ. 0) THEN
  SET WRITE KEY (IWRITE) TO ZERO.
ENDIF

IF (IERR) THEN
  SET WRITE ERROR (IWRITE) TO TRUE.
ENDIF

EQUATE LOCAL VARIABLES TO GLOBAL VARIABLES

RETURN
SUBROUTINE: EMIARY.FLC

SYNTAX: CALL EMIARY(IARRAY,N)

THIS ROUTINE INITIALIZES AN INTEGER ARRAY NAMED "IARRAY" AND OF DIMENSION "N" TO EMPTY.
SUBROUTINES ADFCRE.FLC, TFCRE.FLC, AND TDFCRE.FLC

SYNTAX: CALL ***CRE(TOTFD,IWKEY,IRKEY,NSZ***,IST***)
WHERE *** IS EITHER TF, TDF, OR ADF.

THIS ROUTINE ADDS THE EXTENSION *** TO THE TOTFD FILE TO
CREATE THE TUPLE FILE (TF) DESCRIPTOR, THE ALPHA DATA FILE (ADF),
OR THE TUPLE DESCRIPTOR FILE (TDF). THE APPROPRIATE FILE: EITHER
TF, ADF, OR TDF.IS THEN CREATED AS A CONTIGUOUS FILE OF SIZE
NSZ*** 256 BYTE SECTORS.
FUNCTION EMARRAY (IARRAY, N)

START

DIMENSION IARRAY, J = 0

IF IJSECT J, IARRAY(J) = 0

UNTIL (J .LE. N)

RETURN

TFCRE, ADFCRE, TDFCRE

XXXCRE (FD, FNAME, JNAME, JSTNAME)

START

DEFINE VARIABLES AND COMMON BLOCKS

(EQUD) (FD, JCLASS)

EQUIVALENCE (FD) TO TDFD

UNTIL (J .LE. N)

RETURN

XXX is either TF, ADF, or TDF. These are three identical subroutines used to create the appropriate files.
SUBROUTINE RELFC.FLC

SYNTAX: CALL RELFC(TOTFD,NSZTF,NSZTDF,NSZADF,IFATAL)

THIS ROUTINE IS TITLED THE "RELATIONAL FILE CREATOR" AND ITS PURPOSE IS TO CREATE ALL DATABASE AND TEST MATRIX STORAGE FILES AT THE INITIAL TIME. THE VARIABLE "IFATAL" IS THE FILE CREATION RETURN STATUS INDICATOR. IF "IFATAL"=0 THEN THERE IS NO ERROR. IF IT EQUALS 1 THEN THERE IS NOT ENOUGH ROOM ON THE STORAGE VOLUME.
SUBROUTINES TFDEL.FLC, ADFDEL.FLC, AND TDFDEL.FLC

SYNTAX: CALL ***DEL(TOTFD,IWKEY,IRKEY,IST***)
WHERE *** EQUALS EITHER TF, ADF, OR TDF.

THESE ROUTINES ADD THE EXTENSION *** TO THE TOTFD FILE TO
CREATE THE TUPLE FILE (TF), THE ALPHA DATA FILE (ADF), OR THE
TUPLE DESCRIPTOR FILE (TDF) DESCRIPTOR. THE APPROPRIATE FILE IS
THEN DELETED.
SUBROUTINE RELFD.FLC

SYNTAX: CALL RELFD(IDELST,NAMFIL)

THIS ROUTINES TITLE IS "RELATIONAL FILES DELETER. IT DELETES ALL RELATIONAL DATABASE AND TEST MATRIX FILES ALL AT ONCE. "IDELST" IS A RETURN STATUS INDICATOR. IF IDELST EQUALS 0 THEN THERE IS NO ERROR. IF IDELST EQUALS 1 THEN THERE IS AN UNKNOWN ERROR. THE VARIABLE "NAMFIL" IS THE NAME OF THE RELATION TO BE DELETED. THE FORMAT FOR THIS VARIABLE IS DEFINED IN THE SUBROUTINE "NAME".

THIS ROUTINE IS PRESET TO ALLOW THE USER THREE TRIES TO DELETE THE FILE. IF THE USER IS NOT SUCCESSFUL IT IS ASSUMED THAT HE EITHER DOES NOT KNOW WHAT HE IS DOING OR HE IS TRYING TO DELETE A FILE HE SHOULD NOT BE DELETING. AFTER THREE TRIES THE JOB IS TERMINATED.
RELFD (INEQ. ST. #1 = 1)

START

DEFINE VARIABLES AND COMMON BLOCKS

INITIALIZE FILE DELETION INDICATORS
ISTTF = 20
ISTDF = 20
ISTDF = 20

FILEDS

(TOTFD, NWFD, EKRD, ISECF, SEKRD, XEKRD)

TDFDEL

(IF (ISTTF.EQ.0))

(TOTFD, NWFD, EKRD, ISECF, SEKRD, XEKRD)

TDFDEL

(IF (ISTTDF.EQ.0))

(TOTFD, NWFD, EKRD, ISECF, SEKRD)

WRITE-FILNAME

WRITE-PROJECT

WRITE-FILE

WRITE-UNK

WRITE-FILE-PREP

RETURN
SUBROUTINE EQUIVI.FLC

SYNTAX: CALL EQUIVI(ARRAY1,ARRAY2,N)

THIS ROUTINE EQUIVALENCES THE FIRST N ELEMENT OF INTEGER ARRAY1 TO INTEGER ARRAY2
SUBROUTINES OPNTF.FLC, OPNADF.FLC, AND OPNTDF.FLC.

SYNTAX: CALL OPN*** (TOTFD, IWKEY, IRKEY, ISO***, LU***)
WHERE *** EQUALS TF, ADF, OR TDF.

THESE ROUTINES OPEN THE TUPLE FILE (TF), THE ALPHA DATA FILE
(ADF), OR THE TUPLE DESCRIPTOR FILE (TDF) FOR THE RELATION NAMED
IN THE TOTFD.

********** WARNING ****** WARNING ********

ALL FILES ARE OPENED AS SHAREABLE READ/WRITE
(i.e., IAP-4)

TF FILES ARE LOCATED ON LOGICAL UNIT LU-LUTF.
ADF FILES ARE LOCATED ON LOGICAL UNIT LU-LUADF.
TDF FILES ARE LOCATED ON LOGICAL UNIT LU-LUTDF.
IST IS AN ERROR INDICATOR. FOR MORE INFORMATION REFER TO
THE FTN LIB REAL TIME EXTENSION MANUAL.
EQUIV (AFILE, AFILE, ..., N)

IF (FILEK) THEN FILEK = 0

SET F = 0

DEFINE
VAR. FILES
SET F = 0

J = J + 1
IF (J = N) THEN RETURN

UNTIL (J EQ N)

RETURN

SETUP FILES AS SHAREABLE
READ/WRITE (SPN)

FILEK = (FDNAME, IX)
INITIALIZE FILE DESCRIB.

ADD .EXTENSION

J = 1
UNTIL (J EQ L)

RETURN

IF (NAMEIX.EQ.1) THEN RETURN

IF (NAMEIX.NE.1) THEN RETURN

OPENAR (TDFFD, INKEY, INKEY, ISHAR)

J = J + 1
K = J + 1
FDNAME(J) = FD(K)

UNTIL (J EQ ISHAR)

FILEGV (GNAME, FDNAME, IAP, HRD, ISHAR)

XMM = TR, TDF, OR ADF.

S-17
SUBROUTINE FILERR.FLC

SYNTAX: CALL FILERR(ISTATUS,NOMATCH)

This routine is used to diagnose basic file error codes. "ISTATUS" is the input variable containing the error status code. "NOMATCH" is a return variable that, if equal to 1 indicates that the error is unknown and the user should consult the system operator. If "NOMATCH" does not equal 1 then the appropriate error message is printed on the user's console.
FILEREC (STAT, NOMATCH)

START

NOMATCH = \emptyset

\text{IF } (\text{ISTAT} \leq 50) \land (\text{ISTAT} \geq 51) \text{ Then }

\text{'READ/ WRITE KEY PROTECTION CODE ERROR.'}

\text{IF } (\text{ISTAT} = 65) \land (\text{ISTAT} \neq 64) \text{ Then }

\text{'FILE DOES NOT EXIST.'}

\text{IF } (\text{ISTAT} = 62) \text{ Then }

\text{'SPECIFIED VOLUME NOT MOUNTED.'}

\text{IF } (\text{ISTAT} \neq 68) \lor (\text{ISTAT} \neq 69) \text{ Then }

\text{'INVALID FILENAME/ DESCRIPTOR.'}

NOMATCH = 1

RETURN
SUBROUTINE NAME.FLC

SYNTAX: CALL NAME(I)

THIS ROUTINE REQUESTS THE USERS FILE NAME AND THEN RECONSTRUCTS IT INTO A SUITABLE FORM FOR A FEASIL FILE NAME. THE FILE NAME CAN BE A MAXIMUM OF FORTY-TWO (42) CHARACTERS LONG AND THE FIRST CHARACTER MUST BE ALPHABETIC. THE NAME(I) IS RETURNED IN AN INTEGER ARRAY FORMAT WITH A MAXIMUM OF THIRTEEN ELEMENTS. THE ELEMENTS ARE:

I(1)=NUMBER OF CHARACTERS IN THE NAME,
I(2)=THE FIRST 4 CHARACTERS OF THE FILE NAME,
I(3)=THE SECOND 4 CHARACTERS OF THE FILE NAME,

. . . . . . . . . . . . .
. . . . . . . . . . . . .
I(13)=THE TWELFTH 4 CHARACTERS OF THE FILE NAME.
NAME (I)

START

DEFINE VARIABLES AND COMMON BLOCKS

GET (I,N,WORD, END, FIL)

SUBROUTINE (N, NAM, FIL, LEN, FDIM)

IF (LEN .GE. 42) T

* 201 'NAME TOO LONG - MAX = 42 CHARACTERS MAX - MUST NOT BE LONGER THAN THIS.'

GEICH (NAM, FIL, LEN, FDIM)

IF (ITYPE .EQ. 1) T

RETURN
TPFILECR
PROCEDURES ASSOCIATED
WITH SUBROUTINE RELFD
PAGE 10

TO WRITE - FILE NAME ERROR

SPECIFIED FILE DOES NOT EXIST ON SPECIFIED VOLUME.
RENAME OR CHANGE VOLUMES.

IDELST = 2

RETURN

TO WRITE - FILE PROTECTION ERROR

FILE PROTECTED AND YOUR KEYS DO NOT MATCH !!!!!

SELECT = SELECT + 1
IDELST = 2

RETURN

TO WRITE - FILE OPEN ERROR

FILE IS PRESENTLY OPEN AND CANNOT BE DELETED WHILE OPEN.

IDELST = 2

RETURN

TO WRITE - UNKNOWN ERROR

UNKNOWN ERROR HAS OCCURRED - SEE SYSTEM MESSAGE FILE.

IDELST = 1

RETURN

*170 'YOU HAVE HAD 3 TRIES TO DELETE A PROTECTED FILE.
YOU OBVIOUSLY DO NOT KNOW THE PROPER KEYS. FIND THEM
AND TRY AGAIN LATER.'
TPFILECR
PROCEDURES ASSOCIATED
WITH SUBROUTINE RELFC
PAGE 11

TO-WRITE-
VOLUME-ERROR

TO-WRITE-
SIZE-ERROR

TO-WRITE-
UNK-ERROR

1 SPECIFIED
VOLUME IS NOT
MOUNTED - CORRECT
AND START OVER.

IFATAL
= 2

RETURN

IFATAL
= 1

RETURN

UNKNOWN
ERROR HAS
OCCURRED - SEE
SYSTEM
PROGRAMMER.

IFATAL
= 1

RETURN

TO-WRITE-
FILENAME-ERROR

TO-WRITE-
DEVICE-ERR

FILE NAME
ALL READY EXISTS
ON SPECIFIED
VOLUME, RENAME
OR CHANGE VOLUME.

IFATAL
= 2

RETURN

SPECIFIED
VOLUME IS NOT
A DIRECT-ACCESS
DEVICE!!

IFATAL
= 2

RETURN

8-23/(8-24 blank)
Subroutine: F710

This routine is a collection of machine dependant subroutines that are concerned with file manipulation (opening, closing, copying, etc.) Those shown here are used on the VAX (FAST).
The subroutines are:

FILEOPN- opens a file
FILEDEL- deletes a file
CREAFIL- creates a file
FILEREN- renames a file
FILECLS- closes a file
VAXERR - gives the meaning of VAX error codes.
F710

START

FILEOPEN

Copy FD to filename using 84A format

Open filename on UF as OLD, DIRECT, 64 long

RETURN

FILEDEL

START

FILEOPEN

Open FD on unit 2

Were there any errors?

yes

RETURN

no

Close unit 2 with 'DELETE' status.

RETURN
START

ILR = LRL/4

ILR = 0 ?

yes

no

ENCODE PD using 844 into FILEMN

Open FILEMN on unit 2 as OLD with I/O status STAT

Close unit 2

Is STAT = 0 ?

yes

no

STAT = 314

"Error---file already exists."

RETURN

Open FILEMN on unit 2 as New, Direct. ILR long records, SC initializes also

Close unit 2

RETURN

T-3
START
Copy FN to NEW

INQUIRE name POLD of file on unit LW

Close LW

IST=LIBLENAME (PODL,FNEM)

status = 0

Is MOD (IST,8) not equal to 1?

yes
status = 1

no

Is MOD (IST,8) equal to 2?

Status = MOD (IST/8,255)

yes

no

Is STATUS = 0?

yes
Open NEW on LU as OLD, Direct

no
Open OLD on LU as OLD, Direct.

Were there any IO errors?

yes

Status = -1

no

RETURN
What is the error code?
(conditional)

1.

93.

65, 69, 73, 76, 78, 79.

undefined

NFOUND = 1

else

unknown

NFOUND = 1

RETURN

T-5/(T-6 blank)
DISTRIBUTION

ITT Research Institute
ATTN: GACIAC
10 West 35th Street
Chicago, IL 60616

US Army Materiel System
Analysis Activity
ATTN: AMXSYS-MP
Aberdeen Proving Ground, MD 21005

AMSMI-RD, Dr. McCorkle
   Dr. Rhoades
   -RD-SS-SE, Dr. J. D. Marr
      B. E. Tucker
      T. A. Palmer
      D. C. Pool
      T. N. Long

-RD-CS-R
-RD-CS-T
-RD-GC-IP, Fred Bush

1
1
1
5
1
1
1
1
15
1
1
1

Dist-1/(Dist-2 blank)
END
5-87
DTIC