

AN IMPLEMENTATION OF A FORTRAN SOURCE CODE REARRANGER AND  
DOCUMENTATION GENERATOR PROGRAM/

by

ARTHUR SCOTT MYERS

B.S., Lehigh University, 1969

---

A MASTER'S REPORT

submitted in partial fulfillment of the

requirements of the degree

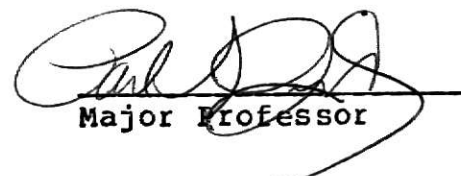
MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY  
Manhattan, Kansas

1983

Approved by:

  
Major Professor

LD  
2668  
R4  
1983  
M93  
C.2

A11202 572290

TABLE OF CONTENTS

|      |                                                 |    |
|------|-------------------------------------------------|----|
| I.   | INTRODUCTION.....                               | 1  |
| II.  | DESIGN REQUIREMENTS.....                        | 6  |
| III. | DESCRIPTION OF IMPLEMENTATION.....              | 12 |
| IV.  | FUTURE WORKS AND EXTENSIONS.....                | 35 |
| V.   | CONCLUSIONS.....                                | 39 |
|      | APPENDIX A - PROGRAM SOURCE CODE LISTING.....   | 43 |
|      | APPENDIX B - USER'S MANUAL.....                 | 44 |
|      | APPENDIX C - EXAMPLES OF ANALYZED PROGRAMS..... | 48 |
|      | BIBLIOGRAPHY.....                               | 49 |

## I. INTRODUCTION

One of the most important, and unfortunately most neglected needs in the software development process today is documentation, especially documentation that aids the programmer or systems analyst in readily understanding computer programs' source code. The intent of this implementation project is to address that need by developing a software tool that will provide documentation showing a program's hierarchical top-down order and alphabetical listing of its modules; it will also rearrange the program's source code into this top-down order of modules.

This documentation should have a very high utility to the programmer because it will concisely show, typically on one or two pages, a complete program's overall structure and detailed inter-module relationships via the report that shows the top-down order of modules. This could be viewed as a table of contents for the program's source code, because it will show each module's location within the source code file. It should be even more useful since the same report will also contain structure, size and inter-module relationship information for each module. The report displaying the alphabetical listing of modules can be thought of as an index to the program, but it too will contain information that goes beyond that function. Furthermore, since the source code file will actually be rearranged into its hierarchical top-down order, an implicit documentation feature will come about because when one reads the source

code it will be in logically related order. Thus, if the reader wants to look at the code of a module called by the module he or she is currently studying, it will usually be a matter of flipping to the next page or two, in most cases, instead of having to make a time consuming search through the entire source code listing.

Because of the documentation and organization capabilities that this software tool will provide, it should prove to be quite valuable to the programmer enabling he or she to improve both productivity and quality with the following benefits:

- (1) ordering of source code in top-down order
- (2) program documentation shown in top-down and alphabetical order
- (3) understanding of program structure
- (4) rudimentary measure of software quality - e.g. program unit sizes, percent of comments, etc.
- (5) improved productivity in developing new programs
- (6) greater ease of maintaining and enhancing existing programs

While using this tool in developing and maintaining one's own software should be of considerable worth, it should be even more useful when one has to work with programs developed by someone else. Furthermore, the rudimentary measures of software quality provided should be of value to the software manager.

Previous work that has been done in this area can be



classified into two approaches: (1) analyzing module relationships of FORTRAN programs which centers around inter-module communications and how the data shared between the modules are affected and (2) identifying basic blocks of FORTRAN programs. The first approach constructs the call graph of a FORTRAN program [2] in order to examine interprocedural communications. This call graph serves as a data structure for providing static representation of interprocedural relations for nonrecursive programs. The algorithm for constructing the call graph is used in the PFORT Verifier [3], a program which checks FORTRAN programs for compliance with a portable subset of FORTRAN. The uses of the call graph in the Verifier are:

- (1) checking the legality of module linkage (i.e. the matching of parameter number, usage, type, and structure)
- (2) checking for unsafe references to interprocedural communications which may have implementation dependent results
- (3) checking the sharing of global data areas
- (4) outputting a depiction of the call graph for use as documentation and as a debugging aid

The construction of a program's call graph differs from this project because it does not actually rearrange the source code, nor does it create the tabular documentation showing the program's top-down order (table of contents) and its alphabetical listing of modules (index). On the other hand,

this project does no checking of parameters passed between modules or detecting of non-portable features in the program.

In the second approach, a FORTRAN program was written to identify basic blocks in FORTRAN programs [1] where a basic block is defined as a sequence of consecutive FORTRAN statements which must be executed consecutively; that is, if one statement in the block is executed, all are executed. Use of this program allows for monitoring of the execution sequence and control path analysis of a program. This approach differs from the source code rearranger about to be described because it must perform a rather complex analysis within modules in order to determine the boundaries of the basic blocks, whereas the source code rearranger is concerned about only module boundaries and any inter-module calls that may reside there. And, as with the first approach, no rearranging of source code and related documentation is obtained via this basic block identifier.

Both of these approaches address related but lower level concerns than does this implementation project. The output from the code rearranger program will be easily understandable to any programmer and possibly non-programmers and will have the primary value of aiding the programmer in developing, maintaining, and enhancing software. Thus this software tool will have legitimate use on a day-to-day basis. In the approach that constructs the call graph, considerable effort was directed toward formalizing the algo-

rithm and proving its correctness. In the identifying basic blocks approach, great care was taken to adhere to the definition of a basic block and to gracefully handle the exceptional conditions that occur. Thus, one could also view the emphasis of this project as being more practical and less theoretical than the two approaches just described.

## II. DESIGN REQUIREMENTS

In considering design requirements for this implementation project, it was decided that this code rearranger program should be able to process both FORTRAN 66 and FORTRAN 77 programs and be capable of handling fairly large programs (up to 4000 lines of source code) that are developed and run in a mini-computer environment. Recognizing that both FORTRAN 66 and FORTRAN 77 are in use today, a higher utility of this program could be achieved by being able to process both versions' source code files. Being able to analyze programs with up to 4000 lines of source code would allow for this program to serve as a software tool for much of the FORTRAN software being written on mini-computers today. With the increasing power, and therefore usage of mini-computers, brought about primarily because of technology gains in hardware, this FORTRAN source code rearranger program could have a large number of potential users, assuming, of course, that it would be portable from one machine to another. Thus, the above considerations were made with the objective of making the results of this project as widely available and useful as possible.

In order to prevent the design of this project from becoming overly complex, it would be clearly stated to any users that satisfactory performance of this program could be guaranteed only if it was used AFTER successful compilation of the FORTRAN program that was about to be analyzed and rearranged. Obviously, if this program had to embody the

capability to detect syntax and semantic errors pertaining to module structure, it would be much more complex than if the resident FORTRAN compiler processed programs prior to the rearranger program doing its analysis. Thus, with the stipulation that the code rearranger program would process only "clean" FORTRAN source code, designing of this post compiler tool became a very doable project.

The program would detect errors that are typically discovered by the linking-loading process, but it would be good and efficient practice to use this program AFTER the linking-loading of the FORTRAN program had successfully been done. By successfully performing the linking-loading step prior to using this program, the most correct program possible would be passed to the source code rearranger program. Thus, the highest probability of successfully rearranging the program on the first attempt would be assured.

Next, it was determined what general tasks this program would have to perform so that it would be a desirable and valuable aid to the programmer. The three general tasks selected were:

- (1) rearrange the program's source code file into hierarchical top-down order of modules. This rearranging would be successful regardless of the order of the modules in the original source code file as long as the original source code file was complete, i.e. contained all of the program's modules.

- (2) generate documentation showing the top-down order of

program modules. This report includes, for each module, the beginning and ending source code line numbers; the level of its call ('main' is defined as level 1); the number of lines of code; the number of comment lines; the number of declaration lines; the number of format lines; and the number of executable lines. Also, this report would print out totals for these module statistics. Thus, this report serves as a table of contents to the now hierarchically ordered source code file.

(3) generate documentation showing the alphabetical listing of the program's modules. This second report shows, for each module, the beginning and ending line numbers; the number of lines; and the number of times called. This report would also display the range and average of modules' size and percent of total lines of code by comment, declaration, format, and executable. This second report would become the index to the rearranged source code file.

With these three outputs of the source code rearranger program: (1) the rearranged source code, (2) the table of contents, and (3) the index, the programmer has a very useful documentation package which has mapped the entire source code's logical structure to its physical structure while also allowing for quickly determining specific module information and relationships. This should prove to be a valuable tool for developing, maintaining, and enhancing FORTRAN

software.

After defining the general tasks, a list of specific considerations was developed that this program would have to be capable of handling. Taking into account the expected environment of the FORTRAN programmer, the following requirements were formulated:

(1) In striving toward the goal of having a useful and practical tool for aiding the FORTRAN programmer, the design requirement of processing FUNCTION modules in addition to MAIN and SUBROUTINE modules was deemed important and necessary. The code rearranger program would give functions the same precedence as subroutine modules in constructing a program's calling hierarchy but would identify each module as "MAIN", "SUBROUTINE", or "FUNCTION" in the subsequently outputted documentation.

(2) Recognizing that virtually all FORTRAN programs contain library calls (calls to modules that exist in the system library, not in the program's source code), it was determined that this program must be able to detect such module calls and process them appropriately. Any call in the source code referencing a module not found in the source code would be assumed to be a library call and hence would simply be ignored when building the module hierarchy.

(3) Being aware that there are now versions of FORTRAN, e.g. HP-3000, that allow for recursive routines (modules

that call themselves), the requirement to detect and properly document recursion became necessary. If recursive calls were not detected and so labeled, then the report showing the hierarchical relationship of modules would grow to infinite length.

(4) A common error that can easily occur when developing FORTRAN programs of even moderate size is duplicate naming of modules, which will not be properly detected by the compiling and linking-loading processes. Because of this fact, they can be especially troublesome to the programmer as he attempts to debug his program. By being able to detect duplicate module names, this program not only assures the integrity of the ultimately rearranged program, but also performs a valuable debugging service.

(5) Another common error that happens when writing FORTRAN software is having extraneous modules in the source code - those which are never called by any other program module. This error, while not fatal, does undermine good housekeeping and space conservation practices. The code rearranger program will naturally expose any extraneous modules as it does its hierarchy analysis.

(6) Because FORTRAN allows for EXTERNAL modules, those passed as arguments in one module, but defined in another module, the capability for proper analysis of the condition must exist in this program. Without this ca-



pability, an incorrect analysis of the hierarchy of module relationships would result.

(7) Finally , keeping in mind that this program must be able to process moderately large programs (up to 4000 lines of code), the determination as to how to handle data storage (arrays in CPU and/or disk files) during program execution was made so as to optimize program execution time. A satisfactory compromise was to handle the input and output source code as a disk file while all other data requirements were to be processed using arrays residing in CPU.

### III. DESCRIPTION OF IMPLEMENTATION

In order to generate the source code file in top-down module order and associated documentation, the computer program that was implemented must create from the input source code file (whose modules can be in any sequence) a table that includes all the module boundaries with all module calls in the proper order within these boundaries. Constructing the table may require up to three passes through the source code, if FUNCTION(S) and/or EXTERNAL subroutine(s) are present. When the table is complete, the rearranged source code file can be output by traversing through this table utilizing a stack in order to determine the hierarchical relationships of the modules. The hierarchy diagrams of Figures 1-8 show an overview of the program design. We will now present a thorough explanation of how this program accomplishes the rearranging of modules and the related report generation:

- (1) A first read of the entire source code file is performed so that a 'reduced' array containing only program unit statements (e.g. SUBROUTINE OR FUNCTION) and CALL statements is built. During this first pass, using subroutines FINDSEGBEGIN and CLASSIFYSTATEMT to analyze each program unit, every statement is classified as comment, declarative, format, or executable and the beginning and ending line numbers are determined. This information is also stored in the 'reduced' array. Also, for each program unit name appearing in a program unit

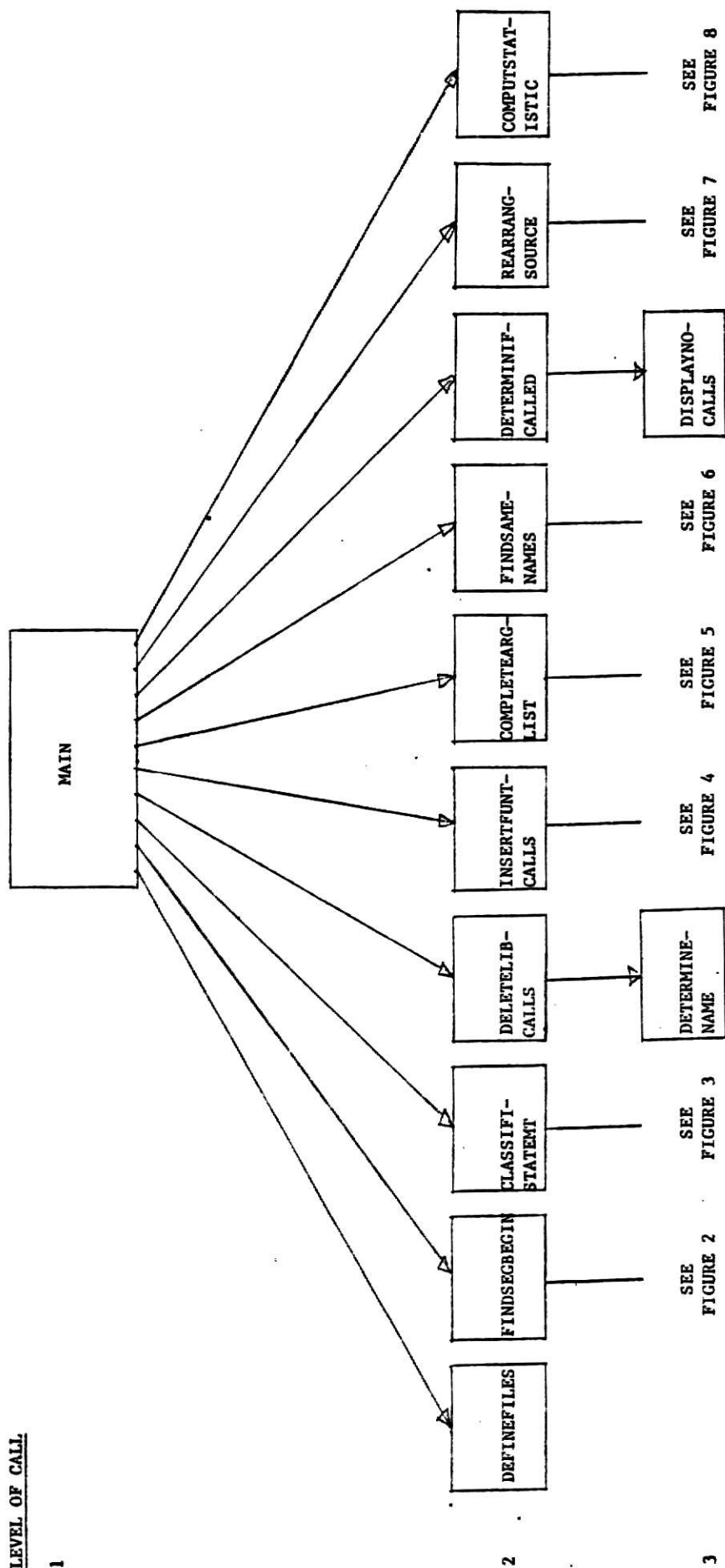


FIGURE 1. HIERARCHY DIAGRAM OF SOURCE CODE REARRANGER PROGRAM

NOTE: MODULES ARE EXECUTED IN VISIT-LEFT-RIGHT SEQUENCE (PREORDER)

LEVEL OF CALL

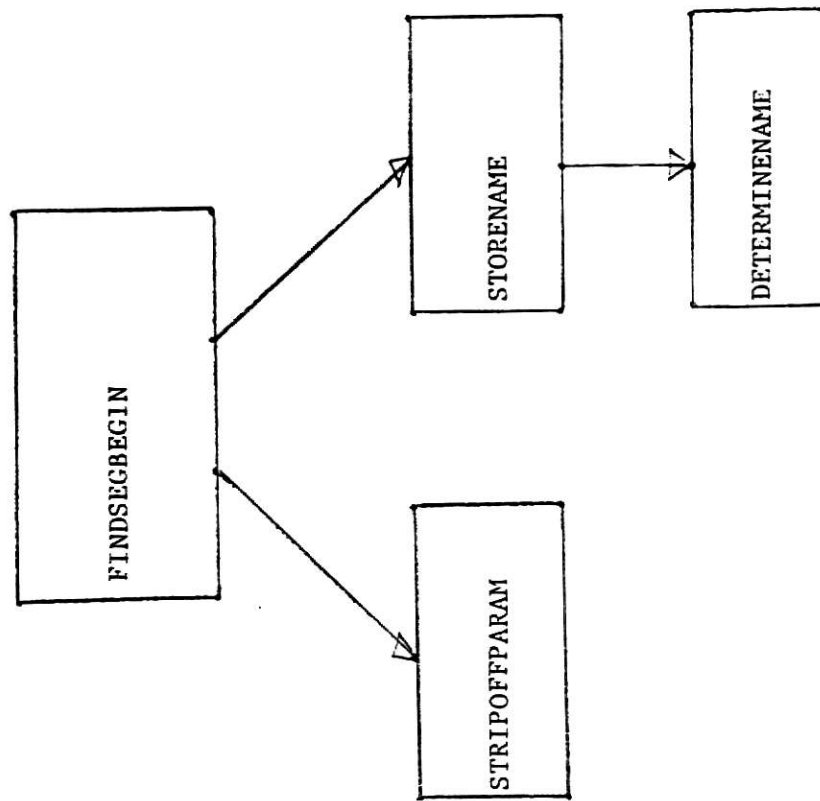


FIGURE 2. HIERARCHY DIAGRAM (CONTINUED)

LEVEL OF CALL

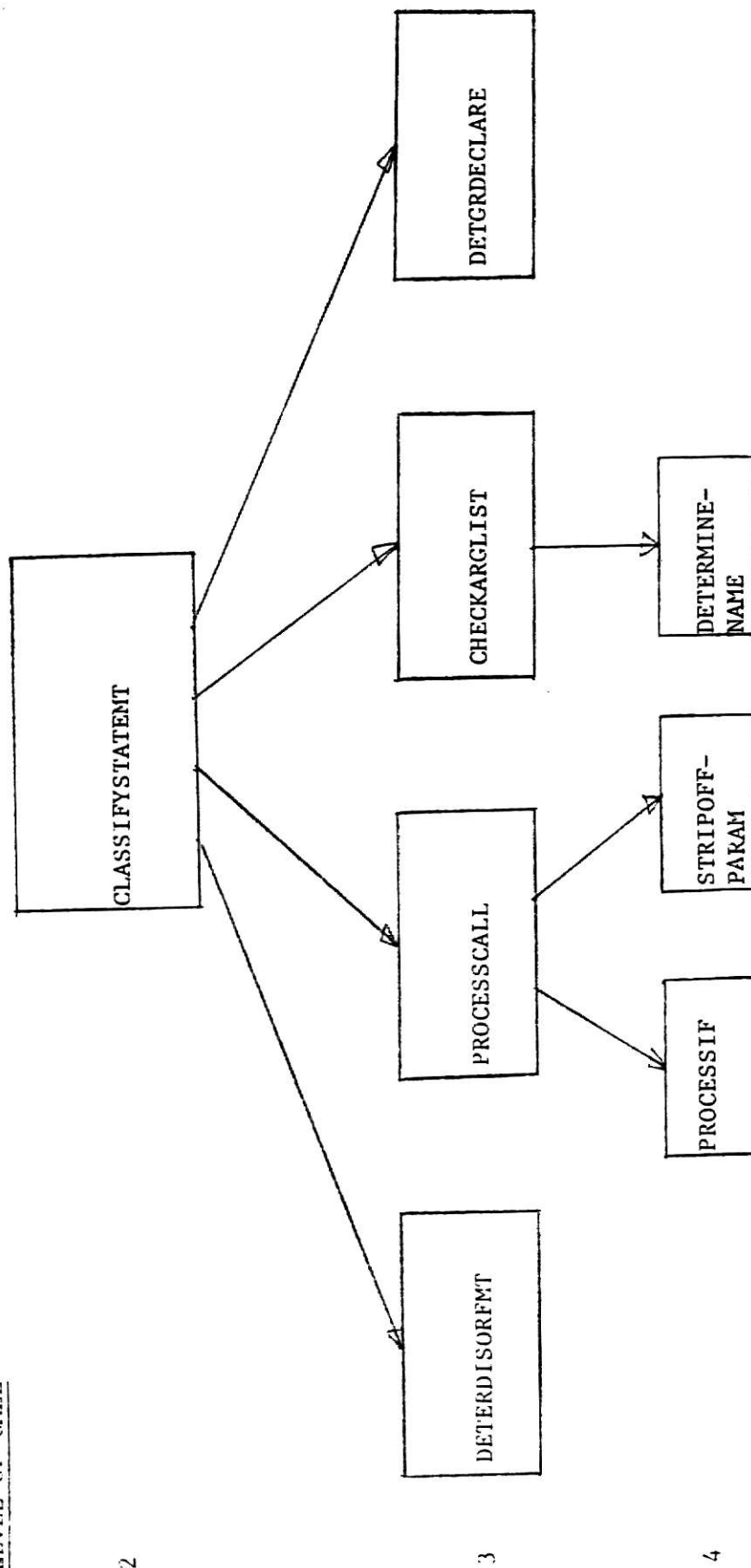


FIGURE 3. HIERARCHY DIAGRAM (CONTINUED)

LEVEL OF CALL

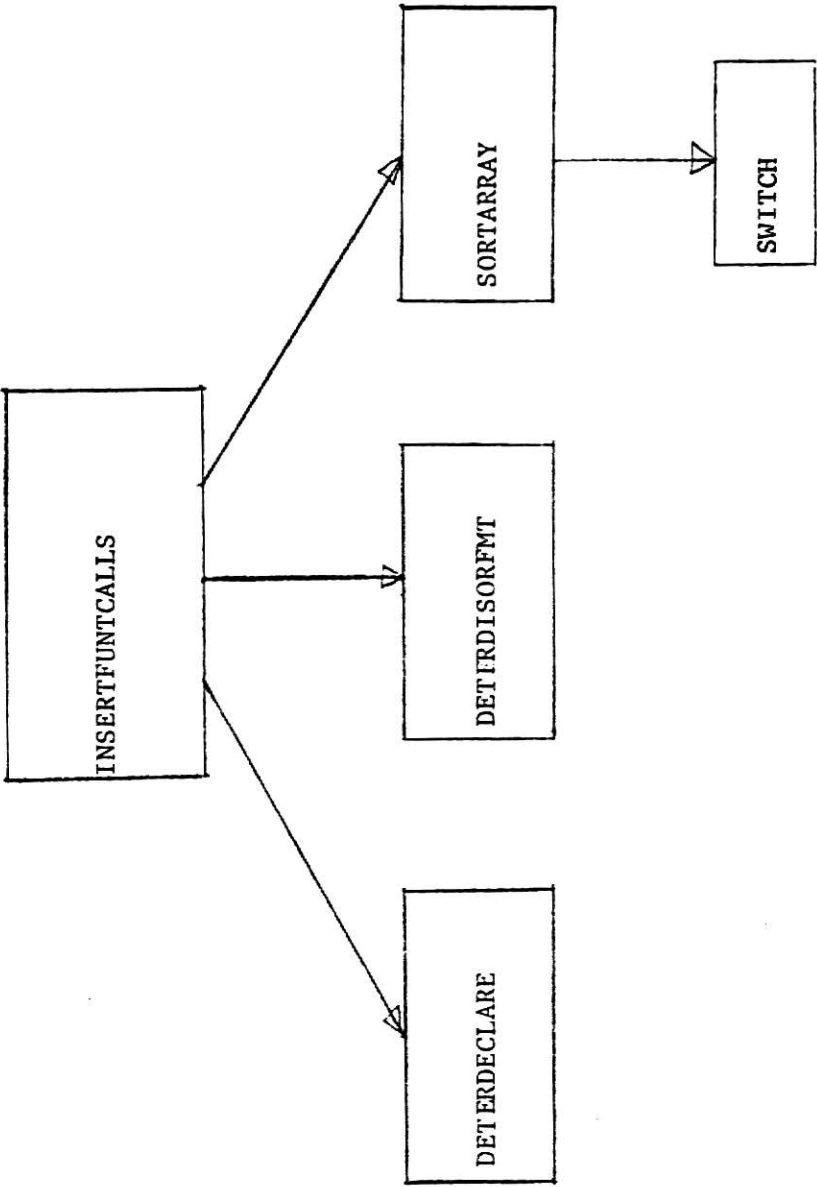
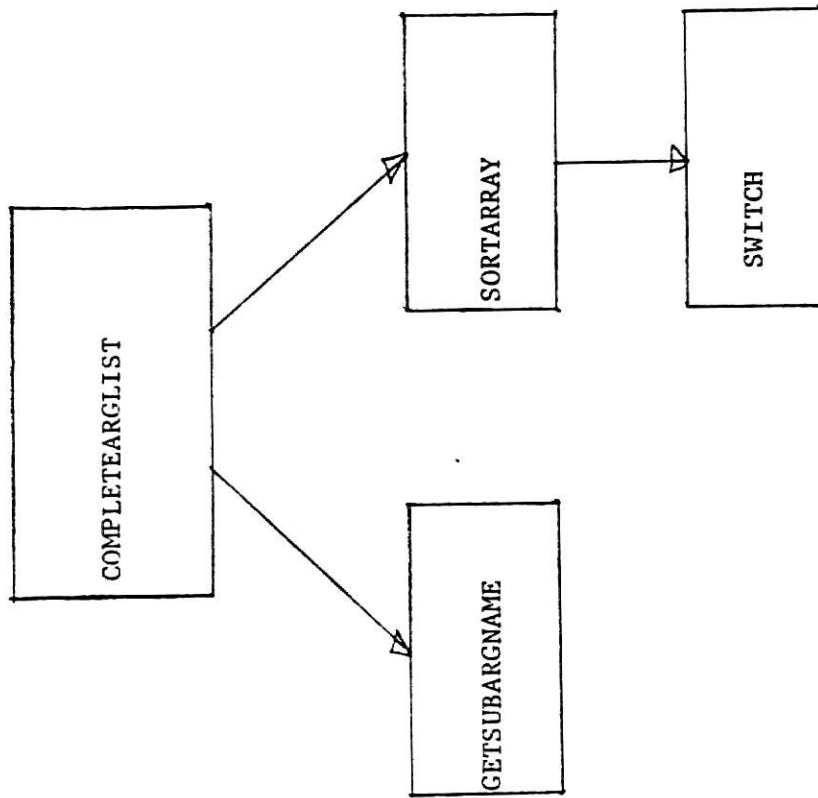


FIGURE 4. HIERARCHY DIAGRAM (CONTINUED)

LEVEL OF CALL

2



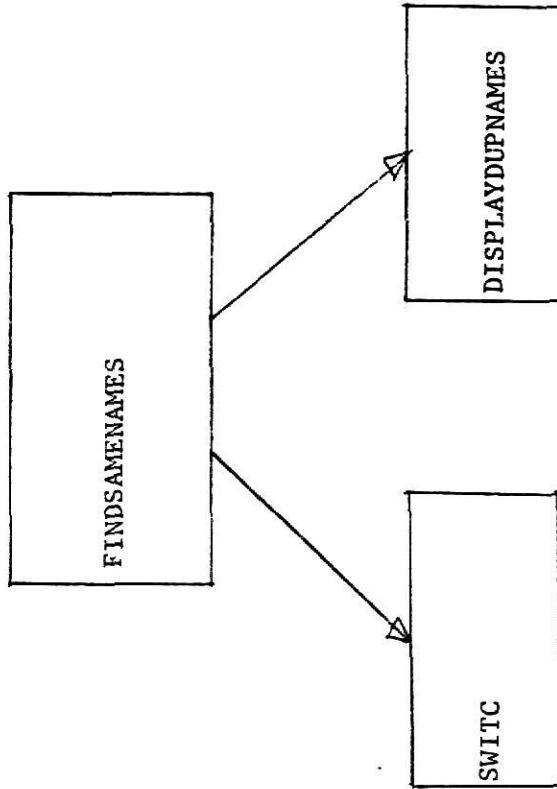
3

4

FIGURE 5. HIERARCHY DIAGRAM (CONTINUED)

LEVEL OF CALL

2



3

FIGURE 6. HIERARCHY DIAGRAM (CONTINUED)



LEVEL OF CALL

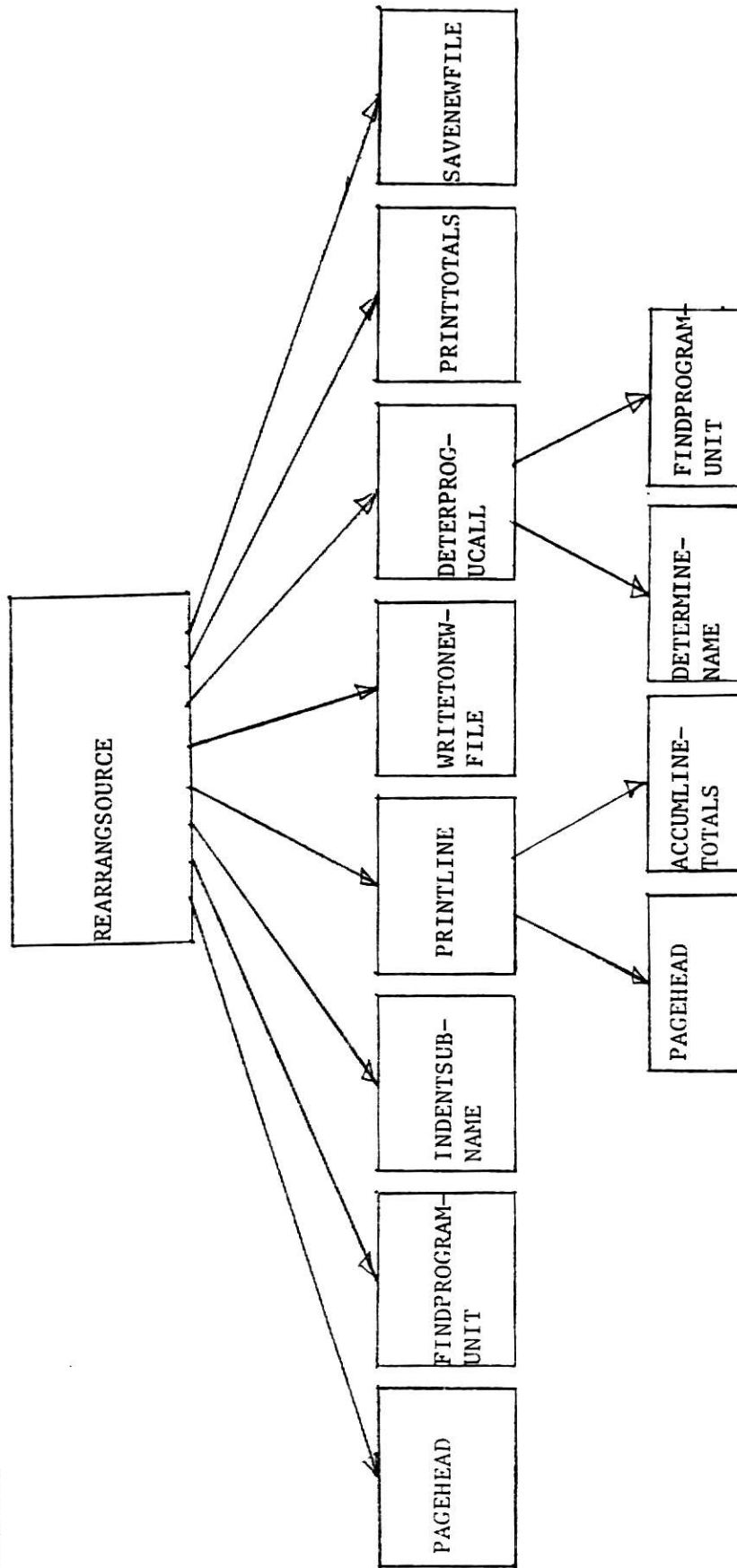


FIGURE 7. HIERARCHY DIAGRAM (CONTINUED)

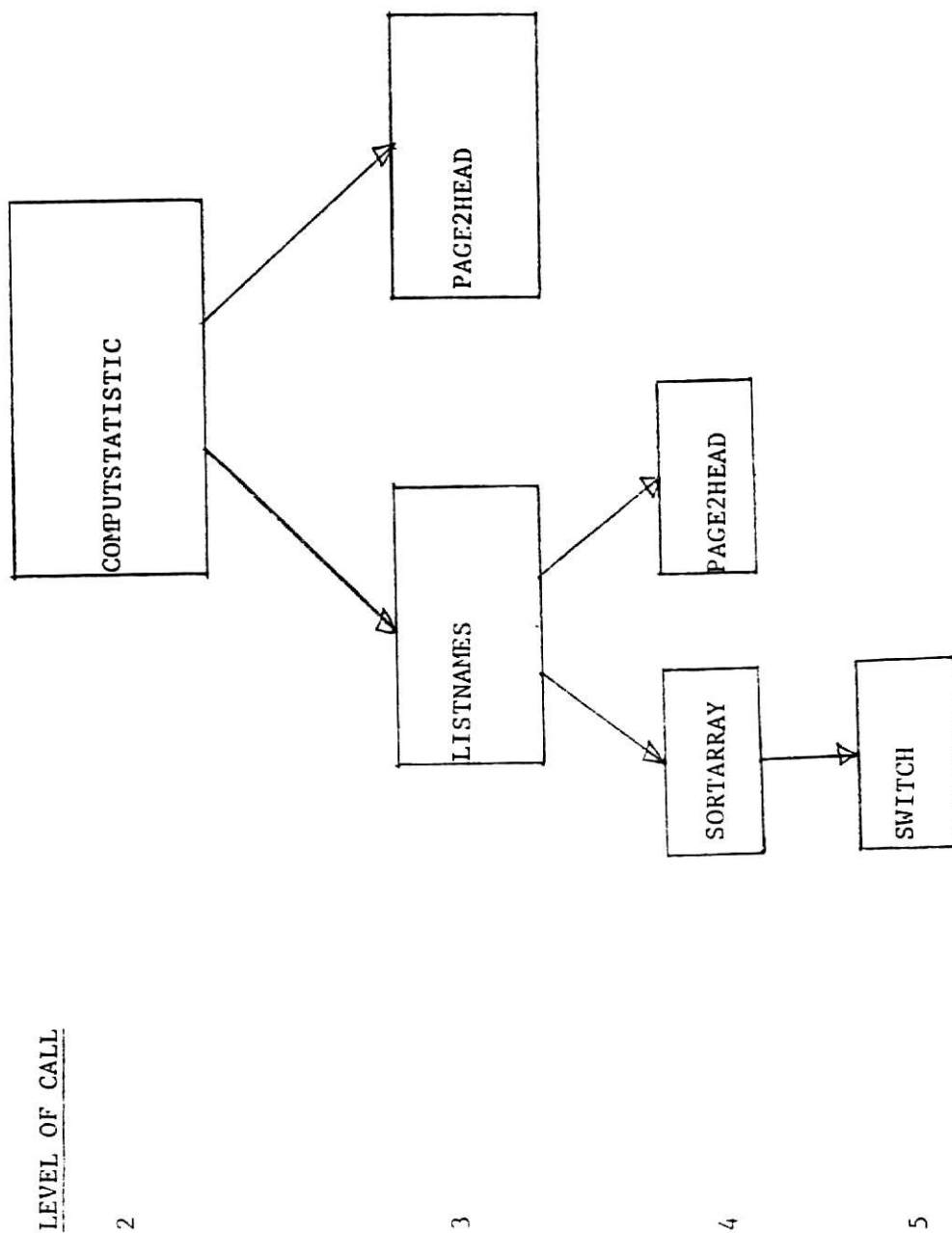


FIGURE 8. HIERARCHY DIAGRAM (CONTINUED)

statement, an entry is made to a 'name' array.

(2) Next, via the subroutine DELETELIBCALLS, any library calls are determined by checking for the existence in the 'name' array of each name found in CALL statements. If the name is not found, then the call is assumed to be to a system library routine and its CALL statement is deleted from the 'reduced' array.

(3) If there are any FUNCTION program units in the source code file, a second read of the file takes place in order to find the function calls (since functions are not called by the explicit CALL statement). These function calls are then inserted into the 'reduced' array. The subroutine INSERTFUNTCALLS performs this task.

(4) A third pass of the source code file becomes necessary if there are any 'EXTERNAL' subroutines. In the third pass, using the subroutine COMPLETEARGLIST, a table is completed that links the actual and formal parameter names of the EXTERNAL subroutines.

(5) At this point the subroutine FINDSAMENAMES detects any duplicate program unit names. If any are found, the name(s) and locations in the unchanged source code file are shown and the program stops.

(6) Next, the subroutine DETERMINIFCALLED checks for the existence of program unit(s) that are not called by at least one other program unit. If any are found, their

name(s) and location(s) in the unchanged source code file are shown and the program stops.

(7) The source code file is now rearranged by the subroutine REARRANGESOURCE into its hierarchical top-down order of program units. Concurrent with the source code file being rearranged, the first report showing this rearranged order is output.

(8) Lastly, the second report, showing the alphabetical listing of program units with location and calling frequency and statistical summary, is output via the subroutine COMPUTSTATISTIC. This causes a normal termination to execution of the program.

The above discussion should give the reader a basic understanding of the purpose and method of this code rearranger program. The narrative about to follow will give a more detailed explanation.

We will now give a brief description of each program module of this code rearranger program, once again referring to the hierarchy diagrams of Figures 1-8:

#### MODULE DESCRIPTIONS

MAIN - Naturally, the program begins with this program module. It declares the necessary global variables and controls the sequential execution of the subroutines that perform the rearranging of the source code file and associated report generation. By looking at the source code and embedded comments of MAIN, the reader should be able to quickly understand, at a very high level, the steps taken by this program to perform its functions.

Subroutine DEFINEFILES (called by MAIN) - This module queries the user for the name of the source code file to be processed so that it can define and initialize the needed input and output disk files for storing of the source code.

Subroutine FINDSEGBEGIN (called by MAIN) - As its name implies, this subroutine finds the first line (ignoring comment lines) of the current program module of the input source code file. The current module is determined by reading the next record of the source code file. (The current record location is passed to this subroutine by MAIN.) It then determines if this program module is MAIN, a function, or a subroutine and stores this information along with the module name and beginning line number in the REDUCED array. If the program unit is a function or subroutine, the program unit name is added to the appropriate name array (FUNCTIONNAME or SUBNAME).

Subroutine STRIPOFFPARAM (called by FINDSEGBEGIN and PROCESSCALL) - This module receives as input either a program unit statement or a CALL statement and removes the argument list, including parentheses, from the statement. It then returns this shortened string to the calling subroutine for eventual entry into the REDUCED array.

Subroutine STORENAME (called by FINDSEGBEGIN) - This short procedure parses out the program unit name from the program unit statement and enters it into the name array passed to it by FINDSEGBEGIN.

Subroutine DETERMINENAME (called by STORENAME, CHECKARGLIST, DELETELIBCALLS, and DETERPROGUCALL) - This module, passed a character string and a beginning location in that string, returns a sixteen character variable NAME.

Subroutine CLASSIFYSTATEMT (called by MAIN) - This program unit classifies each statement of the current program unit as comment, declaration, format, or executable and increments the appropriate counter. If it encounters any CALL statements, they are entered into the REDUCED array after the necessary processing. If an EXTERNAL statement exists in the current program unit, then the CALL statement is processed as required to link any external arguments of the CALL statement to their location in the argument list. When the END statement

of the program unit is reached, the ending line number and statement type counts are stored in the REDUCED array and program control is returned to MAIN.

Subroutine DETERDISORFMT (called by CLASSIFYSTATEMT and INSERTFUNTCALLS) - This subroutine determines if the current source code statement is of type DISPLAY or FORMAT and increments the corresponding counter accordingly.

Subroutine PROCESSCALL (called by CLASSIFYSTATEMT) - This module performs the necessary steps for making the CALL statement suitable for entry into the REDUCED array.

Subroutine PROCESSIF (called by PROCESSCALL) - This subroutine, passed a CALL statement contained in an IF statement of more than one line, removes the argument of the IF statement and thus returns a string that guarantees the presence of the program unit name that follows 'CALL'.

Subroutine CHECKARGLIST (called by CLASSIFYSTATEMT) - This program unit, passed a CALL statement of a program unit that has an EXTERNAL declaration, determines if any arguments of this CALL statement are declared in the EXTERNAL statement. If any are, the following information is stored in the ARGSUBSTITUTION array:

- (1) subroutine name called by this CALL statement
- (2) name(s) in CALL statement's argument list found in EXTERNAL statement
- (3) position number(s) in CALL statement's argument list of argument name(s) found

Then, the CALL statement stored in the REDUCED array is marked with the ARGSUBSTITUTION array position where the above information has just been stored.

Subroutine DETERDECLARE (called by CLASSIFYSTATEMT and INSERTFUNTCALLS) - This module determines if the statement passed to it is declarative, and if so, increments the corresponding counter. Also, if the statement is of type EXTERNAL, it turns on a flag signifying such.

Subroutine DELETEDIBCALLS (called by MAIN) - This module deletes system library calls from the REDUCED array. It does this by checking for the presence in the SUBNAME array of the subroutine name referenced in each CALL statement. If it does not find the name in the SUBNAME

array, then that CALL is assumed to be a system library call and is removed from the REDUCED array.

Subroutine INSERTFUNTCALLS (called by MAIN) - This subroutine is called if one or more FUNCTION modules are present in the input source code file. Its purpose is to read through the entire source code file in order to find FUNCTION calls and their locations and add them to the REDUCED array. To locate FUNCTION calls, this subroutine looks for the presence of any FUNCTION name (stored in array FUNCTIONNAME) in each source code statement (bypassing comment, declarative, and format statements). When a FUNCTION name is located, it insures that it is not its own FUNCTION declaration, and then enters the FUNCTION name and line number location prefixed by 'CALL' into the REDUCED array. After it has processed the last line of source code, it then sorts the REDUCED array by line number location so that all program unit and CALL statements are in the same sequence as the input source code file.

Subroutine SORTARRAY (called by INSERTFUNTCALLS, COMPLETEARGLIST, and LISTNAMES) - This program unit sorts the REDUCED array into the order specified by the values of KEYSTART and KEYLENGTH that are passed to it by the calling subroutine.

Subroutine SWITCH (called by SORTARRAY) - This subroutine simply switches the location of two adjacent records passed to it by the subroutine SORTARRAY.

Subroutine COMPLETEARGLIST (called by MAIN) - This module is called if one or more EXTERNAL statements exist in the input source code file. Its task is to link the actual parameter module name(s) (referenced in the EXTERNAL and CALL statement(s) of a program unit) with the formal parameter name(s) of the module(s) called by the CALL statement(s). To do this, for each module that contains an EXTERNAL statement, the CALL statement that is passing the EXTERNAL parameter name(s) is found in the REDUCED array. This entry contains the pointer to the ARGSUBSTITUTION array (established by subroutine CHECKARGLIST) where the argument name (actual parameter) and its list position in the CALL statement are read. The location of the module called by the CALL statement is found in the REDUCED array. Thus, the argument name of the same list position (formal parameter) of the module statement, determined by reading the correspond-



ing line in the source code file ,is added to the ARGSUBSTITUTION array. Then, the CALL statement using this argument is found in the input source code. This CALL statement, with its source code location and the notation 'PASSED VIA EXTERNAL', is added to the REDUCED file. When all necessary modules have been processed, the REDUCED array is once again sorted by ascending line number location to preserve this array's consistency with the input source code file.

Subroutine GETSUBARGNAME (called by COMPLETEARGLIST) - This subroutine finds the subroutine statement in the input source code for the subroutine referenced by each CALL statement that is passing EXTERNAL parameter(s). Then, it parses out the argument name based on the list position passed to it and links this formal parameter name with the actual parameter name in the ARGSUBSTITUTION array. Lastly, it finds the location of the CALL statement in the source code file that references this formal parameter name.

Subroutine FINDSAMENAMES (called by MAIN) - This program unit determines if any modules are sharing the same name. It does this by merging and sorting the FUNCTION-NAME and SUBNAME arrays. Then, if it finds any two adjacent names to be identical, it displays this information and aborts the program.

Subroutine SWITC (called by FINDSAMENAMES) - Just like SWITCH, this module switches the location of two adjacent records.

Subroutine DISPLAYDUPNAMES (called BY FINDSAMENAMES) - This subroutine gets the input source code's beginning and ending line numbers of dublicately named modules from the REDUCED array and displays the module name and the beginning and ending line numbers for each occurrence of the module name.

Subroutine DETERMIFCALLED (called by MAIN) - This module checks for any program units that exist in the source code but are never called. It does this by finding each CALL statement in the REDUCED array and incrementing the counter for the corresponding function or subroutine name. If any counter still has a value of zero after completion of this process, then the module name(s) and beginning and ending line numbers are displayed and the program is aborted.



Subroutine DISPLAYNOCALLS (called by DETERMIFCALLED) - This program unit displays the name(s) and location(s) of any module(s) present in the source code but never called. Before the display, it checks to see if the module is referenced via an EXTERNAL call by checking the ARGSSUBSTITUTION array.

Subroutine REARRANGESOURCE (called by MAIN) - This module rearranges the input source code file into its hierarchical top-down order of modules, thereby creating an output source code file. Concurrent with this rearranging of modules, a report showing this rearranged structure is generated. This report shows for each program unit name, new beginning and ending line numbers; level of its call (MAIN is defined as level 1); number of lines of comment, declaration, format, and executable lines of code. Also, appropriate source code totals are shown. This module accomplishes its task by using the information contained in the REDUCED array to select out the modules from the input source code file in the proper sequence. The first module of the rearranged source code is always MAIN. With this starting point, traversing through the REDUCED array and employing a stack to keep track of return addresses within the array, the hierarchical rearranging of modules takes place. When rearranging is complete, this module checks that the number of lines of the input source code file equals the number of lines of the just created rearranged output source code file. If such is the case, then the output file overwrites the input file. If this condition is not met, the appropriate message is displayed, and the program is aborted.

Subroutine PAGEHEAD (called by REARRANGESOURCE and PRINTLINE) - This subroutine prints out the title and column headings for the report that shows the rearranged source code information.

Subroutine FINDPROGRAMUNIT (called by REARRANGESOURCE and DETERPROGUCALL) - This program unit finds the program unit statement in the REDUCED array for the program unit name contained in the string passed to it. Also, if the passed string includes 'RECURSIVE CALL' in it, then a flag denoting recursion is turned on.

Subroutine INDENTSUBNAME (called by REARRANGESOURCE) - This module appends zero or more leading blanks to the string passed to it. The number of blanks is determined by the value of the stack pointer which represents one less than the level of the call for the current program

unit about to be displayed.

Subroutine PRINTLINE (called by REARRANGESOURCE) - This subroutine prints the line for the current program unit. If a line for the program unit has been printed previously, only the name and level of the call is output; otherwise, the complete line is printed. A counter is incremented which keeps track of how many time this program unit has been called, line totals are incremented, and the new beginning and ending line numbers, number of lines, and number of times called are entered into the REDUCED array.

Subroutine ACCUMLINETOTALS (called by PRINTLINE) - This program unit simply increments the counters for report totals by the amounts for the current program unit.

Subroutine WRITETONEWFILE (called by REARRANGESOURCE) - This module writes the lines of the current program unit to the rearranged output source code file.

Subroutine DETERPROGUCALL (called by REARRANGESOURCE) - This subroutine is passed the string that contains the current CALL statement and determines if it is a recursive call or if name substitution is required because of passing the program unit name via an EXTERNAL statement. If name substitution is required, the ARGSUBSTITUTION array is accessed as required.

Subroutine PRINTTOTALS (called by REARRANGESOURCE) - This program unit prints the totals for the report that shows the hierarchical top-down order of modules and associated information.

Subroutine SAVENEWFILE (called by REARRANGESOURCE) - This subroutine overwrites the input source code file with the rearranged output source code file after it has determined that the number of lines in each file are the same. If this is not the case, then the appropriate message is displayed and the program is aborted.

Subroutine COMPUTSTATISTIC (called by MAIN) - This module outputs a report showing the alphabetical listing of the program unit names of the source code file. Included in this report are beginning and ending line number and number of times called for each program unit. Also displayed are the largest, smallest, and average

program unit size, and percent of lines by type (comment, declaration, format, and executable).

Subroutine LISTNAMES (called by COMPUTSTATISTIC) - This subroutine extracts the program unit statements from the REDUCED array which contains all the necessary data for outputting the alphabetical listing report. It then sorts this list of program unit statements into alphabetical order by program unit name. After printing the page heading, it then displays this alphabetical listing.

Subroutine PAGE2HEAD (called by LISTNAMES and COMPUTSTATISTIC) - This module prints out the title and column headings for the report that shows the alphabetical listing of source code modules.

#### OTHER IMPLEMENTATION ISSUES

Analysis shows that this implementation meets all the design requirements stated in Section II. This code rearranger program can accommodate both FORTRAN 66 and FORTRAN 77 source code files of up to 4000 lines. When used as a post-compiler, post linker-loader, this program will perform the three general tasks stated in the Design Requirements: rearranging of the input program's source code file into hierarchical top-down order of modules; generating of documentation showing this top-down order; and generating of the report showing the alphabetical listing of the program's modules. The specific tasks listed in Section II are also met: FUNCTION modules are processed appropriately; library calls are detected and eliminated from the module hierarchy; recursive routines are discovered and handled efficiently; duplicately named modules are found and this error condition duly noted; likewise, modules present in the source code but

never called are caught and displayed; EXTERNAL modules are correctly analyzed by constructing a table that relates the actual parameter names with formal parameter names of passed subroutines; and data storage is handled wisely by using disk files for the input and output source code and arrays for all other data requirements.

Extensive testing was done in order to determine that this program would perform as stated in the Design Requirements. A practical approach was taken here - that of processing many programs many times with this code rearranger program. Well over one hundred different source code files (ranging in size from fifty lines to 3,700 lines) were used for ascertaining that this software could perform its designed function in a robust and correct manner. This high volume of testing activity served very effectively to expose programming errors, which were corrected as they were encountered.

Using the prototyping methodology was of significant benefit in implementing this project. This approach to software development, analogous to designing prototypes in engineering applications, allows for quick feedback and early detection and correction of design errors. In fact, using this method was the main reason that extensive testing could be done during all of the software development process. Referring again to the hierarchy diagram of Figure 1, one can see that this program's design is compatible with the iterative testing inherent in prototyping. For example,

it was possible to correctly process programs containing no functions before the code needed for processing of functions (primarily subroutine INSERTFUNTCALLS) was ever written. Likewise, programs containing no EXTERNAL modules were successfully rearranged prior to the existence of any of the code pertaining to EXTERNAL modules (primarily subroutines CHECKARGLIST, COMPLETEARGLIST, and GETSUBARGNAME). Thus, in a matter of only a few weeks, a version of this program was created that could properly rearrange 'simple' FORTRAN programs and validate that the essential software for rearranging source code into its hierarchical top-down order performed correctly. Proceeding from this sound base, extending the program to accommodate all of the remaining design considerations became an orderly and efficient task.

The issue of portability was addressed with this implementation project as this program was required to run on a Hewlett-Packard HP-3000 computer and on a Western Electric 3B20 computer (under the UNIX operating system). The code rearranger program was originally written for the HP-3000 computer whose version of the language is FORTRAN 66 with enhancements added by Hewlett-Packard, such as variable names of up to 15 characters, character variables including substring manipulation, and ACCEPT and DISPLAY statements for processing unformatted input/output. Making this program compatible with the FORTRAN 77 running under UNIX on the 3B20 machine necessitated shortening all variable names to six or less characters, replacing ACCEPT and DISPLAY

statement with READ and WRITE statements, modifying some language dependent features such as file access statements and handling of mixed mode arithmetic, and revamping character declaration and substring manipulation statements because of syntactic and addressing differences. Thus, the source code for the two machines could not be identical because of some of these fundamental language divergencies. But, except for the syntax and character substring differences, only two subroutines required significant modification to any of its statements - DEFINEFILES and SAVENEWFILE - both because of file access statements. Although, true portability of source code was not achieved in this case, the differences were reduced to the absolute minimum.

This program should be able to run on other computer systems that support FORTRAN 66 or FORTRAN 77 without requiring extensive modification to its source code. Any machine running under UNIX should be able to accept the FORTRAN 77 version of the source code with little or no modification. In fact, a considerable effort was made to port this program to KSU's Perkin Elmers Interdata 8/32 computer which runs under UNIX. The only reason portability was not achieved was because of a serious bug in the FORTRAN 77 compiler. When the compiler has been debugged, the porting of the 3B20's source code to the 8/32 should be a straightforward exercise. For other computers, depending on their machine dependent features and customizations to FORTRAN, adaptations probably will be necessary to one or the other



source code files. But since two versions of the source code do exist, one of them should be virtually compatible for almost any mini-computer. Unfortunately, complete portability is not possible at present because of lack of standardization of software and hardware between computer vendors.

As with any implementation project, there are limitations and constraints for this one which are primarily determined by the computer environment in which this program is designed to run. Specifically, as the code rearranger program is presently written, the following maximum capacities apply to the program that is to be rearranged:

- (1) 4000 lines of source code
- (2) 100 subroutines and 100 functions; 150 functions and subroutines combined
- (3) 370 (CALL + SUBROUTINE + FUNCTION) statements

These upper limits were dictated by the allowable maximum run-time data stack of the machine (HP-3000) where the program was developed. Obviously, if the program is transported to another computer environment, these limits should be reviewed and adjusted accordingly. An alternative for increasing the size of programs that can be processed without changing the computer environment would be to modify the program to use disk files in place of some or all of the arrays that are now specified, thereby reducing the required data stack. The user should be aware, however, that such a change will increase the required execution time of the pro-

gram because of higher demands for disk input/output. In any case, this program should be able to be customized to the degree necessary to satisfy the specific user's requirements.



#### IV. FUTURE WORK AND EXTENSIONS

Obviously, this project has opportunities for future work that will enhance its value to the programmer or systems analyst. Naturally, extending this application to process programs of other languages, e.g. PASCAL, C, COBOL, etc. would be both worthwhile and feasible. In the case of PASCAL, considerations relating to forward referencing of procedures would have to be taken into account. Forward references would have to be inserted into the source code, if they did not already exist, so that the hierarchical top-down module order of the rearranged file would compile successfully. Furthermore, these forward references would have to be treated as comment statements belonging to no program unit when the source code rearranging takes place. Otherwise, duplicate module names would be detected and the rearranging process would abort. For the C and COBOL languages, rearranging of code should be as straightforward as for FORTRAN. In any case, two important decisions to be made before extending this project to another language are:

(1) What version(s) of the selected language would the developed software support?

(2) In what version of what language would this source code rearranger software be written?

The answers to these two questions will promulgate specific design considerations and also very much determine portability.

Another approach to extending this system to other programming languages might be writing the source code rearranger program to be table driven. Depending on what language is being processed, the corresponding table containing keywords for identifying call and program unit statements would be accessed. These tables would also contain keywords for enabling classification of statements by comment, declarative, executable, etc. A difficult issue that would arise here would be detecting of comment lines in C, for example, where the delimiters for the comment ('/\*','\*/') can appear anywhere on the line. This certainly would have to be handled differently than in FORTRAN where the comment symbol ('C') must always appear in column 1. Careful analysis of each programming language's calling structure would be necessary so that the hierarchical relationship between modules could be established. Fortunately, this project has already considered both explicit (subroutines) and implicit (functions) types of calls. Thus, it certainly should be practical to extend this approach to other languages such as PASCAL and C which have implicit procedure and function calls. One more problem area for implementing this 'table driven' approach is how to process languages, such as PASCAL and C, which allow multiple statements per line. In order to establish a useful relationship between number of lines and number of statements, as well as to define what parsing capabilities would be needed, ground rules would have to be agreed upon that would standardize

how this situation is to be handled. But the capability of one program to handle a variety of languages would make grappling with issues like the above very justifiable.

Another very useful feature of this project would be the capability to generate graphical outputs, in addition to the report formatted documentation, that would be essentially of the form of the hierarchy diagram shown in Section III. Implementation of this feature would be primarily dependent on defining standard graphical notation. Also, handling such situations as depicting modules whose call invocations are at more than one level and scaling of the graph based on the number of modules, calls, and levels would pose a challenge. Portability would become a difficult issue as another piece of hardware, the graphics output device, would be needed for implementation. But the benefit of having a complete program's structure succinctly displayed in graphical form would certainly justify the development effort.

An additional enhancement to this project would be the incorporation of a data-base, interfaced with this code rearranger program, that would maintain program/module relationships, thereby providing the programmer with valuable information. If this data-base had the capability to catalog detailed information about all of a system's programs' module structure, such as module name, line counts by statement type, number of characters, modification dates, programs used by, etc., then the user would have a powerful

tool for maintaining and controlling software development of his or her source code. For example, if a programmer has to change the code of a specific module, by accessing this data-base, he or she could readily determine what programs are affected and therefore have to be maintained. This could be especially valuable if one is responsible for hundreds of programs, many of which use common subroutines or functions. Also, the user could query this same data-base to ascertain what program units a specific program is comprised of without having to actually look at the program's source code. It might even become possible to create a new program by simply specifying the names and locations of existing routines. An important consideration in implementing this feature is selection of the data-base management system where this data will reside. This choice will greatly influence portability to other computers. As with the other suggested extensions, it appears that the most difficult to achieve software engineering goal will be a high degree of portability. Hopefully, in the years to come, as computer science becomes more of a science, and less of an art, software and hardware incompatibilities will lessen so that portability will be a much more easily realized aim.

## V. CONCLUSIONS

In conclusion, the question should be asked: What are this project's benefits and do these benefits meet the needs and desires stated in the introduction to this paper? To address this question, we will restate the six benefits listed in the Introduction as questions and supply the answers:

(1) Is source code rearranged into top-down module order? Yes. This has clearly been accomplished, as the reader can see by referring to the example programs of Appendix C.

(2) Is program documentation showing top-down and alphabetical module order generated? Yes. Please refer to Appendix C once again and examine the two reports entitled "SUMMARY OF EXECUTION SEQUENCE OF PROGRAM UNITS FOR FORTRAN SOURCE CODE: ..." and "LISTING OF PROGRAM UNITS IN ALPHABETICAL ORDER WITH STATISTICAL SUMMARY FOR SOURCE CODE: ..."

(3) Does the rearranged source code and associated documentation give a better understanding of the program's structure. Yes, in two ways: obviously, having the complete program's top-down module structure contained on a one or two page report lends itself to greatly improved user understanding (at a high level), and the alphabetical listing of modules makes possible quickly locating of a module's source code; but in a more subtle but no

less important way, the mere fact that the program's source code is arranged in the same order as the program's execution sequence, makes it much easier for the programmer to gain a detailed understanding of the program's instructions as he or she follows the flow of control through part or all of the source code listing.

(4) Does the output of the code rearranger program provide a rudimentary measure of software quality? Yes. Referring to the alphabetical listing of program modules for one of the sample programs in Appendix C, one can readily see the largest, smallest, and average program unit sizes as shown at the bottom of this report. Also, the percent of comment lines contained in the source code is prominently displayed. Simple, but important statistics such as these can readily indicate if the program employs modular structure and whether it is adequately commented. Furthermore, the column of this report that shows how many times each module is called could prove worthwhile in analyzing and improving a program's execution time as it would highlight any routines that are called numerous times.

(5) Can one expect improved productivity in developing new programs? Yes, especially if the programmer writes modular code in conjunction with using the prototyping methodology that was discussed in Section III. Then, with each iteration of software development, the pro-

grammer has up to date documentation and properly arranged source code of the in-development program. These tools will naturally lead to more efficient and higher quality software development practices.

(6) Will maintenance and enhancements of existing programs be made easier? Most emphatically, yes! When the occasion arises for a programmer to refer to existing source code, which could be several years old and written by someone else, the fact that this code is in hierarchical top-down order, with the associated 'table of contents' and 'index' reports, will prove to be invaluable for providing quick and in depth understanding of the program's intended function. Thus, the programmer's time and talent will be much more effectively used because he or she will almost immediately be able to start work on developing the necessary software changes or additions instead of having to spend hours or days studying and manually rearranging the source code listing so that it can be understood.

Summarizing, we were able to answer affirmatively that all six benefits stated in the Introduction were achieved with the implementation of this project. Thus, one can deduce that this project began from a well defined problem statement and ended with a successful problem solution.

One aspect of implementing this project that was unexpected was the firsthand realization of how very powerful software tools can be. This power was most strongly felt

during the implementation phase of developing this software. Simply stated, this program aided in its own creation and evolution. As discussed in the previous paragraph, because this tool can improve programmer productivity, especially when coupled with the prototyping approach, the speed and efficiency with which this code developed was greatly attributed to the ability of this program to analyze and rearrange itself. This early on functional capability existed, because of prototyping, long before many of the modules were written (see Section III). Thus, this recursive elegance accelerated each successive iteration of code. Also, the power of this software tool was noticed in using it on a daily basis for its designed function of processing other programs. Being able to perform the rearranging and documentation for a several thousand line program in a matter of minutes can be viewed as somewhat awesome and causes one to wonder just where this capability of 'software analyzing software' will ultimately lead.



## APPENDIX A

### PROGRAM SOURCE CODE LISTING

PAGE 0001 HPJ2102B.01.0V FURMAN/3000 (C) NEWLETT-PACKARD CO. 1982 WED, JUN 15, 1983, 12:43 PM

```

0001000 SCOTTHUL IN11,MAP,LABEL,LOCATIUN
00072 00002000 C
00072 00003000 C PROGRAM NAMES:
00072 00004000 C
00072 00005000 C SOURCE1 REARRANGS
00072 00006000 C OBJECT1 REARRANG
00072 00007000 C
00072 00008000 C
00072 00009000 C CHARACTER CALLS, SUBS12, SUBNAME*10(100), FUNCTIONNAME*10(100),
00072 00010000 C IFUNCTION*10, SOURCEFILE*0, LASTSTATE*TYPE*2, REDUCED*92(500),
00072 00011000 C 2ARGSUBSTITUTION*50(10,0)
00072 00012000 C INTEGER SEGBEGIN, SEGPPOINT, COMMENTCOUNTER, FURNATCOUNTER,
00072 00013000 C IDECLARACOUNTER, FUNCTIONCOUNTER, SUBNAMECOUNTER, ASA, PAGEBU
00072 00014000 C DATA ARGSUBSTITUTION/800 " "
00072 00015000 C SUBS= SUBROUTINE "
00106 00015000 CALL=CALL "
00117 00016000 FUNCTION= " FUNCTION "
00132 00017000 C DEFINE FILES
00132 00018000 CALL DEFINEFILES(SOURCEFILE)
00132 00019000 C MLC=1
00136 00020000 C DETERMINE BEGINNING OF PROGRAM UNIT AND WHETHER IT IS MAIN, FUNCTION,
00136 00021000 C OR SUBROUTINE. ENTER ITS NAME INTO 'REDUCED' ARRAY
00136 00022000 22 CALL FINDSEGBEGIN(MLC, SEGBEGIN, COMMENTCOUNTER, FURNATCOUNTER, CALL,
00136 00023000 IDECLARACOUNTER, LASTSTATE*TYPE, LC, SUB, SUBNAMECOUNTER, SUBNAME, SEGPPOINT,
00136 00024000 REDUCED, FUNCTION, FUNCTIONNAME, FUNCTIONCOUNTER, 999)
00136 00025000 C CLASSIFY EACH STATEMENT OF PROGRAM UNIT AND ENTER ANY 'CALL'
00136 00026000 C STATEMENTS INTO 'REDUCED' ARRAY
00171 00027000 CALL CLASSIFYSTATEMENT(COMMENTCOUNTER, FURNATCOUNTER,
00171 00028000 IDECLARACOUNTER, LASTSTATE*TYPE, CALL, LC, SUB, MLC, SEGPPOINT, REDUCED,
00171 00029000 2ASA, ARGSUBSTITUTION)
00212 00030000 GO TO 22
00213 00031000 C DELETE LIBRARY CALLS FROM 'REDUCED' ARRAY
00213 00032000 99 MLC=MLC-1
00214 00033000 DISPLAY*0 OF STATEMENTS IN 'REDUCED' " MLC, " /500 "
00266 00034000 CALL DELETELIBCALLS(SUBNAME, SUBNAMECOUNTER, CALL, SUB, REDUCED, MLC,
00266 00035000 IFUNCTION)
00302 00036000 C FIND AND INSERT FUNCTION CALLS INTO REDUCED ARRAY
00302 00037000 IF (FUNCTIONCOUNTER.GT.0) CALL INSERTFUNTCALLS(FUNCTIONNAME,
00302 00038000 IFUNCTIONCOUNTER, REDUCED, FUNCTION, MLC, CALL, SUB)
00321 00039000 C IF ANY EXTERNAL STATEMENTS, PLACE ARGUMENT NAME OF SUBROUTINE IN
00321 00040000 C 'ARGSUBSTITUTION' ARRAY
00321 00041000 IF (ASA.GT.0) CALL COMPLETEANGLIST(REDUCED, SUB, CALL, ARGSUBSTITUTION,
00321 00042000 INLC)
00336 00043000 C DETERMINE IF ANY PROGRAM UNITS ARE DUPLICATEDLY NAMED
00336 00044000 CALL FINDSAMESUBNAME(SUBNAME, SUBNAMECOUNTER, FUNCTIONNAME,
00336 00045000 IFUNCTIONCOUNTER, REDUCED, MLC)
00351 00046000 C DETERMINE IF ANY EXISTING PROGRAM UNITS ARE NEVER CALLED
00351 00047000 CALL DETERMINIFCALLED(SUBNAME, SUBNAMECOUNTER, FUNCTIONNAME,
00351 00048000 IFUNCTIONCOUNTER, REDUCED, MLC, ARGSUBSTITUTION, ASA, CALL, SUB, FUNCTION)
00377 00049000 C DISPLAY*NUMBER OF LINES OF INPUT FILE=, LC-1
00443 00050000 C NUMPHUGUNITS=SUBNAMECOUNTER+FUNCTIONCOUNTER+1
00447 00051000 C REARRANGE SOURCE CODE IN TOP-DOWN ORDER
00447 00052000 CALL REARRANGESOURCE(SUB, CALL, SOURCEFILE, LC-1, NUMPHUGUNITS, MLC,
00447 00053000 REDUCED, FUNCTION, ARGSUBSTITUTION, TUTOCMENTS, TUTOCUNITS, TUTODECL,
00447 00054000 2*PAGEBU)
00474 00055000 C COMPUTE AND PRINT OUT STATISTICS REPORT
00474 00056000 CALL COMPUTESTATISTICS(SUB, FUNCTION, SOURCEFILE, LC-1, MLC, REDUCED,
00474 00057000 1TUTOCMENTS, TUTOCUNITS, TUTODECL, NUMPHUGUNITS, PAGEBU)

```

PAGE 0002

00515 00050000 STOP  
00516 00050000 END

## SYMBOL MAP

| NAME            | TYPE      | STRUCTURE  | ADDRESS | NAME                | TYPE       | STRUCTURE  | ADDRESS |
|-----------------|-----------|------------|---------|---------------------|------------|------------|---------|
| ARGSUBSTITUTION | CHARACTER | ARRAY      | 0131 ,1 | ASA                 | INTEGER    | SIMPLE VAR | 0132    |
| CALL            | CHARACTER | SIMPLE VAR | 0131 ,1 | CLASSIFYSTATEMT     | SUBROUTINE | SUBROUTINE |         |
| COMMENTCOUNT    | INTEGER   | SIMPLE VAR | 0131 ,1 | COMPLETARGLIST      | SUBROUTINE | SUBROUTINE |         |
| CUMPUTSTATISTIC | INTEGER   | SUBROUTINE | 0136    | DECLARECOUNT        | INTEGER    | SIMPLE VAR | 0136    |
| DEFINFILES      | INTEGER   | SUBROUTINE |         | DELETETELLCALLS     | SUBROUTINE | SUBROUTINE |         |
| DETERMIFCALLED  | INTEGER   | SUBROUTINE |         | FINDSAMENAMES       | SUBROUTINE | SUBROUTINE |         |
| FINDSEGEGIN     | CHARACTER | SIMPLE VAR | 0132 ,1 | FUNCTIONCOUNT       | INTEGER    | SIMPLE VAR | 0137    |
| FUNCTION        | CHARACTER | ARRAY      | 0132 ,1 | FUNCTIONCOUNT       | INTEGER    | SIMPLE VAR | 0137    |
| FUNCTIONNAME    | CHARACTER | SIMPLE VAR | 0133 ,1 | LC                  | INTEGER    | SIMPLE VAR | 0137    |
| LASTSTATETYPE   | INTEGER   | SIMPLE VAR | 0137    | INSERTFUNCTIONCALLS | INTEGER    | SIMPLE VAR | 0137    |
| NLC             | INTEGER   | SIMPLE VAR | 0131    | LC                  | INTEGER    | SIMPLE VAR | 0137    |
| PAGENO          | INTEGER   | ARRAY      | 0134 ,1 | NUMPRGUNITS         | INTEGER    | SIMPLE VAR | 0137    |
| REDUCED         | CHARACTER | SIMPLE VAR | 0135 ,1 | REARRANGESOURCE     | INTEGER    | SIMPLE VAR | 0137    |
| SEGPOINT        | INTEGER   | SIMPLE VAR | 0135 ,1 | SEGUEGIN            | CHARACTER  | ARRAY      | 0137 ,1 |
| SUB             | CHARACTER | SIMPLE VAR | 0136 ,1 | SOURCEFILE          | CHARACTER  | ARRAY      | 0137 ,1 |
| SUBNAMECOUNT    | INTEGER   | SIMPLE VAR | 0136    | SURNAME             | REAL       | SIMPLE VAR | 0137    |
| TUTDECL         | REAL      | SIMPLE VAR | 0136    | TUTCURRENTS         | REAL       | SIMPLE VAR | 0137    |
|                 |           |            |         | TUTFORMATS          | REAL       | SIMPLE VAR | 0137    |

## LABEL MAP

| STATEMENT LABEL | CODE | STATEMENT LABEL | CODE | STATEMENT LABEL | CODE | STATEMENT LABEL | CODE |
|-----------------|------|-----------------|------|-----------------|------|-----------------|------|
| 22              | 136  | 99              | 213  |                 |      |                 |      |

PROGRAM UNIT MAIN' COMPILED

PAGE 0003 NEWLETT-PACNARD 32102B-01.09 FORTNAN/1000 WED, JUN 15, 1964, 12:24 PM

```

00077 00060000 SCUNTRL INIT MAP, LABEL, LOCATION
00077 00061000 SUBROUTINE DEFINEFILES(SOURCEFILE)
00077 00062000 SYSTEM INTRINSIC COMMAND, UNITCONTROL
00077 00063000 CHARACTER MSG(40), SOURCEFILE(8)
00077 00064000 C ARRAY BY NAME OF "REDUCED" WILL BE USED FOR STORING SUBROUTINE AND
00077 00065000 C FUNCTION NAMES AND CALLS
00077 00066000 C REQUEST NAME OF FORTNAN SOURCE FILE TO BE ANALYZED
00077 00067000 SOURCEFILE="
00111 00068000 2 DISPLAY**
00124 00069000 DISPLAY=ENTER NAME OF FORTNAN SOURCE FILE TO BE ANALYZED"
00174 00070000 ACCEPT SOURCEFILE
00205 00071000 IF(SOURCEFILE.NE."
00221 00072000 DISPLAY=INVALID NAME. PLEASE REENTER."
00260 00073000 GO TO 2
00261 00074000 C DEFINE PERMANENT FILE FOR REQUESTED SOURCE FILE NAME (TO BE
00261 00075000 C DELETED AFTER SOURCE CODE IS REARRANGED)
00261 00076000 1 WRITE(MSG,9002)SOURCEFILE
00304 00077000 9002 FORMAT("FILE FTM46=",AB,".OLD",A15C)
00310 00078000 C CALL COMMAND(MSG,NEKR,NPARR)
00310 00079000 C DEFINE PERMANENT FILE FOR REQUESTED SOURCE FILE NAME (TO STORE
00310 00080000 C REARRANGED MSG AND TO BE SAVED AT COMPLETION OF PROGRAM)
00310 00081000 WRITE(MSG,9004)SOURCEFILE
00310 00082000 9004 FORMAT("FILE FTM41=",AB,".NEWREC=-80.16,F,ASCII7DISC=4000,SAVE"
00310 00083000 1,A15C)
00310 00084000 C CALL COMMAND(MSG,NEKR,NPARR)
00310 00085000 C MOVE RECORD POINTER TO BEGINNING OF FILE FOR FILE # 46
00310 00086000 REWIND 46
00310 00087000 C WRITE EOF AT BEGINNING OF FILE FOR FILE # 41
00310 00088000 CALL UNITCONTROL(41,1)
00310 00089000 RETURN
00310 00090000 END

```

## SYMBOL MAP

| NAME        | TYPE      | STRUCTURE  | ADDRESS | NAME        | TYPE      | STRUCTURE  | ADDRESS |
|-------------|-----------|------------|---------|-------------|-----------|------------|---------|
| COMMAND     | CHARACTER | SUBROUTINE | 0003 /1 | DEFINEFILES | INTEGER   | SUBROUTINE | 0002    |
| MSG         | INTEGER   | SIMPLE VAR | 0001    | NEKR        | CHARACTER | SIMPLE VAR | 0-04 /1 |
| NPARR       |           | SIMPLE VAR |         | SOURCEFILE  |           |            |         |
| UNITCONTROL |           | SUBROUTINE |         |             |           |            |         |

## LABEL MAP

| STATEMENT LABEL | CODE OFFSET | STATEMENT LABEL | CODE OFFSET | STATEMENT LABEL | CODE OFFSET | STATEMENT LABEL | CODE OFFSET |
|-----------------|-------------|-----------------|-------------|-----------------|-------------|-----------------|-------------|
| 1               | 261         | 2               | 111         | 9002 FMT        | 0           | 9004 FMT        | 17          |

PROGRAM UNIT DEFINED BY COMPILER

PAGE 0004 HEMLETT-PACKARD 321020-01-09 FORTMAN/1000 WED, JUN 15, 1983, 12:24 PM

```

00053 00091000 SCOUNTROL INIT,MAP,LABEL,LOCUTION
00054 00092000 SUBROUTINE FINDSEGBEGIN(MLC,SEGBEGIN,COMMENTCOUNTER,FUNCTCOUNTER,
00055 00093000 ICALL,DECLARECOUNTER,LASTSTATYPE,LC,SUB,SUBNAMECOUNTER,SUBNAME,
00056 00094000 ZSEGPUNT,REDUCED,FUNCTION,FUNCTIONNAME,FUNCTCOUNTER,9)
00057 00095000 CHARACTER LASTSTATYPE*2,CALL*5,SUB*12,SUBNAME*16(100),STRING*12,
00058 00096000 INREDCED*92(500),FUNCTION*10,FUNCTIONNAME*16(100)
00059 00097000 INTEGER SEGBEGIN,COMMENTCOUNTER,FUNCTCOUNTER,DECLARECOUNTER,
00060 00098000 ISEGPUNT,FUNCTCOUNTER,SUBNAMECOUNTER
00061 00099000 C BEGIN TO READ PROGRAM UNIT
00062 00100000 22 SEGBEGIN=LC+1
00063 00101000 COMMENTCOUNTER=0
00064 00102000 FUNCTCOUNTER=0
00065 00103000 DECLARECOUNTER=0
00066 00104000 LASTSTATYPE=" "
00067 00105000 2 LC=LC+1
00068 00106000 READ(400LC,1,END=99)STRING
00069 00107000 1 FUNCT(172,514)
00070 00108000 C SKIP BEGINNING LINE(S) IF COMMENT STATEMENT OR CONTROL STATEMENT
00071 00109000 IF(STRING(1:1).EQ."C")COMMENTCOUNTER=COMMENTCOUNTER+1
00072 00110000 IF(STRING(1:1).EQ."C")GO TO 2
00073 00111000 IF(STRING(1:1).EQ."S")GO TO 2
00074 00112000 C DETERMINE IF PROGRAM UNIT IS MAIN, FUNCTION, OR SUBROUTINE AND
00075 00113000 C STORE BEGINNING LINE S AND PROGRAM UNIT NAME IN "REDUCED" ARRAY
00076 00114000 IF(INDEX(STRING,SUB).GT.0.OR.INDEX(STRING,FUNCTION).GT.0)CALL
00077 00115000 ISTMPOFPARAM(STRING,CALL,SUB,FUNCTION)
00078 00116000 IF(INDEX(STRING,SUB).GT.0.OR.INDEX(STRING,FUNCTION).GT.0)WRITE
00079 00117000 1(REDUCED(MLC),1)STRING,SEGBEGIN
00080 00118000 C IF SUBROUTINE, STORE SUBROUTINE NAME IN "SUBNAME" ARRAY
00081 00119000 IF(INDEX(STRING,SUB).GT.0)CALL STORENAME(SUBNAMECOUNTER,SUBNAME,
00082 00120000 INDEX(STRING,SUB),STRING,12)
00083 00121000 C IF FUNCTION, STORE FUNCTION NAME IN "FUNCTIONNAME" ARRAY
00084 00122000 IF(INDEX(STRING,FUNCTION).GT.0)CALL STORENAME(FUNCTCOUNTER,
00085 00123000 FUNCTIONNAME,INDEX(STRING,FUNCTION),STRING,10)
00086 00124000 IF(INDEX(STRING,SUB).EQ.0.AND.INDEX(STRING,FUNCTION).EQ.0)WRITE
00087 00125000 1(REDUCED(MLC),4)SEGBEGIN
00088 00126000 4 FUNCT(164,"MAIN",62X,214)
00089 00127000 SEGPUNT=MLC
00090 00128000 MLC=MLC+1
00091 00129000 LC=LC-1
00092 00130000 RETURN
00093 00131000 99 RETURN 1
00094 00132000 END

```

## SYMBOL MAP

| NAME            | TYPE      | STRUCTURE  | ADDRESS  | NAME           | TYPE      | STRUCTURE  | ADDRESS  |
|-----------------|-----------|------------|----------|----------------|-----------|------------|----------|
| CALL            | CHARACTER | SIMPLE VAR | 0-817 .1 | COMMENTCOUNTER | INTEGER   | SIMPLE VAR | 0-821 .1 |
| DECLARECOUNTER  | INTEGER   | SIMPLE VAR | 0-816 .1 | FINDSEGBEGIN   | CHARACTER | SIMPLE VAR | 0-836 .1 |
| FUNCTCOUNTER    | INTEGER   | SIMPLE VAR | 0-820 .1 | FUNCTION       | CHARACTER | SIMPLE VAR | 0-835 .1 |
| FUNCTIONCOUNTER | INTEGER   | SIMPLE VAR | 0-84 .1  | FUNCTIONNAME   | CHARACTER | SIMPLE VAR | 0-814 .1 |
| LASTSTATYPE     | CHARACTER | SIMPLE VAR | 0-815 .1 | LC             | INTEGER   | SIMPLE VAR | 0-87 .1  |
| MLC             | INTEGER   | SIMPLE VAR | 0-823 .1 | REDUCED        | CHARACTER | SIMPLE VAR | 0-810 .1 |
| SEGBEGIN        | INTEGER   | SIMPLE VAR | 0-822 .1 | SEGPUNT        | INTEGER   | SIMPLE VAR | 0-810 .1 |
| STORENAME       | CHARACTER | SIMPLE VAR | 0-821 .1 | STRING         | CHARACTER | SIMPLE VAR | 0-881 .1 |

PAGE 0005 FINDSEGBEGIN

STRIPOFFPARAM SUBNAME CHARACTER ARRAY SUBROUTING U-311 71 SUB SUBNAMECOUNT CHARACTER INTEGER SIMPLE VAR U-313 71 SIMPLE VAR U-312 71

LABEL MAP

| STATEMENT LABEL | CODE OFFSET | STATEMENT LABEL | CODE OFFSET | STATEMENT LABEL | CODE OFFSET | STATEMENT LABEL | CODE OFFSET |
|-----------------|-------------|-----------------|-------------|-----------------|-------------|-----------------|-------------|
| 1 PNT           | 0           | 2               | 71          | 4 PNT           | 5           | 22              | 53          |
|                 |             |                 |             |                 |             | 99              | 410         |

PROGRAM UNIT FINDSEGBEGIN COMPILED

```

00020 00134000 SCUNTHOL INIT MAP LABEL EDUCATION
00020 00134000 SUBROUTINE STRIPOFFPARAM(STRING,CALL,SUB,FUNCTION)
00020 00135000 CHARACTER STRING(12),CALL(5),SUB(12),NEWSTRING(12),FUNCTION(10)
00020 00136000 IF(INDEX(STRING,CALL).GT.0)INDEXNUM=INDEX(STRING,CALL)+5
00041 00137000 IF(INDEX(STRING,SUB).GT.0)INDEXNUM=INDEX(STRING,SUB)+12
00062 00138000 IF(INDEX(STRING,FUNCTION).GT.0)INDEXNUM=INDEX(STRING,FUNCTION)+10
00093 00139000 1 DO 10 I=INDEXNUM,12
00110 00140000 IF(STRING(I).EQ."1")GO TO 11
00131 00141000 10 CONTINUE
00132 00142000 GO TO 2
00133 00143000 11 DO 20 J=1,12
00140 00144000 STRING(J)= " "
00153 00145000 20 CONTINUE
00154 00146000 C REMOVE ANY EXTRA BLANKS BEFORE NAME
00154 00147000 2 IF(STRING(INDEXNUM+1).NE."1")GO TO 3
00174 00148000 NEWSTRING(1)=INDEXNUM-12
00207 00149000 NEWSTRING(INDEXNUM+12)=STRING(INDEXNUM+12)-INDEXNUM
00223 00150000 STRING=NEWSTRING
00227 00151000 STRING(12)= " "
00240 00152000 GO TO 2
00241 00153000 C IF FUNCTION STATEMENT, REMOVE ANY EXTRA CHARACTERS ON BLANKS BEFORE
00241 00154000 C "FUNCTION"
00241 00155000 3 IF(INDEX(STRING,FUNCTION).EQ.0)RETURN
00252 00156000 INDEXNUM=INDEX(STRING,FUNCTION)
00261 00157000 NEWSTRING(1)=STRING(1)+1
00265 00158000 NEWSTRING(6)=STRING(INDEXNUM)
00277 00159000 STRING=NEWSTRING
00303 00160000 RETURN
00304 00161000 END

```

# SYMBOL MAP

| NAME   | TYPE      | STRUCTURE  | ADDRESS | NAME          | TYPE       | STRUCTURE  | ADDRESS |
|--------|-----------|------------|---------|---------------|------------|------------|---------|
| CALL   | CHARACTER | SIMPLE VAR | 0-86 ,1 | FUNCTION      | CHARACTER  | SIMPLE VAR | 0-84 ,1 |
| I      | INTEGER   | SIMPLE VAR | 0-81    | INDEXNUM      | INTEGER    | SIMPLE VAR | 0-83    |
| J      | INTEGER   | SIMPLE VAR | 0-82    | NEWSTRING     | CHARACTER  | SIMPLE VAR | 0-84 ,1 |
| STRING | CHARACTER | SIMPLE VAR | 0-87 ,1 | STRIPOFFPARAM | SUBROUTINE |            |         |
| SUB    | CHARACTER | SIMPLE VAR | 0-85 ,1 |               |            |            |         |

# LABEL MAP

| STATEMENT LABEL | CODE OFFSET | STATEMENT LABEL | CODE OFFSET | STATEMENT LABEL | CODE OFFSET | STATEMENT LABEL | CODE OFFSET |
|-----------------|-------------|-----------------|-------------|-----------------|-------------|-----------------|-------------|
| 1               | 103         | 2               | 154         | 1               | 241         | 10              | 131         |
|                 |             |                 |             |                 |             | 20              | 153         |
|                 |             |                 |             |                 |             | 133             |             |
|                 |             |                 |             |                 |             | 153             |             |

PROGRAM UNIT STRIPOFFPARAM COMPILED

PAGE 0007    HEWLETT-PACKARD 32102B-01.09    FORTNAN/1000    WED, JUN 15, 1983, 12:24 PM

```

00014 00162000 8 CONTROL INIT MAP, LABEL, LOCATION
00014 00163000  SUBROUTINE STORNAME(NAMECOUNTER, NAME, INDEXNUM, STRING, LEN)
00014 00164000  CHARACTER NAME*10(150), STRING*12
00014 00165000  C FIND BEGINNING OF SUBROUTINE NAME
00014 00166000  INDEXNUM=INDEXNUM+LEN
00014 00167000  NAMECOUNTER=NAMECOUNTER+1
00020 00168000  C DETERMINE NAME
00020 00169000  CALL DETERMINE NAME(INDEXNUM, NAME(NAMECOUNTER), STRING)
00030 00170000  RETURN
00031 00171000  END

```

## SYMBOL MAP

| NAME           | TYPE      | STRUCTURE  | ADDRESS  | NAME     | TYPE       | STRUCTURE  | ADDRESS |
|----------------|-----------|------------|----------|----------|------------|------------|---------|
| DETERMINE NAME | INTEGER   | SUBROUTINE | 0-84 ,1  | INDEXNUM | INTEGER    | SIMPLE VAR | 0-86 ,1 |
| LEN            | INTEGER   | SIMPLE VAR | 0-84 ,1  | NAME     | CHARACTER  | ARRAY      | 0-87 ,1 |
| NAMECOUNTER    | INTEGER   | SIMPLE VAR | 0-810 ,1 | STORNAME | SUBROUTINE | SUBROUTINE |         |
| STRING         | CHARACTER | SIMPLE VAR | 0-85 ,1  |          |            |            |         |

PROGRAM UNIT STORNAME COMPILED



PAGE 0008    HEWLETT-PACKARD 32102H.01.09    FORTRAN/1000    WED, JUN 15, 1961, 12:24 PM

```

00010 00172000 $CONTROL INIT,MAP,LABEL,LOCATION
00010 00173000    SUBROUTINE DETERMINENAME(ISTART,NAME,STRING)
00010 00174000    CHARACTER NAME*16,STRING*12
00010 00175000    C    LOAD NAME INTO 'NAME' VARIABLE
00010 00176000    NAME=STRING(ISTART:16)
00017 00177000    RETURN
00020 00178000    END

```

# SYMBOL MAP

| NAME          | TYPE      | STRUCTURE  | ADDRESS | NAME   | TYPE      | STRUCTURE  | ADDRESS |
|---------------|-----------|------------|---------|--------|-----------|------------|---------|
| DETERMINENAME | CHARACTER | SUBROUTINE | 0-85 ,1 | ISTART | INTEGER   | SIMPLE VAR | 0-86 ,1 |
| NAME          | CHARACTER | SIMPLE VAR | 0-85 ,1 | STRING | CHARACTER | SIMPLE VAR | 0-86 ,1 |

PROGRAM UNIT DETERMINENAME COMPILED

```

PAGE 0009  HEWLETT-PACKARD  J21026.01.09  FORTNAN/JO00  MFD, JUN 15, 1963, 12124 PM

00051 00170000  SCUNTRUL INIT,MAP,LABEL,LOCUTION
00051 00180000  SUBROUTINE CLASSIFYSTATEMENT(COMMENTCOUNTER,FORMATCOUNTER,
00051 00190000  1DECLARFCOUNTER,LASTSTATETYPE,CALL,LC,SUB,MFC,SEPOINT,REDUCED,
00051 00182000  2ASA,ARGSUBSTITUTION)
00051 00184000  CHARACTER STRING*72,END*6,CALL*5,SIN*72,LASTSTATETYPE*2,SUB*12,
00051 00186000  1REDUCED*215007,EXITFLAG*1,ARGSUBSTITUTION*50110.8)
00051 00188000  INTEGER SEGBEGIN,COMMENTCOUNTER,FORMATCOUNTER,DECLARFCOUNTER,
00051 00190000  1SEPOINT,EXITC,ASA
00051 00192000  EXITFLAG=" "
00051 00194000  EXITFLAG=" "
00051 00196000  EXITFLAG=" "
00051 00198000  END=" "
00051 00200000  5 LC=LC+1
00051 00202000  READ(46MFC,1,END=99)STRING
00051 00204000  1 FORMAT(A72,514)
00051 00206000  C SKIP LINE IF COMMENT STATEMENT
00051 00208000  IF(STRING(1:11).EQ."C")COMMENTCOUNTER=COMMENTCOUNTER+1
00051 00210000  C DETERMINE IF CONTINUATION LINE
00051 00212000  IF(STRING(6:11).EQ." ")GO TO 11
00051 00214000  IF(LASTSTATETYPE.EQ."F")FORMATCOUNTER=FORMATCOUNTER+1
00051 00216000  IF(LASTSTATETYPE.EQ."D")DECLARFCOUNTER=DECLARFCOUNTER+1
00051 00218000  IF(LASTSTATETYPE.EQ." ")-GO TO 41
00051 00220000  GO TO 5
00051 00222000  C DETERMINE IF "DISPLAY" OR "FORMAT" STATEMENT
00051 00224000  11 CALL DETERMISFMI(STRING,FORMATCOUNTER,LASTSTATETYPE,CALL,85,85)
00051 00226000  LASTSTATETYPE=" "
00051 00228000  C DETERMINE IF ASSIGNMENT STATEMENT
00051 00230000  IF(INDEX(STRING,"=")-GT.0)GO TO 5
00051 00232000  C IF END STATEMENT, SLORE ENDING LINE 0 IN REDUCED ARRAY AND JUMP
00051 00234000  C TO ANALYZE NEXT PROGRAM UNIT
00051 00236000  IF(INDEX(STRING,END).EQ.0)GO TO 41
00051 00238000  MARK=INDEX(STRING,AND)
00051 00240000  IF(STRING(MARK-515).LT."99999")HEAD(REDUCED(5EPOINT),1)STRING,
00051 00242000  1SEGBEGIN
00051 00244000  IF(STRING(MARK-515).LT."99999")WRITE(REDUCED(5EPOINT),1)STRING,
00051 00246000  1SEGBEGIN,LC,COMMENTCOUNTER,FORMATCOUNTER,DECLARFCOUNTER
00051 00248000  IF(STRING(MARK-515).LT."99999")RETURN
00051 00250000  C IF CALL STATEMENT, ENTER INTO "REDUCED" ARRAY
00051 00252000  41 STRING(6:11)=" "
00051 00254000  NOCALL=1
00051 00256000  IF(INDEX(STRING,CALL).EQ.0)GO TO 74
00051 00258000  LOC=INDEX(STRING,CALL)
00051 00260000  IF(LOC-1).NE.6.AND.STRING(LOC-1:11).NE." ".AND.STRING(LOC-1:11)
00051 00262000  1.NE."")GO TO 74
00051 00264000  CALL PROCESSCALL(STRING,CALL,SUB,MFC,LC,REDUCED)
00051 00266000  NOCALL=0
00051 00268000  C IF THERE IS EXTERNAL STATEMENT IN PROGRAM UNIT, CHECK ARGUMENT
00051 00270000  C LIST OF CALL STATEMENT FOR MARK(S) DECLARED IN EXTERNAL STATEMENT
00051 00272000  IF(EXITFLAG.EQ."F".AND.EXITC.GT.0)CALL CHECKARGLIST(STRING,EXITC,
00051 00274000  1ARGSUBSTITUTION,MFC-1,REDUCED,CALL,LC,ASA)
00051 00276000  C DETERMINE IF DECLARATION STATEMENT
00051 00278000  74 IF(NOCALL.EQ.1)CALL
00051 00280000  1DETERDECLARE(STRING,DECLARFCOUNTER,LASTSTATETYPE,EXITFLAG,LC,EXITC)
00051 00282000  GO TO 5
00051 00284000  99 RETURN
00051 00286000  END
00051 00288000
00051 00290000

```

## PAGE 0010 CLASSIFYSTATEMENT

## SYMBOL MAP

| NAME              | TYPE      | STRUCTURE  | ADDRESS  | NAME           | TYPE       | STRUCTURE  | ADDRESS  |
|-------------------|-----------|------------|----------|----------------|------------|------------|----------|
| ARGSUBSTITUTION   | CHARACTER | ARRAY      | 0-14 ,1  | ADA            | INTEGER    | SIMPLE VAR | 0-85 ,1  |
| CALL              | CHARACTER | SIMPLE VAR | 0-113 ,1 | CHECKANGLIST   | SUBROUTINE | SUBROUTINE | 0-817 ,1 |
| CLASSIFYSTATEMENT | CHARACTER | SUBROUTINE | 0-115 ,1 | COMMENTCOUNTER | INTEGER    | SIMPLE VAR | 0-817 ,1 |
| DECLARACOUNTER    | INTEGER   | SIMPLE VAR | 0-115 ,1 | DECLARACOUNTER | SUBROUTINE | SUBROUTINE | 0-811 ,1 |
| DECLARACOUNTER    | CHARACTER | SUBROUTINE | 0-117 ,1 | END            | INTEGER    | SIMPLE VAR | 0-811 ,1 |
| EXTFLAG           | INTEGER   | SIMPLE VAR | 0-116 ,1 | EATLC          | CHARACTER  | SIMPLE VAR | 0-814 ,1 |
| FORMACOUNTER      | INTEGER   | SIMPLE VAR | 0-112 ,1 | LASTSTATEMENT  | INTEGER    | SIMPLE VAR | 0-85 ,1  |
| LC                | INTEGER   | SIMPLE VAR | 0-114 ,1 | LUC            | INTEGER    | SIMPLE VAR | 0-810 ,1 |
| MARK              | INTEGER   | SIMPLE VAR | 0-111 ,1 | PROCESSCALL    | SUBROUTINE | SUBROUTINE | 0-812 ,1 |
| RECALL            | CHARACTER | ARRAY      | 0-86 ,1  | SEGREGIN       | CHARACTER  | SIMPLE VAR | 0-816 ,1 |
| REDUCED           | INTEGER   | SIMPLE VAR | 0-87 ,1  | STRIN          | CHARACTER  | SIMPLE VAR | 0-811 ,1 |
| SEGPOINT          | CHARACTER | SIMPLE VAR | 0-810 ,1 | SUB            |            |            |          |
| STRING            |           |            |          |                |            |            |          |

## LABEL MAP

| STATEMENT LABEL | CODE OFFSET | STATEMENT LABEL | CODE OFFSET | STATEMENT LABEL | CODE OFFSET |
|-----------------|-------------|-----------------|-------------|-----------------|-------------|
| 1 PMT           | 0           | 5               | 100         | 11              | 220         |
|                 |             |                 |             | 41              | 464         |
|                 |             |                 |             | 614             | 650         |

## PROGRAM UNIT CLASSIFYSTATEMENT COMPILED

PAGE 0011 HELLFT-PACARD 11020.01-09 FORTKAM/1000 WED, JUN 15, 1983, 12:24 P4

```

00010 00235000 SCUNHOL INIT,MAP,LABEL, LOCATION
00010 00236000 SUBROUTINE DETENDISURFMT(STRING,FURNATCOUNTER, LASTSTATETYPE,CALL,
00010 00237000 1,0)
00010 00238000 CHARACTER LASTSTATETYPE*2,CALL*5,STRING*12
00010 00239000 INTEGER FURNATCOUNTER
00010 00240000 LASTSTATETYPE=" "
00017 00241000 IF(INDEX(STRING,"DISPLAY").GT.0.AND.INDEX(STRING,CALL).EQ.0)
00017 00242000 RETURN 2
00062 00243000 IF(INDEX(STRING," FURNAT(").GT.0.AND.INDEX(STRING," FURNAT").EQ.0)
00062 00244000 1)FURNATCOUNTER=FURNATCOUNTER+1
00130 00245000 IF(INDEX(STRING," FURNAT(").GT.0.AND.INDEX(STRING," FURNAT").EQ.0)
00130 00246000 1)LASTSTATETYPE="F"
00205 00247000 IF(INDEX(STRING," FURNAT(").GT.0.AND.INDEX(STRING," FURNAT").EQ.0)
00205 00248000 1)RETURN 1
00263 00249000 RETURN
00264 00250000 END

```

## SYMBOL MAP

| NAME          | TYPE      | STRUCTURE  | ADDRESS | NAME          | TYPE      | STRUCTURE  | ADDRESS |
|---------------|-----------|------------|---------|---------------|-----------|------------|---------|
| CALL          | CHARACTER | SIMPLE VAR | 0-14 .1 | DETENDISURFMT | CHARACTER | SUBROUTINE | 0-15 .1 |
| FURNATCOUNTER | INTEGER   | SIMPLE VAR | 0-16 .1 | LASTSTATETYPE | CHARACTER | SIMPLE VAR | 0-17 .1 |
| STRING        | CHARACTER | SIMPLE VAR | 0-17 .1 |               |           |            |         |

PROGRAM UNIT DETENDISURFMT COMPILED

```

00040 00251000 SCUNTRUL INIT,MAP,LOCUTION,LABEL
00040 00252000 SUBROUTINE PROCESSCALL(STRING,CALL,SUB,MLC,LC,REDUCED)
00040 00253000 CHARACTER STRING*12,CALL*5,SUB*12,MLC*STRLEN*14,NEXTSTRING*12,
00040 00254000 REDUCED*92(500),FUNCTION*10
00040 00255000 IF(INDEX(STRING,CALL).LT.5).AND.INDEX(STRING,"IF").EQ.0)GO TO 2
00070 00256000 C CHECK FOR CONTINUATION OF STATEMENT ON NEXT LINE
00070 00257000 READ(66LC),1)NEXTSTRING
00115 00258000 1 FORMAT(A/2,14)
00115 00259000 IF(NEXTSTRING(1:11).EQ."C")GO TO 2
00126 00260000 IF(NEXTSTRING(6:11).EQ."")GO TO 2
00141 00261000 NEXTSTRING(6:11)=" "
00152 00262000 NEWSTRING(1:12)=NEXTSTRING(6:17)
00162 00263000 NEWSTRING(1:12)=STRING
00166 00264000 C REMOVE ANY EXTRA BLANKS BETWEEN "CALL" AND NAME
00166 00265000 ISTART=INDEX(STRING,CALL)+5
00177 00266000 DO 10 I=ISTART,144
00204 00267000 IF(NEXTSTRING(I:I).NE."")GO TO 11
00225 00268000 10 CONTINUE
00226 00269000 11 IF(1.EQ.1)START=GO TO 12
00232 00270000 NEWSTRING(ISTART:144)=NEWSTRING(1:144-I)
00245 00271000 C ASSUME IF STATEMENT AND REMOVE ARGUMENT
00245 00272000 12 CALL PROCESSIF(NEWSTRING,STRING,CALL)
00251 00273000 2 CALL STRIPUPPARAM(STRING,CALL,SUB,FUNCTION)
00256 00274000 MLC=MLC+1
00307 00275000 RETURN
00310 00276000
00311 00277000 END

```

# SYMBOL MAP

| NAME         | TYPE      | STRUCTURE  | ADDRESS  | NAME      | TYPE       | STRUCTURE  | ADDRESS  |
|--------------|-----------|------------|----------|-----------|------------|------------|----------|
| CALL         | CHARACTER | SIMPLE VAR | 0-110 .1 | FUNCTION  | CHARACTER  | SIMPLE VAR | 0-111 .1 |
| I            | INTEGER   | SIMPLE VAR | 0-111    | ISTART    | INTEGER    | SIMPLE VAR | 0-112    |
| LC           | INTEGER   | SIMPLE VAR | 0-115 .1 | NEWSTRING | CHARACTER  | SIMPLE VAR | 0-114 .1 |
| NEXTSTRING   | CHARACTER | SIMPLE VAR | 0-115 .1 | MLC       | INTEGER    | SIMPLE VAR | 0-116 .1 |
| PROCESSCALL  | CHARACTER | SUBROUTINE | 0-116    | PROCESSIF | SUBROUTINE | SUBROUTINE | 0-117 .1 |
| REDUCED      | CHARACTER | ARRAY      | 0-116    | STRING    | CHARACTER  | SIMPLE VAR | 0-118 .1 |
| STRIPUPPARAM | CHARACTER | SUBROUTINE | 0-116    | SUB       | CHARACTER  | SIMPLE VAR | 0-119 .1 |

# LABEL MAP

| STATEMENT LABEL | CODE OFFSET | STATEMENT LABEL | CODE OFFSET | STATEMENT LABEL | CODE OFFSET |
|-----------------|-------------|-----------------|-------------|-----------------|-------------|
| 1 FMT           | 0           | 2               | 251         | 10              | 225         |
|                 |             |                 |             | 11              | 226         |
|                 |             |                 |             | 12              | 445         |

PROGRAM UNIT PROCESSCALL COMPILED

```

00020 00278000 SCURROL INIT,MAP,INDEX,LOCUTION
00020 00279000 SUBROUTINE PROCESSIF(NEWSTRING,STRING,CALL)
00020 00280000 CHARACTER IFF'J,STRING'72,NEWSTRING'149,CALL'5
00020 00281000 IFF=" IF"
00030 00282000 IF(INDEX(NEWSTRING,IFF),GT,0)GO TO 1
00041 00283000 DISPLAY EXPECTED IF STATEMENT WITH CALL. INSTEAD GOT"
00110 00284000 DISPLAY NEWSTRING
00121 00285000 DISPLAY"THIS PROGRAM IS TERMINATED."
00157 00286000 STOP
00160 00287000 C REMOVE ARGUMENT OF IF STATEMENT
00160 00288000 1 ISTART=INDEX(NEWSTRING,CALL)
00167 00289000 STRING(16:31)=IFF
00175 00290000 STRING(19:11)=" "
00206 00291000 STRING(10:61)=NEWSTRING(1:START:61)
00217 00292000 RETURN
00220 00293000 END

```

# SYMBOL MAP

| NAME      | TYPE      | STRUCTURE  | ADDRESS | NAME      | TYPE      | STRUCTURE  | ADDRESS |
|-----------|-----------|------------|---------|-----------|-----------|------------|---------|
| CALL      | CHARACTER | SIMPLE VAR | 0-34 ,1 | IFF       | CHARACTER | SIMPLE VAR | 0-12 ,1 |
| ISTART    | INTEGER   | SIMPLE VAR | 0-1     | NEWSTRING | CHARACTER | SIMPLE VAR | 0-16 ,1 |
| PROCESSIF |           | SUBROUTINE |         | STRING    | CHARACTER | SIMPLE VAR | 0-15 ,1 |

# LABEL MAP

| STATEMENT LABEL | CODE | STATEMENT LABEL | CODE | STATEMENT LABEL | CODE |
|-----------------|------|-----------------|------|-----------------|------|
| 1               | 100  |                 |      |                 |      |

PROGRAM UNIT PROCESSIF COMPILED

```

PAGE 0014  HEWLETT-PACKARD J21026.01.09  FURTHAN/1000  WED, JUN 15, 1983, 12:25 PM

00062 00294000 SCOUTNUL INIT,MAP,LABEL,LOCUTION
00062 00295000 C THIS SUBROUTINE DETERMINES IF ANY ARGUMENTS IN THIS CALL STATEMENT
00062 00296000 C ARE DECLARED IN THE EXTERNAL STATEMENT OF THIS PROGRAM UNIT.
00062 00297000 C IF TRUE, THEN THE FOLLOWING IS STORED IN 'ARGSUBSTITUTION'.
00062 00298000 C ARRAY:
00062 00299000 C (1) SUBROUTINE NAME CALLED BY THIS CALL STATEMENT
00062 00300000 C (2) ARGUMENT NAME FOUND IN EXTERNAL STATEMENT
00062 00301000 C (3) NUMBER OF POSITION IN ARGUMENT LIST OF ARGUMENT NAME
00062 00302000 C THEN THE CALL STATEMENT STORED IN 'REDUCED' ARRAY IS MARKED WITH THE
00062 00303000 C ARRAY POSITION WHERE THE ABOVE INFORMATION HAS JUST BEEN STORED
00062 00304000 C SUBROUTINE CHECKANGLIST,EXTLC,ARGSUBSTITUTION,RLC,
00062 00305000 C REDUCED,CALL,LC,ASA)
00062 00306000 C CHARACTER REDUCED*92(500),STRING*72,ARGSUBSTITUTION*50(10,8),
00062 00307000 C 1CALL*5,EXTSTRING*72,SUBNAME*16,CALLSTRING*72,ARG*16(50)
00062 00308000 C INTEGER EXTLC,MUMCHAR(50),ASA
00062 00309000 C =0
00064 00310000 C DETERMINE SUBROUTINE NAME CALLED
00064 00311000 C INDEXNUM=INDEX(ISHING,CALL)*5
00064 00312000 C CALL DETERMINE(INDEXNUM,SUBNAME,STRING)
00075 00313000 C READ EXTERNAL STATEMENT AND CALL STATEMENT FROM SOURCE FILE
00101 00314000 C READ(466,EXTLC,11)EXTSTRING
00101 00315000 C 111 FORMAT(A72,214)
00125 00316000 C HEAD(466,LC,11)CALLSTRING
00125 00317000 C DETERMINE # OF CHARACTERS FUN NAMES IN EXTERNAL STATEMENT
00147 00318000 C ISTART=INDEX(EXTSTRING," EXTERNAL ")+10
00147 00319000 C I=0
00176 00320000 C 39 I=I+1
00177 00321000 C DO 30 J=ISTART,ISTART+15
00205 00322000 C IF(EXTSTRING(I:J).EQ." DM.EXTSTRING(I:J).EQ." )JCU TO 31
00246 00323000 C 30 CONTINUE
00247 00324000 C 31 MUMCHAR(I)=J-ISTART
00253 00325000 C ARG(I)=EXTSTRING(ISTART:MUMCHAR(I))
00263 00326000 C ISTART=J+1
00272 00327000 C IF(EXTSTRING(I:J).NE." ")JCU TO 39
00312 00328000 C DETERMINE IF ANY EXTERNAL NAMES APPEAR IN CALL STATEMENT
00312 00329000 C DO 10 K=1,I
00317 00330000 C IF(INDEX(CALLSTRING,ARG(K))>MUMCHAR(K))JCU TO 10
00340 00331000 C DETERMINE POSITION OF NAME IN CALL LIST
00340 00332000 C INDEXNUM=INDEX(CALLSTRING,ARG(K))-(MUMCHAR(K)-1)
00357 00333000 C LISTPOS=I.
00361 00334000 C DO 300 L=1,INDEXNUM
00366 00335000 C IF(CALLSTRING(L:11).EQ." ")LISTPOS=LISTPOS+1
00406 00336000 C 300 CONTINUE
00407 00337000 C STORE INFO IN ARGSUBSTITUTION ARRAY
00407 00338000 C N=M+1
00410 00339000 C IF(N.EQ.1)ASA=ASA+1
00414 00340000 C WRITE(ARGSUBSTITUTION(ASA,N,7))SUBNAME,ARG(N),LISTPOS
00460 00341000 C 77 FORMAT(2A16,16X,12)
00460 00342000 C 10 CONTINUE
00461 00343000 C WRITE 'ARGSUBSTITUTION' ARRAY'S POSITION TO CALL STATEMENT IN
00461 00344000 C REDUCED FILE, IF ANY EXTERNAL NAME'S FOUND IN CALL STATEMENT
00461 00345000 C IF(N.EQ.0)GO TO 98
00466 00346000 C WRITE(REDUCED(466),111)STRING,LC,ASA
00521 00347000 C 98 RETURN
00523 00348000 C END

```

## PAGE 0015 CHECKARGLIST

## SYMBOL MAP

| NAME       | TYPE      | STRUCTURE  | ADDRESS | NAME            | TYPE       | STRUCTURE  | ADDRESS |
|------------|-----------|------------|---------|-----------------|------------|------------|---------|
| ARG        | CHARACTER | ARRAY      | 0-12 .1 | ARGSUBSTITUTION | CHARACTER  | ARRAY      | 0-11 .1 |
| ASA        | INTEGER   | SIMPLE VAR | 0-13 .1 | CALL            | CHARACTER  | SIMPLE VAR | 0-10 .1 |
| CALLSTRING | CHARACTER | SIMPLE VAR | 0-14 .1 | CHECKARGLIST    | SUBROUTINE | SUBROUTINE |         |
| DETERMINE  | CHARACTER | SUBROUTINE |         | EXITC           | INTEGER    | SIMPLE VAR | 0-12 .1 |
| EXITSTRING | CHARACTER | SIMPLE VAR | 0-13 .1 | I               | INTEGER    | SIMPLE VAR | 0-13    |
| INDEXNUM   | INTEGER   | SIMPLE VAR | 0-12    | ISTART          | INTEGER    | SIMPLE VAR | 0-10    |
| J          | INTEGER   | SIMPLE VAR | 0-16    | K               | INTEGER    | SIMPLE VAR | 0-11    |
| L          | INTEGER   | SIMPLE VAR | 0-15    | LC              | INTEGER    | SIMPLE VAR | 0-15 .1 |
| LASTPOS    | INTEGER   | SIMPLE VAR | 0-17    | N               | INTEGER    | SIMPLE VAR | 0-14    |
| NLC        | INTEGER   | SIMPLE VAR | 0-10 .1 | NUMCHAR         | ARRAY      | ARRAY      | 0-11 .1 |
| REDUCED    | CHARACTER | ARRAY      | 0-17 .1 | STRING          | CHARACTER  | SIMPLE VAR | 0-13 .1 |
| SUBNAME    | CHARACTER | SIMPLE VAR | 0-12 .1 |                 |            |            |         |

## LABEL MAP

| STATEMENT LABEL | CODE OFFSET | STATEMENT LABEL | CODE OFFSET | STATEMENT LABEL | CODE OFFSET | STATEMENT LABEL | CODE OFFSET |
|-----------------|-------------|-----------------|-------------|-----------------|-------------|-----------------|-------------|
| 10              | 400         | 30              | 240         | 39              | 176         | 98              | 521         |
| 111 FMT         | 0           | 300             | 406         |                 |             | 5               |             |
|                 |             |                 |             | 17 FMT          |             |                 |             |

PROGRAM UNIT CHECKARGLIST COMPILED



| SYMBOL MAP  |           |            |          |              |           |            |          |
|-------------|-----------|------------|----------|--------------|-----------|------------|----------|
| NAME        | TYPE      | STRUCTURE  | ADDRESS  | NAME         | TYPE      | STRUCTURE  | ADDRESS  |
| COUNTER     | INTEGER   | SIMPLE VAR | 0-110 ,1 | DETERDECLARE |           | SUBROUTINE |          |
| EXTFLAG     | CHARACTER | SIMPLE VAR | 0-16 ,1  | EXTLC        | INTEGER   | SIMPLE VAR | 0-14 ,1  |
| LASTSTATYPE | CHARACTER | SIMPLE VAR | 0-17 ,1  | LC           | INTEGER   | SIMPLE VAR | 0-15 ,1  |
| UDDCOUNTER  | INTEGER   | SIMPLE VAR | 0-11     | STRING       | CHARACTER | SIMPLE VAR | 0-111 ,1 |

PAGE 0017 DETENDECLANE

LABEL MAP

| STATEMENT<br>LABEL | CODE<br>OFFSET | STATEMENT<br>LABEL | CODE<br>OFFSET | STATEMENT<br>LABEL | CODE<br>OFFSET | STATEMENT<br>LABEL | CODE<br>OFFSET | STATEMENT<br>LABEL | CODE<br>OFFSET |
|--------------------|----------------|--------------------|----------------|--------------------|----------------|--------------------|----------------|--------------------|----------------|
| 15                 | 740            |                    |                |                    |                |                    |                |                    |                |

PROGRAM UNIT DETENDECLANE COMPILED

PAGE 0018    HEMLETT-PACKARD 32102B-01-09    FURTHAN/3000    WED, JUN 15, 1993, 12:25 PM

```

00041 00391000    SCOUNTER, INIT, MAX, LABEL, LOCATION
00041 00392000    SUBROUTINE DELETED(CALLS(SUBNAME, SUBNAMECOUNT, CALL, SUB, REDUCED,
00041 00393000    IMLC, FUNCTION))
00041 00394000    CHARACTER SUBNAME*16(100), STRING*72, CALL*5, NAME*16, SUB*12,
00041 00395000    REDUCED*92(500), FUNCTION*10
00041 00396000    INTEGER SEGBEGIN, SEGEND, COMMENTCOUNT, FURNATCOUNT,
00041 00397000    DECLARECOUNT, SUBNAMECOUNT
00041 00398000    LC=0
00043 00399000    DO 20 J=1,MLC
00050 00400000    READ(REDUCED(J),9)STRING,SEGBEGIN,SEGEND,COMMENTCOUNT,
00050 00401000    IFURNATCOUNT,DECLARECOUNT
00111 00402000    9 FORMAT(A72,5I4)
00111 00403000    C DETERMINE IF PROGRAM UNIT STATEMENT
00111 00404000    IF(STRING(1:4).EQ."MAIN")WRITE(REDUCED(LC+1),9)STRING,SEGBEGIN,
00111 00405000    ISEGEND,COMMENTCOUNT,FURNATCOUNT,DECLARECOUNT
00171 00406000    IF(INDEX(STRING,SUB).GT.0)WRITE(REDUCED(LC+1),9)STRING,SEGBEGIN,
00171 00407000    ISEGEND,COMMENTCOUNT,FURNATCOUNT,DECLARECOUNT
00246 00408000    IF(INDEX(STRING,FUNCTION).GT.0)WRITE(REDUCED(LC+1),9)STRING,
00246 00409000    ISEGBEGIN,SEGEND,COMMENTCOUNT,FURNATCOUNT,DECLARECOUNT
00323 00410000    IF(INDEX(STRING,SUB).GT.0)LC=LC+1
00323 00411000    IF(INDEX(STRING,SUB).GT.0)LC=LC+1
00361 00412000    1,FUNCTION).GT.0)GO TO 20
00417 00413000    C DETERMINE TO BE 'CALL' STATEMENT; DETERMINE BEGINNING OF CALL NAME
00417 00414000    IENDNUMBER=INDEX(STRING,CALL)+5
00430 00415000    C DETERMINE NAME OF CALL
00430 00416000    CALL DETERMINE(INDXNUM,NAME,STRING)
00430 00417000    C DETERMINE IF CALL NAME EXISTS AS A SUBROUTINE NAME
00430 00418000    DO 10 I=1,SUBNAMECOUNT
00441 00419000    IF(NAME.EQ.SUBNAME(I))WRITE(REDUCED(LC+1),9)STRING,SEGBEGIN,SEGEND
00512 00420000    IF(NAME.EQ.SUBNAME(I))LC=LC+1
00524 00421000    IF(NAME.EQ.SUBNAME(I))GO TO 20
00537 00422000    10 CONTINUE
00540 00423000    C SUBROUTINE NAME NOT FOUND--THEREFORE, CALL STATEMENT NOT WRITTEN TO
00540 00424000    C "REDUCED" ARRAY
00540 00425000    20 CONTINUE
00540 00426000    99 MLC=LC
00544 00427000    RETURN
00546 00428000    END
00547 00429000

```

## SYMBOL MAP

| NAME         | TYPE      | STRUCTURE  | ADDRESS  | NAME         | TYPE      | STRUCTURE  | ADDRESS  |
|--------------|-----------|------------|----------|--------------|-----------|------------|----------|
| CALL         | CHARACTER | SIMPLE VAR | 0-810 ,1 | CURRENTCOUNT | INTEGER   | SIMPLE VAR | 0-81 ,1  |
| DECLARFCOUNT | INTEGER   | SIMPLE VAR | 0-812 ,1 | DELETEDCALLS | INTEGER   | SIMPLE VAR | 0-811 ,1 |
| DETERMINE    | INTEGER   | SUBROUTINE | 0-812 ,1 | FURNATCOUNT  | INTEGER   | SIMPLE VAR | 0-811 ,1 |
| FUNCTION     | CHARACTER | SIMPLE VAR | 0-84 ,1  | I            | INTEGER   | SIMPLE VAR | 0-811 ,1 |
| INDEXNUM     | INTEGER   | SIMPLE VAR | 0-810 ,1 | J            | INTEGER   | SIMPLE VAR | 0-811 ,1 |
| LC           | INTEGER   | SIMPLE VAR | 0-86 ,1  | NAME         | CHARACTER | ARRAY      | 0-86 ,1  |
| MLC          | INTEGER   | SIMPLE VAR | 0-85 ,1  | REDUCED      | INTEGER   | SIMPLE VAR | 0-86 ,1  |
| SEGBEGIN     | INTEGER   | SIMPLE VAR | 0-84 ,1  | SEGEND       | INTEGER   | SIMPLE VAR | 0-86 ,1  |
| STRING       | CHARACTER | SIMPLE VAR | 0-812 ,1 | SUB          | CHARACTER | SIMPLE VAR | 0-81 ,1  |
| SUBNAME      | CHARACTER | ARRAY      | 0-812 ,1 | SUBNAMECOUNT | INTEGER   | SIMPLE VAR | 0-811 ,1 |

PAGE 0019 DELETED/CALLS

LABEL MAP

| STATEMENT<br>LABEL | CODE<br>OFFSET | STATEMENT<br>LABEL | CODE<br>OFFSET | STATEMENT<br>LABEL | CODE<br>OFFSET | STATEMENT<br>LABEL | CODE<br>OFFSET | STATEMENT<br>LABEL | CODE<br>OFFSET |
|--------------------|----------------|--------------------|----------------|--------------------|----------------|--------------------|----------------|--------------------|----------------|
| 9 FMT              | 0              | 10                 | 537            | 20                 | 540            | 99                 | 544            |                    |                |

PROGRAM UNIT DELETED/CALLS COMPILED

```

PAGE 0020  HENLETT-PACKARD 321020.01.09  FORTNAN/1000  WED, JUN 15, 1983, 12:25 PM

00061 00430000 SCUNROL INIT,MAP,LABEL,LOCUTION
00061 00431000 C THIS SUBROUTINE FINDS FUNCTION CALLS AND ADDS THEM TO REDUCED ARRAY
00061 00432000 SUBROUTINE INSTEADFUNCTIONCALLS(FUNCTIONNAME,FUNCTIONCOUNTER,REDUCED,
00061 00433000 IFUNCTION,NLC,CALL,SUB)
00061 00434000 CHARACTER FUNCTIONNAME*10(100),FUNCTION*10,CALL*5,REDUCED*92(500),
00061 00435000 ISUB*12,STRING*72,LASTSTATE*2,EXTFLAG*1
00061 00436000 INTEGER FUNCTIONCOUNTER,FORMATCOUNTER,NUNCHAR(100),DECLARECOUNTER,
00061 00437000 IEXTLC
00061 00438000 LC=0
00061 00439000 C DETERMINE # OF CHARACTERS IN EACH FUNCTION NAME
00061 00440000 DO 20 I=1,FUNCTIONCOUNTER
00061 00441000 DO 30 J=1,16
00061 00442000 IF(FUNCTIONNAME(I):J).EQ." " GO TO 31
00061 00443000 30 CONTINUE
00061 00444000 31 NUNCHAR(I)=J-1
00061 00445000 20 CONTINUE
00061 00446000 C REWIND SOURCE CODE FILE
00061 00447000 REWIND 46
00061 00448000 C READ LINE FROM SOURCE FILE
00061 00449000 2 LC=LC+1
00061 00450000 READ(46,LC,1,END=99)STRING
00061 00451000 1 FORMAT(72,19)
00061 00452000 C SKIP IF COMMENT OR CONTROL STATEMENT
00061 00453000 IF(STRING(1:11).EQ."C" GO TO 2
00061 00454000 IF(STRING(1:11).EQ."S" GO TO 2
00061 00455000 C DETERMINE IF CONTINUATION LINE
00061 00456000 IF(STRING(6:11).EQ." " GO TO 1
00061 00457000 C IF PREVIOUS STATEMENT IS DECLAR. OR FORMAT, SKIP CONTINUATION LINE
00061 00458000 IF(LASTSTATE.EQ."F" OR LASTSTATE.EQ."DE") GO TO 2
00061 00459000 C SKIP IF DECLARATIVE STATEMENT
00061 00460000 3 LASTSTATE=" "
00061 00461000 IF(INDEX(STRING,CALL).EQ.0 AND INDEX(STRING," IF(").EQ.0)
00061 00462000 ICALL DETERMINE(STRING,DECLARECOUNTER,LASTSTATE,EXTFLAG,LC,
00061 00463000 IEXTLC)
00061 00464000 IF(LASTSTATE.EQ."DE") GO TO 2
00061 00465000 C SKIP IF "FORMAT" STATEMENT
00061 00466000 CALL DETERMINE(STRING,FORMATCOUNTER,LASTSTATE,EXTFLAG,LC,
00061 00467000 29 LASTSTATE=" "
00061 00468000 C DETERMINE IF STATEMENT CONTAINS FUNCTION NAME
00061 00469000 DO 10 I=1,FUNCTIONCOUNTER
00061 00470000 IF(INDEX(STRING,FUNCTIONNAME(I):NUNCHAR(I)):GT.0) GO TO 111
00061 00471000 GO TO 10
00061 00472000 C DETERMINE THAT NAME IS NOT INCLUDED IN A LONGER NAME
00061 00473000 111 INDEXNUM=INDEX(STRING,FUNCTIONNAME(I):NUNCHAR(I))-1
00061 00474000 IF(STRING(INDEXNUM+1:LC,"A".AND.STRING(INDEXNUM+1:LC,"Z"))
00061 00475000 10 GO TO 10
00061 00476000 C DETERMINE THAT "(" IS IMMEDIATELY TO RIGHT OF FUNCTION NAME
00061 00477000 11 INDEXNUM=INDEX(STRING,FUNCTIONNAME(I):NUNCHAR(I)):NUNCHAR(I)
00061 00478000 IF(INDEX(STRING(INDEXNUM+1:LC,"(").EQ.0) GO TO 10
00061 00479000 C STATEMENT DETERMINED TO BE FUNCTION CALL; ADD TO REDUCED FILE
00061 00480000 NLC=NLC+1
00061 00481000 #RITE(STRING,1//FUNCTIONNAME(I)
00061 00482000 // FORMAT(1 CALL ",A16)
00061 00483000 #RITE(REDUCED(NLC),1)STRING,LC
00061 00484000 10 CONTINUE
00061 00485000 GO TO 2
00061 00486000 C SORT REDUCED FILE SO THAT FUNCTION CALL STATEMENTS ARE IN SAME

```

## PAGE 0021 INSERTFUNCTIONCALLS

```

00577 00480000 C SEQUENCE IN REDUCED ARRAY AS THEY ARE IN SOURCE CODE FILE
00577 00480000 99 CALL SURTARRAY(REDUCED,MLC,7,3,4)
00611 00480000 RETURN
00612 00480000 END

```

## SYMBOL MAP

| NAME                | TYPE      | STRUCTURE  | ADDRESS | NAME          | TYPE      | STRUCTURE  | ADDRESS |
|---------------------|-----------|------------|---------|---------------|-----------|------------|---------|
| CALL                | CHARACTER | SIMPLE VAR | 0-15 .1 | DECLARCOUNTEN | INTEGER   | SIMPLE VAR | 0-14    |
| DETENDCLARE         | CHARACTER | SUBROUTINE | 0-11 .1 | DETENDISURFMT | INTEGER   | SUBROUTINE | 0-13    |
| EXTFLAG             | CHARACTER | SIMPLE VAR | 0-11 .1 | EXTLC         | INTEGER   | SIMPLE VAR | 0-13    |
| FORMATCOUNTEN       | INTEGER   | SIMPLE VAR | 0-10    | FUNCTION      | CHARACTER | NULL       | 0-13 .1 |
| FUNCTIONCOUNTEN     | INTEGER   | SIMPLE VAR | 0-11 .1 | FUNCTIONNAME  | CHARACTER | ARRAY      | 0-12 .1 |
| I                   | INTEGER   | SIMPLE VAR | 0-12    | INDEXNUM      | INTEGER   | SIMPLE VAR | 0-13    |
| INSERTFUNCTIONCALLS | CHARACTER | SUBROUTINE | 0-11 .1 | J             | INTEGER   | SIMPLE VAR | 0-13    |
| LASTSTATETYPE       | CHARACTER | SIMPLE VAR | 0-11 .1 | LC            | INTEGER   | SIMPLE VAR | 0-13    |
| MLC                 | INTEGER   | SIMPLE VAR | 0-16 .1 | NUMCHAN       | INTEGER   | ARRAY      | 0-11 .1 |
| REDUCED             | CHARACTER | ARRAY      | 0-10 .1 | SURTARRAY     | INTEGER   | SUBROUTINE | 0-13 .1 |
| STRING              | CHARACTER | SIMPLE VAR | 0-12 .1 | SUB           | CHARACTER | NULL       | 0-14 .1 |

## LABEL MAP

| STATEMENT LABEL | CODE OFFSET | STATEMENT LABEL | CODE OFFSET | STATEMENT LABEL | CODE OFFSET | STATEMENT LABEL | CODE OFFSET |
|-----------------|-------------|-----------------|-------------|-----------------|-------------|-----------------|-------------|
| 1 FMT           | 0           | 2               | 133         | 3               | 241         | 10              | 5/2         |
| 29              | 331         | 30              | 122         | 31              | 123         | 11              | 99          |
|                 |             |                 |             |                 |             | 11              | 44/         |
|                 |             |                 |             |                 |             | 20              | 5/7         |
|                 |             |                 |             |                 |             | 111             | 12/         |
|                 |             |                 |             |                 |             |                 | 36/         |

PROGRAM UNIT INSERTFUNCTIONCALLS COMPILED

PAGE 0022    HEWLETT-PACKARD 321020.01.09    FORTKAW/J000    4ED, JUN 15, 1983, 12:25 PM

```

00014 00491000 SCNTRL INIT,MAP,LABEL,LOCATON
00014 00492000 SUBROUTINE SORTARRAY(REDUCED,MLC,KEYSTART,KEYLENGTH)
00014 00493000 CHARACTER REDUCED*92(500)
00014 00494000 C SORT REDUCED ARRAY INTO ASCENDING LINE NUMBER ORDER
00014 00495000 DO 100 I=1,MLC
00021 00496000 JJ=1
00023 00497000 101 IF(JJ.GE.MLC)GO TO 100
00027 00498000 IF(REDUCED(JJ)(KEYSTART:KEYLENGTH).GT.REDUCED(JJ+1)(KEYSTART:
00027 00499000 KEYLENGTH))CALL SWITCH(REDUCED(JJ),REDUCED(JJ+1))
00066 00500000 JJ=JJ+1
00067 00501000 GO TO 101
00070 00502000 100 CONTINUE
00071 00503000 RETURN
00072 00504000 END

```

# SIMBOL MAP

| NAME      | TYPE    | STRUCTURE  | ADDRESS | NAME     | TYPE      | STRUCTURE  | ADDRESS |
|-----------|---------|------------|---------|----------|-----------|------------|---------|
| I         | INTEGER | SIMPLE VAR | 0-81    | JJ       | INTEGER   | SIMPLE VAR | 0-82    |
| KEYLENGTH | INTEGER | SIMPLE VAR | 0-84    | KEYSTART | INTEGER   | SIMPLE VAR | 0-85    |
| MLC       | INTEGER | SIMPLE VAR | 0-86    | REDUCED  | CHARACTER | ARRAY      | 0-91    |
| SORTARRAY |         | SUBROUTINE |         | SWITCH   |           | SUBROUTINE |         |

# LABEL MAP

| STATEMENT LABEL | CODE OFFSET | STATEMENT LABEL | CODE OFFSET | STATEMENT LABEL | CODE OFFSET |
|-----------------|-------------|-----------------|-------------|-----------------|-------------|
| 100             | 70          | 101             | 23          |                 |             |

PROGRAM UNIT SORTARRAY COMPILED

PAGE 0023 HEMLETT-PACKARD 121020.01.09 FURTRAM/3000 WED, JUN 15, 1983, 12:25 PM

```

00020 00505000 SCURFUL INIT,NAP,LAUER,LOCATION
00020 00506000 SUBROUTINE SWITCH(RECORDJ,RECORDJ1)
00020 00507000 CHARACTER RECORDJ*92,RECORDJ1*92,TEM*92
00020 00508000 TEM=RECORDJ
00024 00509000 RECORDJ=RECORDJ1
00030 00510000 RECORDJ1=TEM
00034 00511000 RETURN
00035 00512000 END

```

SYMBOL MAP

| NAME              | TYPE      | STRUCTURE  | ADDRESS | NAME            | TYPE      | STRUCTURE  | ADDRESS |
|-------------------|-----------|------------|---------|-----------------|-----------|------------|---------|
| RECORDJ<br>SWITCH | CHARACTER | SIMPLE VAN | U-85 ,1 | RECORDJ1<br>TEM | CHARACTER | SIMPLE VAN | U-84 ,1 |
|                   |           | SUBROUTINE |         |                 |           | SIMPLE VAN | U-81 ,1 |

PROGRAM UNIT SWITCH COMPILED



```

PAGE 0024  HE=LETT-PACKARD J21020-01-09  FURTHAN/1000  #ED, JUN 15, 1983, 12125 PM

00120 00513000 SCNTROL INIT,MAP LABEL,LOCATON
00120 00514000 SUBROUTINE COMPLETECARGLIST(REDUCED,SUB,CALL,ARGSUBSTITUTION,MLC)
00120 00515000 CHARACTER STRING*72,CALL*5,NAME*16,SUB*12,REDUCED*92(500),
00120 00516000 IFUNCTION*10,ARGSUBSTITUTION*50(10,0),ARGNAME*16,SUBARGNAME*16,
00120 00517000 2CONSUBNAME*28
00120 00518000 INTEGER SEGBEGIN,SEGENO,ULC
00120 00519000 LC=0
00122 00520000 ULC=MLC
00124 00521000 DO 20 J=1,MLC
00131 00522000 READ(REDUCED(J),9)STRING,SEGBEGIN,LOC
00164 00523000 9 FORMAT(A72,2I4)
00164 00524000 C DETERMINE IF CALL STATEMENT WITHIN PROGRAM UNIT WITH EXTERNAL STATEMENT
00164 00525000 IF(INDEX(STRING,CALL).GT.0.AND,LOC.GT.0)GO TO 11
00200 00526000 GO TO 20
00201 00527000 11 ISTART=INDEX(STRING,CALL)+5
00212 00528000 NAME=STRING(ISTART:16)
00221 00529000 C FIND LOCATION OF SUBROUTINE BY SERIALY HEADING "REDUCED" FILE
00250 00530000 65 FORNAT(A12,A16)
00250 00531000 DO 10 I=1,MLC
00255 00532000 READ(REDUCED(I),9)STRING,SEGBEGIN,SEGENO
00310 00533000 IF(INDEX(STRING,CONSUBNAME).GT.0)GO TO 12
00322 00534000 10 CONTINUE
00323 00535000 DISPLAY NAME," NOT FOUND AS SUBROUTINE IN 'REDUCED' FILE. ERROR"
00400 00536000 STOP
00401 00537000 C READ ARGUMENT NAME OF CALL STATEMENT AND ITS LIST POSITION FROM
00401 00538000 C ARGSUBSTITUTION ARRAY
00401 00539000 12 DO 100 N=1,N
00406 00540000 77 READ(ARGSUBSTITUTION(LOC,N),17)NAME,ARGNAME,LISTPOS
00446 00541000 IF(LISTPOS.EQ.0)GO TO 20
00453 00542000 C GET ARGUMENT NAME FROM SAME LIST POSITION OF SUBROUTINE STATEMENT IN
00453 00543000 C SOURCE FILE AND STORE IN "ARGSUBSTITUTION" ARRAY AND FIND WHERE
00453 00544000 C CALLED IN THIS CALLED SUBROUTINE
00453 00545000 CALL GETSUBARGNAME(ARGSUBSTITUTION,ARGNAME,LISTPOS,SEGBEGIN,
00453 00546000 1)LOC,CALL,SUB,NAME,LOC,N,SUBARGNAME)
00471 00547000 C ADD SUBARGNAME TO "REDUCED" FILE IF IT IS NOT ALREADY THERE
00471 00548000 IF(OLC.EQ.NLC)GO TO 987
00475 00549000 DO 950 L=OLC+1,NLC
00475 00550000 READ(REDUCED(L),9)STRING,SEGBEGIN
00503 00551000 IF(INDEX(STRING,SUBARGNAME).GT.0)GO TO 100
00534 00552000 950 CONTINUE
00551 00553000 987 NLC=NLC+1
00552 00554000 WRITE(REDUCED(NLC),9)SUBARGNAME,LOC,CALL
00553 00555000 91 FORNAT(6X,"CALL ",A16,"PASSED VIA EXTERNAL",20X,I4)
00604 00556000 100 CONTINUE
00605 00557000 C SORT REDUCED FILE SO THAT JUST INSERTED CALLS ARE IN SAME SEQUENCE
00612 00558000 C IN REDUCED ARRAY AS THEY ARE IN SOURCE CODE FILE
00612 00559000 CALL SORTARRAY(REDUCED,NLC,7,4)
00624 00560000 RETURN
00625 00561000 END

```

## PAGE 0025 COMPLETEANGLIST

## SYMBOL MAP

| NAME       | TYPE      | STRUCTURE  | ADDRESS | NAME            | TYPE       | STRUCTURE  | ADDRESS  |
|------------|-----------|------------|---------|-----------------|------------|------------|----------|
| ARGNAME    | CHARACTER | SIMPLE VAR | 0415 ,1 | ARGSUBSTITUTION | CHARACTER  | ARRAY      | 0-45 ,1  |
| CALL       | CHARACTER | SIMPLE VAR | 0-16 ,1 | COMPLETEANGLIST | SUBROUTINE |            |          |
| CUNSUBNAME | CHARACTER | SIMPLE VAR | 0421 ,1 | FUNCTION        | NULL       |            |          |
| GETSUBNAME |           | SUBROUTINE |         | I               | CHARACTER  |            |          |
| ISTANT     | INTEGER   | SIMPLE VAR | 0411    | J               | INTEGER    | SIMPLE VAR | 0411     |
| L          | INTEGER   | SIMPLE VAR | 0413    | J               | INTEGER    | SIMPLE VAR | 0416     |
| LISTPLUS   | INTEGER   | SIMPLE VAR | 0417    | J               | INTEGER    | SIMPLE VAR | 0410     |
| LUCCALL    | INTEGER   | SIMPLE VAR | 0414    | LUC             | INTEGER    | SIMPLE VAR | 0413     |
| NAME       | CHARACTER | SIMPLE VAR | 0420 ,1 | N               | INTEGER    | SIMPLE VAR | 0412     |
| OLC        | INTEGER   | SIMPLE VAR | 0412    | NLC             | INTEGER    | SIMPLE VAR | 0-84 ,1  |
| SEGECIN    | INTEGER   | SIMPLE VAR | 0415    | REDUCED         | CHARACTER  | ARRAY      | 0-110 ,1 |
| SORTANAY   | INTEGER   | SUBROUTINE |         | SEND            | CHARACTER  | SIMPLE VAR | 0414     |
| SUB        | CHARACTER | SIMPLE VAR | 0-37 ,1 | STRING          | CHARACTER  | SIMPLE VAR | 0417 ,1  |
|            |           |            |         | SUBNAME         | CHARACTER  | SIMPLE VAR | 0416 ,1  |

## LABEL MAP

| STATEMENT LABEL | CODE OFFSET | STATEMENT LABEL | CODE OFFSET | STATEMENT LABEL | CODE OFFSET | STATEMENT LABEL | CODE OFFSET |
|-----------------|-------------|-----------------|-------------|-----------------|-------------|-----------------|-------------|
| 9 FMT           | 0           | 10              | 122         | 11              | 201         | 12              | 401         |
| 17 FMT          | 12          | 91 FMT          | 21          | 100             | 604         | 987             | 551         |
|                 |             |                 |             |                 |             | 65 FMT          | 552         |
|                 |             |                 |             |                 |             |                 | 5           |

## PROGRAM UNIT COMPLETEANGLIST COMPILED

PAGE 0026 NEWLEET-PACKARD 321020-01-09 FORTNAD/1000 WED, JUN 15, 1961, 12125 PM

```

00044 00565000 SCUNTHOL INIT,MAP,LAHPL,LOCUTION
00044 00566000 C THIS SUBROUTINE FINDS ARGUMENT NAME IN SUBROUTINE STATEMENT
00044 00567000 C THAT THE CALLING PROGRAM UNIT HAS DECLARED AS "EXTERNAL"
00044 00568000 C AND STORES IT IN "ARGSUBSTITUTION" ARRAY
00044 00569000 C THEN THE CALL STATEMENT USING THIS ARGUMENT IS FOUND AND ITS LOCATION
00044 00570000 C IN THE SOURCE CODE FILE NOTED
00044 00571000 SUBROUTINE GETSUBARGNAME(ARGSUBSTITUTION,ARGNAME,LISTPOS,SEGREGIN,
00044 00572000 ILOCALL,CALL,SUB,SUBNAME,ILOC,M,SUBARGNAME)
00044 00573000 CHARACTER STRING*72,ARGSUBSTITUTION*50(10,8),SUB*12,SUBNAME*10,
00044 00574000 ICALL*5,ARGNAME*10,CALLSTRING*72,SUBARGNAME*10,CUNCALLNAME*21
00044 00575000 INTEGER FOUND,SEGREGIN
00044 00576000 LC=SEGREGIN-1
00044 00577000 95 LC=LC*31
00044 00578000 C FIND SUBROUTINE STATEMENT
00044 00579000 READ(469LC,11)STRING
00044 00580000 111 FORMAT(A/2)
00044 00581000 C IF(INDEX(STRING,SUB).EQ.0)GO TO 95
00044 00582000 C GET ARGUMENT NAME FROM ARGUMENT LIST OF SUBROUTINE STATEMENT
00044 00583000 FOUND=0
00044 00584000 DO 400 K=INDEX(STRING,SUB),72
00044 00585000 IF(STRING(K:11).NE."("AND.STRING(K:11).NE."")GO TO 400
00044 00586000 FOUND=FOUND+1
00044 00587000 IF(FOUND.EQ.LISTPOS)GO TO 301
00044 00588000 400 CONTINUE
00044 00589000 301 ISTART=K+1
00044 00590000 DO 30 J=ISTART,ISTART+15
00044 00591000 IF(STRING(J:11).EQ."")OR.STRING(J:11).EQ."")GO TO 31
00044 00592000 30 CONTINUE
00044 00593000 31 NUNCHAR=J-ISTART
00044 00594000 SUBARGNAME=STRING(ISTART:NUNCHAR)
00044 00595000 SUBARGNAME(NUNCHAR+1:10-NUNCHAR)=" "
00044 00596000 C STORE INFO IN ARGSUBSTITUTION ARRAY
00044 00597000 WRITE(ARGSUBSTITUTION(LOC,M),7)SUBNAME,ARGNAME,SUBARGNAME,LISTPOS
00044 00598000 77 FORMAT(Ia10,i2)
00044 00599000 10 CONTINUE
00044 00600000 C FIND CALL STATEMENT THAT HAS THIS SUBARGNAME
00044 00601000 WRITE(CUNCALLNAME,66)CALL,SUBARGNAME
00044 00602000 66 FORMAT(A5,A10)
00044 00603000 DO 800 M=LC*31,1000
00044 00604000 READ(469M,111,END=99)STRING
00044 00605000 IF(INDEX(STRING,CUNCALLNAME(1:NUNCHAR+5)).GT.0)GO TO
00044 00606000 1801
00044 00607000 800 CONTINUE
00044 00608000 801 LOCALL=M
00044 00609000 96 RETURN
00044 00610000 99 DISPLAY*END OF SOURCE CODE FILE IN SUBROUTINE COMPLETE*ANG. ERROR*
00044 00611000 STOP
00044 00612000 END

```

## SYMBOL MAP

| NAME    | TYPE      | STRUCTURE  | ADDRESS  | NAME            | TYPE      | STRUCTURE | ADDRESS  |
|---------|-----------|------------|----------|-----------------|-----------|-----------|----------|
| ARGNAME | CHARACTER | SIMPLE VAR | 0-415 ,1 | ARGSUBSTITUTION | CHARACTER | ARRAY     | 0-416 ,1 |
| CALL    | CHARACTER | SIMPLE VAR | 0-411 ,1 | CALLSTRING      | CHARACTER | NULL      |          |

PAGE 0027 GETSUBANGNAME

| CUNCALLNAME  | CHARACTER | SIMPLE VAR | UNIO | 1 |
|--------------|-----------|------------|------|---|
| GETSUBARNAME | CHARACTER | SIMPLE VAR | UNIO | 1 |
| J            | INTEGER   | SIMPLE VAR | UNIO | 1 |
| LC           | INTEGER   | SIMPLE VAR | UNIO | 1 |
| LOC          | INTEGER   | SIMPLE VAR | UNIO | 1 |
| N            | INTEGER   | SIMPLE VAR | UNIO | 1 |
| MURCHAN      | INTEGER   | SIMPLE VAR | UNIO | 1 |
| STRING       | CHARACTER | SIMPLE VAR | UNIO | 1 |
| SUBARNAME    | CHARACTER | SIMPLE VAR | UNIO | 1 |
| FOUND        | INTEGER   | SIMPLE VAR | UNIO | 1 |
| ISTART       | INTEGER   | SIMPLE VAR | UNIO | 1 |
| K            | INTEGER   | SIMPLE VAR | UNIO | 1 |
| LISTSUS      | INTEGER   | SIMPLE VAR | UNIO | 1 |
| LOCALL       | INTEGER   | SIMPLE VAR | UNIO | 1 |
| N            | INTEGER   | SIMPLE VAR | UNIO | 1 |
| SEGUECIN     | INTEGER   | SIMPLE VAR | UNIO | 1 |
| SUB          | CHARACTER | SIMPLE VAR | UNIO | 1 |
| SUMAME       | CHARACTER | SIMPLE VAR | UNIO | 1 |

**LABEL MAP**

[illegible]

PROGRAM UNIT GETS A NAME COMPILED

```

00041 00611000 SCNTROL INIT,MAP, LABEL, LOCATION
00041 00614000 SUBROUTINE FINDSARNAME(SUBNAME, SUBNAMECOUNT, FUNCTIONNAME,
00041 00615000 IF FUNCTIONCOUNT, REDUCED, NLC)
00041 00616000 CHARACTER SUBNAME*16(100), FUNCTIONNAME*16(100), REDUCED*9*1500),
00041 00617000 ISTRING*72, NAMELIST*16(150)
00041 00618000 INTEGER FUNCTIONCOUNT, SUBNAMECOUNT, NUNTOTALUNITS
00041 00619000 C MERGE FUNCTION AND SUBROUTINE NAMES INTO ONE ARRAY
00041 00620000 DO 10 I=1, FUNCTIONCOUNT
00041 00621000 NAMELIST(I)=FUNCTIONNAME(I)
00046 00622000 10 CONTINUE
00062 00622000 DO 20 J=1, SUBNAMECOUNT
00063 00623000 NAMELIST(FUNCTIONCOUNT+J)=SUBNAME(J)
00070 00624000 20 CONTINUE
00105 00625000 NUNTOTALUNITS=FUNCTIONCOUNT+SUBNAMECOUNT
00110 00627000 C SORT COMPLETE ARRAY INTO ASCENDING ORDER BY NAME
00111 00628000 DO 100 I=1, NUNTOTALUNITS
00116 00629000 JJ=1
00120 00630000 101 IF(JJ.GE. NUNTOTALUNITS) GO TO 100
00124 00631000 IF(NAMELIST(JJ).GT. NAMELIST(JJ+1)) CALL SWTC
00124 00632000 I=NAMELIST(JJ), NAMELIST(JJ+1)
00156 00633000 JJ=JJ+1
00157 00634000 GO TO 101
00160 00635000 100 CONTINUE
00161 00636000 C CHECK SORTED ARRAY FOR ANY DUPLICATE PROGRAM UNIT NAMES
00161 00637000 NUNFOUND=0
00163 00638000 IF(NUNTOTALUNITS.EQ.0) RETURN
00167 00639000 DO 30 I=1, NUNTOTALUNITS-1
00175 00640000 IF(NAMELIST(I).NE. NAMELIST(I+1)) GO TO 30
00214 00641000 C DUPLICATE NAME FOUND! DISPLAY APPROPRIATE MESSAGE
00214 00642000 CALL DISPLAYDUPNAMES(REDUCED, NAMELIST(I), NLC)
00226 00643000 NUNFOUND=NUNFOUND+1
00227 00644000 30 CONTINUE
00230 00645000 C IF ANY DUPLICATE PROGRAM UNIT NAMES FOUND, STOP PROGRAM
00230 00646000 IF(NUNFOUND.EQ.0) RETURN
00234 00647000 DISPLAY**
00247 00648000 DISPLAY"PROGRAM IS STOPPED BECAUSE OF THE ABOVE DUPLICATED NAMES"
00324 00649000 DISPLAY"PROGRAM UNITS."
00353 00650000 STOP
00354 00651000 END

```

SYMBOL MAP

| NAME            | TYPE      | STRUCTURE  | ADDRESS  | NAME         | TYPE      | STRUCTURE  | ADDRESS |
|-----------------|-----------|------------|----------|--------------|-----------|------------|---------|
| DISPLAYDUPNAMES | INTEGER   | SUBROUTINE | 0-60 ,1  | FINDSARNAME  | CHARACTER | SUBROUTINE | 0-87 ,1 |
| FUNCTIONCOUNT   | INTEGER   | SIMPLE VAR | 0-60 ,1  | FUNCTIONNAME | INTEGER   | ARRAY      | 0-87 ,1 |
| I               | INTEGER   | SIMPLE VAR | 0-82     | J            | CHARACTER | SIMPLE VAR | 0-88 ,1 |
| JJ              | INTEGER   | SIMPLE VAR | 0-83     | NAMELIST     | INTEGER   | ARRAY      | 0-88 ,1 |
| NLC             | INTEGER   | SIMPLE VAR | 0-84 ,1  | NUNFOUND     | INTEGER   | SIMPLE VAR | 0-85 ,1 |
| NUNTOTALUNITS   | INTEGER   | SIMPLE VAR | 0-86     | REDUCED      | CHARACTER | ARRAY      | 0-85 ,1 |
| STRING          | CHARACTER | NULL       |          | SUBNAME      | CHARACTER | ARRAY      | 0-88 ,1 |
| SUBNAMECOUNT    | INTEGER   | SIMPLE VAR | 0-810 ,1 | SWTC         | CHARACTER | SUBROUTINE | 0-88 ,1 |

PAGE 0019 FINDSAMESAMES

LABEL MAP

| STATEMENT<br>LABEL | CODE<br>OFFSET | STATEMENT<br>LABEL | CODE<br>OFFSET | STATEMENT<br>LABEL | CODE<br>OFFSET | STATEMENT<br>LABEL | CODE<br>OFFSET | STATEMENT<br>LABEL | CODE<br>OFFSET | STATEMENT<br>LABEL | CODE<br>OFFSET |
|--------------------|----------------|--------------------|----------------|--------------------|----------------|--------------------|----------------|--------------------|----------------|--------------------|----------------|
| 10                 | 62             | 20                 | 105            | 30                 | 227            | 100                | 160            | 101                | 120            |                    |                |

PROGRAM UNIT FINDSAMESAMES COMPILED

PAGE 0030 HEBLETT-PACKARD J2102B.01.09 FORTKAM/1000 WED, JUN 15, 1983, 12:26 PM

```

00020 00652000 SCUNTHUL INIT,MAP,LABEL,LOCATIOM
00020 00653000 SUBROUTINE SWITC(RECORDJ,RECORDJ1)
00020 00654000 CHARACTER RECORDJ*16,RECORDJ1*16,TEM*16
00020 00655000 TEM=RECORDJ
00024 00656000 RECORDJ=RECORDJ1
00030 00657000 RECORDJ1=TEM
00034 00658000 RETURN
00035 00659000 END

```

SYMBOL MAP

| NAME    | TYPE      | STRUCTURE  | ADDRESS | NAME     | TYPE      | STRUCTURE  | ADDRESS |
|---------|-----------|------------|---------|----------|-----------|------------|---------|
| RECORDJ | CHARACTER | SIMPLE VAR | 0-85 ,1 | RECORDJ1 | CHARACTER | SIMPLE VAR | 0-84 ,1 |
| SWITC   |           | SUBROUTINE |         | TEM      | CHARACTER | SIMPLE VAR | 0-81 ,1 |

PROGRAM UNIT SWITC COMPILED

PAGE 0031 HEMLETT-PACKARD 32102B-01.09 FORTKAW/1000 MED, JUN 15, 1983, 12:26 PM

```

00031 00660000 SCUNTRUL INIT,MAP,LABEL,LOCUTION
00031 00661000 SUBROUTINE DISPLAYUPNAMES(REDUCED,NAME,MLC)
00031 00662000 CHARACTER NAME*16,REDUCED*2(1500),STRING*72
00031 00663000 INTEGER SEGUEGIN,SEGENO
00031 00664000 C GET BEGINNING AND ENDING LINE NUMBERS OF DUPLICATEDLY NAMED PROGRAM
00031 00665000 C UNITS
00031 00666000 DO 50 J=1,MLC
00031 00667000 READ(REDUCED(J),//STRING,SEGUEGIN,SEGENO
00036 00668000 77 FORMAT(A72.514)
00071 00669000 IF(INDEX(STRING,NAME).EQ.0)GO TO 50
00102 00670000 IF(INDEX(STRING," CALL ",JGT-0)GO TO 50
00124 00671000 DISPLAY NAME," (LINES ",SEGUEGIN,"-",SEGENO,") IS DUPLICATEDLY"
00124 00672000 1,"-NAMED."
00223 00673000 50 CONTINUE
00224 00674000 RETURN
00225 00675000 END

```

## SYMBOL MAP

| NAME           | TYPE      | STRUCTURE  | ADDRESS | NAME     | TYPE      | STRUCTURE  | ADDRESS |
|----------------|-----------|------------|---------|----------|-----------|------------|---------|
| DISPLAYUPNAMES | CHARACTER | SUBROUTINE | 0-15 ,1 | J        | INTEGER   | SIMPLE VAR | 0-13    |
| NAME           | CHARACTER | SIMPLE VAR | 0-15 ,1 | MLC      | INTEGER   | SIMPLE VAR | 0-14 ,1 |
| REDUCED        | CHARACTER | ARRAY      | 0-16 ,1 | SEGUEGIN | INTEGER   | SIMPLE VAR | 0-12    |
| SEGENO         | INTEGER   | SIMPLE VAR | 0-11    | STRING   | CHARACTER | SIMPLE VAR | 0-14 ,1 |

## LABEL MAP

| STATEMENT | CODE   | STATEMENT | CODE   | STATEMENT | CODE   | STATEMENT | CODE   |
|-----------|--------|-----------|--------|-----------|--------|-----------|--------|
| LABEL     | OFFSET | LABEL     | OFFSET | LABEL     | OFFSET | LABEL     | OFFSET |
| 50        | 223    | 77        | FAT    | 0         |        |           |        |

PROGRAM UNIT DISPLAYUPNAMES COMPILED



```

PAGE 0032  HEMLETT-PACKARD 321026.01.09  FORTRAN/1000  WED, JUN 15, 1963, 12:26 PM

00063 00676000 SCUNTHOL INIT,MAP, LABEL, LOCATION
00063 00677000 SUMMONTIME DETERMINIFCALLED(SUBNAME, SUBNAMECOUNT, FUNCTIONNAME,
00063 00678000 IF(FUNCTIONCOUNTER, REDUCED, NLC, ARGSUBSTITUTION, ASA, CALL, SUB, FUNCTION)
00063 00679000 CHARACTER SUBNAME*16(100), FUNCTIONNAME*16(100), REDUCED*92(500),
00063 00680000 IARGSUBSTITUTION*50(10,9), STRING*2, CALL*5, SUB*12, FUNCTION*10,
00063 00681000 2NAME*16, PROGRAM*20
00063 00682000 INTEGER FUNCTIONCOUNTER, SUBNAMECOUNT, ASA, FUNCTIONCALL(100),
00063 00683000 ISUBCALL(150)
00063 00684000 C DETERMINE WHAT PROGRAM UNITS ARE CALLED
00063 00685000 DO 10 I=1, NLC
00063 00686000 READ(REDUCED(1), /) STRING
00063 00687000 // FORMAT(A2,5I4)
00063 00688000 INDEXNUM=INDEX(STRING, CALL)
00063 00689000 IF((STRING(INDEXNUM-1:11).NE." "-AND.STRING(INDEXNUM-1:11).NE.""))
00063 00690000 1. OK. INDEXNUM.EQ.0) GO TO 10
00063 00691000 C CALL STATEMENT FOUND IN "REDUCED" ARRAY; DENOTE THAT THIS PROGRAM
00063 00692000 C UNIT IS CALLED
00063 00693000 NAME=STRING(INDEXNUM+5:16)
00063 00694000 DO 20 J=1, FUNCTIONCOUNTER
00063 00695000 IF(FUNCTIONNAME(J).NE.NAME) GO TO 20
00063 00696000 FUNCTIONCALL(J)=FUNCTIONCALL(J)+1
00063 00697000 GO TO 10
00063 00698000 20 CONTINUE
00063 00699000 DO 30 K=1, SUBNAMECOUNT
00063 00700000 IF(SUBNAME(K).NE.NAME) GO TO 30
00063 00701000 SUBCALL(K)=SUBCALL(K)+1
00063 00702000 GO TO 10
00063 00703000 30 CONTINUE
00063 00704000 DISPLAY NAME, " CALLED BUT PROGRAM UNIT NOT FOUND. SHOULD"
00063 00705000 DISPLAY HAVE BEEN HANDLED AS A LIBRARY CALL. THUS PROGRAM IS"
00063 00706000 DISPLAY "STOPPED BECAUSE OF THIS ERROR."
00063 00707000 STOP
00063 00708000 10 CONTINUE
00063 00709000 C DISPLAY ANY FUNCTIONS THAT ARE NEVER CALLED
00063 00710000 PROGRAMUNIT1:10)=FUNCTION
00063 00711000 PROGRAMUNIT2:12)= "
00063 00712000 CALL DISPLAYNCALLS(FUNCTIONNAME, FUNCTIONCOUNTER, FUNCTIONCALL,
00063 00713000 111, MOCALL, REDUCED, PROGRAMIT, NLC, ARGSUBSTITUTION, ASA)
00063 00714000 C DISPLAY ANY SUBROUTINES THAT ARE NEVER CALLED
00063 00715000 PROGRAMUNIT1:12)=SUB
00063 00716000 CALL DISPLAYNCALLS(SUBNAME, SUBNAMECOUNT, SUBCALL,
00063 00717000 113, MOCALL, REDUCED, PROGRAMIT, NLC, ARGSUBSTITUTION, ASA)
00063 00718000 C IF ANY PROGRAM UNITS ARE NEVER CALLED, STOP PROGRAM
00063 00719000 IF(MOCALL.EQ.0) RETURN
00063 00720000 DISPLAY ""
00063 00721000 DISPLAY "PROGRAM IS STOPPED BECAUSE IF "MOCALL," PROGRAM UNIT(S)"
00063 00722000 DISPLAY "PRESENT IN SOURCE CODE BUT NEVER CALLED BY ANY OTHER"
00063 00723000 DISPLAY "PROGRAM UNIT(S)"
00063 00724000 STOP
00063 00725000 END

```

## PAGE 0033 DETERMINIFCALLED

## SYMBOL MAP

| NAME            | TYPE      | STRUCTURE  | ADDRESS  | NAME             | TYPE       | STRUCTURE  | ADDRESS  |
|-----------------|-----------|------------|----------|------------------|------------|------------|----------|
| ANGSUBSTITUTION | CHARACTER | ARRAY      | 0-810 ,1 | ASA              | INTEGER    | SIMPLE VAR | 0-81 ,1  |
| CALL            | CHARACTER | SIMPLE VAR | 0-86 ,1  | DETERMINIFCALLED | SUBROUTINE | SUBROUTINE | 0-84 ,1  |
| DISPLAYNCALLS   | INTEGER   | SUBROUTINE | 0-81 ,1  | FUNCTION         | SIMPLE VAR | SIMPLE VAR | 0-813 ,1 |
| FUNCTIONCALL    | CHARACTER | ARRAY      | 0-814 ,1 | FUNCTIONCOUNTER  | INTEGER    | SIMPLE VAR | 0-813 ,1 |
| FUNCTIONNAME    | CHARACTER | ARRAY      | 0-817 ,1 | J                | INTEGER    | SIMPLE VAR | 0-884    |
| INDEXNUM        | INTEGER   | SIMPLE VAR | 0-817    | NAME             | CHARACTER  | SIMPLE VAR | 0-812 ,1 |
| K               | INTEGER   | SIMPLE VAR | 0-886    | NCALL            | INTEGER    | SIMPLE VAR | 0-885    |
| NLC             | INTEGER   | SIMPLE VAR | 0-811 ,1 | REDUCED          | CHARACTER  | ARRAY      | 0-812 ,1 |
| PROCUNIT        | CHARACTER | SIMPLE VAR | 0-810 ,1 | SUB              | CHARACTER  | SIMPLE VAR | 0-85 ,1  |
| STRING          | CHARACTER | SIMPLE VAR | 0-811 ,1 | SUBNAME          | CHARACTER  | ARRAY      | 0-816 ,1 |
| SUBCALL         | INTEGER   | ARRAY      | 0-817 ,1 |                  |            |            |          |
| SUBNAMECOUNT    | INTEGER   | SIMPLE VAR | 0-815 ,1 |                  |            |            |          |

## LABEL MAP

| STATEMENT LABEL | CODE OFFSET | STATEMENT LABEL | CODE OFFSET | STATEMENT LABEL | CODE OFFSET | STATEMENT LABEL | CODE OFFSET |
|-----------------|-------------|-----------------|-------------|-----------------|-------------|-----------------|-------------|
| 10              | 445         | 20              | 231         | 30              | 257         | // PMT          | 0           |

## PROGRAM UNIT DETERMINIFCALLED COMPILED

PAGE 0034 NEWLETT-PACKARD J21020.01.09 FORTNAN/1000 WED, JUN 15, 1983, 12:26 PM

```

00053 00726000 SCUNTHUL INIT,MAP,LABEL,LOCUTION
00053 00727000 SUBROUTINE DISPLAYMOCALLS(UNITNAME,UNITCOUNTER,UNITCALL,IBEG,
00053 00728000 IMOCALL,REDUCED,PRGUNIT,MLC,ARGSUBSTITUTION,ASA)
00053 00729000 CHARACTER UNITNAME*16(100),REDUCED*2(500),PRGUNIT*20,STRING*12,
00053 00730000 IARGSUBSTITUTION*50(10,0),ARGNAME*10
00053 00731000 INTEGER UNITCOUNTER,UNITCALL(150),SEGBEGIN,SEGEN,ASA
00053 00732000 C DISPLAY ANY PROGRAM UNITS THAT HAVE NOT BEEN CALLED
00053 00733000 DO 40 I=1,UNITCOUNTER
00053 00734000 IF(UNITCALL(I).GT.0)GO TO 40
00053 00735000 C DETERMINE IF PROGRAM UNIT REFERENCED VIA "EXTERNAL" CALL
00053 00736000 IF(ASA.EQ.0)GO TO 101
00053 00737000 DO 10 M=1,ASA
00053 00738000 DO 20 N=1,N
00053 00739000 HEAD(ARGSUBSTITUTION(M,N),66)ARGNAME
00053 00740000 66 FORMAT(1X,A16) "IGD TO 20
00053 00741000 IF(ARGSUBSTITUTION(M,N).EQ."
00053 00742000 IF(ARGSUBSTITUTION(M,N).EQ.UNITNAME(I))GO TO 40
00053 00743000 20 CONTINUE
00053 00744000 10 CONTINUE
00053 00745000 C GET BEGINNING AND ENDING LINE OF PROGRAM UNIT NOT CALLED
00053 00746000 101 PRGUNIT(1)MOCALL=UNITNAME(I)
00053 00747000 DO 50 J=1,MLC
00053 00748000 HEAD(REDUCED(J),1)STRING,SEGBEGIN,SEGEN
00053 00749000 77 FORMAT(A72,S14)
00053 00750000 IF(LINDEX(STRING,PRGUNIT).EQ.0)GO TO 50
00053 00751000 DISPLAY PRGUNIT," (LINES ",SEGBEGIN,"-",SEGEN,") IS IN SOURCE",
00053 00752000 1"FILE BUT IS NEVER CALLED."
00053 00753000 MOCALL=MOCALL+1
00053 00754000 GO TO 40
00053 00755000 50 CONTINUE
00053 00756000 40 CONTINUE
00053 00757000 RETURN
00053 00758000 END

```

## SYMBOL MAP

| NAME     | TYPE      | STRUCTURE  | ADDRESS  | NAME            | TYPE      | STRUCTURE  | ADDRESS  |
|----------|-----------|------------|----------|-----------------|-----------|------------|----------|
| ARGNAME  | CHARACTER | SIMPLE VAR | 0-87 ,1  | ARGSUBSTITUTION | CHARACTER | ARRAY      | 0-85 ,1  |
| ASA      | INTEGER   | SIMPLE VAR | 0-84 ,1  | DISPLAYMOCALLS  | INTEGER   | SUBROUTINE | 0-812 ,1 |
| I        | INTEGER   | SIMPLE VAR | 0-81     | IBEG            | INTEGER   | SIMPLE VAR | 0-812 ,1 |
| J        | INTEGER   | SIMPLE VAR | 0-85     | M               | INTEGER   | SIMPLE VAR | 0-86     |
| N        | INTEGER   | SIMPLE VAR | 0-82     | MLC             | INTEGER   | SIMPLE VAR | 0-86 ,1  |
| VOCALL   | INTEGER   | SIMPLE VAR | 0-811 ,1 | PRGUNIT         | CHARACTER | SIMPLE VAR | 0-87 ,1  |
| REDUCED  | CHARACTER | ARRAY      | 0-810 ,1 | SEGBEGIN        | INTEGER   | SIMPLE VAR | 0-84     |
| SEGEN    | INTEGER   | SIMPLE VAR | 0-83     | STRING          | CHARACTER | SIMPLE VAR | 0-810 ,1 |
| UNITCALL | INTEGER   | ARRAY      | 0-813 ,1 | UNITCOUNTER     | INTEGER   | SIMPLE VAR | 0-814 ,1 |
| UNITNAME | CHARACTER | ARRAY      | 0-815 ,1 |                 |           |            |          |

PAGE 0035 DISPLAYCALLS

LABEL MAP

| STATEMENT<br>LABEL | CODE<br>OFFSET | STATEMENT<br>LABEL | CODE<br>OFFSET | STATEMENT<br>LABEL | CODE<br>OFFSET | STATEMENT<br>LABEL | CODE<br>OFFSET | STATEMENT<br>LABEL | CODE<br>OFFSET |
|--------------------|----------------|--------------------|----------------|--------------------|----------------|--------------------|----------------|--------------------|----------------|
| 10                 | 204            | 20                 | 203            | 40                 | 407            | 50                 | 406            | 66 FMT             | 0              |
| 101                | 205            |                    |                |                    |                |                    |                | 77 FMT             | 5              |

PROGRAM UNIT DISPLAYCALLS COMPILED

```

PAGE 0036  HEMLETT-PACKARD 32102B.01.09  FURTKRM/3000  WED, JUN 15, 1983, 12:26 PM

00063 00759000  $CONTROL INIT, MAP, LABEL, LOCATION
00063 00760000  SUBROUTINE REARRANGESOURCE(SUB, CALL, SOURCEFILE, NUMINPUNITIVES,
00063 00761000  INUMPRGUNIT, NLC, REDUCED, FUNCTION, ARGSUBSTITUTION, TOTCOMMENTS,
00063 00762000  2TOTFUNNATS, TOTDECL, PAGENO)
00063 00763000  C THIS SUBROUTINE PRINTS OUT A SUMMARY OF THE FURTKRM SOURCE CODE'S
00063 00764000  C EXECUTION SEQUENCE IN TOP-DOWN HIERARCHICAL ORDER AND REARRANGES
00063 00765000  C THE PERMANENT SOURCE CODE FILE INTO THIS ORDER
00063 00766000  CHARACTER SUB*12, CALL*5, STRING*72, PRGUNIT*28(12), SOURCEFILE*96,
00063 00767000  IREDUCED*92(500), NAME*16, FUNCTION*10, ARGSUBSTITUTION*50(10,8),
00063 00768000  2UNITNAME*16(10)
00063 00769000  INTEGER SEGBEGIN, SEGEN, ADDRESSSTACK(99), STKPTR, USED(450),
00063 00770000  IPAGENO, NLC, LEVELLOC(10)
00063 00771000  PRGUNIT(1)= 'MAIN'
00063 00772000  C WRITE PAGE HEADINGS
00063 00773000  CALL PAGEHEAD(LINECOUNTER, PAGENO, SOURCEFILE)
00063 00774000  C FIND PROGRAM UNIT NAME IN 'REDUCED' ARRAY
00063 00775000  2 CALL FINDPRGUNIT(1, PRGUNIT, REDUCED, STRING, SEGBEGIN, SEGEN,
00063 00776000  1, NUMCOMMENTS, NUMFUNNATS, NUMDECL, NLC, LC, SUB, FUNCTION, UNITNAME,
00063 00777000  2STKPTR, $11)
00063 00778000  C INDENT PROGRAM UNIT NAME BASED ON ITS LEVEL (LEVEL OF CALL=STKPTR+1)
00063 00779000  11 CALL INDENTSUBNAME(STRING, STKPTR, LASTLEVEL, LINECOUNTER)
00063 00780000  C PRINT OUT LINE FOR PROGRAM UNIT STATISTICS
00063 00781000  CALL PRINTLINE(USED, LINECOUNTER, PAGENO, SOURCEFILE, STRING, LC, STKPTR
00063 00782000  1, TOTLINES, SEGEN, SEGBEGIN, TOTCOMMENTS, TOTFUNNATS, TOTDECL,
00063 00783000  2NUMCOMMENTS, NUMFUNNATS, NUMDECL, NEWDECL, REDUCED, $1)
00063 00784000  C WRITE PROGRAM UNIT TO NEW FILE
00063 00785000  CALL WRITETOWNEWFILE(SEGBEGIN, SEGEN, NLC)
00063 00786000  13 IF INDEX(STRING, "RECURSIVE CALL", 1, 0) GO TO 12
00063 00787000  LC=LC+1
00063 00788000  C DETERMINE NEXT CALL, IF ANY, IN PROGRAM UNIT
00063 00789000  3 IF (LC-CT.NLC) GO TO 12
00063 00790000  HEAD(REDUCED(LC), 1) STRING, LL, L
00063 00791000  1 FORMAT(A72, 514)
00063 00792000  IF (INDEX(STRING, SUB), 0, 0) INDEX(STRING, " MAIN ") -GT, 0, 0, INDEX
00063 00793000  1(STRING, FUNCTION), 0, 0, 0 GO TO 12
00063 00794000  C DETERMINE PROGRAM UNIT CALLED
00063 00795000  CALL DETERPRGUNIT(STRING, CALL, NAME, ARGSUBSTITUTION, L, M, STKPTR,
00063 00796000  1LEVELLOC, REDUCED, PRGUNIT, NLC, SUB, FUNCTION, LC, UNITNAME(STKPTR+1))
00063 00797000  C PUSH RETURN ADDRESS OF 'REDUCED' ARRAY ON TO STACK
00063 00798000  STKPTR=STKPTR+1
00063 00799000  ADDRESSSTACK(STKPTR)=LCV1
00063 00800000  PRGUNIT(1)(1:12)=SUB
00063 00801000  PRGUNIT(1)(13:16)=NAME
00063 00802000  PRGUNIT(2)(1:10)=FUNCTION
00063 00803000  PRGUNIT(2)(11:16)=NAME
00063 00804000  GO TO 2
00063 00805000  12 IF (STKPTR-10, 0) GO TO 99
00063 00806000  C POP NEW CURRENT ADDRESS OF 'REDUCED' ARRAY OFF STACK
00063 00807000  LC=ADDRESSSTACK(STKPTR)
00063 00808000  STKPTR=STKPTR-1
00063 00809000  GO TO 1
00063 00810000  C PRINT LINE TOTALS
00063 00811000  99 CALL PRINTTOTALS(TOTLINES, TOTCOMMENTS, TOTFUNNATS, TOTDECL,
00063 00812000  1, INUMPRGUNIT)
00063 00813000  C DELETE OLD SOURCE FILE AND SAVE NEW SOURCE FILE OF SAME NAME
00063 00814000  C IF NUMBER OF LINES FOR BOTH FILES IS EQUAL
00063 00815000  CALL SAVENWFILE(INDIRINPUNITIVES, TOTLINES)

```

PAGE 0037 NEARRANGESOURCE

00501 00010000 RETURN  
 00502 00017000 END

## SYMBOL MAP

| NAME          | TYPE      | STRUCTURE  | ADDRESS  | NAME            | TYPE       | STRUCTURE  | ADDRESS  |
|---------------|-----------|------------|----------|-----------------|------------|------------|----------|
| ADDRESSSTACK  | INTEGER   | ARRAY      | 0-011 ,1 | ANGSUBSTITUTION | CHARACTER  | ARRAY      | 0-010 ,1 |
| CALL          | CHARACTER | SIMPLE VAR | 0-017 ,1 | DEFERPRUGOCALL  | SUBROUTINE | SIMPLE VAR | 0-011 ,1 |
| FINDPRUGRUNIT | CHARACTER | SUBROUTINE |          | FUNCTION        | CHARACTER  | SIMPLE VAR | 0-017 ,1 |
| INDENTSUBNAME | INTEGER   | SUBROUTINE |          | L               | INTEGER    | SIMPLE VAR | 0-015    |
| LASTLEVEL     | INTEGER   | SIMPLE VAR | 0-023    | LC              | INTEGER    | SIMPLE VAR | 0-015    |
| LEVELLOC      | INTEGER   | ARRAY      | 0-022    | LINCOUNTER      | INTEGER    | SIMPLE VAR | 0-017    |
| LL            | INTEGER   | SIMPLE VAR | 0-022    | M               | INTEGER    | SIMPLE VAR | 0-017    |
| NAME          | CHARACTER | SIMPLE VAR | 0-015 ,1 | MEMEND          | INTEGER    | SIMPLE VAR | 0-014    |
| MLC           | INTEGER   | SIMPLE VAR | 0-012    | NUMCOMMENTS     | INTEGER    | SIMPLE VAR | 0-020    |
| NUMDECL       | INTEGER   | SIMPLE VAR | 0-021    | NUMFORMATS      | INTEGER    | SIMPLE VAR | 0-016    |
| NUMINPUTLINES | INTEGER   | SIMPLE VAR | 0-015 ,1 | NUMPRUGUNITS    | INTEGER    | SIMPLE VAR | 0-014 ,1 |
| PAGEHEAD      | INTEGER   | SUBROUTINE |          | PAGEND          | INTEGER    | SIMPLE VAR | 0-014 ,1 |
| PRINTLINE     | CHARACTER | SUBROUTINE |          | PRINTTOTALS     | SUBROUTINE | SUBROUTINE |          |
| PRUGUNIT      | CHARACTER | ARRAY      | 0-014 ,1 | REARRANGESOURCE | SUBROUTINE | SUBROUTINE |          |
| REDUCED       | CHARACTER | ARRAY      | 0-012 ,1 | RLC             | INTEGER    | SIMPLE VAR | 0-013 ,1 |
| SAVEREMFILE   | INTEGER   | SUBROUTINE |          | SEGUEGIN        | INTEGER    | SIMPLE VAR | 0-013    |
| SEGEN         | INTEGER   | SIMPLE VAR | 0-011    | SOURCEFILE      | CHARACTER  | SIMPLE VAR | 0-016 ,1 |
| STKPTR        | INTEGER   | SIMPLE VAR | 0-010    | SINING          | CHARACTER  | SIMPLE VAR | 0-026 ,1 |
| SUB           | CHARACTER | SIMPLE VAR | 0-020 ,1 | TOTCOMMENTS     | REAL       | SIMPLE VAR | 0-017 ,1 |
| TOTDECL       | REAL      | SIMPLE VAR | 0-015 ,1 | TOTFORMATS      | REAL       | SIMPLE VAR | 0-017 ,1 |
| TOTLINES      | REAL      | SIMPLE VAR | 0-024    | UNITNAME        | CHARACTER  | ARRAY      | 0-015 ,1 |
| USED          | INTEGER   | ARRAY      | 0-013 ,1 | WHITETUREFILE   | SUBROUTINE | SUBROUTINE |          |

## LABEL MAP

| STATEMENT LABEL | CODE OFFSET | STATEMENT LABEL | CODE OFFSET | STATEMENT LABEL | CODE OFFSET | STATEMENT LABEL | CODE OFFSET |
|-----------------|-------------|-----------------|-------------|-----------------|-------------|-----------------|-------------|
| 1 FMT           | 0           | 2               | 117         | 3               | 243         | 11              | 150         |
| 99              | 470         |                 |             |                 |             | 12              | 457         |
|                 |             |                 |             |                 |             | 13              | 212         |

PROGRAM UNIT NEARRANGESOURCE COMPILED

PAGE 0039 HEWLETT-PACKARD 22102B-01-09 FORTRAN/1000 WED, JUN 15, 1961, 12:26 PM

```

00150 00010000 SCUNFOL INIT,MAP,LABEL,LOCUTION
00150 00010000 SUMROUTINE PAGEHEAD(LINECOUNTER,PAGENO,SOURCEFILE%)
00150 00020000 CHARACTER DATE(27),SOURCEFILE%
00150 00021000 INTEGER PAGENO
00150 00022000 CALL DATELINE(DATE)
00152 00023000 PAGENO=PAGENO+1
00153 00024000 WRITE(6,1)DATE(6:12),PAGENO
00200 00025000 1 FORMAT(//,101,1001,A12," PAGE ",12)
00200 00026000 1 SOURCE CODE?*,SOURCEFILE
00200 00027000 1 DISPLAY-----
00270 00028000 1-----
00300 00030000 1 DISPLAY**
00373 00031000 1 WRITE(6,76)
00373 00032000 76 FORMAT(1X,"PROGRAM UNIT NAME
00411 00033000 1," LEVEL # OF LINES # OF COMM. # OF DECLAR. # OF FORMATS
00411 00034000 2," # OF NEW LINES")
00411 00035000 1 DISPLAY-----
00411 00036000 1,-----
00411 00037000 2,-----
00556 00038000 LINECOUNTER=0
00560 00039000 RETURN
00561 00040000 END

```

# SYMBOL MAP

| NAME        | TYPE      | STRUCTURE  | ADDRESS | NAME       | TYPE       | STRUCTURE | ADDRESS |
|-------------|-----------|------------|---------|------------|------------|-----------|---------|
| DATE        | CHARACTER | SIMPLE VAR | 0-81 .1 | DATELINE   | SUBROUTINE |           |         |
| LINECOUNTER | INTEGER   | SIMPLE VAR | 0-86 .1 | PAGEHEAD   | SUBROUTINE |           |         |
| PAGENO      | INTEGER   | SIMPLE VAR | 0-85 .1 | SOURCEFILE | SIMPLE VAR | 0-99 .1   |         |

# LABEL MAP

| STATEMENT LABEL | CODE OFFSET | STATEMENT LABEL | CODE OFFSET | STATEMENT LABEL | CODE OFFSET |
|-----------------|-------------|-----------------|-------------|-----------------|-------------|
| 1 FMT           | 0           | 76 FMT          | 20          |                 |             |

PROGRAM UNIT PAGEHEAD COMPILED

```

00031 00841000 SCOUTROL UNIT,MAP,LAHEL,LOCUTION
00031 00842000 SUBROUTINE FINDPROGRAMUNIT(FUNCTION,REDUCED,STRING,SECUREGIN,SEGEND
00031 00843000 1,MUMCOMMENTS,MUMFURNATS,MUMDECL,MLC,LC,SUB,FUNCTION,UNITNAME,
00031 00844000 2,STKPTR,*)
00031 00845000 CHARACTER STRING*72,REDUCED*92(500),PROGRAMUNIT*2(2),SUB*12,
00031 00846000 IFUNCTION*10,UNITNAME*16(30)
00031 00847000 INTEGER SECUREGIN,SEGEND,MLC,STKPTR,MFCUR
00031 00848000 C CHECK FOR RECURSION
00031 00849000 RECUR=0
00031 00850000 IF(INDEX(STRING,"RECURSIVE CALL").GT.0)RECUR=1
00031 00851000 C FIND PROGRAM UNIT NAME IN "REDUCED" ARRAY
00031 00852000 DO 10 LC=1,MLC
00031 00853000 HEAD(REDUCED(LC),1)STRING,SECUREGIN,SEGEND,MUMCOMMENTS,MUMFURNATS,
00031 00854000 1,MUMDECL
00031 00855000 1 FORMAT(A/2,514)
00031 00856000 IF(INDEX(STRING,PROGRAMUNIT(1)).GT.0)GO TO 11
00031 00857000 IF(INDEX(STRING,PROGRAMUNIT(2)).GT.0)GO TO 11
00031 00858000 10 CONTINUE
00031 00859000 RETURN 1
00031 00860000 C ISOLATE PROGRAM UNIT NAME
00031 00861000 11 IF(INDEX(STRING,SUB).GT.0)UNITNAME(STKPTR)=STRING(1:16)
00031 00862000 IF(INDEX(STRING,FUNCTION).GT.0)UNITNAME(STKPTR)=STRING(16:16)
00031 00863000 IF(RECUR.EQ.1)STRING(17:114)=-RECURSIVE CALL-
00031 00864000 RETURN
00031 00865000 END

```

# SYMBOL MAP

| NAME            | TYPE      | STRUCTURE  | ADDRESS  | NAME        | TYPE      | STRUCTURE  | ADDRESS  |
|-----------------|-----------|------------|----------|-------------|-----------|------------|----------|
| FINDPROGRAMUNIT | INTEGER   | SUBROUTINE | 0-810 ,1 | FUNCTION    | CHARACTER | SIMPLE VAR | 0-86 ,1  |
| LC              | INTEGER   | SIMPLE VAR | 0-812 ,1 | MUMCOMMENTS | INTEGER   | SIMPLE VAR | 0-814 ,1 |
| MUMDECL         | CHARACTER | SIMPLE VAR | 0-821 ,1 | MUMFURNATS  | INTEGER   | SIMPLE VAR | 0-813 ,1 |
| PROGRAMUNIT     | CHARACTER | ARRAY      | 0-820 ,1 | RECUR       | INTEGER   | SIMPLE VAR | 0-811 ,1 |
| REDUCED         | CHARACTER | ARRAY      | 0-816 ,1 | MLC         | INTEGER   | SIMPLE VAR | 0-815 ,1 |
| SECUREGIN       | INTEGER   | SIMPLE VAR | 0-814 ,1 | SEGEND      | INTEGER   | SIMPLE VAR | 0-817 ,1 |
| STKPTR          | INTEGER   | SIMPLE VAR | 0-814 ,1 | STRING      | CHARACTER | SIMPLE VAR | 0-817 ,1 |
| SUB             | CHARACTER | SIMPLE VAR | 0-817 ,1 | UNITNAME    | CHARACTER | ARRAY      | 0-85 ,1  |

# LABEL MAP

| STATEMENT LABEL | CODE OFFSET | STATEMENT LABEL | CODE OFFSET | STATEMENT LABEL | CODE OFFSET |
|-----------------|-------------|-----------------|-------------|-----------------|-------------|
| 1 FAT           | 0           | 10              | 165         | 11              | 177         |

PROGRAM UNIT FINDPROGRAMUNIT COMPILED



PAGE 0040    HEWLETT-PACKARD 32102B-01-09    FURTHAN/1000    WED, JUN 15, 1983, 12:26 PM

```

00020 00866000 $CONTROL 1611-MAP, LABEL, LOCATION
00020 00867000 SUBROUTINE INDENTSUBNAME$STRING, STKPTR, LASTLEVEL, LINECOUNTER)
00020 00868000 CHARACTER STRING*12, ASTRING*12
00020 00869000 INTEGER STKPTR
00020 00870000 C IF LEVEL OF CALL IS 1 (MAIN PROGRAM UNIT), NO INDENTATION IS DONE
00020 00871000 IF (STKPTR.EQ.0) GO TO 99
00020 00872000 C INDENT SUBROUTINE NAME # OF SPACES EQUAL TO ITS LEVEL OF CALL
00024 00873000 ASTRING(1:67-STKPTR)=STRING(1:67-STKPTR)
00024 00874000 DO 10 I=1, STKPTR
00044 00875000 STRING(5+I:11)= ' '
00060 00876000 10 CONTINUE
00061 00877000 STRING(16+STKPTR:67-STKPTR)=ASTRING(1:67-STKPTR)
00075 00878000 IF (LASTLEVEL.LT.STKPTR) GO TO 98
00101 00879000 DISPLAY=
00147 00880000 LINECOUNTER=LINECOUNTER+1
00150 00881000 98 LASTLEVEL=STKPTR
00152 00882000 RETURN
00153 00883000 END

```

## SYMBOL MAP

| NAME          | TYPE       | STRUCTURE  | ADDRESS | NAME      | TYPE    | STRUCTURE  | ADDRESS |
|---------------|------------|------------|---------|-----------|---------|------------|---------|
| ASTRING       | CHARACTER  | SIMPLE VAR | 0-12 ,1 | I         | INTEGER | SIMPLE VAR | 0-11    |
| INDENTSUBNAME | SUBROUTINE |            |         | LASTLEVEL | INTEGER | SIMPLE VAR | 0-85 ,1 |
| LINECOUNTER   | INTEGER    | SIMPLE VAR | 0-84 ,1 | STKPTR    | INTEGER | SIMPLE VAR | 0-86 ,1 |
| STRING        | CHARACTER  | SIMPLE VAR | 0-11 ,1 |           |         |            |         |

## LABEL MAP

| STATEMENT LABEL | CODE OFFSET | STATEMENT LABEL | CODE OFFSET | STATEMENT LABEL | CODE OFFSET | STATEMENT LABEL | CODE OFFSET |
|-----------------|-------------|-----------------|-------------|-----------------|-------------|-----------------|-------------|
| 10              | 60          | 98              | 150         |                 |             |                 |             |

PROGRAM UNIT INDENTSUBNAME COMPILED



PAGE 0042 PRINTLINE

FORMATS USED REAL INTEGER SIMPLE VAN U-812 ,I REAL SIMPLE VAN U-816 ,I  
ARRAY U-825 ,I

LABEL MAP

| STATEMENT LABEL | CODE | STATEMENT LABEL | CODE | STATEMENT LABEL | CODE | STATEMENT LABEL | CODE | STATEMENT LABEL | CODE |
|-----------------|------|-----------------|------|-----------------|------|-----------------|------|-----------------|------|
| 1 FMT           | 20   | 14              | 227  | 66 FMT          | 0    | // FMT          | 4    |                 |      |

PROGRAM UNIT PRINTLINE COMPILED

PAGE 0043    HEWLETT-PACKARD 32102B.01.09    FORTKAR/1000    WED, JUN 15, 1993, 12:26 PM

```

00010 00922000    SCURTOL INIT MAP, LABEL, LOCATION
00010 00923000    SUBROUTINE ACCUMLINE TOTALS (TUTLINES, TUTCUMMENTS, TUTFURNMATS,
00010 00924000    INDECL, NUMCUMMENTS, NUMFURNMATS, NUMDECL, TUTDECL)
00010 00925000    TUTLINES=TUTLINES+NUMCUMMENTS
00014 00926000    TUTCUMMENTS=TUTCUMMENTS+NUMCUMMENTS
00020 00927000    TUTFURNMATS=TUTFURNMATS+NUMFURNMATS
00024 00928000    TUTDECL=TUTDECL+NUMDECL
00030 00929000    RETURN
00031 00930000    END

```

## SYMBOL MAP

| NAME             | TYPE    | STRUCTURE  | ADDRESS  | NAME        | TYPE    | STRUCTURE  | ADDRESS  |
|------------------|---------|------------|----------|-------------|---------|------------|----------|
| ACCUMLINE TOTALS | INTEGER | SUBROUTINE | 0-95 ,1  | NUMCUMMENTS | INTEGER | SIMPLE VAR | 0-97 ,1  |
| NUMDECL          | INTEGER | SIMPLE VAR | 0-96 ,1  | NUMFURNMATS | INTEGER | SIMPLE VAR | 0-98 ,1  |
| NUMCUMMENTS      | INTEGER | SIMPLE VAR | 0-99 ,1  | TUTCUMMENTS | REAL    | SIMPLE VAR | 0-912 ,1 |
| TUTDECL          | REAL    | SIMPLE VAR | 0-94 ,1  | TUTFURNMATS | REAL    | SIMPLE VAR | 0-911 ,1 |
| TUTLINES         | REAL    | SIMPLE VAR | 0-913 ,1 |             |         |            |          |

PROGRAM UNIT ACCUMLINE TOTALS COMPILED

PAGE 0044    NEWLETT-PACKARD 32102B.01.09    FORTKAM/1000    WED, JUN 15, 1983, 12:26 PM

```

00040 00931000    SCONTROL INIT,MAP,LABEL,LOCUTION
00040 00932000       SUBROUTINE WRITETONEFILE(SEGREGIN,SEGEND,MLC)
00040 00933000       INTEGER SEGREGIN,SEGEND
00040 00934000       CHARACTER STRING*72,LINENUM*5
00040 00935000    C    WRITE PROGRAM UNIT TO NEW FILE
00040 00936000       DO 30 I=SEGREGIN,SEGEND
00045 00937000          READ(4691,1)STRING
00071 00938000          1 FORMAT(A72.514)
00071 00939000          MLC=MLC+1
00072 00940000          LINENUM=STR(MLC,5)
00105 00941000          DO 50 N=1,4
00112 00942000          IF(LINENUM(N).NE." ")GO TO 51
00133 00943000          LINENUM(N)=0
00146 00944000          50 CONTINUE
00147 00945000          51 WRITE(41,4)STRING,LINENUM
00174 00946000          4 FORMAT(A72.A5,"000")
00174 00947000          30 CONTINUE
00175 00948000          RETURN
00176 00949000          END

```

SYMBOL MAP

| NAME     | TYPE      | STRUCTURE  | ADDRESS | NAME          | TYPE       | STRUCTURE  | ADDRESS |
|----------|-----------|------------|---------|---------------|------------|------------|---------|
| I        | INTEGER   | SIMPLE VAR | 0+81    | K             | INTEGER    | SIMPLE VAR | 0+82    |
| LINENUM  | CHARACTER | SIMPLE VAR | 0+83 ,1 | MLC           | INTEGER    | SIMPLE VAR | 0+84 ,1 |
| SEGREGIN | INTEGER   | SIMPLE VAR | 0+86 ,1 | SEGEND        | INTEGER    | SIMPLE VAR | 0+85 ,1 |
| STRING   | CHARACTER | SIMPLE VAR | 0+84 ,1 | WRITETONEFILE | SUBROUTINE |            |         |

LABEL MAP

| STATEMENT LABEL | CODE OFFSET | STATEMENT LABEL | CODE OFFSET | STATEMENT LABEL | CODE OFFSET | STATEMENT LABEL | CODE OFFSET |
|-----------------|-------------|-----------------|-------------|-----------------|-------------|-----------------|-------------|
| 1 FMT           | 0           | 4 FMT           | 5           | 30              | 174         | 51              | 147         |

PROGRAM UNIT WRITETONEFILE COMPILED

PAGE 0045 NEMLETT-PACKAND 321020.01.09 FURTHAN/1000 WED, JUN 15, 1964, 12120 PM

```

00052 00950000 SCNTROL INIT MAP, LABEL, LOCATION
00052 00951000 SUBROUTINE DETEPRUCGALL(STRING, CALL, NAME, ARGSUBSTITUTION,
00052 00952000 ILN, STAPTR, LEVELLOC, REDUCED, PROGNIT, HLC, SUB, FUNCTION, LC, UNITNAME)
00052 00953000 CHARACTER SUBN12, CALLS, STRING12, ARGSUBSTITUTION*50(10,8),
00052 00954000 IACTUALNAME*10, FORMALNAME*16, NAME*16, REDUCED*92(500), PROGNIT*28(2)
00052 00955000 2, FUNCTION*10, UNITNAME*16, DUNITNAME*16(1)
00052 00956000 INTEGER STAPTR, LEVELLOC(30), HLC
00052 00957000 J=STAPTR+1
00055 00958000 C STONE "REDUCED" ARRAY'S ADDRESS OF CURRENT CALL STATEMENT
00055 00959000 LEVELLOC(J)=LC
00055 00960000 LOCALLEX=LEVELLOC(J)
00060 00961000 C CHECK FOR "ARGSUBSTITUTION" POINTER
00063 00962000 IF(LC.GT.0)M=L
00070 00963000 C DETERMINE PROGNIT CALLED
00070 00964000 INDEX=INDEX(STRING, CALL)+5
00101 00965000 CALL DETERMINE(INDEX, NAME, STRING)
00105 00966000 C CHECK FOR RECURSIVE CALL
00105 00967000 IF(NAME=NE.UNITNAME)GO TO 44
00113 00968000 STRING(17:141)="RECURSIVE CALL"
00113 00969000 RETURN
00133 00970000 C CHECK FOR NAME SUBSTITUTION VIA USE OF "EXTERNAL" STATEMENT
00133 00971000 44 IF(INDEX(STRING, "PASSED VIA EXTERNAL").EQ.0)RETURN
00166 00972000 C FIND ACTUAL SUBROUTINE NAME IN "ARGSUBSTITUTION" ARRAY
00166 00973000 112 DO 10 I=1,8
00173 00974000 HEAD(ARGSUBSTITUTION(M,I),/7)ACTUALNAME, FORMALNAME
00231 00975000 77 FORMAT(16X, 2A16)
00231 00976000 IF(NAME.EQ.FORMALNAME)GO TO 11
00240 00977000 10 CONTINUE
00241 00978000 11 PROGNIT(1)11:12)=SUB
00251 00979000 PROGNIT(1)11:16)=ACTUALNAME
00263 00980000 PROGNIT(2)11:10)=FUNCTION
00273 00981000 PROGNIT(2)11:18)=ACTUALNAME
00307 00982000 CALL FINDPROGNARUNIT(PROGNIT, REDUCED, STRING, 11, 12, 13, 14, 15, HLC,
00307 00983000 1LC, SUB, FUNCTION, DUNITNAME, IDUN, 599)
00340 00984000 C SUBSTITUTE ACTUAL NAME FOR FORMAL NAME
00340 00985000 111 NAME=ACTUALNAME
00344 00986000 RETURN
00345 00987000 C PROGRAM UNIT NOT FOUND/ GET "ARGSUBSTITUTION" POINTER FROM
00345 00988000 C CALLING PROGRAM UNIT
00345 00989000 99 IF(J.EQ.0)GO TO 98
00351 00990000 HEAD(REDUCED(LOCALLEX),/7)M
00376 00991000 77 FORMAT(7X, 14)
00376 00992000 J=J-1
00377 00993000 LOCALLEX=LEVELLOC(J)
00402 00994000 GO TO 112
00403 00995000 98 DISPLAY"NO PROGRAM UNIT FOUND FOR ", NAME, ". PROGRAM IS STOPPED"
00470 00996000 STOP
00471 00997000 END

```

## SYMBOL MAP

| NAME       | TYPE      | STRUCTURE  | ADDRESS  | NAME            | TYPE      | STRUCTURE  | ADDRESS  |
|------------|-----------|------------|----------|-----------------|-----------|------------|----------|
| ACTUALNAME | CHARACTER | SIMPLE VAR | 0-814 ,1 | ARGSUBSTITUTION | CHARACTER | ARRAY      | 0-817 ,1 |
| CALL       | CHARACTER | SIMPLE VAR | 0-821 ,1 | DETERMINE NAME  | CHARACTER | SUBROUTINE | 0-817 ,1 |

PAGE 0046 DETENTRICUALLY

[illegible]

2400 7:35 PM

| STATEMENT LABEL | CODE<br>OFFSET | STATEMENT LABEL | CODE<br>OFFSET | STATEMENT LABEL | CODE<br>OFFSET | STATEMENT LABEL | CODE<br>OFFSET | STATEMENT LABEL | CODE<br>OFFSET | STATEMENT LABEL | CODE<br>OFFSET |
|-----------------|----------------|-----------------|----------------|-----------------|----------------|-----------------|----------------|-----------------|----------------|-----------------|----------------|
| 10              | 240            | 11              | 241            | 44              | 133            | 11 FMT          | 0              | FM              | 403            | 99              | 345            |
| 11              | 340            | 112             | 166            | 717 FMT         | 5              |                 |                |                 |                |                 |                |

PROGRAM UNIT DETERMINED CALL COMPILED

PAGE 0047 HEMLETT-PACKARD 121020.01.09 FURTRAN/1000 WED, JUN 15, 1983, 12:20 PM

```

00042 00598000 $CUNTRUL INIT,MAP,LABEL,LOCUTION
00042 00599000 SUBROUTINE PRINTTOTALS(TOTLINES,TOTCOMMENTS,TOTFORMATS,TOTDECL,
00042 01000000 INAPROGUNIT)
00042 01001000 TOTLINES=TOTLINES-TOTCOMMENTS-TOTFORMATS-TOTDECL
00042 01002000 WRITE(6,1)INAPROGUNIT,TOTLINES,TOTCOMMENTS,TOTDECL,TOTFORMATS,
00052 01003000 1 TOTLINES
00052 01004000 1 FORMAT(," TOTALS FOR ",13," PROGRAM UNITS",J2A,4I12,117)
00103 01005000 RETURN
00103 01005000 END
00104 01006000

```

## SYMBOL MAP

| NAME        | TYPE    | STRUCTURE  | ADDRESS | NAME        | TYPE | STRUCTURE  | ADDRESS  |
|-------------|---------|------------|---------|-------------|------|------------|----------|
| INAPROGUNIT | INTEGER | SIMPLE VAR | 0-34 ,1 | PRINTTOTALS | REAL | SUBROUTINE | 0-35 ,1  |
| TOTCOMMENTS | REAL    | SIMPLE VAR | 0-37 ,1 | TOTDECL     | REAL | SIMPLE VAR | 0-36 ,1  |
| TOTFORMATS  | REAL    | SIMPLE VAR | 0-36 ,1 | TOTLINES    | REAL | SIMPLE VAR | 0-310 ,1 |
| TOTDEU      | REAL    | SIMPLE VAR | 0-31    |             |      |            |          |

## LABEL MAP

| STATEMENT LABEL | CODE OFFSET | STATEMENT LABEL | CODE OFFSET | STATEMENT LABEL | CODE OFFSET | STATEMENT LABEL | CODE OFFSET |
|-----------------|-------------|-----------------|-------------|-----------------|-------------|-----------------|-------------|
| 1 FMT           | 0           |                 |             |                 |             |                 |             |

PROGRAM UNIT PRINTTOTALS COMPILED



PAGE 0048 NEWLETT-PACKARD 32102H.01.09 FURTHAN/1000 WED, JUN 15, 1983, 12:26 PM

```

00010 01007000 $CONTROL INIT,MAP,LABEL,LOCATION
00010 01008000 $SOURCEFILE(SOURCEFILE,LOCATIONS,TOTALLINES)
00010 01009000 $SYSTEM INTRINSIC FROM,FCLOSE
00010 01010000 $NUMERICALS=TOTALS
00015 01011000 C DETERMINE IF # OF OUTPUT LINES = # OF INPUT LINES
00015 01012000 IF (NUMERICALS.EQ.NUMERICALS)GO TO 77
00021 01013000 DISPLAY**
00021 01014000 DISPLAY**
00034 01015000 C DISPLAY ERROR MESSAGE BECAUSE INPUT LINES DO NOT EQUAL OUTPUT LINES
00047 01016000 C AND TERMINATE PROGRAM WITHOUT CREATING NEW FILE
00047 01017000 DISPLAY"NUMBER OF LINES IN INPUT FILE =" ,NUMERICALS
00047 01018000 DISPLAY"NUMBER OF LINES IN OUTPUT FILE =" ,NUMERICALS
00111 01019000 DISPLAY**
00153 01020000 DISPLAY"THUS OUTPUT FILE OF REARRANGED CODE IS NOT SAVED."
00166 01021000 DISPLAY**
00237 01022000 DISPLAY**
00252 01023000 DISPLAY"PROGRAM IS TERMINATED BECAUSE OF THIS ERROR."
00320 01024000 STOP
00321 01025000 C NUMBER OF INPUT LINES=NUMBER OF OUTPUT LINES
00321 01026000 C SAVE PERMANENT FILE OF SOURCE CODE NAME WITH REARRANGED OUTPUT FILE
00331 01027000 77 CALL FCLOSE(FROM,4,0)
00331 01028000 RETURN
00332 01029000 END

```

## SYMBOL MAP

| NAME        | TYPE    | STRUCTURE  | ADDRESS | NAME       | TYPE    | STRUCTURE  | ADDRESS |
|-------------|---------|------------|---------|------------|---------|------------|---------|
| FCLOSE      | INTEGER | SUBROUTINE | 0101    | FROM       | INTEGER | FUNCTION   | 0-15 ,1 |
| NUMERICALS  | INTEGER | SIMPLE VAR | 0101    | NUMERICALS | INTEGER | SIMPLE VAR | 0-15 ,1 |
| SAVENEWFILE | INTEGER | SUBROUTINE | 0101    | TOTALLINES | INTEGER | SIMPLE VAR | 0-15 ,1 |

| STATEMENT LABEL | CODE OFFSET | STATEMENT LABEL | CODE OFFSET | STATEMENT LABEL | CODE OFFSET |
|-----------------|-------------|-----------------|-------------|-----------------|-------------|
| 77              | 321         |                 |             |                 |             |

## LABEL MAP

| STATEMENT LABEL | CODE OFFSET | STATEMENT LABEL | CODE OFFSET | STATEMENT LABEL | CODE OFFSET |
|-----------------|-------------|-----------------|-------------|-----------------|-------------|
| 77              | 321         |                 |             |                 |             |

PROGRAM UNIT SAVENEWFILE COMPILED

PAGE 0049 HEMLEFF-PACKARD 321028,01.09 FORTRAN/1000 WED, JUN 15, 1965, 12:26 PM

```

00046 01029000 SCUNROL IMIT,MAP,LABEL,LOCUTION
00046 01030000 SUBROUTINE COMPUTSTATISTICS(SUB,FUNCTION,SOURCEFILE,LC,MLC,REDUCED,
00046 01031000 ITOFCOMMENTS,TOTFORMATS,TOTDECL,NOMPROGUNIT,S,PAGE)
00046 01032000 INTEGER PAGE
00046 01033000 CHARACTER SUB*12,FUNCTION*10,SOURCEFILE*8,REDUCED*9*2(500)
00046 01034000 C PRINT OUT ALPHABETICAL LISTING OF PROGRAM UNIT NAMES
00046 01035000 CALL LISTNAMES(SUB,FUNCTION,MLC,REDUCED,PAGE,NOMPROGUNIT,MAXUNIT,
00046 01036000 IMINUNIT,LINECOUNTER,TOTALCALLS)
00046 01037000 C PRINT OUT TOTALS LINE
00046 01038000 WRITE(6,1)NOMPROGUNIT,LC,TOTALCALLS
00046 01039000 1 FORMAT(/,' TOTALS FOR ',11,' PROGRAM UNITS',21X,11Z,11B,/)
00046 01040000 C COMPUTE AND PRINT OUT PROGRAM'S STATISTICS
00046 01041000 IF(LINECOUNTER.GT.40)CALL PAGEHEAD(LINECOUNTER,PAGE,NOMPROGUNIT,
00046 01042000 SOURCEFILE)
00046 01043000 DISPLAY**
00046 01044000 DISPLAY=LARGEST PROGRAM UNIT SIZE =",MAXUNIT,"LINES"
00046 01045000 DISPLAY=SMALLEST PROGRAM UNIT SIZE =",MINUNIT,"LINES"
00046 01046000 AVGNUNIT=FLOAT(LC)/NOMPROGUNIT
00046 01047000 DISPLAY=AVERAGE PROGRAM UNIT SIZE =",INT(AVGUNIT+.5),"LINES"
00046 01048000 DISPLAY**
00046 01049000 PERCENTCOMMENTS=100.*TOTCOMMENTS/LC
00046 01050000 PERCENTDECL=100.*TOTDECL/LC
00046 01051000 PERCENTFMT=100.*TOTFORMATS/LC
00046 01052000 PERCENTDEU=100.*(LC-TOTCOMMENTS-TOTDECL-TOTFORMATS)/LC
00046 01053000 DISPLAY=" COMMENT LINES =",INT(PERCENTCOMMENTS+.5)
00046 01054000 DISPLAY=" DECLARATION LINES =",INT(PERCENTDECL+.5)
00046 01055000 DISPLAY=" FORMAT LINES =",INT(PERCENTFMT+.5)
00046 01056000 DISPLAY=" EXECUTABLE LINES =",INT(PERCENTDEU+.5)
00046 01057000 RETURN
00046 01058000 END

```

## SYMBOL MAP

| NAME            | TYPE      | STRUCTURE  | ADDRESS  | NAME            | TYPE      | STRUCTURE  | ADDRESS  |
|-----------------|-----------|------------|----------|-----------------|-----------|------------|----------|
| AVGNUNIT        | REAL      | SIMPLE VAR | 0-112 ,1 | COMPUTSTATISTIC |           |            |          |
| FUNCTION        | CHARACTER | SIMPLE VAR | 0-115 ,1 | LC              | INTEGER   | SIMPLE VAR | 0-113 ,1 |
| LINECOUNTER     | INTEGER   | SIMPLE VAR | 0-11 ,1  | LISTNAMES       |           |            |          |
| MAXUNIT         | INTEGER   | SIMPLE VAR | 0-112 ,1 | MINUNIT         | INTEGER   | SIMPLE VAR | 0-113 ,1 |
| MLC             | INTEGER   | SIMPLE VAR | 0-112 ,1 | NOMPROGUNIT     | INTEGER   | SIMPLE VAR | 0-113 ,1 |
| PAGE2HEAD       | REAL      | SUBROUTINE | 0-110    | PAGE            | INTEGER   | SIMPLE VAR | 0-114 ,1 |
| PERCENTCOMMENTS | REAL      | SIMPLE VAR | 0-11 ,1  | PERCENTDECL     | REAL      | SIMPLE VAR | 0-116 ,1 |
| PERCENTFMT      | REAL      | SIMPLE VAR | 0-11 ,1  | PERCENTDEU      | REAL      | SIMPLE VAR | 0-114 ,1 |
| REDUCED         | CHARACTER | ARRAY      | 0-111 ,1 | SOURCEFILE      | CHARACTER | SIMPLE VAR | 0-114 ,1 |
| SUB             | CHARACTER | SIMPLE VAR | 0-116 ,1 | TOTALCALLS      | REAL      | SIMPLE VAR | 0-116 ,1 |
| TOTCOMMENTS     | REAL      | SIMPLE VAR | 0-110 ,1 | TOTDECL         | REAL      | SIMPLE VAR | 0-116 ,1 |
| TOTFORMATS      | REAL      | SIMPLE VAR | 0-117 ,1 |                 |           |            |          |

## LABEL MAP

| STATEMENT LABEL | CODE OFFSET | STATEMENT LABEL | CODE OFFSET | STATEMENT LABEL | CODE OFFSET |
|-----------------|-------------|-----------------|-------------|-----------------|-------------|
|-----------------|-------------|-----------------|-------------|-----------------|-------------|

1 FMT 0

PAGE 0050 COMPUTATISTIC

PROGRAM UNIT COMPUTATISTIC COMPILED

PAGE 0051 HEWLETT-PACKARD 32102B-01-09 FORTNAN/3000 WED, JUN 15, 1983, 12:27 PM

```

00055 01054000 $CONTROL INIT,MAP,LABEL,LOCATION
00055 01055000 SUBROUTINE LISTNAMES(SUB,FUNCTION,MLC,REDUCED,PAGENO,SOURCEFILE,
00055 01060000 1,MAXUNIT,MINUNIT,LINECOUNTER,TOTALCALLS)
00055 01061000 CHARACTER STRING*72,NAME*16,SUB*12,FUNCTION*10,REDUCED*4(500),
00055 01062000 1,SOURCEFILE*8
00055 01063000 INTEGER SEGBEGIN,SEGEN,NUMLINES,MUNITSCALLED,PAGENO
00055 01064000 LC=0
00057 01065000 MAXUNIT=0
00061 01066000 MINUNIT=5000
00063 01067000 TOTALCALLS=0.
00065 01068000 DO 20 J=1,MLC
00072 01069000 READ(REDUCED(J),9)STRING,SEGBEGIN,SEGEN,NUMLINES,MUNITSCALLED
00131 01070000 9 FORMAT(A72,4I4)
00131 01071000 C DETERMINE IF PROGRAM UNIT STATEMENT
00131 01072000 IF(STRING(1:4).EQ."MAIN")WRITE(REDUCED(LC+1),9)STRING,SEGBEGIN,
00131 01073000 1,SEGEN,NUMLINES,MUNITSCALLED
00210 01074000 IF(INDEX(STRING,SUB).GT.0)WRITE(REDUCED(LC+1),9)STRING,SEGBEGIN,
00263 01075000 1,SEGEN,NUMLINES,MUNITSCALLED
00263 01076000 IF(INDEX(STRING,FUNCTION).GT.0)WRITE(REDUCED(LC+1),9)STRING(1:15)
00263 01077000 1,STRING(16:55),SEGBEGIN,SEGEN,NUMLINES,MUNITSCALLED
00344 01078000 1,STRING(1:15),2A,A55,4I4)
00344 01079000 IF(INDEX(STRING,SUB).GT.0)WRITE(REDUCED(LC+1),9)STRING(1:4).EQ."MAIN".UN.INDEX(STRING
00344 01080000 1,FUNCTION).GT.0)LC=LC+1
00402 01081000 C IF NOT PROGRAM UNIT STATEMENT, THEN NOT WRITTEN TO 'REDUCED' ARRAY
00402 01082000 20 CONTINUE
00403 01083000 MLC=LC
00405 01084000 C SORT PROGRAM UNITS INTO ALPHABETICAL ORDER
00405 01085000 CALL SORTARRAY(REDUCED,MLC,19,16)
00417 01086000 C PRINT PAGE HEADING
00417 01087000 CALL PAGE2HEAD(LINECOUNTER,PAGENO,SOURCEFILE)
00423 01088000 C DISPLAY ALPHABETICAL LISTING OF PROGRAM UNIT NAMES
00423 01089000 DO 100 I=1,MLC
00430 01090000 READ(REDUCED(I),9)STRING,SEGBEGIN,SEGEN,NUMLINES,MUNITSCALLED
00467 01091000 LINECOUNTER=LINECOUNTER+1
00470 01092000 IF(LINECOUNTER.GT.50)CALL PAGE2HEAD(LINECOUNTER,PAGENO,SOURCEFILE)
00477 01093000 4R(16,77)STRING(16:111),SEGBEGIN,SEGEN,NUMLINES,MUNITSCALLED
00532 01094000 77 FORMAT(1X,A30,19,112,112,116)
00532 01095000 C CHECK FOR LARGEST AND SMALLEST PROGRAM UNIT
00532 01096000 IF(NUMLINES.GT.MAXUNIT)MAXUNIT=NUMLINES
00537 01097000 IF(NUMLINES.LT.MINUNIT)MINUNIT=NUMLINES
00544 01098000 TOTALCALLS=TOTALCALLS+MUNITSCALLED
00550 01099000 100 CONTINUE
00551 01100000 RETURN
00554 01101000 END

```

## SYMBOL MAP

| NAME     | FUNCTION    | TYPE      | STRUCTURE  | ADDRESS  | NAME         | TYPE    | STRUCTURE  | ADDRESS  |
|----------|-------------|-----------|------------|----------|--------------|---------|------------|----------|
| J        | FUNCTION    | CHARACTER | SINGLE VAR | 0-814 ,1 | 1            | INTEGER | SIMPLE VAR | 0-811    |
| L        | LINECOUNTER | INTEGER   | SIMPLE VAR | 0-811    | LC           | INTEGER | SIMPLE VAR | 0-815    |
| MAXUNIT  | MAXUNIT     | INTEGER   | SIMPLE VAR | 0-85 ,1  | LISTNAMES    | INTEGER | SUBROUTINE |          |
| MINUNIT  | MINUNIT     | INTEGER   | SIMPLE VAR | 0-87 ,1  | MINUNIT      | INTEGER | SIMPLE VAR | 0-86 ,1  |
| NAME     | NAME        | CHARACTER | NULL       |          | ALL          | INTEGER | SIMPLE VAR | 0-813 ,1 |
| NUMLINES | NUMLINES    | INTEGER   | SIMPLE VAR | 0-81     | MUNITSCALLED | INTEGER | SIMPLE VAR | 0-880    |



```

00120 $COUNTRUL INIT,MAP,LABEL,LOCUTION
01102000 SUBROUTINE PAGEHEAD(LINECOUNTER,PAGENO,SOURCEFILE)
01103000 CHARACTER DATE*27,SOURCEFILE*9
01104000 INTEGER PAGENO
01105000 CALL DATELINE(DATE)
01106000 PAGENO=PAGENO+1
01107000 WRITE(6,1)DATE(6:12),PAGENO
01108000 1 FORMAT(//,1H1,10X,A12, PAGE ",I2)
01109000 DISPLAY=LISTING OF PROGRAM UNITS IN ALPHABETICAL ORDER WITH STATUS
01110000 1 DISPLAY SUMMARY FOR SOURCE CODE: ",SOURCEFILE
01111000 1 DISPLAY
01112000 1-----
01113000 1-----
01114000 1-----
01115000 1-----
01116000 1-----
01117000 1-----
01118000 1-----
01119000 1-----
01120000 1-----
01121000 1-----
01122000 1-----
01123000 1-----
01124000 1-----
01125000 1-----
01126000 1-----
01127000 1-----
01128000 1-----
01129000 1-----
01130000 1-----
01131000 1-----
01132000 1-----
01133000 1-----
01134000 1-----
01135000 1-----
01136000 1-----
01137000 1-----
01138000 1-----
01139000 1-----
01140000 1-----
01141000 1-----
01142000 1-----
01143000 1-----
01144000 1-----
01145000 1-----
01146000 1-----
01147000 1-----
01148000 1-----
01149000 1-----
01150000 1-----
01151000 1-----
01152000 1-----
01153000 1-----
01154000 1-----
01155000 1-----
01156000 1-----
01157000 1-----
01158000 1-----
01159000 1-----
01160000 1-----
01161000 1-----
01162000 1-----
01163000 1-----
01164000 1-----
01165000 1-----
01166000 1-----
01167000 1-----
01168000 1-----
01169000 1-----
01170000 1-----
01171000 1-----
01172000 1-----
01173000 1-----
01174000 1-----
01175000 1-----
01176000 1-----
01177000 1-----
01178000 1-----
01179000 1-----
01180000 1-----
01181000 1-----
01182000 1-----
01183000 1-----
01184000 1-----
01185000 1-----
01186000 1-----
01187000 1-----
01188000 1-----
01189000 1-----
01190000 1-----
01191000 1-----
01192000 1-----
01193000 1-----
01194000 1-----
01195000 1-----
01196000 1-----
01197000 1-----
01198000 1-----
01199000 1-----
01200000 1-----
01201000 1-----
01202000 1-----
01203000 1-----
01204000 1-----
01205000 1-----
01206000 1-----
01207000 1-----
01208000 1-----
01209000 1-----
01210000 1-----
01211000 1-----
01212000 1-----
01213000 1-----
01214000 1-----
01215000 1-----
01216000 1-----
01217000 1-----
01218000 1-----
01219000 1-----
01220000 1-----
01221000 1-----
01222000 1-----
01223000 1-----
01224000 1-----
01225000 1-----
01226000 1-----
01227000 1-----
01228000 1-----
01229000 1-----
01230000 1-----
01231000 1-----
01232000 1-----
01233000 1-----
01234000 1-----
01235000 1-----
01236000 1-----
01237000 1-----
01238000 1-----
01239000 1-----
01240000 1-----
01241000 1-----
01242000 1-----
01243000 1-----
01244000 1-----
01245000 1-----
01246000 1-----
01247000 1-----
01248000 1-----
01249000 1-----
01250000 1-----
01251000 1-----
01252000 1-----
01253000 1-----
01254000 1-----
01255000 1-----
01256000 1-----
01257000 1-----
01258000 1-----
01259000 1-----
01260000 1-----
01261000 1-----
01262000 1-----
01263000 1-----
01264000 1-----
01265000 1-----
01266000 1-----
01267000 1-----
01268000 1-----
01269000 1-----
01270000 1-----
01271000 1-----
01272000 1-----
01273000 1-----
01274000 1-----
01275000 1-----
01276000 1-----
01277000 1-----
01278000 1-----
01279000 1-----
01280000 1-----
01281000 1-----
01282000 1-----
01283000 1-----
01284000 1-----
01285000 1-----
01286000 1-----
01287000 1-----
01288000 1-----
01289000 1-----
01290000 1-----
01291000 1-----
01292000 1-----
01293000 1-----
01294000 1-----
01295000 1-----
01296000 1-----
01297000 1-----
01298000 1-----
01299000 1-----
01300000 1-----
01301000 1-----
01302000 1-----
01303000 1-----
01304000 1-----
01305000 1-----
01306000 1-----
01307000 1-----
01308000 1-----
01309000 1-----
01310000 1-----
01311000 1-----
01312000 1-----
01313000 1-----
01314000 1-----
01315000 1-----
01316000 1-----
01317000 1-----
01318000 1-----
01319000 1-----
01320000 1-----
01321000 1-----
01322000 1-----
01323000 1-----
01324000 1-----
01325000 1-----
01326000 1-----
01327000 1-----
01328000 1-----
01329000 1-----
01330000 1-----
01331000 1-----
01332000 1-----
01333000 1-----
01334000 1-----
01335000 1-----
01336000 1-----
01337000 1-----
01338000 1-----
01339000 1-----
01340000 1-----
01341000 1-----
01342000 1-----
01343000 1-----
01344000 1-----
01345000 1-----
01346000 1-----
01347000 1-----
01348000 1-----
01349000 1-----
01350000 1-----
01351000 1-----
01352000 1-----
01353000 1-----
01354000 1-----
01355000 1-----
01356000 1-----
01357000 1-----
01358000 1-----
01359000 1-----
01360000 1-----
01361000 1-----
01362000 1-----
01363000 1-----
01364000 1-----
01365000 1-----
01366000 1-----
01367000 1-----
01368000 1-----
01369000 1-----
01370000 1-----
01371000 1-----
01372000 1-----
01373000 1-----
01374000 1-----
01375000 1-----
01376000 1-----
01377000 1-----
01378000 1-----
01379000 1-----
01380000 1-----
01381000 1-----
01382000 1-----
01383000 1-----
01384000 1-----
01385000 1-----
01386000 1-----
01387000 1-----
01388000 1-----
01389000 1-----
01390000 1-----
01391000 1-----
01392000 1-----
01393000 1-----
01394000 1-----
01395000 1-----
01396000 1-----
01397000 1-----
01398000 1-----
01399000 1-----
01400000 1-----
01401000 1-----
01402000 1-----
01403000 1-----
01404000 1-----
01405000 1-----
01406000 1-----
01407000 1-----
01408000 1-----
01409000 1-----
01410000 1-----
01411000 1-----
01412000 1-----
01413000 1-----
01414000 1-----
01415000 1-----
01416000 1-----
01417000 1-----
01418000 1-----
01419000 1-----
01420000 1-----
01421000 1-----
01422000 1-----
01423000 1-----
01424000 1-----
01425000 1-----
01426000 1-----
01427000 1-----
01428000 1-----
01429000 1-----
01430000 1-----
01431000 1-----
01432000 1-----
01433000 1-----
01434000 
```

**SYMBOL MAP**

| NAME       | TYPE      | STRUCTURE | ADDRESS  | NAME       | TYPE       | STRUCTURE | ADDRESS |
|------------|-----------|-----------|----------|------------|------------|-----------|---------|
| DATE       | CHARACTER | SIMPLE    | VAR 0-11 | DATELINE   | SUBROUTINE |           |         |
| LINENUMBER | INTEGER   | SIMPLE    | VAR 0-16 | PAGE2HEAD  | SUBROUTINE |           |         |
| PAGENO     | INTEGER   | SIMPLE    | VAR 0-15 | SOURCEFILE | SIMPLE     | VAR 0-14  | 1       |

**LABEL MAP**

| STATEMENT LABEL | CODE   | STATEMENT LABEL | CODE   | STATEMENT LABEL | CODE   | STATEMENT LABEL | CODE   |
|-----------------|--------|-----------------|--------|-----------------|--------|-----------------|--------|
| OFFSET          | OFFSET | OFFSET          | OFFSET | OFFSET          | OFFSET | OFFSET          | OFFSET |
| 1 FMT           | 0      | 76 FMT          | 20     |                 |        |                 |        |

PROGRAM UNIT PAGE2HEAD COMPILED

PAGE 0054    HEWLETT-PACKARD 32162B-01-02    FORTKAW/1000    WED, JUN 15, 1983, 1212/ PM

\*\*\*\* GLOBAL STATISTICS \*\*\*\*  
\*\*\*\* NO ERRORS, NO WARNINGS \*\*\*\*  
TOTAL COMPIATION TIME 0:01:14  
TOTAL ELAPSED TIME 0:03:06  
TOTAL SYMBOL TABLE USED 8 11

## USER'S MANUAL FOR FORTRAN SOURCE CODE REARRANGER PROGRAM

### I. INTRODUCTION

This discussion assumes that the object code of the FORTRAN source code rearranger program named 'REARRANG', is accessible to the user.

Before using this program to analyze and rearrange the source code file of a FORTRAN program, the following conditions must be met:

(1) all modules of the FORTRAN program must exist in one file. The name of this source code file can be from one to eight characters where the first character must be alphabetical.

(2) each module of the program should have been successfully compiled.

(3) all compiled modules of the program should have been successfully linked- loaded.

(4) the size of the program's source code cannot exceed 4000 lines.

(5) the number of subroutines and the number of functions must not exceed 100. The sum of functions plus subroutines cannot exceed 150.

(6) the total number of (CALL + SUBROUTINE + FUNCTION) statements cannot exceed 370.



## II. EXECUTING THE PROGRAM

To execute the source code rearranger program, simply type 'RUN REARRANG'. The program will then ask for the name of the FORTRAN source code file to be analyzed. Respond by typing the name of the FORTRAN source code file. If the file name does not exist or if the file is not accessible, the program will terminate displaying the appropriate file system error. If the file name is valid and the file is accessible, no further user input is required.

If the code rearranger program executes successfully, it will actually overwrite the new rearranged file to the old source code file. This can be done because the file size does not change, but is simply rearranged. If program execution is unsuccessful, for any reason, the appropriate message is displayed and the original source code file is unchanged.

If the code rearranger is unsuccessful, it will be because of one of the following reasons:

(1) Duplicate module names have been detected. The name(s) and locations in the unchanged source code file are shown and the program is terminated.

(2) One or more modules exist in the source code but are not called by at least one other module. The name(s) and locations of the the extraneous modules in the unchanged source code file are shown and the program is terminated.

(Although presence of not used modules does not prevent successful execution of a program, this detection does serve as a useful housekeeping and file space conservation utility.)

The user can then edit the source code file based on the above error detection information that has been given. Then, the code rearranger program can be executed again.

When the code rearranger program is successful, the source code file is rearranged concurrent with the outputting of the first of two reports, which shows the hierarchical top-down order of the modules of the FORTRAN source code. Included in this report are the beginning and ending line number in the newly rearranged file for each module. Also, for each module, the level of its call, the total number, number of comment, declaration, format, and executable lines of code are shown.

A second report is then output showing all the modules listed in alphabetical order with beginning and ending line number and number of times called for each module. Also, this report shows the number of lines for the largest, smallest, and average module size and the percent of lines of code classified as comment, declaration, format, and executable.

Upon successful execution of this program, it is suggested that a compiled listing of the rearranged source code file be generated. Then the output of the code rearranger program filed with this compiled listing will serve as both

a table of contents and an index for the FORTRAN program.

Whenever any program maintenance is done, the code rearranger program should be run for the affected FORTRAN source code and a new compiled listing generated.

## APPENDIX C

### EXAMPLES OF ANALYZED PROGRAMS

IKUN REARRANG

ENTER NAME OF FORTRAN SOURCE FILE TO BE ANALYZED 'REARRANGS

# OF STATEMENTS IN 'REDUCED' = 92 / 300  
NUMBER OF LINES OF INPUT FILE = 1123

SUMMARY OF EXECUTION SEQUENCE OF PROGRAM UNITS FOR FORTRAN SOURCE CODE: REARRANGS

| PROGRAM UNIT NAME           | BEG. LINE | END LINE | LEVEL | # OF LINES | # OF COMM. | # OF DECLAR. | # OF FORMATS | # OF XEQ LINES |
|-----------------------------|-----------|----------|-------|------------|------------|--------------|--------------|----------------|
| MAIN                        | 1         | 59       | 1     | 59         | 19         | 6            | 0            | 34             |
| SUBROUTINE DEFINEFILES      | 60        | 90       | 2     | 31         | 9          | 3            | 3            | 16             |
| SUBROUTINE FINOSEGBEGIN     | 91        | 132      | 2     | 42         | 4          | 7            | 2            | 27             |
| SUBROUTINE STRIPOFFPARAM    | 133       | 161      | 3     | 29         | 3          | 2            | 0            | 24             |
| SUBROUTINE SIORENAME        | 162       | 171      | 3     | 10         | 2          | 2            | 0            | 6              |
| SUBROUTINE DETERMINE        | 172       | 178      | 4     | 7          | 1          | 2            | 0            | 4              |
| SUBROUTINE SIORENAME        |           |          | 3     |            |            |              |              |                |
| SUBROUTINE DETERMINE        |           |          | 4     |            |            |              |              |                |
| SUBROUTINE CLASSIFYSTATM1   | 179       | 234      | 2     | 56         | 10         | 7            | 1            | 38             |
| SUBROUTINE DETERDISORFMT    | 235       | 250      | 3     | 16         | 0          | 4            | 0            | 12             |
| SUBROUTINE PROCESCALL       | 251       | 277      | 3     | 27         | 3          | 3            | 1            | 20             |
| SUBROUTINE PROCESSIF        | 278       | 293      | 4     | 16         | 1          | 2            | 0            | 13             |
| SUBROUTINE STRIPOFFPARAM    |           |          | 4     |            |            |              |              |                |
| SUBROUTINE CHECKARGLIST     | 294       | 348      | 3     | 55         | 17         | 5            | 2            | 31             |
| SUBROUTINE DETERMINE        |           |          | 4     |            |            |              |              |                |
| SUBROUTINE DETERDECLARE     | 349       | 390      | 3     | 42         | 2          | 4            | 0            | 36             |
| SUBROUTINE DELETEDIMCALLS   | 391       | 429      | 2     | 39         | 6          | 6            | 1            | 26             |
| SUBROUTINE DETERMINE        |           |          | 3     |            |            |              |              |                |
| SUBROUTINE INSERTFUNTCALLS  | 430       | 490      | 2     | 61         | 15         | 6            | 2            | 38             |
| SUBROUTINE DETERDECLARE     |           |          | 3     |            |            |              |              |                |
| SUBROUTINE DETERDISORFMT    | 491       | 504      | 3     | 14         | 1          | 2            | 0            | 11             |
| SUBROUTINE SORTARRAY        | 505       | 512      | 4     | 8          | 0          | 2            | 0            | 6              |
| SUBROUTINE SWITCH           |           |          |       |            |            |              |              |                |
| SUBROUTINE COMPLETEARGLIST  | 513       | 564      | 2     | 52         | 10         | 5            | 4            | 33             |
| SUBROUTINE GETSUBARGNAME    | 565       | 612      | 3     | 48         | 9          | 5            | 3            | 31             |
| SUBROUTINE SORTARRAY        |           |          | 3     |            |            |              |              |                |
| SUBROUTINE SWITCH           |           |          | 4     |            |            |              |              |                |
| SUBROUTINE FINISARGNAMES    | 613       | 651      | 2     | 39         | 5          | 5            | 0            | 29             |
| SUBROUTINE SWITCH           | 652       | 659      | 3     | 8          | 0          | 2            | 0            | 6              |
| SUBROUTINE DISPLAYUPNAMES   | 660       | 675      | 3     | 16         | 2          | 2            | 1            | 11             |
| SUBROUTINE DETERMINIFCALLED | 676       | 725      | 2     | 50         | 6          | 7            | 1            | 36             |
| SUBROUTINE DISPLAYNOCALLS   | 726       | 758      | 3     | 33         | 3          | 3            | 2            | 25             |
| SUBROUTINE REARRANGESOURCE  | 759       | 817      | 2     | 59         | 15         | 8            | 1            | 35             |
| SUBROUTINE PAGEHEAD         | 818       | 840      | 3     | 23         | 0          | 3            | 4            | 16             |
| SUBROUTINE FINDPROGRAMUNIT  | 841       | 865      | 3     | 25         | 3          | 4            | 1            | 15             |
| SUBROUTINE FINDSUBNAME      | 866       | 883      | 3     | 18         | 2          | 3            | 0            | 13             |
| SUBROUTINE PRINTLINE        | 884       | 921      | 3     | 38         | 8          | 5            | 3            | 22             |
| SUBROUTINE PAGEHEAD         |           |          | 4     |            |            |              |              |                |
| SUBROUTINE ACCUMLINETOTALS  | 922       | 930      | 4     | 9          | 0          | 2            | 0            | 7              |

JUN 16, 1983 PAUL 2

SUMMARY OF EXECUTION SEQUENCE OF PROGRAM UNITS FOR FORTRAN SOURCE CODE: REARRANS

| PROGRAM UNIT NAME           | BEG. LINE # | END LINE # | LEVEL | # OF LINES | # OF CORR. | # OF DECLAR. | # OF FORMATS | # OF XED LINES |
|-----------------------------|-------------|------------|-------|------------|------------|--------------|--------------|----------------|
| SUBROUTINE WRITEONCHFILE    | 931         | 949        | 3     | 19         | 1          | 3            | 2            | 13             |
| SUBROUTINE DETERPRODCALL    | 950         | 997        | 3     | 48         | 9          | 6            | 2            | 31             |
| SUBROUTINE DETERMINE NAME   |             |            | 4     |            |            |              |              |                |
| SUBROUTINE FINDPROGRAMUNIT  |             |            | 4     |            |            |              |              |                |
| SUBROUTINE PRINTTOTALS      | 998         | 1006       | 3     | 9          | 0          | 2            | 1            | 6              |
| SUBROUTINE SAVENCHFILE      | 1007        | 1028       | 3     | 22         | 5          | 2            | 0            | 15             |
| SUBROUTINE COMPUTSTATISTIC  | 1029        | 1057       | 2     | 29         | 3          | 4            | 1            | 21             |
| SUBROUTINE LIST NAMES       | 1058        | 1101       | 3     | 44         | 6          | 5            | 3            | 30             |
| SUBROUTINE SORTARRAY        |             |            | 4     |            |            |              |              |                |
| SUBROUTINE SWITCH           |             |            | 5     |            |            |              |              |                |
| SUBROUTINE PAGE2HEAD        | 1102        | 1123       | 4     | 22         | 0          | 3            | 3            | 16             |
| SUBROUTINE PAGE2HEAD        |             |            | 4     |            |            |              |              |                |
| SUBROUTINE PAGE2HEAD        |             |            | 3     |            |            |              |              |                |
| TOTALS FOR 36 PROGRAM UNITS |             |            |       | 1123       | 182        | 144          | 44           | 753            |

## LISTING OF PROGRAM UNITS IN ALPHABETICAL ORDER WITH STATISTICAL SUMMARY FOR SOURCE CODE: REARRANS

| PROGRAM UNIT NAME           | DEG. LINE | END LINE | # OF LINES | # OF TIMES CALLED |
|-----------------------------|-----------|----------|------------|-------------------|
| MAIN                        | 1         | 59       | 59         | 1                 |
| SUBROUTINE ACCUMLINETOTALS  | 922       | 930      | 9          | 1                 |
| SUBROUTINE CHECKARGLIST     | 294       | 348      | 55         | 1                 |
| SUBROUTINE CLASSIFYSTATENT  | 179       | 234      | 56         | 1                 |
| SUBROUTINE COMPLETEARGLIST  | 513       | 564      | 52         | 1                 |
| SUBROUTINE COMPUTSTATISTIC  | 1029      | 1057     | 29         | 1                 |
| SUBROUTINE DEFINETFILES     | 60        | 90       | 31         | 1                 |
| SUBROUTINE DELETETOTALS     | 391       | 429      | 39         | 1                 |
| SUBROUTINE DETEDECLARE      | 349       | 390      | 42         | 2                 |
| SUBROUTINE DETEDTSORFHT     | 235       | 250      | 16         | 2                 |
| SUBROUTINE DETERMINEFNAME   | 172       | 178      | 7          | 5                 |
| SUBROUTINE DETERMINEFNAME   | 674       | 725      | 50         | 1                 |
| SUBROUTINE DETEPROGUCALL    | 950       | 997      | 48         | 1                 |
| SUBROUTINE DISPLAYDUPNAMES  | 660       | 675      | 16         | 1                 |
| SUBROUTINE DISPLAYNOCALLS   | 726       | 758      | 33         | 2                 |
| SUBROUTINE FINDPROGRAMUNIT  | 841       | 865      | 25         | 2                 |
| SUBROUTINE FINDSNAMEFNAME   | 613       | 651      | 39         | 1                 |
| SUBROUTINE FINDSEGMENT      | 91        | 132      | 42         | 1                 |
| SUBROUTINE GETSUBARGNAME    | 565       | 612      | 48         | 1                 |
| SUBROUTINE INDENTSUBNAME    | 866       | 883      | 18         | 1                 |
| SUBROUTINE INSERTFUNTCALLS  | 430       | 490      | 61         | 1                 |
| SUBROUTINE LISTNAMES        | 1058      | 1101     | 44         | 1                 |
| SUBROUTINE PAGEHEAD         | 1102      | 1123     | 22         | 3                 |
| SUBROUTINE PAGEHEAD         | 818       | 840      | 23         | 2                 |
| SUBROUTINE PRINTLINE        | 884       | 921      | 38         | 1                 |
| SUBROUTINE PRINTTOTALS      | 998       | 1006     | 9          | 1                 |
| SUBROUTINE PROCESSCALL      | 251       | 277      | 27         | 1                 |
| SUBROUTINE PROCESSIF        | 278       | 293      | 16         | 1                 |
| SUBROUTINE REARRANGESOURCE  | 759       | 817      | 59         | 1                 |
| SUBROUTINE SAVENEMFILE      | 1007      | 1028     | 22         | 1                 |
| SUBROUTINE SORTARRAY        | 491       | 504      | 14         | 3                 |
| SUBROUTINE STORENAME        | 162       | 171      | 10         | 2                 |
| SUBROUTINE STRIPOFFPARAM    | 133       | 161      | 29         | 2                 |
| SUBROUTINE SWITCH           | 652       | 659      | 8          | 1                 |
| SUBROUTINE SWITCH           | 505       | 512      | 8          | 3                 |
| SUBROUTINE WRITETONEFILE    | 931       | 949      | 19         | 1                 |
| TOTALS FOR 36 PROGRAM UNITS |           |          | 1123       | 53                |

LARGEST PROGRAM UNIT SIZE = 41 LINES  
 SMALLEST PROGRAM UNIT SIZE = 7 LINES  
 AVERAGE PROGRAM UNIT SIZE = 31 LINES

% COMMENT LINES = 14  
 % DECLARATION LINES = 13  
 % FORMAT LINES = 4  
 % EXECUTABLE LINES = 67

END OF PROGRAM  
 IUYE

CFU-223. CONNECT-9. THU, JUN 16, 1983, 10:07 AM



RUN REARRANG.

ENTER NAME OF FORTRAN SOURCE FILE TO BE ANALYZED /RESUL303

# OF STATEMENTS IN 'REDUCED' = 145 /506

NUMBER OF LINES OF INPUT FILE = 1741

## SUMMARY OF EXECUTION SEQUENCE OF PROGRAM UNITS FOR FORTRAN SOURCE CODE: RESUL308

| PROGRAM UNIT NAME          | BEG. LINE | END LINE | LEVEL | # OF LINES | # OF COMM. | # OF DECLAR. | # OF FORMATS | # OF REQ LINES |
|----------------------------|-----------|----------|-------|------------|------------|--------------|--------------|----------------|
| MAIN                       | 1         | 83       | 1     | 83         | 22         | 14           | 0            | 47             |
| SUBROUTINE READGPPAR       | 84        | 106      | 2     | 23         | 0          | 5            | 1            | 17             |
| SUBROUTINE READDEPTPAR     | 107       | 147      | 3     | 41         | 2          | 4            | 0            | 35             |
| SUBROUTINE DEFINEFILES     | 148       | 168      | 2     | 21         | 0          | 3            | 5            | 13             |
| SUBROUTINE DATEDETERMINE   | 169       | 271      | 2     | 103        | 8          | 9            | 4            | 82             |
| SUBROUTINE SSWITCH         | 272       | 279      | 3     | 8          | 0          | 2            | 0            | 6              |
| SUBROUTINE DETERMINWEEK    | 280       | 322      | 3     | 43         | 5          | 4            | 1            | 33             |
| SUBROUTINE ENTERHEKNO      | 323       | 346      | 2     | 24         | 1          | 2            | 0            | 21             |
| SUBROUTINE CHANGEADJ       | 347       | 377      | 2     | 31         | 3          | 4            | 0            | 24             |
| SUBROUTINE TRUCKHOURS      | 378       | 396      | 2     | 19         | 1          | 4            | 0            | 14             |
| SUBROUTINE NOBALHOURS      | 397       | 415      | 2     | 19         | 1          | 4            | 0            | 14             |
| SUBROUTINE READDATA        | 416       | 463      | 2     | 48         | 4          | 8            | 0            | 36             |
| SUBROUTINE ASKFORCHANGES   | 464       | 527      | 3     | 64         | 5          | 10           | 0            | 49             |
| SUBROUTINE SHOROTOTALS     | 528       | 560      | 4     | 33         | 1          | 6            | 0            | 26             |
| SUBROUTINE ACCEPTCHANGES   | 561       | 624      | 4     | 64         | 1          | 7            | 0            | 56             |
| SUBROUTINE SHOROTOTALS     |           |          | 5     |            |            |              |              |                |
| SUBROUTINE COMPUTENEMDATA  | 625       | 656      | 4     | 32         | 1          | 9            | 0            | 22             |
| SUBROUTINE STOREDATA       | 657       | 689      | 3     | 33         | 1          | 8            | 0            | 24             |
| SUBROUTINE CHANGEINV       | 690       | 710      | 2     | 21         | 1          | 4            | 0            | 16             |
| SUBROUTINE GETBH           | 711       | 759      | 2     | 49         | 2          | 9            | 2            | 36             |
| SUBROUTINE LUMQUAN         | 760       | 796      | 3     | 37         | 1          | 10           | 0            | 26             |
| SUBROUTINE GETQUAN         | 797       | 824      | 3     | 28         | 0          | 9            | 0            | 19             |
| SUBROUTINE CONCATENATE     | 825       | 834      | 3     | 10         | 0          | 3            | 1            | 6              |
| SUBROUTINE CUMHOURSBI      | 835       | 920      | 3     | 86         | 8          | 11           | 0            | 67             |
| SUBROUTINE CONCATENATE     |           |          | 4     |            |            |              |              |                |
| SUBROUTINE CUMHOURS        | 921       | 964      | 3     | 44         | 6          | 6            | 0            | 32             |
| SUBROUTINE FINDCOMPONENT   | 965       | 1008     | 4     | 44         | 5          | 10           | 0            | 29             |
| SUBROUTINE COMPONENTHOURS  | 1009      | 1040     | 5     | 32         | 3          | 7            | 0            | 22             |
| SUBROUTINE CONCATENATE     |           |          | 6     |            |            |              |              |                |
| SUBROUTINE FINDCOMPONENT   |           |          | 5     |            |            |              |              |                |
| SUBROUTINE CHECKFORBOTHFAC | 1041      | 1061     | 4     | 21         | 2          | 5            | 0            | 14             |
| SUBROUTINE COMPUTEIREC     | 1062      | 1114     | 2     | 53         | 4          | 6            | 1            | 42             |
| SUBROUTINE SORTANDDISPLA   | 1115      | 1149     | 3     | 35         | 2          | 3            | 3            | 27             |
| SUBROUTINE SWITCH          | 1150      | 1157     | 4     | 8          | 0          | 2            | 0            | 6              |
| SUBROUTINE COMPUTEIREC     | 1158      | 1185     | 2     | 28         | 3          | 10           | 0            | 15             |
| SUBROUTINE DETERHEKNO      | 1186      | 1202     | 3     | 17         | 2          | 4            | 0            | 11             |
| SUBROUTINE ACCUMHOURS      | 1203      | 1215     | 3     | 13         | 3          | 2            | 0            | 8              |
| SUBROUTINE COMPUTINCHG     | 1216      | 1242     | 2     | 27         | 1          | 6            | 0            | 20             |
| SUBROUTINE COMPUTESETUPHRS | 1243      | 1270     | 2     | 28         | 4          | 9            | 0            | 15             |



## LISTING OF PROGRAM UNITS IN ALPHABETICAL ORDER WITH STATISTICAL SUMMARY FOR SOURCE CODE: RESUL305

| PROGRAM UNIT NAME           | BEG. LINE | END LINE | # OF LINES | # OF TIMES CALLED |
|-----------------------------|-----------|----------|------------|-------------------|
| MAIN                        | 1         | 83       | 83         | 1                 |
| SUBROUTINE ACCEPTCHANGES    | 561       | 624      | 64         | 1                 |
| SUBROUTINE ACCUMCHGHS       | 1301      | 1317     | 17         | 1                 |
| SUBROUTINE ACCUMHOURS       | 1203      | 1215     | 13         | 1                 |
| SUBROUTINE ASKFORCHANGES    | 464       | 527      | 64         | 1                 |
| SUBROUTINE CHANGEADJ        | 347       | 377      | 31         | 1                 |
| SUBROUTINE CHANGEINV        | 690       | 710      | 21         | 1                 |
| SUBROUTINE CHECKFORBOTHFAC  | 1041      | 1061     | 21         | 1                 |
| SUBROUTINE COMPCHGHS        | 1271      | 1300     | 30         | 1                 |
| SUBROUTINE COMPINCHG        | 1216      | 1242     | 27         | 1                 |
| SUBROUTINE COMPONENTHOURS   | 1009      | 1040     | 32         | 1                 |
| SUBROUTINE COMPUTAREC       | 1158      | 1185     | 28         | 1                 |
| SUBROUTINE COMPUTEMEMDATA   | 625       | 656      | 32         | 1                 |
| SUBROUTINE COMPUTEDEC       | 1052      | 1114     | 63         | 1                 |
| SUBROUTINE COMPUTESETUPHRS  | 1243      | 1270     | 28         | 1                 |
| SUBROUTINE CONCATENATE      | 825       | 834      | 10         | 3                 |
| SUBROUTINE CUMHOURS         | 921       | 964      | 44         | 1                 |
| SUBROUTINE CUMHOURS81       | 835       | 920      | 86         | 1                 |
| SUBROUTINE DATEDETERMINE    | 149       | 271      | 103        | 1                 |
| SUBROUTINE DEFINETFILES     | 148       | 168      | 21         | 1                 |
| SUBROUTINE DETERMINEEEK     | 280       | 322      | 43         | 1                 |
| SUBROUTINE DETERMINEKOKO    | 1186      | 1202     | 17         | 3                 |
| SUBROUTINE ENTERHEENKO      | 323       | 346      | 24         | 1                 |
| SUBROUTINE ENTERHEENKO      | 965       | 1008     | 44         | 2                 |
| SUBROUTINE FINDCOMPONENT    | 711       | 759      | 49         | 1                 |
| SUBROUTINE GETBH            | 797       | 824      | 28         | 1                 |
| SUBROUTINE GETQUAN          | 1701      | 1713     | 13         | 2                 |
| SUBROUTINE LINETYPE1        | 1714      | 1741     | 28         | 14                |
| SUBROUTINE LINETYPE2        | 760       | 796      | 37         | 1                 |
| SUBROUTINE LUMPQUAN         | 397       | 415      | 19         | 1                 |
| SUBROUTINE NOBALHOURS       | 1498      | 1613     | 116        | 1                 |
| SUBROUTINE OUTPUTEARN       | 1318      | 1428     | 111        | 1                 |
| SUBROUTINE OUTPUTEFF        | 1483      | 1497     | 15         | 1                 |
| SUBROUTINE PREVEFF          | 416       | 463      | 48         | 1                 |
| SUBROUTINE READDATA         | 107       | 147      | 41         | 1                 |
| SUBROUTINE READDEPTPAR      | 84        | 106      | 23         | 1                 |
| SUBROUTINE READGRPAR        | 528       | 560      | 33         | 2                 |
| SUBROUTINE SHOWTOTALS       | 1115      | 1149     | 35         | 1                 |
| SUBROUTINE SORTANDDISPLA    | 272       | 279      | 8          | 1                 |
| SUBROUTINE SWITCH           | 657       | 689      | 33         | 2                 |
| SUBROUTINE STOREDATA        | 1429      | 1482     | 54         | 1                 |
| SUBROUTINE SUM              | 1614      | 1700     | 87         | 1                 |
| SUBROUTINE SUMEARN          | 1150      | 1157     | 8          | 1                 |
| SUBROUTINE SWITCH           | 378       | 396      | 19         | 1                 |
| SUBROUTINE TRUCKHOURS       |           |          |            |                   |
| TOTALS FOR 44 PROGRAM UNITS |           | 1741     | 65         |                   |

LISTING OF PROGRAM UNITS IN ALPHABETICAL ORDER WITH STATISTICAL SUMMARY FOR SOURCE CODE 1 RESULTS

|                              |           |
|------------------------------|-----------|
| LARGEST PROGRAM UNIT SIZE =  | 116 LINES |
| SMALLEST PROGRAM UNIT SIZE = | 8 LINES   |
| AVERAGE PROGRAM UNIT SIZE =  | 40 LINES  |
| % COMMENT LINES              | 7         |
| % DECLARATION LINES          | 17        |
| % FORMAT LINES               | 3         |
| % EXECUTABLE LINES           | 73        |
| END OF PROGRAM               |           |

## BIBLIOGRAPHY

1. Fosdick, L.D., 'BRNANL, a Fortran program to identify basic blocks in Fortran programs', Report CU-CS040-74 Department of CS, University of Colorado.
2. Ryder, Barbara G., 'Constructing the Call Graph of a Program', IEEE Transactions on Software Engineering, Vol. 5, No. 3, May, 1979, pp 216-226.
3. Ryder, Barbara G., 'The PFORT Verifier', Software Practice and Experience, Vol. 4 pp 359-377, Oct.-Dec. 1974.

AN IMPLEMENTATION OF A FORTRAN SOURCE CODE REARRANGER AND  
DOCUMENTATION GENERATOR PROGRAM

by

ARTHUR SCOTT MYERS

B.S., Lehigh University, 1969

---

AN ABSTRACT OF A MASTER'S REPORT

submitted in partial fulfillment of the

requirements of the degree

MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY  
Manhattan, Kansas

1983

With the explosive growth in the size of software, the analysis and documentation of module relationships for programs written in high level programming languages are undoubtedly useful and needed capabilities. This paper describes an implementation project for processing the source code of FORTRAN programs which produces such an analysis and documentation. The system that resulted from this implementation analyzes and rearranges a complete FORTRAN source code file into its hierarchical top-down order of modules. This system also provides two reports, which contain relevant information about each module's structure and size. The first report effectively shows the hierarchical arrangement of modules and can be thought of as a table of contents for the just rearranged program; the second report lists alphabetically by name the program's modules and therefore is the index. The paper discusses the requirements for implementing this project, the design decisions, and the actual implementation. Future work and extensions pertaining to this project are also considered.