AD-A067 203 CONSTRUCTION ENGINEERING RESEARCH LAB (ARMY) CHAMPAI--ETC F/G 9/2
THE AUTOMATED DOCUMENTATION SYSTEM--USER MANUAL.(U)
FEB 79 L LAWRIE

UNCLASSIFIED CERL-TR-E-147 NL

|OF|
AD
A067203

END
DATE
FILMED
6-79
DDC

construction engineering research laboratory

United States Army Corps of Engineers
...Serving the Army
...Serving the Nation

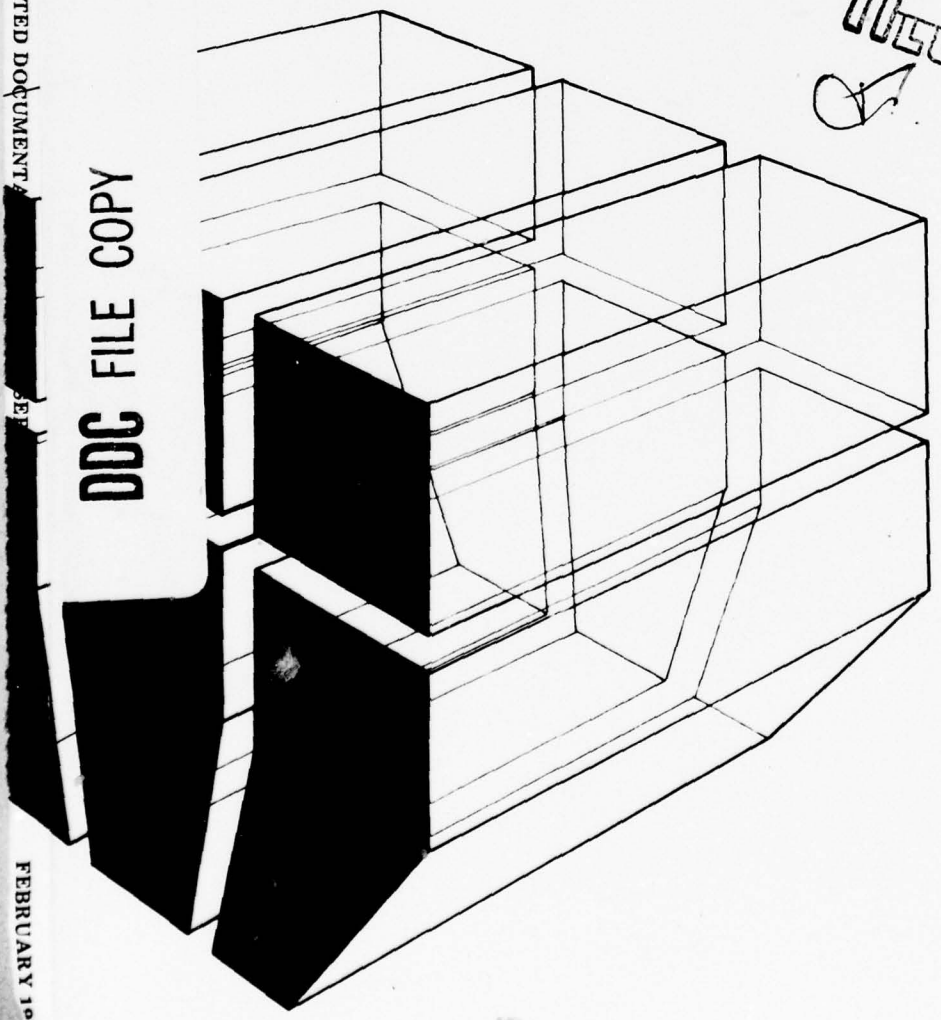
TECHNICAL REPORT E-147
February 1979

12

LEVEL II

THE AUTOMATED DOCUMENTATION SYSTEM—USER MANUAL

by
Linda Lawrie



CERL

CERL-TR-E-147
THE AUTOMATED DOCUMENTATION SYSTEM—USER MANUAL
FEBRUARY 1979

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: CERL-TR-E-147
2. GOVT ACCESSION NO.:
3. RECIPIENT'S CATALOG NUMBER:
4. TITLE (and Subtitle): THE AUTOMATED DOCUMENTATION SYSTEM--USER MANUAL
5. TYPE OF REPORT & PERIOD COVERED: FINAL rept.
6. PERFORMING ORG. REPORT NUMBER:
7. AUTHOR(s): Linda Lawrie
8. CONTRACT OR GRANT NUMBER(s):
9. PERFORMING ORGANIZATION NAME AND ADDRESS:
U.S. Army
CONSTRUCTION ENGINEERING RESEARCH LABORATORY
P.O. Box 4005, Champaign, IL 61820
10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 4A762725AT11-02 P2
11. CONTROLLING OFFICE NAME AND ADDRESS:
12. REPORT DATE: February 1979
13. NUMBER OF PAGES: 84
14. MONITORING AGENCY NAME & ADDRESS *(If different from Controlling Office)*: 87p.
15. SECURITY CLASS. *(of this report)*: Unclassified
15a. DECLASSIFICATION/DOWNGRADING SCHEDULE:

16. DISTRIBUTION STATEMENT *(of this Report)*
Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES
Copies are obtainable from National Technical Information Service
Springfield, VA 22151

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*
software development
FORTRAN
computer documentation

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

The Automated Documentation System (ADS) is a computer program and user procedure designed to facilitate management of the development of software and the production of final documentation for FORTRAN programs. The ADS system can be used in two ways.

(1) At any point during development of the software, the status of the development process can be determined by application of the program

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

405 279

UNCLASSIFIED

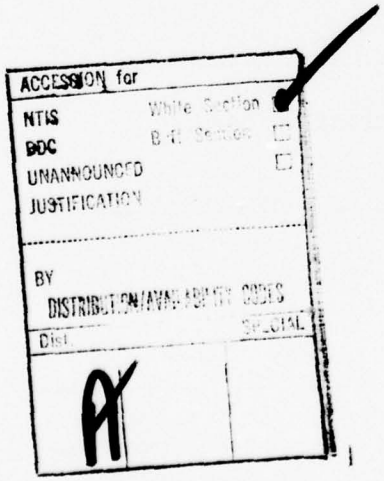
SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

Block 20 continued.

to the source code under development. Flow charts and internal documentation are summarized for the program manager.

(2) After the software is complete, external documentation can be produced from the internal documentation and compilation maps by running the ADS program.

The ADS program is written in Control Data Corporation (CDC) FORTRAN extended, version 4.6 and can be used on CDC 6000/7000 series computers with few or no modifications. This report provides detailed user instructions for ADS.



FOREWORD

This research was conducted by the Energy and Habitability Division (EH) of the U.S. Army Construction Engineering Research Laboratory (CERL) for the Engineering and Scientific Division of the Engineering and Data Systems Office, Department of the Army, under RDT&E Program 6.27.25A, Project 4A762725AT11, "Engineering Software Development," Task 02, "Comprehensive Standard for Software Development."

Mr. Gene Manning was the Technical Monitor. Mr. Douglas C. Hittle was the CERL principal investigator. Administrative support was provided by Dr. D. J. Leverenz and Mr. R. G. Donaghy, Chief of EH.

The ADS program was authored by Ms. L. Lawrie, Mr. R. Lidral, Mr. D. Herron, Mr. K. Morgan, Mr. A. Itzkowitz, and Mr. D. Burn; background to the ADS system was provided by Mr. W. Struebing. Appreciation is expressed to M. L. Scala for her help in writing this report.

COL J. E. Hays is Commander and Director of CERL, and Dr. L. R. Shaffer is Technical Director.

CONTENTS

TABLE

FIGURES

THE AUTOMATED DOCUMENTATION
SYSTEM -- USER MANUAL

1 INTRODUCTION

Good documentation can appreciably extend the useful life of a software tool and increase its efficiency by allowing individuals not involved in its initial development to use it effectively. There are, however, no universal standards for good documentation. Indeed, what is effective documentation for one type of software may be completely insufficient to the needs of another type of software. In addition, programmers are generally unable or unwilling to prepare documentation until after completion of the final development of software tools -- long after details intrinsic to the development process are forgotten. As a result, it is almost impossible for software managers to obtain accurate, up-to-date reports on the ongoing development process.

The Automated Documentation System (ADS) was developed to provide high-quality documentation while simplifying the jobs of both the program developer and the software manager. In effect, ADS becomes part of the software tool, thereby allowing internal and external documentation to proceed simultaneously with software development. ADS can create documentation tailored to the unique needs of a software tool, relieve programmers of the tedious task of "after-the-fact" documentation, and provide software managers with an effective method of supervising the software development process.

Objective

The overall objective of this study was to provide a comprehensive standard for software development, and to provide user instructions for the ADS method of producing internal and external documentation for software projects.

Outline of Report

Chapter 2 contains an overview of ADS; Chapter 3 describes ADS input format code; Chapter 4 details ADS program control, ADS commands, and various options available with ADS; and Chapter 5 outlines ADS job control and data files used by ADS.

Mode of Technology Transfer

ADS will be available from Boeing Computer Services Company, under the U.S. Army Corps of Engineers contract for scientific and engineering teleprocessing.

2 ADS OVERVIEW

Background

Essentially, ADS is an organizational tool -- a program for reporting on the development of another program. Its function is that of the journalist/librarian of a program under development: first, it gathers information supplied to it by the programmer and computer; it then files, organizes, and cross-references this information. Finally, it summarizes the information in a report.

At present, ADS can only be used to document FORTRAN programs, and will only interface with compiler reference maps produced by the Control Data Corporation (CDC).

This chapter presents an overview of ADS -- how it works and how it can be manipulated by the user. Although ADS is itself a program, to avoid confusion the terms "program," "subprogram," "routine," "source code," etc., will refer only to the program under development and being documented by ADS.

Using ADS on a program under development is easy. The basic requirement is that ADS comment cards be implanted in the FORTRAN source code of the program being documented by ADS. Thus, ADS allows documentation to proceed simultaneously with software development.

Beginning the Documentation Process With ADS

ADS initially requires three types of input (see Figure 1):

1. All or part of the source code implanted with ADS comment cards

2. ADS commands

3. The compiler reference map for all or part of the program under development.

The source code is the actual FORTRAN code of the program being documented by ADS; ADS can use all or part of this code. ADS commands are user instructions to ADS which tell ADS what to do with other inputs and what reports to produce.

The third ADS input, the compiler reference map, is part of the output produced by the FORTRAN compiler on CDC computers and usually follows the FORTRAN source code listing also produced by the compiler. The compiler reference map is used by ADS because it is, in effect, an

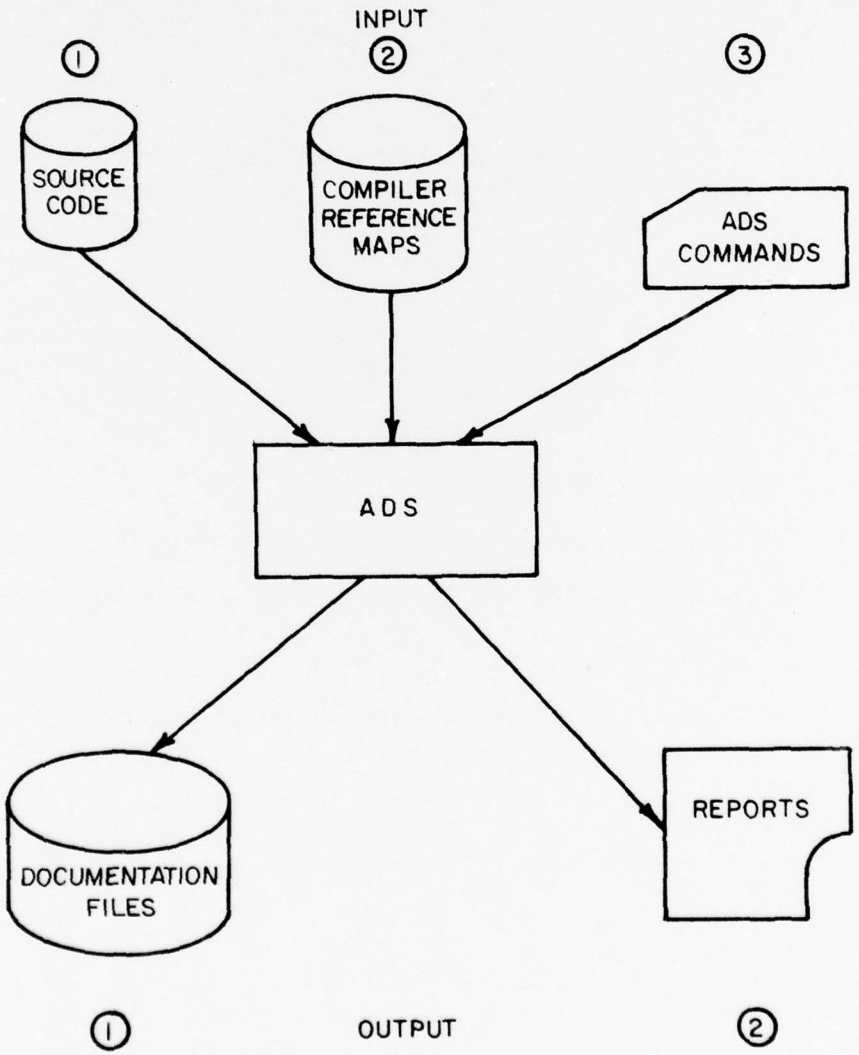


Figure 1. ADS input and output.

index of the program (subroutine, common block) under development. It includes information pertinent to the program, such as:

1. Variable names

2. Routines called within the program or subprogram

3. Common blocks referenced within the program or subprogram

4. Entry points for the program or subprogram

5. Files referenced by the program or subprogram

6. Inline functions used by the program or subprogram

7. Equivalence classes for variables

8. The length of the program, subprogram, common blocks, arrays, etc.

(Note: FORTRAN programs or subroutines must be compiled [with or without errors] before ADS can be used to document them.)

The compiler reference map not only provides information useful for the documentation of the source code of the program under development, but provides a means by which ADS can verify the comprehensiveness and accuracy of the ADS comment cards implanted in the program source code by the user.

ADS produces two types of output from the input described above:

1. External documentation reports

2. Internal documentation files.

External documentation reports are the program documentation produced by ADS; their content depends upon the commands given ADS by the user. These commands are detailed in Chapter 4. Documentation files are stored computer files and contain data regarding the program under development that is being documented by ADS. These data are derived from ADS comment cards and the compiler reference map. Since documentation files are stored online in the computer, they can be retained indefinitely and used to create reports and/or new documentation files as the program is refined and extended. The only requirement of the programmers is that they exercise ADS frequently enough to keep the documentation files current with the software development activity.

Expanding and Updating Documentation

ADS is intended for use during the initial as well as intermediate and final stages of software development. Since ADS is designed to update and expand its documentation, it can be implemented even in the "skeletal" phase of program development, i.e., when a program consists of routines yet to be fully developed, routines which though largely developed will require revision, as well as routines in final form. Updates and expansion of ADS documentation do not require the programmer to redo documentation input for software elements that do not change.

When in the update mode, ADS uses the source code and compiler reference map for new and revised portions of the program under development, as well as information from the documentation files pertaining to software elements which do not change. The source code, reference map, and "old" documentation files are input to ADS to produce new documentation files and/or new external documentation reports.

This process should be repeated as often as necessary to assure up-to-date documentation of the program. The process is usually inexpensive since data relating to unchanged software elements can pass without manipulation from old to new ADS documentation files. The user and ADS need only process data relating to new or revised portions of the program such as new subroutines, changed common blocks, etc. (See Figure 2.)

Documentation Output

External documentation reports for the program under development can be produced by running ADS using the documentation files without referring to either the source code or the compiler reference map (see Figure 3). The most recent documentation files are used by ADS to produce the report in the form dictated to it by the user in the ADS commands. This option of retrieving documentation from ADS at any time is especially useful to software project managers and team leaders because it allows them to monitor the progress of a software development project without disturbing the program developers.

Example

Figures 4, 5, and 6 show the FORTRAN source code for a main program, the compiler reference map for that program, and the ADS commands which will activate the ADS documentation files and write external reports, respectively. Figure 7 is a sample of a report produced by ADS. Figures 8, 9, and 10 are the source code, reference map, and ADS commands, respectively, necessary to update documentation by adding one

subroutine. Figure 11 is the ADS commands which are used only for producing external documentation reports. Finally, Figure 12 is a sample of a second external documentation report option available from ADS -- the static calling structure or tree diagram of the program under development.

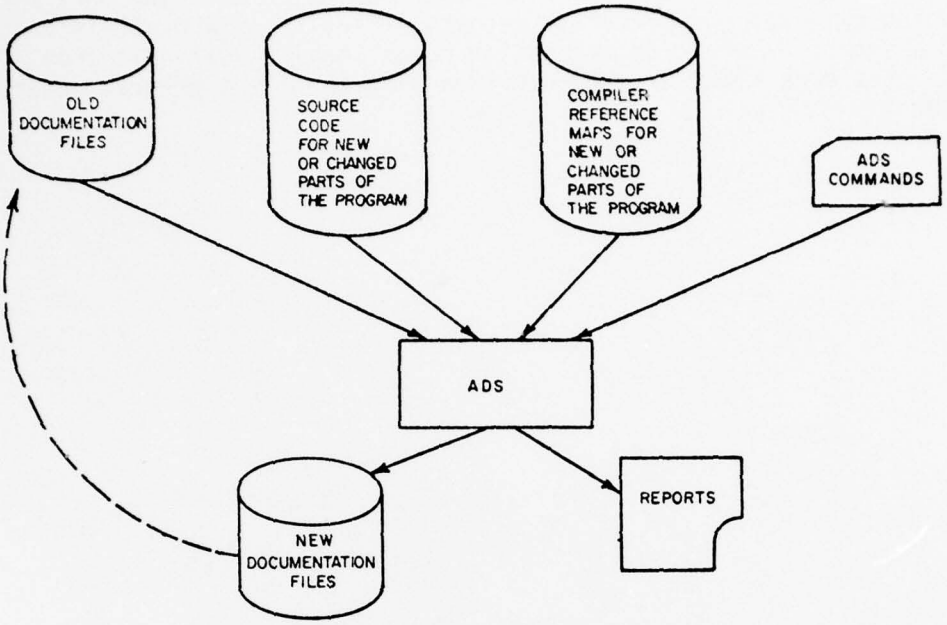


Figure 2. Expansion and update of ADS documentation.

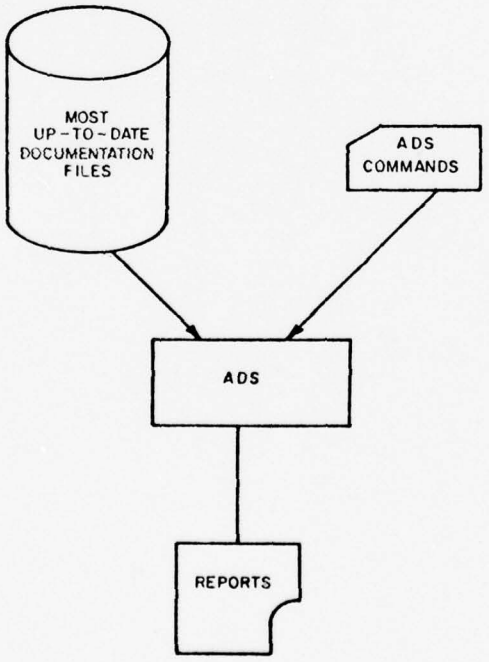


Figure 3. Production of reports directly from documentation files.

```
      PROGRAM MAIN(INPUT,OUTPUT)
CD    TITLE:=
CD    MAIN - MAIN PROGRAM OF MANUAL EXAMPLE
CD    AUTHOR:= THIS SYSTEM PROGRAMMER
CD    PURPOSE:=
CD    THE PURPOSE OF THIS PROGRAM IS TO ILLUSTRATE HOW A PROGRAM UNDER
CD    DEVELOPMENT MIGHT UTILIZE ADS.  THE PROGRAM IS OUTLINES BY THE
CD    MAIN PROGRAM.  THEN AS OTHER ROUTINES ARE DEVELOPED THEY WILL BE
CD    ADDED TO THE DOCUMENTATION FILES.
C
      CALL SUB1
C
      CALL SUB2
C
      CALL SUB3
C
      STOP
      END
```

Figure 4. FORTRAN source code for a main program.

```
PROGRAM MAIN     7600_7600 OPT=1


 1                    PROGRAM MAIN(INPUT,OUTPUT)
                CD    TITLE:=
                CD    MAIN - MAIN PROGRAM OF MANUAL EXAMPLE
                CD    AUTHOR:= THIS SYSTEM PROGRAMMER
 5              CD    PURPOSE:=
                CD    THE PURPOSE OF THIS PROGRAM IS TO ILLUSTRATE HOW A PROGRAM UNDER
                CD    DEVELOPMENT MIGHT UTILIZE ADS.  THE PROGRAM IS OUTLINES BY THE
                CD    MAIN PROGRAM.  THEN AS OTHER ROUTINES ARE DEVELOPED THEY WILL BE
                CD    ADDED TO THE DOCUMENTATION FILES.
10              C
                      CALL SUB1
                C
                      CALL SUB2
                C
15                    CALL SUB3
                C
                      STOP
                      END


      SYMBOLIC REFERENCE MAP (R=3)

ENTRY POINTS     DEF LINE     REFERENCES
 2106  MAIN           1

FILE NAMES        MODE
    0  INPUT
 1041  OUTPUT

EXTERNALS          TYPE   ARGS       REFERENCES
       SUB1                 0            11
       SUB2                 0            13
       SUB3                 0            15

STATISTICS
  PROGRAM LENGTH                78         7
  BUFFER LENGTH              21028      1090
```

Figure 5. Compiler reference map for program in Figure 4.

```
25 JUL 78      11.55.54      CERL - ADS - MESSAGE OUTPUT          PAGE    1

   TITLE THE ADS USER MANUAL EXAMPLE;
   PRINT NARROW (DUMP) FOR MAIN;
   END;
```

Figure 6. Sample ADS commands.

```
25 JUL 78    11.55.54       CERL - ADS - VERSION  1.0          PAGE    1
                    THE ADS USER MANUAL EXAMPLE
DUMP                                              AS OF 25 JUL 78-MAIN


TITLE.
MAIN - MAIN PROGRAM OF MANUAL EXAMPLE

AUTHOR.
THIS SYSTEM PROGRAMMER

PURPOSE.
THE PURPOSE OF THIS PROGRAM IS TO ILLUSTRATE HOW A PROGRAM UNDER
DEVELOPMENT MIGHT UTILIZE ADS.  THE PROGRAM IS OUTLINES BY THE
MAIN PROGRAM.  THEN AS OTHER ROUTINES ARE DEVELOPED THEY WILL BE
ADDED TO THE DOCUMENTATION FILES.

     FILES USED IN MAIN    ARE --
INPUT      OUTPUT

VARIABLE DICTIONARY FOR ROUTINE MAIN    -- ** NONE **

THIS A MAIN PROGRAM OF LENGTH        7 WORDS.

     ROUTINES CALLED BY MAIN    ARE --
SUB1       SUB2       SUB3

     COMMON BLOCKS CALLED BY MAIN    ARE -- ** NONE **

     THE ROUTINES WHICH CALL MAIN    ARE -- ** NONE **
```

Figure 7. Sample ADS report.

```
      SUBROUTINE SUB1
CD    TITLE:=
CD    SUB1 - THE FIRST SUBROUTINE OF THE PROGRAM TO BE IMPLEMENTED.
CD    AUTHOR:= THIS SYSTEM PROGRAMMER(S).
CD    PURPOSE:=
CD    THE PURPOSE OF THIS ROUTINES IS TO SHOW HOW ONE MIGHT ADD A
CD    ROUTINE TO A DOCUMENTATION FILE UNDER DEVELOPMENT.
C
      CALL SUBRTE
      RETURN
      END
```

Figure 8. Sample of source code for a subroutine.

```
 SUBROUTINE SUB1     7600_7600 OPT=1

 1                   SUBROUTINE SUB1
                CD   TITLE:=
                CD   SUB1 - THE FIRST SUBROUTINE OF THE PROGRAM TO BE IMPLEMENTED.
                CD   AUTHOR:= THIS SYSTEM PROGRAMMER(S).
 5              CD   PURPOSE:=
                CD   THE PURPOSE OF THIS ROUTINES IS TO SHOW HOW ONE MIGHT ADD A
                CD   ROUTINE TO A DOCUMENTATION FILE UNDER DEVELOPMENT.
                C
                     CALL SUBRTE
10                   RETURN
                     END


      SYMBOLIC REFERENCE MAP (R=3)

ENTRY POINTS     DEF LINE     REFERENCES
    1  SUB1            1         10

EXTERNALS          TYPE   ARGS       REFERENCES
      SUBRTF                0              9

STATISTICS
  PROGRAM LENGTH                  4B        4
```

Figure 9. Sample of reference map for a subroutine.

```
PRINT NARROW (DUMP) FOR UPDATED ROUTINE;
END;
```

Figure 10. Sample ADS commands to update documentation by adding subroutine.

```
                    CERL - ADS - MESSAGE OUTPUT

   PRINT NARROW (DUMP) FOR ALL ROUTINES;
** WARNING   REQUESTED READ OF RECORD (ROUTINE) -- SUB2   . RECORD NOT ON MASTER FILE
** WARNING   REQUESTED READ OF RECORD (ROUTINE) -- SUB3   . RECORD NOT ON MASTER FILE
** WARNING   REQUESTED READ OF RECORD (ROUTINE) -- SUBRTE . RECORD NOT ON MASTER FILE
   DRAW NARROW FULL TREE;
   END;
```

Figure 11. Sample ADS commands used only for production of external documentation reports.

```
                              CERL - ADS - VERSION  1.0
                          THE ADS USER MANUAL EXAMPLE

TREE

    MAIN___
           !_SUB1___
           !        !*SUBRTF
           !*SUB2
           !*SUB3
```

Figure 12. Tree diagram of program under development.

3 ADS SOURCE CODE INPUT -- FORMAT

There are 19 documentation categories in which a programmer may store information regarding a program under development. These categories are listed in Table 1; detailed explanations of each category are found in Appendix A.

Special ADS comment cards ("CD" in columns 1 and 2 of the source code) are used to insert documentation regarding a program under development to ADS. To enter an ADS documentation comment (1) type "CD" in columns 1 and 2 of the source code, (2) type the title of the documentation category selected from the 19 provided by ADS, (3) type ":=", (4) type "CD" in columns 1 and 2 of the source code, followed by the first line of the comment, and (5) repeat step 4 until the comment is complete.

For example, the format for an ADS comment on the purpose of a program is

```
CD  PURPOSE:=
CD  THE PURPOSE OF THIS PROGRAM IS TO GENERATE MOVE DIRECTIVES TO
CD  SORT OLDPL -- UPDATE LIBRARY.  UP TO THREE LEVELS OF DECKS
CD  CAN BE EXCLUDED FROM THE TOTAL SORT (AND THEN ARE SORTED AMONGST
CD  THEMSELVES).  COMDECKS ARE SORTED SEPARATELY FROM MAIN DECKS.  EACH
CD  LEVEL MUST ALSO INPUT A DECK NAME THAT ALL DECKS IN THE LEVEL WILL
CD  FOLLOW -- THE DIRECTIVE WILL BE OF FORM:
```

The above will appear in the external documentation report as:

```
PURPOSE.
THE PURPOSE OF THIS PROGRAM IS TO GENERATE MOVE DIRECTIVES TO
SORT OLDPL -- UPDATE LIBRARY.  UP TO THREE LEVELS OF DECKS CAN
BE EXCLUDED FROM THE TOTAL SORT (AND THEN ARE SORTED AMONGST
THEMSELVES).  COMDECKS ARE SORTED SEPARATELY FROM MAIN DECKS.
EACH LEVEL MUST ALSO INPUT A DECK NAME THAT ALL DECKS IN THE
LEVEL WILL FOLLOW -- THE DIRECTIVE WILL BE OF FORM:
```

The user can control the paragraphing of the comment text by inserting one or all of the following flags in the source code:

CD$_	Begin new line at left margin
CD$	Begin new line; retain tab setting indicated in preceding line
CD	No new line; line may be run on with preceding line
CD$__	Begin new line; reset tab to indent 5 spaces

Table 1

ADS Documentation Categories

TITLE	FLOW
AUTHOR	FILES
DATE WRITTEN	ALGORITHM
REFERENCES	SYSTEM DEPENDENCIES
LOCATION	NON SYSTEM DEPENDENCIES
METHOD	MACHINE DEPENDENCIES
CONTROL CARDS	IMPLEMENTATION DEPENDENCIES
REMARKS	VARIABLE DICTIONARY
SYSTEM	REVISED (date)
PURPOSE	

CD$___ Begin new line; reset tab to indent 10 spaces.

CD$____ Begin new line; reset tab to indent 15 spaces.

For an example of a program source code implanted with ADS comments (documentation category, initial comments, and subsequent paragraphing flags) and the resultant output (external documentation reports), see Figures 13 and 14.

Common Block Input

ADS comments regarding common blocks within a program are strictly limited to two ADS documentation categories: TITLE and VARIABLE DICTIONARY. In addition, ADS comments for common blocks must appear contiguously in the source code. For example:

```
      COMMON/COM1/VAR1,VAR2
CD    TITLE:=
CD    COM1 - TITLE FOR COM1
CD    VARIABLE DICTIONARY:=
CD    VAR1 - DEFINITION OF VAR1
CD    VAR2 - DEFINITION OF VAR2
```

Any attempt to enter ADS comments other than TITLE or VARIABLE DICTIONARY, or to type comments noncontiguously, will probably result in the loss of that common block documentation.

Variable Dictionary Input

Variable dictionary comments (under the heading "VARIABLE DICTIONARY:=") must be definitions for variable names already in the source code. To enter a variable dictionary comment, (1) type "CD" in columns 1 and 2 of the source code followed by "VARIABLE DICTIONARY:=", (2) type "CD" in columns 1 and 2 of the source code, followed by the variable name to be defined and "-" and (3) type the definition.

For example, a variable dictionary comment implanted in the source code as

```
CD    VARIABLE DICTIONARY:=
CD    FLAGV - FLAG VARIABLE
CD$   1--> INPUT IS FROM DISK FILE
CD$   2--> INPUT IS FROM CARDS
```

will appear in the external report as

```
CD      TITLE:=
CD      RDIT - READ A SET OF DECKS TO EXCLUDE FROM GENERAL SORTING
CD      PURPOSE:=
CD      THE PURPOSE OF THIS PROGRAM IS TO GENERATE MOVE DIRECTIVES TO
CD      SORT AN OLDPL -- UPDATE LIBRARY.  UP TO THREE LEVELS OF
CD      DECKS CAN BE EXCLUDED FROM THE TOTAL SORT (AND ARE THEN
CD      SORTED AMONGST THEMSELVES).  COMDECKS ARE SORTED SEPARATELY
CD      FROM MAIN DECKS.  EACH LEVEL MUST ALSO INPUT A DECK NAME
CD      THAT ALL DECKS IN THE LEVEL WILL FOLLOW -- THE DIRECTIVE
CD       WILL BE OF FORM:
CD$     "*MOVE <DECK NAME>, <INPUT DECK NAME>"
CD$     TO LEVEL DECKS ONE PUTS IN CARDS OF FORM:
CD$      COL 1--9.  NAME OF EXCLUDED DECK
CD$     COL. 10  BLANK, 1, OR 2
CD$     DECK NAMES ARE SORTED AND MOVE DIRECTIVES GENERATED
CD      SUCH THAT THE FINAL ORDER OF THE UPDATE OLDPL WILL
CD      BE:
CD$        COMDECKS,
CD$     DECKS (INPUT CARD NAMES) WITH COL 10 = 2,
CD$     DECKS (INPUT CARD NAMES) WITH COL 10 = 1,
CD$     DECKS (NOT COMDECKS) NOT SPECIFIED ON INPUT CARDS.
CD$     NOTE -- DECKS (INPUT CARD NAMES) WITH COL 10 = BLANK
CD      ARE NOT SORTED!  THESE DECKS CAN REMAIN IN PRESENT PLACES
CD      IN THE OLDPL AND BE USED FOR POSITIONING OTHER DECKS.
CD$      THE PROGRAM USES THE BASIC OUTPUT (L=A14) FROM THE UPDATE
CD      PROGRAM AS INPUT (TAPE1).
```

Figure 13. FORTRAN source code implanted with ADS comments.

```
TITLE.
RDIT - READ A SET OF DECKS TO EXCLUDE FROM GENERAL SORTING
PURPOSE.
THE PURPOSE OF THIS PROGRAM IS TO GENERATE MOVE
DIRECTIVES TO SORT OLDPL -- UPDATE LIBRARY.  UP TO
THREE LEVELS OF DECKS CAN BE EXCLUDED FROM THE
TOTAL SORT (AND ARE THEN SORTED AMONGST THEMSELVES).
COMDECKS ARE SORTED SEPARATELY FROM MAIN DECKS.
EACH LEVEL MUST ALSO INPUT A DECK NAME THAT ALL
DECKS IN THE LEVEL WILL FOLLOW -- THE DIRECTIVE
WILL BE OF FORM:

    "*MOVE <DECK NAME>, <INPUT DECK NAME>"
TO LEVEL DECKS ONE PUTS IN CARDS OF FORM:
    COL 1 -- 9 . NAME OF EXCLUDED DECK
    COL. 10  BLANK 1 OR 2
DECK NAMES ARE SORTED AND MOVE DIRECTIVES
GENERATED SUCH THAT THE FINAL ORDER OF THE
UPDATE OLDPL WILL BE:
    COMDECKS,
    DECKS (INPUT CARD NAMES) WITH COL 10 = 2,
    DECKS (INPUT CARD NAMES) WITH COL 10 = 1,
    DECKS (NOT COMDECKS) NOT SPECIFIED ON INPUT CARDS.
    NOTE -- DECKS (INPUT CARD NAMES) WITH
    COL 10 = BLANK ARE NOT SORTED! THESE
    DECKS CAN REMAIN IN PRESENT PLACES
    IN THE OLDPL AND BE USED FOR
    POSITIONING OTHER DECKS.
THE PROGRAM USES THE BASIC OUTPUT (L = A14)
FROM THE UPDATE PROGRAM AS INPUT (TAPE1).
```

Figure 14. ADS Output from Figure 13.

```
VARIABLE DICTIONARY.
FLAGV - FLAG VARIABLE
          1 --> INPUT IS FROM DISK FILE
          2 --> INPUT IS FROM CARDS
```

(Note that variable dictionary comments containing the variable name are indented ten spaces. The "-" typed by the user following the variable name has the effect of indenting the tab setting 10 spaces to the right of the left margin -- however this indenting is not under user control).

4 ADS PROGRAM CONTROL AND OPTIONS

This chapter further defines the syntax of ADS commands and provides examples of various report retrieval options. Words in capital letters must appear as written. Words inclosed in angle brackets (< >) must be supplied by the user from an optional list. Symbols not inclosed in brackets ("-", ";", ",", etc.) must be included as written. COMMON(S) indicates that either COMMON or COMMONS may be used.

External Reports: Filing and Retrieval

The contents of documentation reports generated by ADS may be defined by the user. These report names (and definitions) are stored on the documentation files and may be retrieved, by name, when producing a report (via the PRINT command).

ADS provides two predefined report names on each documentation file: ABSTRACT and DUMP. ABSTRACT provides the ADS documentation categories: TITLE, AUTHOR, PURPOSE, METHOD, ALGORITHM, and REFERENCES. DUMP, on the other hand, is a comprehensive listing of all possible ADS documentation stored for a program or common block.

To define a report, the user types:

REPORT <name> = <section list>;

<Name> is user supplied and must be less than or equal to 10 typed characters. <Section list> is the documentation categories listed in the order in which they are to be printed in the external report; each documentation category is separated from the one preceding it by a comma. The final documentation category typed by the user is followed by ";". For example:

REPORT ABSTRACT = TITLE,AUTHOR,PURPOSE,METHOD,ALGORITHM,REFERENCES;

(NOTE: The above are the elements included in the ADS ABSTRACT report.)

External documentation reports produced by ADS can be used by programmers as final documentation, progress reports, or as information for other programmers. Also, by attaching the proper documentation files, individuals other than the original programmer are able to produce reports on the program(s) under development.

Report names are included in the documentation files and may be defined for each program's documentation files and retrieved, by name, when producing a report.

Additional ADS Documentation Output

In addition to the 19 documentation categories already described (see Table 1), four other types of ADS output can be requested by the user for inclusion in an external documentation report. They are: ENTRY POINTS, ROUTINES CALLED, COMMON BLOCKS CALLED, and CALLED BY.

ENTRY POINTS gives general information about the program: the entry point names (if more than one); the date documentation was entered onto the documentation files; the type of routine (PROGRAM, SUBROUTINE, FUNCTION); and the routine's length. ROUTINES CALLED specifies the routines called by the program plus the inline functions used by the program. COMMON BLOCKS CALLED lists the common blocks called by the program. A maximum of 100 possible items is assumed for both ROUTINES CALLED and COMMON BLOCKS CALLED. CALLED BY gives the names of the routines and common blocks calling the program under development. In addition, CALLED BY lists which submodules use the files of the main program.

Delete Option

Another option of the ADS program input is the ability to delete information from the documentation files about a particular routine, common block, or report name. The format for this command is:

DELETE <record type> (<name list>);

<Record type> requires the user to type one of the following: ALL, ROUTINE, COMMON, or REPORT. <Name list> specifies the routines common blocks, etc., to be deleted of type <record type>. ALL will delete all types of the names in <name list>. For example, "DELETE ROUTINE (MAIN, SUB1);" deletes all references to routines named MAIN and SUB1. "DELETE REPORT (ABSTRACT);" deletes all references to report ABSTRACT. "DELETE ALL (MAIN, COM1);" searches routines, commonblocks, and report lists and deletes all references to MAIN and COM1 entries.

Print Option

To print reports from the documentation file, the user types:

PRINT <report width> <paging> (<report list>) FOR <record list>;

For <report width> the user types either NARROW or WIDE. NARROW reports are suitable for 8 x 10-1/2-in. reproduction. WIDE reports use the full width of the printer paper (14 x 11 in.). <Paging> is one of PAGED or UNPAGED. PAGED reports (default) start each new item in <record list>

on a single page. Reports are also paged within records so a report will be spaced properly over page boundaries. <Report list> requires the user to type the list report names, separated by commas, to be printed for each occurrence of a record in <record list>. When typing <record list> the user may choose one of the following: (<name list>), ALL ROUTINE(S), ALL COMMON(S), UPDATED ROUTINE(S), or UPDATED COMMON(S). The <name list> specifies a set of routines, common blocks, etc. (separated by commas) for which a specified external report will be produced. The ALL prefix prints reports for all routines or common blocks presently on the documentation files. Additionally, warning errors are printed for each report not yet on documentation files. The UPDATED prefix will print reports for each routine or common block included on the documentation files for this run. Thus, ADS can update the documentation files and produce reports in a single run. For each occurrence of a "report name" in <record list>, a report detailing that "report name" will be produced, showing the ADS documentation categories in the order indicated by the "report name". For example, "PRINT NARROW (DUMP) FOR ALL ROUTINES;" produces a DUMP report for all routines defined on the documentation files. "PRINT NARROW (DUMP) FOR (ABSTRACT, DUMP);" produces a REPORT DEFINE report for the two reports ABSTRACT and DUMP. This command will also produce a DUMP report for any routines or common blocks with names ABSTRACT or DUMP.

Tree Option

The "tree" option produces a diagram of the routines on the documentation files of the program under development by using the cross-referencing information available for each routine. The format is fixed by ADS and therefore cannot be modified by the user. The format for requesting the tree option from ADS is:

DRAW <tree type> <report width> TREE <routine> <depth>;

For <tree type>, the user types either FULL or COMPRESSED. For <report width>, the user types either NARROW or WIDE. <Routine> is optional; if <routine> is included, ADS will use <routine> as the root of the tree. If <routine> is not included, ADS will use the main program as the root. If more than one main program resides on the documentation files, a tree will be drawn for each main program. <Depth> specifies the number of levels to be included in the tree. (If <depth> is omitted, full depth is assumed.) A FULL tree includes all routines indicated by the cross-referencing; a COMPRESSED tree does not include each routine of a previously drawn branch. For example, "DRAW FULL WIDE TREE ADSDOC ;" should draw a tree the entire ADS program. However, "DRAW COMPRESSED NARROW TREE ADSRPT 3 ;" will draw only the subtree at routine ADSRPT for three levels of subroutine calls.

The tree option displays the basic calling sequence of the program. A FULL tree shows each routine called at each level. For example, "DRAW FULL NARROW TREE MAIN;" might produce:

```
MAIN
    |_SUB3
    |    |_SUB1
    |    |    |_TIMER
    |    |    |    |_*SECOND
    |_SUB2
    |    |_TIMER
    |    |    |_*SECOND
    |_SUB1
         |_TIMER
              |_*SECOND
```

whereas "DRAW COMPRESSED NARROW TREE MAIN;" would produce

```
MAIN
    |_SUB3
    |    |_SUB1
    |    |    |_TIMER
    |    |    |    |_*SECOND
    |_SUB2
    |    |_TIMER*
    |_SUB1*
```

An "*" before a name indicates the routine is not on the master documentation files. An "*" following a name indicates that the branch has already been displayed.

Other Options

The user also has the option of assigning a title to the documentation files. This title will be printed on the heading of each report and stored in the documentation files. (Additional headings on each report page will contain a page number, the date and time of the run, the report name, record identifer, and date of the record on the documentation files.) To assign a system title, the user types:

TITLE characters;

Up to 80 typed characters are allowed for the title. Once the title is stored on the documentation files it need not be input again. For example:

TITLE THE AUTOMATED DOCUMENTATION SYSTEM (ADS);

The ADS program will accept and process each statement input to it. Processing of the input begins when ADS finds the symbol indicating the end of the statement, i.e., ";". If an error occurs during the parsing

of the statement, ADS skips until it locates the semicolon, effectively ignoring the statement. ADS first processes any reference maps and source code the user has attached before beginning production of an external report.

To terminate the ADS program the user types:

END;

ADS can be run either as a batch or interactive program. ADS reports and various informative messages are written to separate local files. The reports are formatted for printers; however ADS informative messages can be received either on printer or interactive terminal output.

For interactive usage (though, of course, batch runs will accept these statements), three additional statements are allowed:

HELP;

ECHO; (default)

NOECHO;

ECHO and NOECHO turn on and off echoing user input to the ADS program. HELP provides some information about the form of the input statements.

During the execution of ADS, error messages may be printed. These error messages inform the user of invalidities in the input, in source code processing, and in documentation files processing. For example, if the user inputs "PRONT NARROW (DUMP) FOR MAIN;" ADS will respond: "**** SEVERE UNKNOWN COMMAND" and continue to the next input command. If, in the source code, a routine does not have an "end" card, "SEVERE MISSING END CARD, RESTART WITH FOLLOWING CARD" will appear. In the next output line, the card restarting source code parsing will be printed. If a routine is referenced (i.e, called by a routine that has been documented), but is not found on the documentation files, ADS will print: "** WARNING REQUESTED READ OF RECORD (ROUTINE) -- EOF." The message identifies what kind of record (in this case a ROUTINE named EOF) cannot be found in any ROUTINE COMMON or REPORT category of the documentation file. This message will only appear once per execution through the read of the record may be requested many times.

A complete ADS processing run may then look like:

```
ECHO;
TITLE ADS USER MANUAL EXAMPLE;
REPORT TITLES = TITLE;
PRINT NARROW UNPAGED (TITLES) FOR ALL ROUTINES;
PRINT NARROW (DUMP) FOR ALL;
DRAW FULL WIDE TREE;
DRAW COMPRESSED NARROW TREE RDIT;
END;
```

A complete list of ADS error messages and possible causes may be found in Appendix B. Appendix C contains a quick reference to ADS user input.

5 ADS JOB CONTROL

As indicated in Chapter 2, ADS is designed for, and currently executable with, the Control Data Corporation (CDC) FTN version 4.6 compiler. An attempt has been made to make the program portable to computers of other manufacturers; however, reasonably extensive changes in word size, formats, characters, file handling, etc., are necessary before this can be accomplished.

Assuming that the ADS program is ready to be executed at the CDC computer facility chosen by the user, certain control card sequences are necessary to execute the program and attach and save the required files.

The internal names used by the ADS program are:

INPUT- The ADS commands described in Chapter 4.

OUTPUT- The printed output file containing informative diagnostics and optional echo of user-supported ADS commands.

RPTOUT- The reports created by this run of ADS.

REFFL- The reference map information created by the FORTRAN compiler. Only routines on this file are updated this run.

SRCFL- The FORTRAN source code containing the special ADS comment cards. The routine found in the SRCFL must match that of the REFFL.

MASBK- The master random access file containing the documentation records and report definitions for the system. This file should be attached when updates to documentation or reporting of documentation are done. The file MASFL contains old documentation plus any updates performed and should be saved whenever changes to the documentation are to be kept.

INVBK- The master random access file containing the cross-referencing records (see CALLED BY, p26) for routines and common blocks. This file should be attached whenever updates to documentation or reporting of documentation are required. The file INVFL contains old invert records plus any updates performed and should be saved whenever changes to the documentation are to be kept.

MASBK and INVBK are the local file names for the "old" documentation files used as input to ADS. MASFL and INVFL are the new, updated documentation files created by ADS. For a REFFL input example, see Appendix D. For an example of a complete ADS run, see Appendix E.

Although this report outlines general ADS user instructions, it should be noted that the exact control cards used for attaching and saving programs and files differ with various computer sites depending upon the site operating system.

For a typical example using the input outlined above, see Figure 15. In addition, documentation files may be too big to store inexpensively on disks and probably should be kept on tape.

```
<Job cards, accounting cards>

Get/create SRCFL  the source code to be run through ADS (Optional).

FTN,I=SRCFL, L=REFFL,Q, R=<>.

            Invoke FTN for installation using SRCFL,
            creating REFFL for this run.
            R= <>, the user may chose
            the reference map--1 or 2 or 3.

Get ADSDOC  Get executable code of ADS.

Get MASBK   Get master documentation file, if one
            exists.

Get INVBK   Get master invert documentation file, if
            one exists.

ADSDOC.     Execute ADS.

REWIND,RPTOUT.

COPY,RPTOUT,OUTPUT.

Save MASFL / Save Master and invert documentation
            files, if desired.

Save INVFL

7/8/9 (End of Record Card)

ADS Input Directives

6/7/8/9 (End of Information Card)
```

Figure 15. ADS command example.

APPENDIX A:

ADS DOCUMENTATION CATEGORY AND USAGE GUIDE

This appendix is in semitabular format. Each possible documentation category (section title) is shown in capital letters followed by a brief description of information that might be included in a typical documented code. The least allowable format for each documentation category is in capital letters inclosed by parentheses.

ALGORITHM (ALG): denotes a particular type of algorithm being used in a routine. This category is similar to METHOD but might be particularly applicable when the method is not well known or is an adaptation of another method to a new purpose. Also the specification of a particular algorithm may make the documentation action clearer for some applications (e.g. numerical rather than information systems).

Examples:

1. This routine uses the Collins Algorithm for polynomial division as outlined by Knuth, Vol 2.

2. This routine adapts the Taylor series for the sine of an angle to the constraints outlined in the functional requirements of the system.

AUTHOR (AUT): shows the author of the routine. While the AUTHOR section may not be useful in final documentation, it can be very useful to the manager and to other programmers in the software's developmental stages.

CONTROL CARDS (CON): aids the programmer using the system/routine. In general, systems usually need more instructions than a documentation category can provide though, for some systems, this category could indicate catalogued procedure usage. The parameters to the catalogued procedure could be described in more detail in a user's manual. This category could also serve as instruction to a user of a simple program or routine, and could include a user library that contains the program/routine. Since explicit control cards also change from operating system to operating system, this category could become more confusing than useful if an attempt is made to document all systems. Additionally, "JCL" can be used.

DATE WRITTEN (DAT): indicates when the routine was written. This category may be useful to indicate what routines may have been written prior to some operating system change. In developing systems, the date written can aid in displaying the current stage of the development.

FILES (FIL): indicates the files used by the program/system. This category should appear only in a main program because the file documentation record exists as a header record on the documentation files. A maximum of 64 files is allowed for the entire system. In addition to the names of the files, this category can indicate usage of the file, type of access, etc.

Examples:

1. Card input is typically a formatted file.

2. Certain files can be accessed randomly, sequentially, or word addressable.

3. Special formatted files (such as weather tapes from National Atmospheric and Oceanic Administration [NOAA]), may take special consideration. For example, the FILE control card necessary to process the file could be included.

FLOW (FLO): illustrates the logical flow of the code. This category should be used at the top level of system or for a complex submodule. Indentation level options allowed by ADS can be particularly effective in this section. Indenting can aid in displaying "while" loops, etc.

For example:

```
LOGICAL FLOW.
CALL INITIALIZATION ROUTINES (ADINIT, FHINIT, PINIT)
WHILE NOT EOF (REFFL) DO
      READ REFERENCE MAP FOR A ROUTINE.
      SCAN THE SOURCE CODE PARSING DOCUMENTATION FOR ROUTINE.
PERFORM USER INPUT COMMANDS.
PERFORM END-OF-JOB PROCESSING.
```

IMPLEMENTATION DEPENDENCIES (IMP DEP): defines the dependencies of a system/routine implementation. These dependencies could be of several types -- certain operating system characteristics used by the system/routine; assumptions made by the routine about its parameters; new features in the system necessary to implement this code. In each case, the documented code may work only as long as the implied assumptions are in effect. It may not be good programming practice to rely on certain "defects" in the system being used, but it is occasionally necessary. If certain assumptions are implicit in implementing code, then these assumptions should be documented.

LOCATION (LOC): details the location of the source/object code.

MACHINE DEPENDENCIES (MAC DEP): defines the dependencies of the system/routine on a particular machine's architecture. For example, using a certain word size or arithmetic precision can make a system dependent on the machine of its development. Though a goal in software development is transferability (to other machines), often the inefficiencies that can result are too costly to make software even semi-independent. However, these dependencies can at least be documented to aid transferability.

METHOD (MET): describes the method used by the routine. The METHOD category is similar to the ALGORITHM category but may contain more information about the way the code is implemented. For example, the routine may be implementing an in-order tree traversal algorithm, but the method used will depend on the tree data structure. Since FORTRAN is not recursive, the implicit recursion of an in-order tree traversal could be implemented via stacks or some other structure. A METHOD category could explain how the algorithm is implemented.

NONSYSTEM EXTERNALS (NON SYS EXT): defines the various routines used by the system that are not part of the system. Much of this information is provided by ADS for CDC systems. However, these externals could be documented in this category. Also, since the documentation files represent a static version of the actual code (i.e. not directly executable), "dummy" documentation records could be provided for these kind of externals. Typically, these "dummy" records are used to explain not only the usage of the externals, but also why these externals were chosen over similar code within the system under development.

PURPOSE (PUR): describes the purpose of the routine. Often the purpose will be similar to the method used in the routine. (PURPOSE may be a brief description of a simple method whereas the METHOD category would provide a detailed description.)

REFERENCES (REF): details various references that may have been used in developing the routine/system. Another reference could also be the system's user manual. For example, if a complex algorithm cannot readily be explained in sufficient detail to teach the user about it, references are included to indicate where the user can obtain more information about the algorithm or method.

REMARKS (REM): details any documentation which did not fit under other categories; for example, the implementation of a finite state machine in the code or the tabular information of states and transitions.

REVISED (REV): notes revisions made to the code. In addition to date of revision, this category could also contain the name of and reason for the revision.

SYSTEM (SYS): denotes the system of the code. SYSTEM can refer to either the software development system (thus indicating the LOCATION of the most up-to-date version of the code) or the operating system expected by the programmer (thus indicating an inherent IMPLEMENTATION DEPENDENCY).

SYSTEM DEPENDENCIES (SYS DEP): denotes possible dependencies of the documented system.

TITLE (TIT): gives the title of the routine. Typically the title should be the mnemonic meaning of the routine name. The routine name might also be included in the title section. For example, "CD WRLFZB - WRITE LOAD FILE ZONE BLOCK" could be the title line for routine WRLFZB.

(Note: if the word preceding the "-" is the name of a common block, the title text is automatically taken to be the common block title.)

For example:

CD LFEHDR - LOAD FILE ENVIRONMENT HEADER COMMON

Additionally COMMON BLOCK TITLE (COM TIT) may be used for common blocks.

VARIABLE DICTIONARY (VAR): defines the variables in the routine/common block. In addition to indicating the usage of the variable, a definition might also indicate valid values of the variable and these values' meanings.

APPENDIX B:

ADS USER ERROR MESSAGES

If these messages are printed during an ADS execution, the user should correct some of his/her input or bring the output to the ADS consultant. The messages are of three types -- WARNING, SEVERE, or FATAL. WARNING messages are informative to the user; SEVERE messages usually indicate the user must correct something in the code or input; FATAL errors indicate that the ADS system has an unrecoverable error. SEVERE errors in the input commands cause abort of the current input command.

The messages are listed by type; each description includes hints to the user of causes and corrections of ADS errors.

WARNING Messages

INVALID NAME ENTERED --> name, ILLEGAL FOR TYPE --> ROUTINE

A name in the <name list> of a PRINT commmand cannot be found on the master list of routines. User should check spelling, make sure correct documentation files are attached, or check SRCFL and REFFL files to ascertain that such a routine exists.

INVALID NAME ENTERED --> name, ILLEGAL FOR TYPE --> COMMON

A name in the <name list> of a PRINT command cannot be found on the master list of common blocks. User should check spelling, make sure correct documentation files are attached, or check SRCFL for existence of common block documentation.

INVALID NAME ENTERED --> name, ILLEGAL FOR TYPE --> REPORT

A name in the <name list> of a PRINT command cannot be found in the master list of reports. User should check spelling, make sure correct documentation files are attached, or check previous input commands for definition.

NOT FOUND LIST IS FULL, NO MORE CROSS-REFS ALLOWED

The list detailing the names referenced but not found on the master files is full (100 entries allowed) and no more will be entered into the list. This list is output as a byproduct of a "PRINT ... FOR ALL .."

command. If this message occurs, only the first 100 occurrences will be listed.

RESTARTING PARSE WITH FOLLOWING CARD.

Parsing of the source code will resume with the card listed after this message. User should check SRCFL and REFFL for one to one correspondence between modules and reference maps.

ERROR IN READ TAB, SET TO NEAREST -- 1 OR 4

The number of indentation levels has exceeded the limit of four. This message will occur during report processing. User should check documentation "CD" cards for too many "_"s.

DUPLICATE WRITE REQUESTED FOR ROUTINE - name - REQUEST IGNORED

A routine has been documented twice in one run. Only the first will be retained on the documentation files. User should check SRCFL for the duplicate occurrence.

DUPLICATE WRITE REQUESTED FOR COMMON - name - REQUEST IGNORED

A common block has been documented twice in one run. Only the first will be retained on the documentation files. User should check SRCFL for the duplicate documentation occurrence.

REQUESTED READ OF RECORD (ROUTINE) -- name, RECORD NOT ON MASTER FILE

A routine documentation record has been referenced and retrieval has been attempted, but the record of the requested name is not on the documentation files. For example, this message will occur for externals that are part of the FTN library. User should make sure correct documentation files are attached.

REQUESTED READ OF RECORD (COMMON) -- name, RECORD NOT ON MASTER FILE

A common documentation record has been referenced and retrieval has been attempted, but the record of the requested name is not on the documentation files. User should make sure correct documentation files are attached and rerun with documentation for the missing common block.

MISSING ; ASSUMED AT END-OF-CARD

A ";" was expected at the logical end of the statement but was not found on the rest of the card.

MISSING ";" ASSUMED PRESENT

A ";" was expected.

SEVERE Messages

MACRO NAME NOT ON MASTER FILE --> name

A name in <report list> of a PRINT command cannot be found on the master file. User should check spelling or make sure correct documentation files are attached.

UNKNOWN KEYWORD IN DELETE COMMAND

The format of the user DELETE command is incorrect.

MISSING OR MISPLACED (IN DELETE

The format of the user DELETE command is incorrect.

MISSING ROUTINE HEADER, IGNORING CARDS STARTING WITH

A header was found that did not appear to be a valid FORTRAN routine header. Cards are skipped until an END card is found or a valid routine header is encountered.

OVERFLOW OF STRING BUFFER ON OUTPUT

Some combination of the code input has caused the string buffer to overflow in trying to create a documentation record. Additional information is printed that will be useful to a consultant. User should bring outputs (ADS execution, listing of SRCFL) to an ADS consultant.

OVERFLOW OF STRING BUFFER

Additional occurrences of the above error.

TOO MANY VARIABLES IN COMMON BLOCK -- name

The common block displayed has too many variables (limit is 100) for ADS. Either the common block must be redefined or ADS must be changed.

KEYWORD EXPECTED

A keyword is expected in the current input command. User should check the command for proper format.

ROUTINE NAME, DEPTH, OR ; EXPECTED AFTER TREE

A DRAW command has improper format. User should check the command for proper format.

DEPTH EXPECTED

The DRAW command expects a <DEPTH>, but found something entirely different. User should check command for proper format.

; EXPECTED

A ";" is expected in the current input command. User should check the command for proper format.

UNKNOWN COMMAND

The current input command cannot be recognized. User should check input command for proper format.

= EXPECTED AFTER REPORT NAME

A REPORT input command has improper format. User should check the command for proper format.

UNKNOWN ERROR

A REPORT input command has improper format. User should check the command for proper format.

DELIMITER EXPECTED

A REPORT input command has improper format. User should check the command for proper format.

EXTERNALS EXPECTED AFTER NON SYSTEM

A REPORT keyword was not properly specified. User should check the command for format.

SYSTEM EXPECTED AFTER NON

A REPORT keyword was not properly specified. User should check the command for proper format.

FOUND END-OF-FILE WHILE LOOKING FOR -- character

The end of the input file was found while looking for the indicated character.

; FOUND TOO SOON IN PRINT STATEMENT

A ";" terminated the PRINT command too early. User should check the command for proper format.

MISSING -FOR- IN PRINT COMMAND, SKIP TO ;

The FOR keyword was missing in the PRINT command. User should check the command for proper format.

MISSING END CARD. RESTART WITH FOLLOWING CARD

Parsing of the source code will resume with the card listed after this message. User should place an END card in SRCFL before the listed card.

ILLEGAL NAME IN NAMELIST FORMAT, SKIP TO ;

A word which could not be a name (i.e., a character, etc.) was found in a PRINT command. User should check the command for proper format.

DEPENDENCY EXPECTED

The word DEPENDENCY was expected in a REPORT keyword. User should check for proper format.

TOO MANY CHARACTERS IN TITLE, SKIP TO ;

The TITLE is limited to 80 characters. User should check the TITLE card to make sure the title string is followed by a ";", ",", etc.

FATAL Error Messages

On a FATAL error, the indicated messages will be printed. Also supplementary information of use to the consultant will be printed. All applicable printouts should be taken to the consultant.

TOO MANY ITEMS IN LIST (ENTRNO)

TOO MANY ITEMS IN LIST (SCANNO)

Too many items have been entered in an ADS list. The items in the list will be printed as part of the error dump.

TOO MANY EXTERNALS IN ROUTINE

The limit of externals used by one routine is 100.

TOO MANY COMMONS IN ROUTINE

The limit of commons used by one routine is 100.

TOO MANY LOCAL VARIABLES IN ROUTINE

The limit of local variables used by one routine is 100.

APPENDIX C:

QUICK REFERENCE -- ADS USER AND CODE INPUT

ADS User Input

Each type of statement listed is ended by a semicolon (";"). Definitions for the input commands are listed in Chapter 3 and 4. Characters ",", "=", "(",")" must be included as shown; words in all capitals must be included as shown.

```
<INPUT KEYWORD> ::= ECHO | NOECHO | HELP | END

<ADS REPORT DEFINE> ::= REPORT <NAME> = <REPORT SECTION LIST> ;

<ADS REPORT DEFINE> ::= REPORT <NAME> = <REPORT SECTION LIST> ;

<NAME> ::= (user supplied name <= 10 characters)

<REPORT SECTION LIST> ::= <REPORT SECTION NAME>
                          [, <REPORT SECTION NAME>]

<REPORT SECTION NAME> ::= ALGORITHM [ AUTHOR | CALLED | BY
                          COMMON(S) [BLOCKS] CALLED | CONTROL CARDS |
                          DATE WRITTEN | ENTRY POINTS | FILES | FLOW |
                          IMPLEMENTATION DEPENDENCIES |
                          LOCATION | MACHINE DEPENDENCIES |
                          METHOD | NON SYSTEM EXTERNALS |
                          PURPOSE | REFERENCES | REMARKS |
                          REVISED | ROUTINES CALLED | SYSTEM |
                          SYSTEM DEPENDENCIES | TITLE |
                          VARIABLE DICTIONARY

<ADS DELETE> ::= DELETE <RECORD TYPE> ( <NAME LIST> );

<RECORD TYPE> ::= ALL | ROUTINE(S) | COMMON(S) | REPORT(S)

<NAME LIST> ::= <RECORD NAME> [, <RECORD NAME> ]

<RECORD NAME> ::= (any record name -- ROUTINE, COMMON, or REPORT)

<ADS PRINT> ::= PRINT <REPORT WIDTH> <PAGING>
                ( <REPORT LIST> ) FOR <RECORD LIST>;

<REPORT WIDTH> ::= NARROW | WIDE

<PAGING> ::= PAGED | UNPAGED
```

<REPORT LIST> ::= <NAME LIST OF REPORT names>

<RECORD LIST> ::= ALL | ALL ROUTINE(S) | ALL COMMON(S) |
ALL REPORT(S) | UPDATED | UPDATED ROUTINE(S) |
UPDATED COMMON(S) | (<NAME LIST>)

<ADS DRAW> ::= DRAW <TREE TYPE> <REPORT WIDTH> TREE [<ROUTINE>]
[<DEPTH>]

<TREE TYPE> ::= COMPRESSED | FULL

<ROUTINE> ::= (record name for root of tree)

<DEPTH> ::= (number of levels to be drawn)

ADS Code Input

CD$ _ Begin new line at left margin

CD$ Begin new line; retain tab setting indicated in previous line

CD No new line; line may be run on with preceding line

CD$ __ Begin new line; reset tab to indent 5 spaces

CD$ ___ Begin new line; reset tab to indent 10 spaces

CD$ ____ Begin new line; reset tab to indent 15 spaces

ADS text sections are of two types:

1. Text that completely documents a section -- <TYPE 1 TEXT>

2. Text that has a subheader, followed by a separator character, followed by <TYPE 1 TEXT> (reference FILES, VARIABLE DICTIONARY sections). For example: "CD TITLE:= RDIT - PROGRAM TO READ DECKS..." "RDIT - PROGRAM TO ..." is <TYPE 1 TEXT>. The entire statement is a TITLE section. In, "CD VAR1 - VARIABLE TO FLAG ...", "VAR1" is a subheader, "-" is a separator. "VARIABLE TO ..." is <TYPE 1 TEXT>. A separator character is typically the hyphen "-", but may be any of "-", "=", or ":".

The ADS sections are of two types:

1. ADS type 1 section:

CD <TYPE 1 SECTION HEADER>:=
CD <TYPE 1 TEXT>

where the user can insert multiple "CD" cards of <TYPE 1 TEXT>.

2. ADS type 2 section:

```
CD    <TYPE 2 SECTION HEADER> ::=
CD    <TYPE 2 SECTION SUBHEADER> <SEPARATOR>
CD          <TYPE 1 TEXT>
```

where <TYPE 1 TEXT> cards may be repeated for each <TYPE 2 SECTION SUBHEADER> and <TYPE 2 SECTION SUBHEADER> statements may also be repeated as needed.

```
<TYPE 1 SECTION HEADER> ::= ALGORITHM | AUTHOR | CONTROL CARDS |
                            COMMON BLOCK TITLE | DATE WRITTEN | FLOW |
                            IMPLEMENTATION DEPENDENCIES |
                            LOCATION | MACHINE DEPENDENCIES |
                            METHOD | NON SYSTEM EXTERNALS |
                            PURPOSE | REFERENCES | REMARKS |
                            REVISED | SYSTEM |
                            SYSTEM DEPENDENCIES  TITLE

<TYPE 1 TEXT> ::= <TEXT CARDS CONTAINING OPTIONS "$","__">

<TYPE 2 SECTION HEADER> ::= FILES | VA

<SEPARATOR> ::= -|:|=

<TYPE 2 SECTION SUBHEADER> ::= <FILE OR VARIABLE NAME>
```

(Note: No error processing is done during the parsing of the "CD" cards. Therefore, an invalid section header or section subheader will be taken as text of the previous section. These input errors will usually be apparent in the report output by making some documentation categories appear inconsistent, or a variable thought to have been documented may appear with a flag indicating no documentation.)

APPENDIX D:

ADS REFFL INPUT FILE EXAMPLE

This appendix illustrates the ADS REFFL input file for the example shown in Appendix D. Also included is an example CDC FTN compiler listing and reference map. The source code shown is the ADS SRCFL input file.

TREE

```
RDIT____
        |_SORTIN
        |_SORT
        |_SMOUT
        |_EOF
        |_ENDDK
        |_ENDCDK
```

```
PROGRAM RDIT      7600_7600 OPT=1                                FTN 4.6+452/034

 1                PROGRAM RDIT(INPUT,OUTPUT,TAPE1,TAPE2)
                  IMPLICIT INTEGER (A-Z)
            CD    TITLE:=
            CD    RDIT - READ A SET OF DECKS TO EXCLUDE FROM GENERAL SORTING
 5          CD    PURPOSE:=
            CD    THE PURPOSE OF THIS PROGRAM IS TO GENERATE MOVE DIRECTIVES TO
            CD    SORT AN OLDPL -- UPDATE LIBRARY.  UP TO THREE LEVELS OF
            CD    DECKS CAN BE EXCLUDED FROM THE TOTAL SORT (AND ARE THEN
            CD    SORTED AMONGST THEMSELVES).  COMDECKS ARE SORTED SEPARATELY
10          CD    FROM MAIN DECKS.  EACH LEVEL MUST ALSO INPUT A DECK NAME
            CD    THAT ALL DECKS IN THE LEVEL WILL FOLLOW -- THE DIRECTIVE
            CD     WILL BE OF FORM:
            CD$__"*MOVE <DECK NAME>,<INPUT DECK NAME>"
            CD$_  TO LEVEL DECKS ONE PUTS IN CARDS OF FORM:
15          CD$__  COL 1--9.  NAME OF EXCLUDED DECK
            CD$__  COL. 10    BLANK, 1, OR 2
            CD$_ DECK NAMES ARE SORTED AND MOVE DIRECTIVES GENERATED
            CD    SUCH THAT THE FINAL ORDER OF THE UPDATE OLDPL WILL
            CD    BE:
20          CD$__    COMDECKS,
            CD$ DECKS (INPUT CARD NAMES) WITH COL 10 = 1,
            CD$   DECKS (INPUT CARD NAMES) WITH COL 10 = 2,
            CD$   DECKS (NOT COMDECKS) NOT SPECIFIED ON INPUT CARDS.
            CD$   NOTE -- DECKS (INPUT CARD NAMES) WITH COL 10 = BLANK
25          CD    ARE NOT SORTED!  THESE DECKS CAN REMAIN IN PRESENT PLACES
            CD    IN THE OLDPL AND BE USED FOR POSITIONING OTHER DECKS.
            CD$_   THE PROGRAM USES THE BASIC OUTPUT (L=A14) FROM THE UPDATE
            CD    PROGRAM AS INPUT (TAPE1).
            CD    VARIABLE DICTIONARY:=
30          CD    IN - NUMBER OF INPUTS <= 268.
            CD    CDECK - NUMBER OF COMMON DECKS IN UPDATE LIST.
            CD    DECKA - NAME OF "AFTER" DECK FOR CURRENT LEVEL (ON OUTPUT).
            CD    I     - LOOP CONTROL
            CD    IKEY  - LEVEL FOR CURRENT SORT.
35          CD    J     - LOOP CONTROL
            CD    K     - LOOP CONTROL
            CD    NDECK - NUMBER OF DECKS IN LIST.
                  COMMON/DECKS/DECKS(400),DFLAGS(400)
            CD    COMMON BLOCK TITLE:=
40          CD    DECKS - DECK NAMES AND FLAGS
            CD    VARIABLE DICTIONARY:=
            CD    DECKS - THE ARRAY OF DECK NAMES.
            CD    DFLAGS - THE CORRESPONDING ARRAY OF DECK FLAGS 0--3
                  DIMENSION INBUF(132),INFLG(268)
45                COMMON /SCRATCH/ SORTD(400),INUM
                  EQUIVALENCE (SORTD(1),INBUF(1)), (SORTD(133),INFLG(1))
            CD    COMMON BLOCK TITLE:=
            CD    SCRATCH - SCRATCH USE AREA.
            CD    VARIABLE DICTIONARY:=
50          CD    SORTD   - ARRAY CONTAINING NAMES TO BE SORTED. NUMBER OF
            CD              NAMES IS INUM.
            CD    INBUF - INPUT BUFFER AREA FOR DECK LIST FROM UPDATE.
            CD            EQUIVALENCED TO FIRST PART OF SORTD ARRAY.
            CD    INFLG - INPUT DECKS AND FLAGS.
55          CD            EQUIVALENCED TO PART OF SORTD ARRAY.
            CD     INUM  - NUMBER OF NAMES TO BE SORTED BY ROUTINE SORT.
                  LOGICAL ENDDK,ENDCDK
```

```
      PROGRAM RDIT      7600_7600 OPT=1                          FTN 4.6+452/074

            DATA DECKS,DFLAGS,INBUF,INFLG/1200*0/
          C                                          READ INPUT
 60         REWIND 1
            IN = 1
  100       READ 700,INFLG(IN)
            IF (EOF(5LINPUT) .NE. 0) GO TO 105
            IN = IN + 1
 65         IF (IN .GT. 268) STOP "TOO MANY INPUTS"
            GO TO 100
  105       IN = IN - 1
          C
          C                              DEPENDING UPON LIST OPTION TO
 70       C                              UPDATE, HEADERS OF UPDATE OUTPUT
          C                              CAN START IN COL 2 OR COL 11.
          C
  110       READ (1,701) INBUF
            IF (EOF(1) .NE. 0) STOP "ERROR IN DECK LIST"
 75       C                              LOOK FOR WORDS --
          C                              DECKS ARE LISTED IN ...
            IF (INBUF(2) .EQ. 1HD .A. INBUF(3) .EQ. 1HE .A.
           1      INBUF(4) .EQ. 1HC .A. INBUF(5) .EQ. 1HK) GO TO 115
            IF (INBUF(11) .EQ. 1H .A. INBUF(12) .EQ. 1HD .A.
 80        1      INBUF(13) .EQ. 1HE .A. INBUF(14) .EQ. 1HC
           2      .A. INBUF(15) .EQ. 1HK) GO TO 115
            GO TO 110
  115       CONTINUE
          C                                          READY FOR DECK LIST
 85       C                              READ DECK NAMES (STARTING IN COL 11)
            READ(1,701)  INBUF
            IF (EOF(1) .NE. 0) STOP "ERROR IN DECK LIST"
            IF (INBUF(11) .NE. 1H ) GO TO 120
            GO TO 115
 90         120  NDECK = 0
            125  IF (ENDDK(NDECK,INBUF(11),A)) GO TO 130
            126  READ(1,701) INBUF
                 IF (EOF(1) .NE. 0) STOP "ERROR IN DECK LIST"
                 IF (INBUF(1) .EQ. 1H1) GO TO 126
 95              GO TO 125
  130       CONTINUE
          C                                          LOOK FOR COM DECKS
          C                              READ COMDECK NAMES (STARTING IN COL 11)
            READ (1,701) INBUF
100         IF (EOF(1) .NE. 0) GO TO 155
            IF (INBUF(2) .EQ. 1HC .A. INBUF(3) .EQ. 1HO .A.
           1      INBUF(4) .EQ. 1HM) GO TO 135
            IF (INBUF(11) .EQ. 1H  .A. INBUF(12) .EQ. 1HC .A.
           1      INBUF(13) .EQ. 1HO .A. INBUF(14) .EQ. 1HM) GO TO 135
105         GO TO 130
  135       READ(1,701) INBUF
            IF (EOF(1) .NE. 0) STOP "ERROR IN DECK LIST"
            IF (INBUF(11) .NE. 1H ) GO TO 140
            GO TO 135
110         140  CONTINUE
                 CDECK = 1
            147  IF (ENDCDK(NDECK,INBUF(11),A,CDECK)) GO TO 155
            148  READ(1,701) INBUF
                 IF (EOF(1) .NE. 0) STOP "ERROR IN DCCK LIST"
```

```
      PROGRAM RDIT     7600_7600 OPT=1

115               IF (INBUF(1) .EQ. 1H1) GO TO 148
                  GO TO 147
              155 CONTINUE
                C
                C                              IF ANY INPUT NAMES WERE ACCEPTED,
120             C                              THEY ARE POSITIONED IN A LEVEL TO BE
                C                              USED LATER IN SORTING.
                C                              INTERNAL VALUES (STORED) IN DFLAGS ARE:
                C                              CARD COL. 10 = 1, DFLAGS=1
                C                              CARD COL. 10 = 2, DFLAGS=2
125             C                              CARD COL. 10 = BLANK, DFLAGS=4
                C                              COMDECK, DFLAGS=3
                C                              OTHER DECK, DFLAGS=0.
                C                              SORTING IS DONE 0, 1, 2, 3, IN
                C                              DESCENDING ORDER IN ORDER TO POSITION THE
130             C                              DECKS PROPERLY WITH THE MOVE STATEMENTS.
                  IF (IN .EQ. 0) GO TO 170
                  DO 160 I=1,NDECK
                  DO 160 J=1,IN
                  IF (AND(DECKS(I),MASK(54)) .NE. AND(INFLG(J),MASK(54)))
135              1     GO TO 160
                  IF (AND(INFLG(J),778) .EQ. 1R ) DFLAGS(I)=4
                  IF (AND(INFLG(J),778) .EQ. 1R1) DFLAGS(I) = 1
                  IF (AND(INFLG(J),778) .EQ. 1R2) DFLAGS(I)=2
              160 CONTINUE
140           170 CONTINUE
                  PRINT 702
              702 FORMAT(1H1,T40,"GENERATING UPDATE DIRECTIVES")
                  DO 190 I=1,4
                  IKEY=I-1
145               INUM=0
                  DO 180 K=1,NDECK
              180 IF (DFLAGS(K) .EQ. IKEY) CALL SORTIN(DECKS(K))
                  IF (INUM .EQ. 0) GO TO 190
                  READ 700,DECKA
150               IF (EOF(5LINPUT) .NE. 0) DECKA = 10HYANK$$$
                  PRINT 703,DECKA
              703 FORMAT(1H-,   " DIRECTIVES WILL BE OF FORM '*MOVE <DECK NAME>.",
                 1A10,"'")
                  CALL SORT
155               CALL SMOUT(DECKA)
              190 CONTINUE
                  REWIND 2
              700 FORMAT(A10)
              701 FORMAT(132A1)
160               STOP
                  END


      SYMBOLIC REFERENCE MAP (R=3)

ENTRY POINTS     DEF LINE     REFERENCES
 4212  RDIT           1
```

PROGRAM RDIT 7600_7600 OPT=1 FTN 4.6+452/034

VARIABLES	SN	TYPE	RELOCATION									
4335	CDECK	INTEGER			REFS	112	DEFINED	111				
4342	DECKA	INTEGER			REFS	151	155	DEFINED	149	150		
0	DECKS	INTEGER	ARRAY	DECKS	REFS	38	134	147	DEFINED	58		
620	DFLAGS	INTEGER	ARRAY	DECKS	REFS	38	147	DEFINED	58	136	137	138
4336	I	INTEGER			REFS	134	136	137	138	144		
					DEFINED	132	143					
4340	IKEY	INTEGER			REFS	147	DEFINED	144				
4333	IN	INTEGER			REFS	62	64	65	67	131	133	
					DEFINED	61	64	67				
0	INBUF	INTEGER	ARRAY	SCRATCH	REFS	44	46	4*77	5*79	88	91	94
						3*101	4*103	108	112	115	DEFINED	58	73
						86	92	99	106	113		
204	INFLG	INTEGER	ARRAY	SCRATCH	REFS	44	46	134	136	137	138	
					DEFINED	58	62					
620	INUM	INTEGER		SCRATCH	REFS	45	148	DEFINED	145			
4337	J	INTEGER			REFS	134	136	137	138	DEFINED	133	
4341	K	INTEGER			REFS	2*147	DEFINED	146				
4334	NDECK	INTEGER			REFS	91	112	132	146	DEFINED	90	
0	SORTD	INTEGER	ARRAY	SCRATCH	REFS	45	2*46					

FILE NAMES		MODE								
0	INPUT	FMT	READS	62	149					
1041	OUTPUT	FMT	WRITES	141	151					
2102	TAPE1	FMT	READS	73	86	92	99	106	113	
			MOTION	60						
3143	TAPE2		MOTION	157						

EXTERNALS	TYPE	ARGS	REFERENCES							
ENDCDK	LOGICAL	4	57	112						
ENDDK	LOGICAL	3	57	91						
EOF	INTEGER	1	63	74	87	93	100	107	114	150
SYOUT		1	155							
SORT		0	154							
SORTIN		1	147							

INLINE FUNCTIONS	TYPE	ARGS	DEF LINE	REFERENCES			
AND	NO TYPE	0	INTRIN	2*134	136	137	138
MASK	NO TYPE	1	INTRIN	2*134			

STATEMENT LABELS		DEF LINE	REFERENCES		
0	100	62	66		
400000	105	67	63		
400000	110	73	82		
400000	115	83	77	79	89
0	120	90	88		
400000	125	91	95		
0	126	92	94		
0	130	96	91	105	
0	135	106	101	103	109
0	140	110	108		
0	147	112	116		
400000	148	113	115		
400000	155	117	100	112	
400000	160	139	132	133	134
0	170	140	131		
400000	180	147	146		

PROGRAM RDIT 7600_7600 OPT=1 FTN 4.6+452/034

STATEMENT LABELS			DEF LINE	REFERENCES					
0	190		156	143	148				
4312	700	FMT	158	62	149				
4314	701	FMT	159	73	86	92	99	106	113
4262	702	FMT	142	141					
4301	703	FMT	152	151					

COMMON BLOCKS	LENGTH	MEMBERS - BIAS NAME (LENGTH)		
DECKS	800	0 DECKS (400)	400 DFLAGS	(400)
SCRATCH	401	0 SORTD (400)	400 INUM	(1)

EQUIV CLASSES	LENGTH	MEMBERS - BIAS NAME (LENGTH)		
SORTD SORTD	400	0 INBUF (132)	132 INFLG	(268)

STATISTICS

PROGRAM LENGTH	1658	117
BUFFER LENGTH	42048	2180
SCM LABELED COMMON LENGTH	22618	1201

```
FUNCTION ENDDK    7600_7600 OPT=1                          FTN 4.6+452/034

 1                LOGICAL FUNCTION ENDDK(NDECK,INBUF,N)
                  IMPLICIT INTEGER(A-Z)
               CD TITLE:= ENDDK - DETERMINE END OF DECK LIST
               CD PURPOSE:= THIS FUNCTION DETERMINES WHEN THE END OF
 5             CD DECK LIST HAS BEEN REACHED.  IT ALSO DECODES THE LINE
               CD OF INPUT INTO THE DECK LIST TO BE SORTED.
               CD VAR:=
               CD ENDDK - FUNCTION VALUE, WHEN TRUE, END OF DECK LIST
               CD         HAS BEEN REACHED.
10             CD I     - LOOP CONTROL.
               CD INBUF - INPUT LINE (INPUT PARAMETER).
               CD I1    - START CHARACTER POSITION OF DECK NAME
               CD I2    - END CHARACTER POSITION OF DECK NAME.
               CD J     - LOOP CONTROL.
15             CD N     - NUMBER OF POSSIBLE DECKS ON THIS LINE (INPUT PARAMETER).
               CD NDECK - CUMULATIVE COUNT OF NUMBER OF DECKS (I/O PARAMETER).
               CD WORD  - WORD FOUND I1..I2 POSITIONS, IF BLANK END OF LIST.
                  DIMENSION INBUF(1)
                  COMMON/DECKS/DECKS(400),DFLAGS(400)
20                ENDDK=.FALSE.
                  DO 100 I=1,N
                  I1 = (I-1)*10+1
                  I2 = I1+9
                  ENCODE(10,700,WORD) (INBUF(J),J=I1,I2)
25             700 FORMAT(10A1)
                  IF (WORD .EQ. 10H          ) GO TO 400
                  NDECK = NDECK + 1
                  IF (NDECK .GT. 400) STOP "TOO MANY DECKS"
                  DECKS(NDECK) = WORD
30             100 CONTINUE
                  RETURN
               400 ENDDK = .TRUE.
                  RETURN
                  END
```

SYMBOLIC REFERENCE MAP (R=3)

ENTRY POINTS

		DEF LINE	REFERENCES	
4	ENDDK	1	31	33

VARIABLES

	SN	TYPE		RELOCATION								
0	DECKS	INTEGER	ARRAY	DECKS	REFS	19	DEFINED	29				
620	DFLAGS	INTEGER	ARRAY	DECKS	REFS	19						
17	ENDDK	LOGICAL			DEFINED	20	32					
20	I	INTEGER			REFS	22	DEFINED	21				
0	INBUF	INTEGER	ARRAY	F.P.	REFS	18	24	DEFINED	1			
21	I1	INTEGER			REFS	23	24	DEFINED	22			
22	I2	INTEGER			REFS	24	DEFINED	23				
24	J	INTEGER			REFS	24	DEFINED	24				
0	N	INTEGER		F.P.	REFS	21	DEFINED	1				
0	NDECK	INTEGER		F.P.	REFS	27	28	29	DEFINED	1	27	
23	WORD	INTEGER			REFS	26	29	DEFINED	24			

```
          FUNCTION ENDDK     7600_7600 OPT=1                    FTN 4.6+452/034

STATEMENT LABELS                    DEF LINE    REFERENCES
     0  100                             30          21
     0  400                             32          26
    13  700       FMT                   25          24

COMMON BLOCKS    LENGTH      MEMBERS - BIAS NAME(LENGTH)
       DECKS        800                0 DECKS  (400)          400 DFLAGS (400)

STATISTICS
  PROGRAM LENGTH                     308         24
  SCM LABELED COMMON LENGTH         1440B        800
```

```
      FUNCTION ENDCDK   7A00_7600 OPT=1                    FTN 4.6+452/034

    1                 LOGICAL FUNCTION ENDCDK(NDECK,INBUF,N,CDECK)
                      IMPLICIT INTEGER(A-Z)
                CD    TITLE:= ENDCDK - DETERMINE END OF COMDECK LIST
                CD    PURPOSE:= THIS FUNCTION DETERMINES WHEN THE END OF
    5           CD    COMDECK LIST HAS BEEN REACHED.  IT ALSO DECODES THE LINE
                CD    OF INPUT INTO THE COMDECK LIST TO BE SORTED.
                CD    VAR:=
                CD    CDECK  - STARTING POSITION FOR COMDECKS.
                CD    ENDCDK - FUNCTION VALUE, WHEN TRUE, END OF COMDECK LIST
   10           CD    I      - LOOP CONTROL.
                CD    INBUF  - INPUT LINE (INPUT PARAMETER).
                CD    I1     - START CHARACTER POSITION OF DECK NAME
                CD    I2     - END CHARACTER POSITION OF DECK NAME
                CD    J      - LOOP CONTROL
   15           CD    N      - NUMBER OF DECK NAMES ALLOWED IN A LINE.
                CD    NDECK  - CURRENT TOTAL NUMBER OF DECKS.
                CD    WORD   - WORD FOUND I1..I2 POSITIONS. IF BLANK END OF LIST.
                      DIMENSION INBUF(1)
                      COMMON/DECKS/DECKS(400),DFLAGS(400)
   20                 ENDCDK=.FALSE.
                      DO 100 I=1,N
                      I1 = (I-1)*10+1
                      I2 = I1+9
                      ENCODE(10,700,WORD) (INBUF(J),J=I1,I2)
   25          700    FORMAT(10A1)
                      IF (WORD .EQ. 10H          ) GO TO 400
                      DO 101 J=CDECK,NDECK
                      IF (DECKS(J) .NE. WORD) GO TO 101
                      CDECK=J
   30                 DFLAGS(J)=3
                      GO TO 100
                  101 CONTINUE
                  100 CONTINUE
                      RETURN
   35             400 ENDCDK = .TRUE.
                      RETURN
                      END


CARD NR. SEVERITY  DETAILS    DIAGNOSIS OF PROBLEM

      29    I      CDECK      THIS STATEMENT REDEFINES A CURRENT LOOP CONTROL VARIABLE OR PARAMETER.


        SYMBOLIC REFERENCE MAP (R=3)

 ENTRY POINTS      DEF LINE      REFERENCES
     4  ENDCDK          1          34          36

 VARIABLES     SN  TYPE              RELOCATION
     0  CDECK      INTEGER                F.P.        REFS        27    DEFINED        1        29
     0  DECKS      INTEGER     ARRAY      DECKS       REFS        19                   28
```

```
          FUNCTION ENDCDK   7600_7600 OPT=1                    FTN 4.6+452/034

 VARIABLES      SN  TYPE          RELOCATION
   620  DFLAGS      INTEGER   ARRAY    DECKS          REFS      19    DEFINED       30
    17  ENDCDK      LOGICAL                         DEFINED     20       35
    20  I           INTEGER                           REFS      22    DEFINED       21
     0  INBUF       INTEGER   ARRAY    F.P.           REFS      18       24         DEFINED        1
    21  I1          INTEGER                           REFS      23       24         DEFINED       22
    22  I2          INTEGER                           REFS      24    DEFINED       23
    24  J           INTEGER                           REFS      24       28         29        30    DEFINED       24        27
     0  N           INTEGER             F.P.          REFS      21    DEFINED        1
     0  NDECK       INTEGER             F.P.          REFS      27    DEFINED        1
    23  WORD        INTEGER                           REFS      26       28         DEFINED       24

 STATEMENT LABELS                 DEF LINE   REFERENCES
400000  100                          33         21        31
     0  101                          32         27        28
400000  400                          35         26
    13  700      FMT                 25         24

 COMMON BLOCKS   LENGTH     MEMBERS - BIAS NAME(LENGTH)
       DECKS        800              0 DECKS  (400)              400 DFLAGS (400)

 STATISTICS
   PROGRAM LENGTH                    258       21
   SCM LABELED COMMON LENGTH       14408      800
```

```
 SUBROUTINE SMOUT    7600_7600 OPT=1                          FTN 4.6+452/034

  1                SUBROUTINE SMOUT(DECKA)
                   IMPLICIT INTEGER(A-Z)
                CD TITLE:= SMOUT - OUTPUT THE SORTED NAMES IN UPDATE DIRECTIVE
                CD FORM.
  5             CD PURPOSE:= THIS ROUTINE OUTPUTS THE SORTED NAMES (ON TAPE2)
                CD IN UPDATE DIRECTIVE FORM (*MOVE ...).
                CD VAR:=
                CD DECKA  - DECK AFTER WHICH CURRENT LEVEL OF SORTED
                CD          NAMES IS TO BE PLACED.
 10             CD NAME   - DECK NAME IN CHARACTER FORMAT, SINCE UPDATE CAN
                CD          NOT ACCEPT BLANKS BEFORE THE COMMA IN THE *MOVE
                CD          DIRECTIVE.
                   COMMON/SCRATCH/SORTD(400),INUM
                   DIMENSION NAME(10)
 15                DO 100 K=1,INUM
                   DECODE(10,700,SORTD(K)) NAME
                700 FORMAT(10A1)
                   DO 101 I=1,10
                   IF (NAME(I) .EQ. 1H ) GO TO 102
 20                101 CONTINUE
                   I=10
                   102 I=I-1
                   WRITE(   2,701) I,(NAME(J),J=1,I),DECKA
                   701 FORMAT("*MOVE ",=A1,",",A10)
 25                100 CONTINUE
                   400 CONTINUE
                   RETURN
                   END
```

SYMBOLIC REFERENCE MAP (R=3)

ENTRY POINTS

		DEF LINE	REFERENCES
3	SMOUT	1	27

VARIABLES

	SN	TYPE		RELOCATION								
0	DECKA	INTEGER		F.P.	REFS	23	DEFINED	1				
31	I	INTEGER			REFS	19	22	2*23	DEFINED	18	21	22
620	INUM	INTEGER		SCRATCH	REFS	13	15					
32	J	INTEGER			REFS	23	DEFINED	23				
30	K	INTEGER			REFS	16	DEFINED	15				
33	NAME	INTEGER	ARRAY		REFS	14	19	23	DEFINED	16		
0	SORTD	INTEGER	ARRAY	SCRATCH	REFS	13	16					

FILE NAMES

	MODE		
TAPE2	FMT	WRITES	23

STATEMENT LABELS

			DEF LINE	REFERENCES
400000	100		25	15
0	101		20	18
0	102		22	19
0	400	INACTIVE	26	
12	700	FMT	17	16
22	701	FMT	24	23

```
      SUBROUTINE SMOUT     7600_7600 OPT=1                    FTN 4.6+452/014

COMMON BLOCKS   LENGTH     MEMBERS - BIAS NAME(LENGTH)
      SCRATCH     401                 0 SORTD  (400)          400 INUM   (1)

STATISTICS
  PROGRAM LENGTH                  458       37
  SCM LABELED COMMON LENGTH      6218      401
```

```
      SUBROUTINE SORT0     7600_7600 OPT=1                          FTN 4.6+452/034

     1                SUBROUTINE SORT0
                      IMPLICIT INTEGER(A-Z)
                      COMMON/SCRATCH/SORTD(400),INUM
                      INUM=0
     5                RETURN
                      END



       SYMBOLIC REFERENCE MAP (R=3)

ENTRY POINTS      DEF LINE      REFERENCES
    1  SORT0             1          5

VARIABLES     SN  TYPE            RELOCATION
  620  INUM       INTEGER              SCRATCH       REFS          3    DEFINED        4
    0  SORTD      INTEGER    ARRAY     SCRATCH       REFS          3

COMMON BLOCKS   LENGTH      MEMBERS - BIAS NAME(LENGTH)
       SCRATCH     401                0 SORTD  (400)              400 INUM   (1)

STATISTICS
  PROGRAM LENGTH                    28       2
  SCM LABELED COMMON LENGTH       6218     401
```

```
      SUBROUTINE SORTIN   7600_7600 OPT=1                    FTN 4.6+452/034

 1                SUBROUTINE SORTIN(DECKN)
                  IMPLICIT INTEGER(A-Z)
            CD    TITLE:= SORTIN - INPUT NAMES TO BE SORTED.
            CD    PURPOSE:= THIS ROUTINE IS CALLED ONCE FOR EACH NAME
 5          CD    TO BE SORTED.  THIS ROUTINE ACCUMULATES THOSE NAMES
            CD    INTO A SORT LIST.
            CD    VAR:=
            CD    DECKN  - DECK NAME TO BE PLACED IN LIST (INPUT PARAMETER).
                  COMMON/SCRATCH/SORTD(400),INUM
10                INUM = INUM+1
                  IF (INUM .GT. 400) STOP "TOO MANY FOR SORT"
                  SORTD(INUM)=DECKN
                  RETURN
                  END
```

```
      SYMBOLIC REFERENCE MAP (R=3)

ENTRY POINTS      DEF LINE      REFERENCES
    3  SORTIN          1        13

VARIABLES     SN  TYPE            RELOCATION
    0  DECKN      INTEGER              F.P.        REFS       12    DEFINED        1
  620  INUM       INTEGER           SCRATCH        REFS        9         10       11        12    DEFINED       10
    0  SORTD      INTEGER    ARRAY  SCRATCH        REFS        9    DEFINED       12

COMMON BLOCKS   LENGTH      MEMBERS - BIAS NAME(LENGTH)
       SCRATCH     401              0 SORTD  (400)              400 INUM   (1)

STATISTICS
  PROGRAM LENGTH                  108       8
  SCM LABELED COMMON LENGTH      6218     401
```

```
SUBROUTINE SORT     7600_7600 OPT=1                    FTN 4.6+452/034

      1               SUBROUTINE SORT
                      IMPLICIT INTEGER(A-Z)
                CD    TITLE := SORT - SORT A LIST OF NAMES
                CD    PURPOSE:= THIS ROUTINE ACCOMPLISHES THE ACTUAL SORT ON
      5         CD    THE INPUT LIST OF NAMES.
                CD    THE LIST IS SORTED IN DESCENDING ORDER SO THAT
                CD    THE EVENTUAL OUTPUT WILL PLACE THE DECKS ON
                CD    THE OLDPL IN PROPER ORDER.
                CD    METHOD:=
     10         CD    THE METHOD OF THIS SORT IS A SELECTION SORT.
                CD    AT EACH STEP THE ITEM WITH THE HIGHEST (LOWEST)
                CD    VALUE IS SELECTED FROM THOSE REMAINING.
                CD    REFERENCES:=
                CD$    KNUTH, VOL III: SEARCHING AND SORTING;
     15         CD$    ELSON, DATA STRUCTURES;
                CD$    MANY, MANY MORE.
                CD    VAR:=
                CD    KEY   - CURRENT KEY
                CD    N1    - LOOP CONTROL (NUMBER OF DECKS - 1)
     20         CD    I1    - LOOP CONTROL (I+1)
                CD    I     - LOOP CONTROL
                CD    J     - LOOP CONTROL
                CD    K     - LOOP CONTROL
                      COMMON/SCRATCH/SORTD(400),INUM
     25               IF (INUM .EQ. 1) RETURN
                      N1 = INUM-1
                      DO 100 I=1,N1
                           K=I
                           KEY=SORTD(I)
     30                    I1=I+1
                           DO 101 J=I1,INUM
                                IF (SORTD(J) .LE. KEY) GO TO 101
                                 K=J
                                 KEY = SORTD(J)
     35          101       CONTINUE
                           SORTD(K)=SORTD(I)
                           SORTD(I)=KEY
                100   CONTINUE
                      RETURN
     40               END
```

SYMBOLIC REFERENCE MAP (R=3)

ENTRY POINTS		DEF LINE	REFERENCES							
1	SORT	1	25	39						

VARIABLES	SN	TYPE	RELOCATION							
3	I	INTEGER		REFS	28	29	30	36	37	
				DEFINED	27					
620	INUM	INTEGER	SCRATCH	REFS	24	25	26	31		
6	I1	INTEGER		REFS	31	DEFINED	30			
7	J	INTEGER		REFS	32	33	34	DEFINED	31	
4	K	INTEGER		REFS	36	DEFINED	28	33		

```
      SUBROUTINE SORT     7600_7600 OPT=1                    FTN 4.6+452/034

 VARIABLES     SN  TYPE          RELOCATION
     5  KEY       INTEGER                          REFS      32      37    DEFINED    29      34
     2  NI        INTEGER                          REFS      27  DEFINED       26
     0  SORTD     INTEGER    ARRAY    SCRATCH      REFS      24      29        32      34      36
                                                   DEFINED   36      37

 STATEMENT LABELS              DEF LINE   REFERENCES
     0  100                       38        27
400000  101                       35        31        32

 COMMON BLOCKS   LENGTH     MEMBERS - BIAS NAME(LENGTH)
       SCRATCH     401              0 SORTD  (400)              400 INUM   (1)

 STATISTICS
   PROGRAM LENGTH                  108       8
   SCM LABELED COMMON LENGTH      6218     401
```

APPENDIX E:

ADS EXECUTION EXAMPLE

This appendix illustrates a complete ADS execution. Pages titled "CERL - ADS - MESSAGE" show the user input and any warning, severe or fatal messages produced by ADS. The remaining pages display the results of the user input.

TREE AS OF 4 MAY 78-RDIT

```
RDIT____
        |_SORTIN
        |_SORT
        |_SMOUT
        |_EOF
        |_ENDDK
        |_ENDCDK
```

```
TITLE.
RDIT - READ A SET OF DECKS TO EXCLUDE FROM GENERAL SORTING

PURPOSE.
THE PURPOSE OF THIS PROGRAM IS TO GENERATE MOVE DIRECTIVES TO
SORT AN OLDPL -- UPDATE LIBRARY.  UP TO THREE LEVELS OF DECKS CAN BE
EXCLUDED FROM THE TOTAL SORT (AND ARE THEN SORTED AMONGST
THEMSELVES).  COMDECKS ARE SORTED SEPARATELY FROM MAIN DECKS.  EACH
LEVEL MUST ALSO INPUT A DECK NAME THAT ALL DECKS IN THE LEVEL WILL
FOLLOW -- THE DIRECTIVE WILL BE OF FORM:
     "*MOVE <DECK NAME>,<INPUT DECK NAME>"
TO LEVEL DECKS ONE PUTS IN CARDS OF FORM:
     COL 1--9.  NAME OF EXCLUDED DECK
     COL. 10    BLANK. 1. OR 2
DECK NAMES ARE SORTED AND MOVE DIRECTIVES GENERATED SUCH THAT THE
FINAL ORDER OF THE UPDATE OLDPL WILL BE:
     COMDECKS.
     DECKS (INPUT CARD NAMES) WITH COL 10 = 1.
     DECKS (INPUT CARD NAMES) WITH COL 10 = 2.
     DECKS (NOT COMDECKS) NOT SPECIFIED ON INPUT CARDS.
     NOTE -- DECKS (INPUT CARD NAMES) WITH COL 10 = BLANK ARE NOT
     SORTED!  THESE DECKS CAN REMAIN IN PRESENT PLACES IN THE OLDPL
     AND BE USED FOR POSITIONING OTHER DECKS.
THE PROGRAM USES THE BASIC OUTPUT (L=A14) FROM THE UPDATE PROGRAM AS
INPUT (TAPE1).

VARIABLE DICTIONARY FOR ROUTINE RDIT    .

     CDECK    - NUMBER OF COMMON DECKS IN UPDATE LIST.

     DECKA    - NAME OF "AFTER" DECK FOR CURRENT LEVEL (ON OUTPUT).

     I        - LOOP CONTROL

     IKEY     - LEVEL FOR CURRENT SORT.

     IN       - NUMBER OF INPUTS <= 268.

     J        - LOOP CONTROL

     K        - LOOP CONTROL

     NDECK    - NUMBER OF DECKS IN LIST.

THIS A MAIN PROGRAM OF LENGTH      117 WORDS.
```

ROUTINES CALLED BY RDIT ARE --

ENDCDK ENDDK EOF SMOUT SORT SORTIN
AND MASK

COMMON BLOCKS CALLED BY RDIT ARE --

DECKS SCRATCH

THE ROUTINES WHICH CALL RDIT ARE -- ** NONE **

TITLE.
ENDCDK - DETERMINE END OF COMDECK LIST

PURPOSE.
THIS FUNCTION DETERMINES WHEN THE END OF COMDECK LIST HAS BEEN REACHED. IT ALSO DECODES THE LINE OF INPUT INTO THE COMDECK LIST TO BE SORTED.

VARIABLE DICTIONARY FOR ROUTINE ENDCDK .

CDECK - STARTING POSITION FOR COMDECKS.

ENDCDK - FUNCTION VALUE, WHEN TRUE, END OF COMDECK LIST

I - LOOP CONTROL.

INBUF - INPUT LINE (INPUT PARAMETER).

I1 - START CHARACTER POSITION OF DECK NAME

I2 - END CHARACTER POSITION OF DECK NAME

J - LOOP CONTROL

N - NUMBER OF DECK NAMES ALLOWED IN A LINE.

NDECK - CURRENT TOTAL NUMBER OF DECKS.

WORD - WORD FOUND I1..I2 POSITIONS, IF BLANK END OF LIST.

THIS IS A LOGICAL FUNCTION OF LENGTH 21 WORDS.

ROUTINES CALLED BY ENDCDK ARE -- ** NONE **

COMMON BLOCKS CALLED BY ENDCDK ARE --
DECKS

THE ROUTINES WHICH CALL ENDCDK ARE --
RDIT

```
TITLE.
ENDDK - DETERMINE END OF DECK LIST

PURPOSE.
THIS FUNCTION DETERMINES WHEN THE END OF DECK LIST HAS BEEN REACHED.
IT ALSO DECODES THE LINE OF INPUT INTO THE DECK LIST TO BE SORTED.

VARIABLE DICTIONARY FOR ROUTINE ENDDK  .

     ENDDK   - FUNCTION VALUE, WHEN TRUE, END OF DECK LIST HAS BEEN
               REACHED.

     I       - LOOP CONTROL.

     INBUF   - INPUT LINE (INPUT PARAMETER).

     I1      - START CHARACTER POSITION OF DECK NAME

     I2      - END CHARACTER POSITION OF DECK NAME.

     J       - LOOP CONTROL.

     N       - NUMBER OF POSSIBLE DECKS ON THIS LINE (INPUT
               PARAMETER).

     NDECK   - CUMULATIVE COUNT OF NUMBER OF DECKS (I/O PARAMETER).

     WORD    - WORD FOUND I1..I2 POSITIONS. IF BLANK END OF LIST.

THIS IS A LOGICAL    FUNCTION OF LENGTH      24 WORDS.

     ROUTINES CALLED BY ENDDK   ARE -- ** NONE **

     COMMON BLOCKS CALLED BY ENDDK   ARE --
DECKS

     THE ROUTINES WHICH CALL ENDDK   ARE --
RDIT
```

```
TITLE.
SMOUT - OUTPUT THE SORTED NAMES IN UPDATE DIRECTIVE FORM.

PURPOSE.
THIS ROUTINE OUTPUTS THE SORTED NAMES (ON TAPE2) IN UPDATE DIRECTIVE
FORM (*MOVE ...).

VARIABLE DICTIONARY FOR ROUTINE SMOUT  .

     DECKA    - DECK AFTER WHICH CURRENT LEVEL OF SORTED NAMES IS TO
                BE PLACED.

     I        - ** NONE *

     J        - ** NONE *

     K        - ** NONE *

     NAME     - DECK NAME IN CHARACTER FORMAT. SINCE UPDATE CAN
                NOT ACCEPT BLANKS BEFORE THE COMMA IN THE *MOVE
                DIRECTIVE.

THIS SUBROUTINE HAS A LENGTH OF       37 WORDS.

     ROUTINES CALLED BY SMOUT   ARE -- ** NONE **

     COMMON BLOCKS CALLED BY SMOUT   ARE --
SCRATCH

     THE ROUTINES WHICH CALL SMOUT   ARE --
RDIT
```

```
TITLE.
SORT - SORT A LIST OF NAMES

REFERENCES.
KNUTH, VOL III: SEARCHING AND SORTING;
ELSON, DATA STRUCTURES;
MANY, MANY MORE.

METHOD.
THE METHOD OF THIS SORT IS A SELECTION SORT. AT EACH STEP THE ITEM
WITH THE HIGHEST (LOWEST) VALUE IS SELECTED FROM THOSE REMAINING.

PURPOSE.
THIS ROUTINE ACCOMPLISHES THE ACTUAL SORT ON THE INPUT LIST OF NAMES.
THE LIST IS SORTED IN DESCENDING ORDER SO THAT THE EVENTUAL OUTPUT
WILL PLACE THE DECKS ON THE OLDPL IN PROPER ORDER.

VARIABLE DICTIONARY FOR ROUTINE SORT    .

     I        - LOOP CONTROL

     I1       - LOOP CONTROL (I+1)

     J        - LOOP CONTROL

     K        - LOOP CONTROL

     KEY      - CURRENT KEY

     N1       - LOOP CONTROL (NUMBER OF DECKS - 1)

THIS SUBROUTINE HAS A LENGTH OF        8 WORDS.

     ROUTINES CALLED BY SORT    ARE -- ** NONE **

     COMMON BLOCKS CALLED BY SORT    ARE --
SCRATCH

     THE ROUTINES WHICH CALL SORT    ARE --
RDIT
```

TITLE.
SORTIN - INPUT NAMES TO BE SORTED.

PURPOSE.
THIS ROUTINE IS CALLED ONCE FOR EACH NAME TO BE SORTED. THIS ROUTINE ACCUMULATES THOSE NAMES INTO A SORT LIST.

VARIABLE DICTIONARY FOR ROUTINE SORTIN .

DECKN - DECK NAME TO BE PLACED IN LIST (INPUT PARAMETER).

THIS SUBROUTINE HAS A LENGTH OF 8 WORDS.

ROUTINES CALLED BY SORTIN ARE -- ** NONE **

COMMON BLOCKS CALLED BY SORTIN ARE --
SCRATCH

THE ROUTINES WHICH CALL SORTIN ARE --
RDIT

VARIABLE DICTIONARY FOR ROUTINE SORT0 -- ** NONE **

THIS SUBROUTINE HAS A LENGTH OF 2 WORDS.

ROUTINES CALLED BY SORT0 ARE -- ** NONE **

COMMON BLOCKS CALLED BY SORT0 ARE --
SCRATCH

THE ROUTINES WHICH CALL SORT0 ARE -- ** NONE **

THE FOLLOWING RECORDS WERE NOT FOUND ON THE MASTER FILE --
EOF

THE ROUTINES WHICH CALL EOF ARE --
RDIT

```
TITLE.
DECKS - DECK NAMES AND FLAGS

VARIABLE DICTIONARY FOR COMMON BLOCK DECKS  .

     DECKS   - THE ARRAY OF DECK NAMES.

     DFLAGS  - THE CORRESPONDING ARRAY OF DECK FLAGS 0--3

     THE ROUTINES WHICH CALL DECKS   ARE --
RDIT      ENDDK     ENDCDK
```

```
TITLE.
SCRATCH - SCRATCH USE AREA.

VARIABLE DICTIONARY FOR COMMON BLOCK SCRATCH.

     INBUF   - INPUT BUFFER AREA FOR DECK LIST FROM UPDATE.
               EQUIVALENCED TO FIRST PART OF SORTD ARRAY.

     INFLG   - INPUT DECKS AND FLAGS. EQUIVALENCED TO PART OF SORTD
               ARRAY.

     INUM    - NUMBER OF NAMES TO BE SORTED BY ROUTINE SORT.

     SORTD   - ARRAY CONTAINING NAMES TO BE SORTED, NUMBER OF
               NAMES IS INUM.

     THE ROUTINES WHICH CALL SCRATCH ARE --
RDIT      SMOUT     SORT0     SORTIN    SORT
```

DEFINITION AS OF 4 MAY 78-ABSTRACT

REPORT ABSTRACT PRINTS THE FOLLOWING SECTIONS
IN THE ORDER LISTED

TITLE
AUTHOR
PURPOSE
METHOD
ALGORITHM
REFERENCES

REPORT DUMP PRINTS THE FOLLOWING SECTIONS
IN THE ORDER LISTED

TITLE
AUTHOR
DATE WRITTEN
REFERENCES
LOCATION
METHOD
CONTROL CARDS
REMARKS
SYSTEMS
PURPOSE
ALGORITHM
DATE REVISED
NON SYSTEM EXTERNALS
MACHINE DEPENDENCIES
SYSTEM DEPENDENCIES
IMPLEMENTATION DEPENDENCIES
FLOW
FILES
VARIABLE DICTIONARY
IODUM
ENTRY POINTS
ROUTINES CALLED
COMMON BLOCKS CALLED
CALLED BY ROUTINES
IODUM1(1)
IODUM1(2)
IODUM1(3)
IODUM1(4)
IODUM1(5)
IODUM1(6)

```
TITLE.
RDIT - READ A SET OF DECKS TO EXCLUDE FROM GENERAL SORTING

TITLE.
ENDCDK - DETERMINE END OF COMDECK LIST

TITLE.
ENDDK - DETERMINE END OF DECK LIST

TITLE.
SMOUT - OUTPUT THE SORTED NAMES IN UPDATE DIRECTIVE FORM.

TITLE.
SORT - SORT A LIST OF NAMES

TITLE.
SORTIN - INPUT NAMES TO BE SORTED.
```

THE FOLLOWING RECORDS WERE NOT FOUND ON THE MASTER FILE --
EOF

THE ROUTINES WHICH CALL EOF ARE --
RDIT

```
TITLE.
DECKS - DECK NAMES AND FLAGS

TITLE.
SCRATCH - SCRATCH USE AREA.
```

DEFINITION AS OF 4 MAY 78-**********

REPORT ABSTRACT PRINTS THE FOLLOWING SECTIONS
IN THE ORDER LISTED

TITLE
AUTHOR
PURPOSE
METHOD
ALGORITHM
REFERENCES

REPORT DUMP PRINTS THE FOLLOWING SECTIONS
IN THE ORDER LISTED

TITLE
AUTHOR
DATE WRITTEN
REFERENCES
LOCATION
METHOD
CONTROL CARDS
REMARKS
SYSTEMS
PURPOSE
ALGORITHM
DATE REVISED
NON SYSTEM EXTERNALS
MACHINE DEPENDENCIES
SYSTEM DEPENDENCIES
IMPLEMENTATION DEPENDENCIES
FLOW
FILES
VARIABLE DICTIONARY
IODUM
ENTRY POINTS
ROUTINES CALLED
COMMON BLOCKS CALLED
CALLED BY ROUTINES
IODUM1(1)
IODUM1(2)
IODUM1(3)
IODUM1(4)
IODUM1(5)
IODUM1(6)

DEFINITION

AS OF 4 MAY 78-**********

REPORT TITLES PRINTS THE FOLLOWING SECTIONS
IN THE ORDER LISTED

TITLE

CERL DISTRIBUTION

Chief of Engineers
ATTN: DAEN-ASI-L (2)
ATTN: DAEN-DSE/R. A. McMurrer
ATTN: DAEN-MPO-B
ATTN: DAEN-MPO
ATTN: DAEN-MPO-U
ATTN: DAEN-MPZ-A
ATTN: DAEN-MPR
ATTN: DAEN-RDL
Dept of the Army
WASH DC 20314

Chief of Engineers
ATTN: DAEN-PMS
Dept of the Army
WASH DC 20314
for forwarding to:
International Organization
for Standards
Central Secretariat
1, Rue de Varembe, 1211
Geneva, Switzerland

US Army Materiel Development
and Readiness Command
ATTN: DRCDE-DK (3)
5001 Eisenhower Ave
Alexandria, VA 22333

Commander
US Army Electronics Command
ATTN: DRSEL-TL-M/Mr. Tenzer
Fort Monmouth, NJ 07703

DOD ADP Policy Committee
Assistant Secretary of Defense
(Comptroller)
WASH DC 20301

DOD Standardization Area
Computer Aided Design and
Numerical Control
Naval Ship Engineering Center
Hyattsville, MD 20782

DOD Standardization Program
For Information Processing
Standards for Computers (IPSC)
Directorate of Data Automation
(AF/KRAX)
HQ, USAF
WASH DC 20330

Commander
HQ, XVIII Airborne Corps and
Fort Bragg
ATTN: AFZA-FE-EE
Fort Bragg, NC 28307

HQ, 7th Army Training Command
ATTN: AETTG-DEH (5)
APO New York 09114

Commander
HQ USAEREUR and 7th Army
ODCS/Engineer
ATTN: AEAEN-EH (4)
APO New York 09403

Commander
7th Army Combined Arms Training
Center
ATTN: AETTM-HRD-EHD
APO New York 09407

US Army Engr Div, Europe
ATTN: Technical Library (3)
APO New York 09757

Commander
V Corps
ATTN: AETVDEH
APO New York 09079

American National Standards
Institute
1430 Broadway
New York, NY 10018

ANSI X3 Committee
C/O CBEMA
1828 L Street, NW
WASH DC 20036

DOD Working Group on Computer
Documentation Standards
DOD/DOCN
The Pentagon
WASH DC 20301

DOD Working Group on Computer-
Generated Military Symbology
DOD/DISPLAY
The Pentagon
WASH DC 20301

Federal Information Processing
Standards Coordinating and
Advisory Committee
Dept of Commerce
WASH DC 20234

Library of Congress
Exchange and Gift Division
ATTN: Federal Documents Section
WASH DC 20540

Interagency Committee on Automatic
Data Processing
National Bureau of Standards
WASH DC 20234

National Bureau of Standards
Institute for Computer Sciences
and Technology
WASH DC 20234

Defense Documentation Center
ATTN: TCA (12)
Cameron Station
Alexandria, VA 22314

Commander
VII Corps
ATTN: AETSDEH
APO New York 09154

Commander
21st Support Command
ATTN: AEREH
APO New York 09325

Commander
US Army Berlin
ATTN: AEBA-EN
APO New York 09742

Commander
US Army Southern European Task Force
ATTN: AESE-ENG
APO New York 09168

Commander
US Army Installation Support
Activity, Europe
ATTN: AEUES-RP
APO New York 09403

LT Neil B. Hall, CEC, USNR (Code 100)
884-6366
U.S. Navy Public Works Center
Box 6, FPO San Francisco 96651

Lawrie, Linda
The automated documentation system : user manual. -- Champaign, IL : Construction Engineering Research Laboratory ; Springfield, VA : available from NTIS, 1979.
83 p. ; 27 cm. (Technical report ; E-147)

1. Electronic data processing documentation. I. Title. II. Series: U.S. Construction Engineering Research Laboratory. Technical report ; E-147.

DEPARTMENT OF THE ARMY
CONSTRUCTION ENGINEERING RESEARCH LABORATORY
CORPS OF ENGINEERS
P.O. BOX 4005
CHAMPAIGN, ILLINOIS 61820

OFFICIAL BUSINESS
PENALTY FOR PRIVATE USE $300

POSTAGE AND FEES PAID
DEPARTMENT OF THE ARMY
DOD - 314

THIRD CLASS