AFWAL-TR-85-3066
Volume III

CADS - A COMPUTER AIDED DESIGN SYSTEM
Volume III - Program Maintenance Manual

Michael C. Less
Susan Manuel
Rockwell International
North American Aircraft Operations (NAAO)
El Segundo, California 90009

October 1986

Final Report for Period December 1981 - May 1985

Approved for public release; distribution unlimited

FLIGHT DYNAMICS LABORATORY
AIR FORCE WRIGHT AERONAUTICAL LABORATORIES
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433-6553

86 11 19 047
NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture use, or sell any patented invention that may in any way be related thereto.

This report has been reviewed by the Office of Public Affairs (ASD/PA) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.

VICTORIA A. TISCHLER
Project Engineer
Design & Analysis Methods Group

FREDERICK A. PICCHIONI, Lt Col, USAF
Chief, Analysis & Optimization Branch

FOR THE COMMANDER

ROBERT M. BADER
Acting Chief
Structures & Dynamics Division

"If your address has changed, if you wish to be removed from our mailing list, or if the addressee is no longer employed by your organization please notify AFWAL/FRBA, W-PAFB, OH 45433 to help us maintain a current mailing list".

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.
This is the program maintenance manual for the Computer Aided Design System, "CADS." CADS is a pre and post processor for structural analysis and optimization programs based on the finite element method. The system supports five functional modules controlled by an Executive Monitor. All of these modules communicate with a database through a data manager. In addition a post output translator, CADSPP, is available which processes output from finite element programs, e.g. NASTRAN, directly into the data base. This report gives a detailed description of the internal structure of CADS for use in future maintenance and enhancement of the code. CADS uses two random access files to store the geometry and analysis program results for a finite element model. The geometry (GEOM) data base is used to store all the elements, grid points, and similar model information. The POST data base is used to store analysis results such as element stresses and forces, grid displacements and modal output. Detailed descriptions of the individual data records for the GEOM and POST data bases are given. In addition each subroutine or function is described. (CONTINUED ON REVERSE SIDE)
Subroutine descriptions include an outline of its purpose and approach, routine inputs and outputs, error messages, external calls, the argument list, a key variable list and a list of common blocks. A brief discussion on the installation of CADS is also included.
FOREWORD

This final report was prepared by Rockwell International, North American Aircraft Operations (NAAO), El Segundo, California for the Structures and Dynamics Division, Flight Dynamics Laboratory, (FDL) Wright-Patterson Air Force Base, Dayton, Ohio. The work was performed under Contract No. F33615-81-C-3229 which was initiated under Project No. 2401. Mrs. V. Tischler was the FDL project engineer for this effort.


The Rockwell program manager for this effort was Mr. M. C. Less, NAAO Advanced Structures and Materials Department. He was supported by Mrs. S. Manuel of the same department.

The work described in this report was begun in December 1981 and completed in May 1985. This report was submitted for publication in May 1985.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>2.0</td>
<td>INSTALLATION</td>
<td>3</td>
</tr>
<tr>
<td>3.0</td>
<td>CADS MAINTENANCE</td>
<td>8</td>
</tr>
<tr>
<td>4.0</td>
<td>DATA BASE DESCRIPTION</td>
<td>9</td>
</tr>
<tr>
<td>5.0</td>
<td>SUBROUTINE DESCRIPTIONS</td>
<td>29</td>
</tr>
<tr>
<td>6.0</td>
<td>ERROR MESSAGES</td>
<td>305</td>
</tr>
<tr>
<td></td>
<td>REFERENCES</td>
<td>306</td>
</tr>
</tbody>
</table>
### ILLUSTRATIONS

<table>
<thead>
<tr>
<th>Figure</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Modular Nature of CADS Software</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>CADSCOPY.COM File Listing</td>
<td>5</td>
</tr>
</tbody>
</table>

### TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>VAX Supplied CADS Files</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>Non-VAX Supplied CADS Files</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>GEOM Model Header</td>
<td>11</td>
</tr>
<tr>
<td>4</td>
<td>Node Header Record</td>
<td>12</td>
</tr>
<tr>
<td>5</td>
<td>Node Data Records</td>
<td>13/14</td>
</tr>
<tr>
<td>6</td>
<td>Element Header Record</td>
<td>16</td>
</tr>
<tr>
<td>7</td>
<td>Element Data Records</td>
<td>17/19</td>
</tr>
<tr>
<td>8</td>
<td>Property Data Records</td>
<td>20/23</td>
</tr>
<tr>
<td>9</td>
<td>Material Data Records</td>
<td>24/25</td>
</tr>
<tr>
<td>10</td>
<td>POST Data Base Header Record</td>
<td>27</td>
</tr>
<tr>
<td>11</td>
<td>POST Master Record</td>
<td>27</td>
</tr>
<tr>
<td>12</td>
<td>POST Node Data Records</td>
<td>28</td>
</tr>
<tr>
<td>13</td>
<td>POST Element Data Records</td>
<td>28</td>
</tr>
</tbody>
</table>
1.0 INTRODUCTION

The widespread use of a large variety of finite element (FE) analysis codes to perform structural analysis tasks has focused attention on a common Air Force and industry problem: the relatively large amount of time and effort required to perform data preparation, data validation, and resultant FE analysis tasks with existing state-of-the-art codes. This problem is further aggravated by the relatively slow, interactive response of mainframe time-sharing computer processing systems. To reduce time and effort, a computer-aided, advanced interactive graphics, minicomputer-based, finite element modeling system has been developed. This system includes mesh generation and validation capabilities as preprocessing functions as well as interactive graphic features for postprocessing the analysis code output data.

The Computer Aided Design System (cADS) software's most important aspects are that it is targeted for 32-bit minicomputer hardware, makes use of FORTRAN 77 and device independent graphics, and supports the definition of composite material elements. The CADS program utilizes VAX 11/780 hardware with secondary testing for transportability, having been performed on IBM 4341 and PRIME 850 hardware. CADS is modular in nature with various functional modules accessed through a common executive monitor and makes use of common data base routines, as shown in Figure 1.


This program Maintenance Manual contains information concerning the data bases, error handling, and installation procedures for the CADS and CADSPP programs. However, the majority of this manual is the detailed subroutine descriptions for the CADS and CADSPP software.
Figure 1. Modular Nature of CADS Software
2.0 INSTALLATION

The installation of the CADS software requires the loading of the CADS files to on-line disk storage. Depending on the hardware being used, the program source may have to be compiled and linked. Details of the necessary procedures are provided in the following paragraphs.

2.1 STANDARD SOURCE TAPE (VAX SYSTEM)

The CADS software is generally supplied as a standard 9-track, 1600 bpi magnetic tape containing a number of different files. It is created using the standard VAX/VMS COPY command to copy the original VAX files to the tape. The file numbers, names, and descriptions are given in Table 1. The source code, object decks, compiled listings, and test data files for the software are supplied. The VAX/VMS MOUNT and COPY commands are used to load the CADS files from the tape to disk. The MOUNT command is used to mount and attach the tape to a drive. The tape is VAX labeled as CADS and should be mounted on drive MTA0. Once mounted, the command

```
COPY MTA0:CADSCOPY.COM CADSCOPY.COM
```

will copy the first file from the tape to disk. This CADSCOPY.COM command file can then be executed to copy the remaining files from tape to disk. In all cases the files are named as given in Table 1. The CADSCOPY.COM file is listed in Figure 2.

After these files are on the disk, the CADS software is ready for link-editing and user testing. This is a straightforward process using the VAX LINK command. The library name for the DI-3000 graphics package is required in order to link the software for execution. This name is used with the /LIBR keyword of the LINK command to resolve the DI-3000 graphics package call statements. If the DI-3000 package has been installed following the standard vendor directions the link can be performed using the following command:

```
DI3LOAD CADSV11,CADSV12,CADSV13,CADSV14,CADSV15, T14
```
This command will link the CADS object decks (files 8-12, Table 1) with the correct DI-3000 core routines and Tektronix 4014 device driver. After linking the CADS object decks basic test cases should be executed to ensure that the executable load module is ready for general use. The CADSPP object deck should now be linked. It does not use auxiliary libraries. The software can be released for general use once this testing is complete.

**TABLE 1**

VAX SUPPLIED CADS FILES

<table>
<thead>
<tr>
<th>Number</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CADSCOPY.COM</td>
<td>Command file to copy the remaining files</td>
</tr>
<tr>
<td>2</td>
<td>CADSV11.FOR</td>
<td>CADS source code by alphabetic routine name</td>
</tr>
<tr>
<td>3</td>
<td>CADSV12.FOR</td>
<td>CADS source code</td>
</tr>
<tr>
<td>4</td>
<td>CADSV13.FOR</td>
<td>Object decks for source code</td>
</tr>
<tr>
<td>5</td>
<td>CADSV14.FOR</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>CADSV15.FOR</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>CADSPPP.FOR</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>CADSV11.OBJ</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>CADSV12.OBJ</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>CADSV13.OBJ</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>CADSV14.OBJ</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>CADSV15.OBJ</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>CADSPPP.OBJ</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>CADSV11.LIS</td>
<td>Compiled listings for source code</td>
</tr>
<tr>
<td>15</td>
<td>CADSV12.LIS</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>CADSV13.LIS</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>CADSV14.LIS</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>CADSV15.LIS</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>CADSPPP.LIS</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>NATUIN1.DAT</td>
<td>NATURAL generator input test cases</td>
</tr>
<tr>
<td>21</td>
<td>NATUIN2.DAT</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>NATUIN3.DAT</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>NAST tin1.DAT</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>NASTIN2.DAT</td>
<td>NASTRAN bulk input test cases</td>
</tr>
<tr>
<td>25</td>
<td>NASTIN3.DAT</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>ANALIN.DAT</td>
<td>ANALYZE bulk input test case</td>
</tr>
<tr>
<td>27</td>
<td>OPTIN.DAT</td>
<td>OPTSTAT bulk input test case</td>
</tr>
<tr>
<td>28</td>
<td>NASTBULK.DAT</td>
<td>NATUIN1 output as NASTRAN data</td>
</tr>
<tr>
<td>29</td>
<td>ANALBULK.DAT</td>
<td>NATUIN3 output as ANALYZE data</td>
</tr>
<tr>
<td>30</td>
<td>OPTBULK.DAT</td>
<td>NATUIN2 output as OPTSTAT data</td>
</tr>
<tr>
<td>31</td>
<td>NASTOUT3.DAT</td>
<td>NASTRAN analysis file output for input NASTIN3.DAT</td>
</tr>
<tr>
<td>32</td>
<td>ANALOUT.DAT</td>
<td>ANALYZE analysis file output</td>
</tr>
<tr>
<td>33</td>
<td>OPTOUT.DAT</td>
<td>OPTSTAT analysis file output</td>
</tr>
<tr>
<td>34</td>
<td>CADSCOMP.COM</td>
<td>Compiles and links CADS source</td>
</tr>
</tbody>
</table>

4
The CADS software is also available as source code and example data decks in card image format on magnetic tape for installation on non-VAX hardware. The files supplied for non-VAX systems are described in Table 2. The files are supplied as an unlabelled, 1600 bpi, 9-track, fixed block, multi-file magnetic tape. The blocks contain 20 records of 80 bytes (1 card) each with each file listed in Table 2 making up a separate tape file.
TABLE 2

NON-VAX SUPPLIED CADS FILES

<table>
<thead>
<tr>
<th>Number</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CADSV11.FOR</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>CADSV12.FOR</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>CADSV13.FOR</td>
<td>CADS source code by alphabetic routine name</td>
</tr>
<tr>
<td>4</td>
<td>CADSV14.FOR</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>CADSV15.FOR</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>CADSPP.FOR</td>
<td>CADSPP source code</td>
</tr>
<tr>
<td>7</td>
<td>NATUIN1.DAT</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>NATUIN2.DAT</td>
<td>NATURAL generator input test cases</td>
</tr>
<tr>
<td>9</td>
<td>NATUIN3.DAT</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>NASTIN1.DAT</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>NASTIN2.DAT</td>
<td>NASTRAN bulk input test cases</td>
</tr>
<tr>
<td>12</td>
<td>NASTIN3.DAT</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>OPTIN.DAT</td>
<td>OPTSTAT bulk input test case</td>
</tr>
<tr>
<td>14</td>
<td>ANALIN.DAT</td>
<td>ANALYZE bulk input test case</td>
</tr>
<tr>
<td>15</td>
<td>NASTOUT3.DAT</td>
<td>NASTRAN analysis file output for input</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NASTIN3.DAT</td>
</tr>
<tr>
<td>16</td>
<td>OPTOUT.DAT</td>
<td>OPTSTAT analysis file output</td>
</tr>
<tr>
<td>17</td>
<td>ANALOUT.DAT</td>
<td>ANALYZE analysis file output</td>
</tr>
<tr>
<td>18</td>
<td>NASTBULK.DAT</td>
<td>NATUIN1 output as NASTRAN data</td>
</tr>
<tr>
<td>19</td>
<td>OPTBULK.DAT</td>
<td>NATUIN2 output as OPTSTAT data</td>
</tr>
<tr>
<td>20</td>
<td>ANALBULK.DAT</td>
<td>NATUIN3 output as ANALYZE data</td>
</tr>
</tbody>
</table>

2.3 DESCRIPTIONS OF FILES

The CADSCOPY.COM file is a command list which copies the remaining tape
files to an on-line disk. CADSV11.FOR through CADSV15.FOR files are the source
code for the CADS software. These files contain all of the CADS routines in
alphabetical order with approximately 4500 source statements per file. The
CADSPP.FOR file contains the source statements for the CADSPP program routines
in alphabetical order. The CADSV11.OBJ through CADSPP.OBJ files contain the
object decks of the respective compiled source files, while CADSV11.LIS through
CADSPP.LIS contain the compiled listing files.
A series of test data files are contained in the next 14 files. These files were used to test the CADS software while it was in development and provided the examples for the Volume II "Users Guide" sample sessions. The NATUIN1.DAT through NATUIN3.DAT files contain NATURAL generation data test cases. The NASTIN1.DAT through NASTIN3.DAT are NASTRAN bulk data decks while the OPTIN.DAT and ANALIN.DAT files contain OPTSTAT and ANLAYZE input bulk data decks respectively. The NASTOUT3.DAT through ANALOUT.DAT files are output files containing finite element analysis results for input to CADSPP. The NASTBULK.DAT, OPTBULK.DAT, and ANALBULK.DAT are files which contain bulk data decks from CADS of the NATUIN1.DAT through NATUIN3.DAT data, respectively.

Finally, the last file is a command file which will compile the CADS and CADSPP source on the VAX and link them into executable files named CADS.EXE and CADSPP.EXE. This command file assumes that the DI-3000 graphics library is available to the user name performing the compilation.
3.0 CADS MAINTENANCE

The CADS software has been extensively documented internally through the use of comment statements embedded in the source code. The code is standard FORTRAN 77 and has proven readily transportable in the past. The only known system-dependent problem concerns the use of the record length parameter on the OPEN statement. IBM uses bytes for this parameter while DEC, PRIME, and CDC use words for the record length. In section 5.0 overview descriptions of each of the CADS subroutines are provided.

The general purpose, hidden line program developed by NASA and available through COSMIC (reference 3) was used to provide the basis for the CADS hidden line functions. Those routines have proven to be fairly stable; however, any problems encountered in them can be reported to COSMIC for evaluation and distribution to other users.

The DI-3000 device independent graphics package, from Precision Visuals, Incorporated (PVI), has been used to provide the interactive graphics functions for CADS. This proprietary, commercially available, package is maintained and supported by PVI. The individual installation license for DI-3000 would contain details of support for that installation. CADS uses release 4.0 of DI-3000.

Generally, the CADS code would be stored on a system disk accessible to those responsible for configuration control of the software for the user community. Changes and enhancements made to the code would typically be made on a routine-by-routine basis with validation testing before release for production use. For this reason, it would be efficient to store the CADS code as a library so that routines can be changed, compiled, and relinked on an individual basis. The executable would be stored as a file which would be accessible to all users. It should also have a test version for use in validation and verification testing. The standard DEC, IBM, or other system commands and procedures should be used to make the library, perform editing on the routines, recompile, and relink the code.
4.0 DATA BASE DESCRIPTION

4.1 BACKGROUND

The CADS program uses two random access files to store the geometry information and analysis program output results for a finite element model. The geometry (GEOM) data base is used to store all of the element, grid point, and similar model information. It is the data base required for the display and pre-processing CADS activities. It will be generated by CADS when an existing GEOM data base is not available.

The second data base is used for the display of various analysis program results. This is the POST data base and contains the element stresses and forces and the grid displacements and eigenvectors depending upon the user request and analysis program. This data base is generated by the CADSPP program for use by the CADS program.

Each data base is laid out in the same basic format of a master record pointing to header records, which in turn, point to the actual data records. Both data bases are accessed by the same set of input/output routines; IOHEAD, IOPAC, and IODB. The IODB routine performs the actual Fortran READ/WRITE direct access I/O functions on the data bases. The IOPAC routine packs and unpacks data arrays of multi-record information for I/O to the data base. The IOHEAD routine takes care of the header record information and full data base copy functions.

4.2 GEOMETRY DATA BASE

The geometry (GEOM) data base is attached to unit 1 during the CADS program execution. It can be saved as a permanent file for use in later sessions depending upon the user requirements. Each record contains 990 words of information which in turn depend upon the particular types of data being read into CADS and stored to the data base. These records are not stored in any particular order on the data base and their location depends upon the order of information fed into CADS during the READ module operations. That is, the GEOM data base is not specifically blocked so that certain sections contain only element materials and other sections contain only node coordinates and so on.
However, if all of the node data is read in or generated, and then all of the element data is defined then the resulting GEOM data base records will be packed one after the other.

Detailed descriptions of the individual data records for the GEOM data base are given in the following paragraphs. The order of the records is arbitrary with the obvious restriction that the correct pointers to the data records be stored in the appropriate header and master records so that the data elements may be retrieved as needed.

4.2.1 GEOM MODEL HEADER RECORD

The first record in the GEOM data base is the model header record. It has provisions for various model data and contains pointers to the node and element data stored in the data base. Table 3 lists the individual data elements of the model header. The pointers in the record point to the node and element data headers described next.

4.2.2 NODE DATA HEADER RECORD

The node header record is listed in Table 4. It is made up of a series of 14 word pointer key sets which point to the various types of node data or attributes stored for a particular model. For example, the fourth word points to the base coordinate system data for the nodes defined in this set of keys. CADS can use this key to go pick up the base coordinate information when it is required for displays and/or output.

4.2.3 NODE DATA RECORDS

The actual node data components are stored in various data records. Each record is 990 words long and is split into different sized matrices depending upon the data components being stored. Table 5 lists the components in each of these different node data records.
### TABLE 3

**GEOM MODEL HEADER**

<table>
<thead>
<tr>
<th>WORD</th>
<th>NAME</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-2</td>
<td>DATELA</td>
<td>Date of Last Access</td>
</tr>
<tr>
<td>3-4</td>
<td>DATEMC</td>
<td>Date Model was Created</td>
</tr>
<tr>
<td>5-6</td>
<td>TIMELA</td>
<td>Time of Last Access</td>
</tr>
<tr>
<td>7-8</td>
<td>NAMELU</td>
<td>Name of Last User (not-used)</td>
</tr>
<tr>
<td>9-10</td>
<td>NAMEUC</td>
<td>Name of Model Creator (not-used)</td>
</tr>
<tr>
<td>11-18</td>
<td>TITLE</td>
<td>Model Title</td>
</tr>
<tr>
<td>19</td>
<td>NSUB</td>
<td>Number of Model Substructures</td>
</tr>
<tr>
<td>20</td>
<td>NVRLA</td>
<td>Next Valid Record Number</td>
</tr>
<tr>
<td>21-22</td>
<td>NAMESH</td>
<td>Substructure Name (not-used)</td>
</tr>
<tr>
<td>23</td>
<td>NOSUBH</td>
<td>Substructure Number (not-used)</td>
</tr>
<tr>
<td>24</td>
<td>NODREC</td>
<td>Number of Node Header Record</td>
</tr>
<tr>
<td>25</td>
<td>NGEMP</td>
<td>Number of Element Header Record</td>
</tr>
<tr>
<td>26</td>
<td>NGROUP</td>
<td>Number of Substructure Element Groups</td>
</tr>
<tr>
<td>27</td>
<td>NONODE</td>
<td>Number of Substructure Nodes</td>
</tr>
<tr>
<td>28</td>
<td>MXNODE</td>
<td>Highest Node Number</td>
</tr>
<tr>
<td>29</td>
<td>NELMOD</td>
<td>Number of Model Elements</td>
</tr>
<tr>
<td>30</td>
<td>NTNODE</td>
<td>Number of Model Nodes</td>
</tr>
</tbody>
</table>

**NOTE:** Words 21 through 30 repeat for each substructure to a total of 97 substructures for the model.
<table>
<thead>
<tr>
<th>WORD</th>
<th>NAME</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NODE</td>
<td>First Node Number in Data Record</td>
</tr>
<tr>
<td>2</td>
<td>NODEL</td>
<td>Last Node Number in Data Record</td>
</tr>
<tr>
<td>3</td>
<td>NODENO</td>
<td>Number of Nodes in Data Record</td>
</tr>
<tr>
<td>4</td>
<td>KBSYS</td>
<td>Key to Base Coordinate Data</td>
</tr>
<tr>
<td>5</td>
<td>KINSYS</td>
<td>Key to Input Coordinate Data</td>
</tr>
<tr>
<td>6</td>
<td>KNODE</td>
<td>Key to Node IDs and Constraints</td>
</tr>
<tr>
<td>7</td>
<td>KLOAD</td>
<td>Key to External Load Data</td>
</tr>
<tr>
<td>8</td>
<td>NLOAD</td>
<td>Number of External Loads</td>
</tr>
<tr>
<td>9</td>
<td>KMOMNT</td>
<td>Key to External Moment Data</td>
</tr>
<tr>
<td>10</td>
<td>NMOMNT</td>
<td>Number of External Moments</td>
</tr>
<tr>
<td>11</td>
<td>KCOORD</td>
<td>Key to Coordinate System Data</td>
</tr>
<tr>
<td>12</td>
<td>KSEQGP</td>
<td>Key to NASTRAN SEQGP Data</td>
</tr>
<tr>
<td>13</td>
<td>KSPC1</td>
<td>Key to NASTRAN SPC1 Data</td>
</tr>
<tr>
<td>14</td>
<td>LCNO</td>
<td>Case Number for Loads/Moments</td>
</tr>
</tbody>
</table>

**NOTE:** Words 1 through 14 repeat up to 39 times as needed to meet the node data requirements. This provides for a total of some 12,000 nodes per model.
### TABLE 5

**NODE DATA RECORDS**

<table>
<thead>
<tr>
<th>WORD</th>
<th>NAME</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X</td>
<td>Base Coordinate System X Value</td>
</tr>
<tr>
<td>2</td>
<td>Y</td>
<td>Base Coordinate System Y Value</td>
</tr>
<tr>
<td>3</td>
<td>Z</td>
<td>Base Coordinate System Z Value</td>
</tr>
<tr>
<td>1</td>
<td>X,R,R</td>
<td>Input System - First Axis Value</td>
</tr>
<tr>
<td>2</td>
<td>Y,T,T</td>
<td>Input System - Second Axis Value</td>
</tr>
<tr>
<td>3</td>
<td>Z,Z,P</td>
<td>Input System - Third Axis Value</td>
</tr>
<tr>
<td>1</td>
<td>ID</td>
<td>Node Identification Number</td>
</tr>
<tr>
<td>2</td>
<td>NCONT</td>
<td>Packed Switch of Constraints</td>
</tr>
<tr>
<td>3</td>
<td>NCNCTC</td>
<td>Constraint Coordinate System ID</td>
</tr>
<tr>
<td>1</td>
<td>ID</td>
<td>Node Identification Number</td>
</tr>
<tr>
<td>2</td>
<td>SEQGP</td>
<td>NASTRAN SEQGP Number</td>
</tr>
<tr>
<td>3</td>
<td>CP</td>
<td>Node Coordinate System Number</td>
</tr>
</tbody>
</table>

**NOTE:** Words 1-3 are repeated up to 330 times per data record until each node has a set of base coordinate values.

<table>
<thead>
<tr>
<th>WORD</th>
<th>NAME</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SID</td>
<td>Case Identification Number</td>
</tr>
<tr>
<td>2</td>
<td>NODE</td>
<td>Node Number for External Force</td>
</tr>
<tr>
<td>3</td>
<td>CID</td>
<td>Coordinate System of Force</td>
</tr>
<tr>
<td>4</td>
<td>SCALE</td>
<td>Scale Factor for Force Vector</td>
</tr>
<tr>
<td>5</td>
<td>X</td>
<td>Force Vector Direction - X</td>
</tr>
<tr>
<td>6</td>
<td>Y</td>
<td>Force Vector Direction - Y</td>
</tr>
<tr>
<td>7</td>
<td>Z</td>
<td>Force Vector Direction - Z</td>
</tr>
</tbody>
</table>

**NOTE:** Words 1-7 are repeated up to 141 times per data record until all of the external load or moment values have been stored.
TABLE 5 (Continued)

NODE DATA RECORDS

<table>
<thead>
<tr>
<th>WORD</th>
<th>NAME</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CID</td>
<td>Coordinate System Id. Number</td>
</tr>
<tr>
<td>2</td>
<td>ISYS</td>
<td>Coordinate System Type Switch</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-1=Rect., -2=Cyl., -3=Sphere.</td>
</tr>
<tr>
<td>3</td>
<td>NODE1</td>
<td>First System Definition Node</td>
</tr>
<tr>
<td>4</td>
<td>NODE2</td>
<td>Second System Definition Node</td>
</tr>
<tr>
<td>5</td>
<td>NODE3</td>
<td>Third System Definition Node</td>
</tr>
<tr>
<td>6-12</td>
<td>Blank</td>
<td>Not Used</td>
</tr>
</tbody>
</table>

Note: Words 1-12 are repeated up to 82 times per data record until all of the coordinate system data is defined.

<table>
<thead>
<tr>
<th>WORD</th>
<th>NAME</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SID</td>
<td>SPC1 Set Identification Number</td>
</tr>
<tr>
<td>2</td>
<td>ICONT</td>
<td>Packed Word of Constraint Switch</td>
</tr>
<tr>
<td>3</td>
<td>NONODE</td>
<td>Number of Nodes with ICONT</td>
</tr>
<tr>
<td>4-n</td>
<td>NODES</td>
<td>Node Numbers of SPC1 Set Nodes</td>
</tr>
</tbody>
</table>

Note: Words 1-n are repeated until the SPC1 set is defined using up to 990 words per data record.
4.2.4 ELEMENT HEADER RECORD

The element header record maintains the pointers to the element attributes, such as the connectivity, sizes, and materials. These pointers are then used to retrieve the specific data records containing the requested data elements. This header/pointer procedure reduces data retrieval times since the software just has to search one pointer and one data record for a specific piece of data. Without the pointer procedure the entire set of element data records would have to be searched for a specific piece of data. Table 6 describes the element header record details.

4.2.5 ELEMENT DATA RECORDS

The element data records are described in Table 7. These records contain the connectivity information for the elements being stored.

4.2.6 PROPERTY DATA RECORDS

The property data records are described in Table 8. These records contain the size or property values for the property tables. These tables are used to apply the size data to the individual elements and are referenced by the individual element connectivity data records.

4.2.7 MATERIAL DATA RECORDS

The material data records are described in Table 9. These records contain the material component values for the material tables defined for the model. They are pointed to by the individual element connectivity data records to specify the materials of a particular element.

4.3 POST DATA BASE

The POST data base contains the results of an analysis program execution as stored by the CADSPP program for use in the CADS program. It uses the pointer record concept like the GEOM data base and contains four types of data records. The first two are the grid displacement and eigenvector data record types while the second two contain the element stress and forces data types.
# TABLE 6

## ELEMENT HEADER RECORD

<table>
<thead>
<tr>
<th>WORD</th>
<th>NAME</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NGRP</td>
<td>Group Number of the Elements</td>
</tr>
<tr>
<td>2</td>
<td>NETYPE</td>
<td>Element Type for the Group</td>
</tr>
<tr>
<td>3</td>
<td>NEL</td>
<td>Number of Elements in the Group</td>
</tr>
<tr>
<td>4</td>
<td>KECON</td>
<td>Key to Connectivities for the Group</td>
</tr>
<tr>
<td>5</td>
<td>KEID</td>
<td>Key to Element Numbers for the Group</td>
</tr>
<tr>
<td>6</td>
<td>KPROP</td>
<td>Key to Properties for the Group</td>
</tr>
<tr>
<td>7</td>
<td>MNNODE</td>
<td>Minimum Node Number in the Group</td>
</tr>
<tr>
<td>8</td>
<td>MXNODE</td>
<td>Maximum Node Number in the Group</td>
</tr>
<tr>
<td>9</td>
<td>KEYEX</td>
<td>Key to Extra BEAM or Layer Data</td>
</tr>
</tbody>
</table>

Note: Words 1-9 are repeated up to 99 times per data record until each group in the model has been defined.

| 10,1 | KM1 | Key to MAT1 Material Table |
| 10,2 | NM1 | Number of MAT1 Table Entries |
| 10,11 | KM2 | Key to MAT2 Material Table |
| 10,12 | NM2 | Number of MAT2 Table Entries |
| 10,41 | KM4 | Key to MAT4 Material Table |
| 10,42 | NM4 | Number of MAT4 Table Entries |
| 10,51 | KM5 | Key to MAT5 Material Table |
| 10,52 | NM5 | Number of MAT5 Table Entries |
| 10,61 | KMC | Key to MATC Material Table |
| 10,62 | NMC | Number of MATC Table Entries |

Note: Words in column ten of the element header record (i.e., 10,1 and 10,2, etc.) are used to point to the material property tables for the elements.
## TABLE 7

### ELEMENT DATA RECORDS

<table>
<thead>
<tr>
<th>WORD</th>
<th>NAME</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>EID</td>
<td>Element Number</td>
</tr>
<tr>
<td>2</td>
<td>MID</td>
<td>Material Number for Element</td>
</tr>
<tr>
<td>3</td>
<td>KMAT</td>
<td>Key to the Material Data Record</td>
</tr>
<tr>
<td>4</td>
<td>MATOFF</td>
<td>Offset in Record for Material</td>
</tr>
<tr>
<td>1</td>
<td>EID</td>
<td>Element Number</td>
</tr>
<tr>
<td>2</td>
<td>NANG</td>
<td>Number of Angles for the Element</td>
</tr>
<tr>
<td>3</td>
<td>KMAT</td>
<td>Key to Composite Material Table</td>
</tr>
<tr>
<td>4</td>
<td>LINOFF</td>
<td>Key/Offset to the Layer Data</td>
</tr>
<tr>
<td>1</td>
<td>NLAY</td>
<td>Number of Plies for this Angle</td>
</tr>
<tr>
<td>2</td>
<td>COID</td>
<td>Composite Material ID for Angle</td>
</tr>
<tr>
<td>3</td>
<td>LA</td>
<td>Orientation Angle</td>
</tr>
<tr>
<td>4</td>
<td>MINL</td>
<td>Minimum Number of Plies for Angle</td>
</tr>
<tr>
<td>5</td>
<td>MAXL</td>
<td>Maximum Number of Plies for Angle</td>
</tr>
</tbody>
</table>

Note: Repeat words 1 to 4 up to 240 times per data record to specify the element materials for the model group now being defined.

| 1    | B2CID                 | PIPE Element Continuation Number                |
| 2    | PA                    | Pin Flag for PIPE Element                       |
| 3    | PM                    | Moment Pin Flag for PIPE Element                |

Note: Repeat words 1 to 3 up to 330 times per data record to store continuation data for PIPE elements.

| 1    | BCID                  | BAR Element Continuation Number                 |
| 2    | PA                    | Pin Flag at end A for BAR Element               |
| 3    | PB                    | Pin Flag at end B for BAR Element               |
| 4    | Z1A                   | Bar Offsets at end A                            |
| 5    | Z2A                   | Bar Offsets at end B                            |
| 6    | Z3A                   | Bar Offsets at end B                            |
| 7    | Z1B                   | Bar Offsets at end B                            |
| 8    | Z2B                   | Bar Offsets at end B                            |
| 9    | Z3B                   | Bar Offsets at end B                            |

Note: Repeat words 1 to 9 up to 110 times per data record to store continuation data for BAR elements.
TABLE 7 (Continued)

ELEMENT DATA RECORDS

<table>
<thead>
<tr>
<th>WORD</th>
<th>NAME</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>G1</td>
<td>First Connectivity Node Number</td>
</tr>
<tr>
<td>2</td>
<td>G2</td>
<td>Second Connectivity Node Number</td>
</tr>
</tbody>
</table>

Note: Repeat words 1 to 2 up to 495 times per data record to specify the element connectivity for two noded groups of elements.

| 1    | G1   | First Connectivity Node Number                   |
| 2    | G2   | Second Connectivity Node Number                  |
| 3    | G3   | Third Connectivity Node Number                   |

Note: Repeat words 1 to 3 up to 330 times per data record to specify the element connectivity for three noded PIPE elements.

| 1    | G1   | First Connectivity Node Number                   |
| 2    | G2   | Second Connectivity Node Number                  |
| 3    | G3   | Third Connectivity Node Number                   |
| 4    | TH   | Material Orientation Angle                       |

Note: Repeat words 1 to 4 up to 247 times per data record to specify the element connectivity for three noded triangular elements.

| 1    | G1   | First Connectivity Node Number                   |
| 2    | G2   | Second Connectivity Node Number                  |
| 3    | G3   | Bar Axis Vector Direction Node                   |
| 4    | CBOFF| Bar Offset Information Pointer                   |

Note: Repeat words 1 to 4 up to 247 times per data record to specify the element connectivity for the two noded BAR elements.

| 1    | G1   | First Connectivity Node Number                   |
| 2    | G2   | Second Connectivity Node Number                  |
| 3    | G3   | Third Connectivity Node Number                   |
| 4    | G4   | Fourth Connectivity Node Number                  |

Note: Repeat words 1 to 4 up to 247 times per data record to specify the element connectivity for the four noded shear, twist, and tetrahedron elements.
TABLE 7 (Concluded)

ELEMENT DATA RECORDS

<table>
<thead>
<tr>
<th>WORD</th>
<th>NAME</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>G1</td>
<td>First Connectivity Node Number</td>
</tr>
<tr>
<td>2</td>
<td>G2</td>
<td>Second Connectivity Node Number</td>
</tr>
<tr>
<td>3</td>
<td>G3</td>
<td>Third Connectivity Node Number</td>
</tr>
<tr>
<td>4</td>
<td>G4</td>
<td>Fourth Connectivity Node Number</td>
</tr>
<tr>
<td>5</td>
<td>TH</td>
<td>Material Orientation Angle</td>
</tr>
</tbody>
</table>

Note: Repeat words 1 to 5 up to 198 times per data record to specify the element connectivity for the four noded quadrilateral plate and membrane elements.

<table>
<thead>
<tr>
<th>WORD</th>
<th>NAME</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>G1</td>
<td>First Connectivity Node Number</td>
</tr>
<tr>
<td>2</td>
<td>G2</td>
<td>Second Connectivity Node Number</td>
</tr>
<tr>
<td>3</td>
<td>G3</td>
<td>Third Connectivity Node Number</td>
</tr>
<tr>
<td>4</td>
<td>G4</td>
<td>Fourth Connectivity Node Number</td>
</tr>
<tr>
<td>5</td>
<td>G5</td>
<td>Fifth Connectivity Node Number</td>
</tr>
<tr>
<td>6</td>
<td>G6</td>
<td>Sixth Connectivity Node Number</td>
</tr>
<tr>
<td>7</td>
<td>TH</td>
<td>Material Orientation Angle</td>
</tr>
</tbody>
</table>

Note: Repeat words 1 to 7 up to 141 times per data record to specify the element connectivity for six noded TRIM6 higher order triangular membrane elements.

<table>
<thead>
<tr>
<th>WORD</th>
<th>NAME</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>G1</td>
<td>First Connectivity Node Number</td>
</tr>
<tr>
<td>2</td>
<td>G2</td>
<td>Second Connectivity Node Number</td>
</tr>
<tr>
<td>3</td>
<td>G3</td>
<td>Third Connectivity Node Number</td>
</tr>
<tr>
<td>4</td>
<td>G4</td>
<td>Fourth Connectivity Node Number</td>
</tr>
<tr>
<td>5</td>
<td>G5</td>
<td>Fifth Connectivity Node Number</td>
</tr>
<tr>
<td>6</td>
<td>G6</td>
<td>Sixth Connectivity Node Number</td>
</tr>
<tr>
<td>7</td>
<td>G7</td>
<td>Seventh Connectivity Node Number</td>
</tr>
<tr>
<td>8</td>
<td>G8</td>
<td>Eighth Connectivity Node Number</td>
</tr>
<tr>
<td>9</td>
<td>TH</td>
<td>Material Orientation Angle</td>
</tr>
</tbody>
</table>

Note: Repeat words 1 to 9 up to 110 times per data record to specify the element connectivity for eight noded QM8 higher order quadrilateral membrane elements.

<table>
<thead>
<tr>
<th>WORD</th>
<th>NAME</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>G1-G20</td>
<td>Connectivity Node Numbers 1-20 for the WEDGE, HEX1, and HEX2 Elements</td>
</tr>
</tbody>
</table>

Note: Repeat words 1 to 20 to define the solid element types for the model. Up to 165, 123, or 49 elements can be defined per data record depending upon the number of nodes required to specify the given element type. For instance 123 eight noded HEX1 elements can be defined, or 49 twenty noded HEX2 elements, or 165 six noded WEDGE elements.
### TABLE 8

**PROPERTY DATA RECORDS**

<table>
<thead>
<tr>
<th>WORD</th>
<th>NAME</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
<td>Cross-Sectional Rod Area</td>
</tr>
<tr>
<td>2</td>
<td>J</td>
<td>Torsional Constant for Rod</td>
</tr>
<tr>
<td>3</td>
<td>C</td>
<td>Torsional Stress Coefficient</td>
</tr>
<tr>
<td>4</td>
<td>NSM</td>
<td>Non-Structural Mass</td>
</tr>
<tr>
<td>5</td>
<td>MIN</td>
<td>Minimum Area</td>
</tr>
<tr>
<td>6</td>
<td>MAX</td>
<td>Maximum Area</td>
</tr>
</tbody>
</table>

**Note:** Repeat words 1 to 6 up to 165 times per data record to specify the axial rod properties for these elements.

| 1    | T1   | Thickness at Node 1 |
| 2    | T3   | Thickness at Node 3 |
| 3    | T5   | Thickness at Node 5 |
| 4    | NSM  | Non-Structural Mass |
| 5    | MIN  | Minimum Thickness |
| 6    | MAX  | Maximum Thickness |

**Note:** Repeat words 1 to 6 up to 165 times per data record to specify the 6-noded triangular membrane properties.

| 1    | T    | Element Thickness |
| 2    | NSM  | Non-Structural Mass |
| 3    | MIN  | Minimum Thickness |
| 4    | MAX  | Maximum Thickness |

**Note:** Repeat words 1 to 4 up to 247 times per data record to specify the properties for the membrane, shear, twist, and type 2 (CQUAD2, CTRIA2) bending elements.

| 1    | T1   | Membrane Sheet Thickness |
| 2    | MID2 | Material number for Bending |
| 3    | I    | Area Moment of Inertia |
| 4    | MID3 | Material number for Transverse Shear |
| 5    | T3   | Transverse Shear Thickness |
| 6    | NSM  | Non-Structural Mass |
| 7    | Z1   | Fiber Distance - 1 Side |
| 8    | Z2   | Fiber Distance - 2 Side |
| 9    | MIN  | Minimum T1 Thickness |
| 10   | MAX  | Maximum T1 Thickness |

**Note:** Repeat words 1 to 10 up to 99 times per data record to specify the properties for the TRIA1 and QUAD1 bending elements.
TABLE 8 (Continued)

PROPERTY DATA RECORDS

<table>
<thead>
<tr>
<th>WORD</th>
<th>NAME</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>T1</td>
<td>Thickness at Node 1</td>
</tr>
<tr>
<td>2</td>
<td>T3</td>
<td>Thickness at Node 3</td>
</tr>
<tr>
<td>3</td>
<td>T5</td>
<td>Thickness at Node 5</td>
</tr>
<tr>
<td>4</td>
<td>T7</td>
<td>Thickness at Node 7</td>
</tr>
<tr>
<td>5</td>
<td>NSM</td>
<td>Non-Structural Mass</td>
</tr>
<tr>
<td>6</td>
<td>MIN</td>
<td>Minimum Thickness</td>
</tr>
<tr>
<td>7</td>
<td>MAX</td>
<td>Maximum Thickness</td>
</tr>
</tbody>
</table>

Note: Repeat words 1 to 7 up to 141 times per data record to specify the values for 8-noded quadrilateral elements.

<table>
<thead>
<tr>
<th>WORD</th>
<th>NAME</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>K</td>
<td>Scalar Spring Value</td>
</tr>
<tr>
<td>2</td>
<td>GE</td>
<td>Damping Coefficient</td>
</tr>
<tr>
<td>3</td>
<td>S</td>
<td>Stress Coefficient</td>
</tr>
</tbody>
</table>

Note: Repeat words 1 to 3 up to 330 times per data record to specify the properties for the ELAS spring elements.

<table>
<thead>
<tr>
<th>WORD</th>
<th>NAME</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CID</td>
<td>Coordinate System ID for Element Material Reference System</td>
</tr>
<tr>
<td>2</td>
<td>NIP</td>
<td>No. of Integration Points/Side</td>
</tr>
<tr>
<td>3</td>
<td>AR</td>
<td>Maximum Aspect Ratio</td>
</tr>
<tr>
<td>4</td>
<td>ALFA</td>
<td>Maximum Angle Between Normals of two Subtriangles of a Face</td>
</tr>
<tr>
<td>5</td>
<td>BETA</td>
<td>Maximum Angle Between the Vector Connecting a Corner Point to an Adjacent Midside Point and the Vector Connecting that Point to the Other Midside or Corner Point.</td>
</tr>
</tbody>
</table>

Note: Repeat words 1 to 5 up to 198 times per data record to specify the properties for the CIHEX1 and CIHEX2 solid elements.
<table>
<thead>
<tr>
<th>WORD</th>
<th>NAME</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>OD</td>
<td>Outside Diameter of PIPE</td>
</tr>
<tr>
<td>2</td>
<td>T</td>
<td>PIPE Wall Thickness</td>
</tr>
<tr>
<td>3</td>
<td>NSM</td>
<td>Non-Structural Mass</td>
</tr>
<tr>
<td>4</td>
<td>P</td>
<td>Internal Pressure</td>
</tr>
<tr>
<td>5</td>
<td>C1</td>
<td>Stress Recovery Coefficients</td>
</tr>
<tr>
<td>6</td>
<td>C2</td>
<td>Stress Recovery Coefficients</td>
</tr>
<tr>
<td>7</td>
<td>D1</td>
<td>Minimum OD Value</td>
</tr>
<tr>
<td>8</td>
<td>D2</td>
<td>Maximum OD Value</td>
</tr>
<tr>
<td>9</td>
<td>E1</td>
<td>Stress Recovery Coefficients</td>
</tr>
<tr>
<td>10</td>
<td>E2</td>
<td>Stress Recovery Coefficients</td>
</tr>
<tr>
<td>11</td>
<td>F1</td>
<td>Stress Recovery Coefficients</td>
</tr>
<tr>
<td>12</td>
<td>F2</td>
<td>Stress Recovery Coefficients</td>
</tr>
<tr>
<td>13</td>
<td>MIN</td>
<td>Minimum OD Value</td>
</tr>
<tr>
<td>14</td>
<td>MAX</td>
<td>Maximum OD Value</td>
</tr>
</tbody>
</table>

Note: Repeat words 1 to 14 up to 70 times per data record to specify the properties for the PIPE elements.

<table>
<thead>
<tr>
<th>WORD</th>
<th>NAME</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
<td>Cross-Sectional Area of BAR</td>
</tr>
<tr>
<td>2</td>
<td>I1</td>
<td>Area Moment of Inertia - 1</td>
</tr>
<tr>
<td>3</td>
<td>I2</td>
<td>Area Moment of Inertia - 2</td>
</tr>
<tr>
<td>4</td>
<td>J</td>
<td>Torsional Constant</td>
</tr>
<tr>
<td>5</td>
<td>NSM</td>
<td>Non-Structural Mass</td>
</tr>
<tr>
<td>6</td>
<td>C1</td>
<td>Stress Recovery Coefficients</td>
</tr>
<tr>
<td>7</td>
<td>C2</td>
<td>Stress Recovery Coefficients</td>
</tr>
<tr>
<td>8</td>
<td>D1</td>
<td>Stress Recovery Coefficients</td>
</tr>
<tr>
<td>9</td>
<td>D2</td>
<td>Stress Recovery Coefficients</td>
</tr>
<tr>
<td>10</td>
<td>E1</td>
<td>Stress Recovery Coefficients</td>
</tr>
<tr>
<td>11</td>
<td>E2</td>
<td>Stress Recovery Coefficients</td>
</tr>
<tr>
<td>12</td>
<td>F1</td>
<td>Stress Recovery Coefficients</td>
</tr>
<tr>
<td>13</td>
<td>F2</td>
<td>Stress Recovery Coefficients</td>
</tr>
<tr>
<td>14</td>
<td>K1</td>
<td>Area Factor for Shear - 1</td>
</tr>
<tr>
<td>15</td>
<td>K2</td>
<td>Area Factor for Shear - 2</td>
</tr>
<tr>
<td>16</td>
<td>I12</td>
<td>Area Moment of Inertia</td>
</tr>
<tr>
<td>17</td>
<td>MIN</td>
<td>Minimum A Value</td>
</tr>
<tr>
<td>18</td>
<td>MAX</td>
<td>Maximum A Value</td>
</tr>
</tbody>
</table>

Note: Repeat words 1 to 18 up to 55 times per data record to specify the properties for the BAR elements.
### TABLE 8 (Concluded)

#### PROPERTY DATA RECORDS

<table>
<thead>
<tr>
<th>WORD</th>
<th>NAME</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PHI1-PHI14</td>
<td>Aximuthal Coordinates (degrees) for Stress Recovery.</td>
</tr>
</tbody>
</table>

Note: Repeat words 1 to 14 up to 70 times per data record to specify up to 14 stress recovery coefficients for the TRIAAX and TRAPAX elements.

### TABLE 9

#### MATERIAL DATA RECORDS

<table>
<thead>
<tr>
<th>WORD</th>
<th>NAME</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>MID</td>
<td>Material Identification Number</td>
</tr>
<tr>
<td>2</td>
<td>E</td>
<td>Young's Modulus</td>
</tr>
<tr>
<td>3</td>
<td>G</td>
<td>Shear Modulus</td>
</tr>
<tr>
<td>4</td>
<td>U</td>
<td>Poisson's Ratio</td>
</tr>
<tr>
<td>5</td>
<td>RHO</td>
<td>Density</td>
</tr>
<tr>
<td>6</td>
<td>A</td>
<td>Thermal Coefficient</td>
</tr>
<tr>
<td>7</td>
<td>TREF</td>
<td>Reference Temperature</td>
</tr>
<tr>
<td>8</td>
<td>GE</td>
<td>Damping Coefficient</td>
</tr>
<tr>
<td>9</td>
<td>ST</td>
<td>Tension Stress Allowable</td>
</tr>
</tbody>
</table>

Note: Repeat words 1 to 15 up to 66 times per data record to specify the properties for TRIA3 and QUAD4 elements.
### TABLE 9 (Concluded)

**MATERIAL DATA RECORDS**

<table>
<thead>
<tr>
<th>WORD</th>
<th>NAME</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>SC</td>
<td>Compression Stress Allowable</td>
</tr>
<tr>
<td>11</td>
<td>SS</td>
<td>Shear Stress Allowable</td>
</tr>
</tbody>
</table>

Note: Repeat words 1 to 11 for up to 99 times until all MAT1 isotropic materials are stored for the model.

| 1    | MID  | Material Identification Number |
| 2    | A11  | Elastic Modulus Array following |
| 3    | A12  | the NASTRAN Naming Convention |
| 4    | A13  | |
| 5    | A22  | $\sigma = \begin{bmatrix} A11 & A12 & A13 \\ A22 & A23 & \varepsilon \\ A33 & & \end{bmatrix}$ |
| 6    | A23  | |
| 7    | A33  | |
| 8    | RHO  | Density |
| 9    | A1   | |
| 10   | A2   | Array of Thermal Expansion Coefficient |
| 11   | A3   | |
| 12   | TREF | Reference Temperature |
| 13   | GE   | Damping Coefficient |
| 14   | ST   | Tension Stress Allowable |
| 15   | SC   | Compression Stress Allowable |
| 16   | SS   | Shear Stress Allowable |

Note: Repeat words 1 to 16 for up to 61 times until all MAT2 anisotropic materials are stored for the model.

<table>
<thead>
<tr>
<th>WORD</th>
<th>NAME</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>MID</td>
<td>Material Identification Number</td>
</tr>
<tr>
<td>2</td>
<td>K</td>
<td>Thermal Conductivity</td>
</tr>
<tr>
<td>3</td>
<td>CP</td>
<td>Thermal Capacity/Unit Volume</td>
</tr>
</tbody>
</table>

Note: Repeat words 1 to 3 for up to 330 times until all MAT4 isotropic thermal materials are stored for the model.
### TABLE 9 (Concluded)

**MATERIAL DATA RECORDS**

<table>
<thead>
<tr>
<th>WORD</th>
<th>NAME</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>MID</td>
<td>Material Identification Number</td>
</tr>
<tr>
<td>2</td>
<td>KXX</td>
<td>Thermal Conductivity Matrix</td>
</tr>
<tr>
<td>3</td>
<td>KXY</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>KXZ</td>
<td>KXX KXY KXZ</td>
</tr>
<tr>
<td>5</td>
<td>KYY</td>
<td>KXY KYY KYZ</td>
</tr>
<tr>
<td>6</td>
<td>KYZ</td>
<td>KXZ KYZ KZZ</td>
</tr>
<tr>
<td>7</td>
<td>KZZ</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>CP</td>
<td>Thermal Capacity/Unit Volume</td>
</tr>
</tbody>
</table>

Note: Repeat words 1 to 8 for up to 123 times until all MAT5 anisotropic thermal materials are defined and stored.

<table>
<thead>
<tr>
<th>WORD</th>
<th>NAME</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>MID</td>
<td>Material Identification Number</td>
</tr>
<tr>
<td>2</td>
<td>ET</td>
<td>Transverse Uniaxial Modulus</td>
</tr>
<tr>
<td>3</td>
<td>EL</td>
<td>Longitudinal Uniaxial Modulus</td>
</tr>
<tr>
<td>4</td>
<td>GL</td>
<td>Shear (LT) Uniaxial Modulus</td>
</tr>
<tr>
<td>5</td>
<td>UL</td>
<td>Uniaxial Poisson's Ratio (LT)</td>
</tr>
<tr>
<td>6</td>
<td>T</td>
<td>Layer Thickness</td>
</tr>
<tr>
<td>7</td>
<td>M</td>
<td>Reference Percent Moisture</td>
</tr>
<tr>
<td>8</td>
<td>TE</td>
<td>Reference Temperature</td>
</tr>
<tr>
<td>9</td>
<td>AL</td>
<td>Longitudinal Thermal Expansion Coefficient</td>
</tr>
<tr>
<td>10</td>
<td>AT</td>
<td>Transverse Thermal Expansion Coefficient</td>
</tr>
<tr>
<td>11</td>
<td>BL</td>
<td>Longitudinal Moisture Expansion Coefficient</td>
</tr>
<tr>
<td>12</td>
<td>BT</td>
<td>Transverse Moisture Expansion Coefficient</td>
</tr>
<tr>
<td>13</td>
<td>DEN</td>
<td>Density of a Layer</td>
</tr>
<tr>
<td>14</td>
<td>FTL</td>
<td>Longitudinal Tension Allowable</td>
</tr>
<tr>
<td>15</td>
<td>FTT</td>
<td>Transverse Tension Allowable</td>
</tr>
<tr>
<td>16</td>
<td>FLT</td>
<td>Shear (LT) Allowable</td>
</tr>
<tr>
<td>17</td>
<td>FCL</td>
<td>Longitudinal Compression Allowable</td>
</tr>
<tr>
<td>18</td>
<td>FCT</td>
<td>Transverse Compression Allowable</td>
</tr>
<tr>
<td>19</td>
<td>IB</td>
<td>Balanced Laminate Clue</td>
</tr>
<tr>
<td>20</td>
<td>Blank</td>
<td>Not Used</td>
</tr>
</tbody>
</table>

Note: Repeat words 1 to 20 up to 49 times per data record until the MATC layered composite materials are stored.
4.3.1 POST HEADER RECORD

The POST header record is described in Table 10. The POST header record contains the counters, analysis program name, data types loaded and corresponding master record pointers for the POST data base.

4.3.2 POST MASTER DATA RECORD

The MASTER record is the pointer to the actual data records for a particular data type. Table 11 describes the format of the MASTER records. The MASTER records basically contain sets of condition numbers, identifiers, data record numbers, and counter information for the particular analysis data type for that master record.

4.3.3 POST DATA RECORDS - GRIDS

The POST data records for grid data are described in Table 12. It must be recognized that only those data types supported by a specific analysis program can be stored. For example, NASTRAN can provide eigenvector and element force data while ANALYZE and OPTSTAT will only support the displacement and stress data types.

4.3.4 POST DATA RECORDS - ELEMENTS

The POST data records for element data are described in Table 13. It must be recognized that only those data types supported by a specific analysis program can be stored. For example, NASTRAN can provide eigenvector and element force data while ANALYZE and OPTSTAT will only support the displacement and stress data types. The terminology used for the component names is the same as that used for Tables 6-8 in the "CADS User's Guide," Volume II of this final report.
### TABLE 10

**POST DATA BASE HEADER RECORD**

<table>
<thead>
<tr>
<th>WORD</th>
<th>NAME</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NOCOND</td>
<td>Number of Conditions Stored</td>
</tr>
<tr>
<td>2</td>
<td>NVREC</td>
<td>Next Valid Record</td>
</tr>
<tr>
<td>3</td>
<td>NSTEP</td>
<td>Number of Time Steps Stored</td>
</tr>
<tr>
<td>4</td>
<td>NCOND</td>
<td>Condition Number of Next Keys</td>
</tr>
<tr>
<td>5</td>
<td>KFORCE</td>
<td>Key to Force Master Record</td>
</tr>
<tr>
<td>6</td>
<td>KSTRES</td>
<td>Key to Stress Master Record</td>
</tr>
<tr>
<td>7</td>
<td>KDISP</td>
<td>Key to Displacement Master Record</td>
</tr>
<tr>
<td>8</td>
<td>KEIGEN</td>
<td>Key to Eigenvector Master Record</td>
</tr>
<tr>
<td>9-17</td>
<td>Blank</td>
<td>Not Used; for Expansion Keys</td>
</tr>
<tr>
<td>18</td>
<td>MONTH</td>
<td>Month Data was Loaded</td>
</tr>
<tr>
<td>19</td>
<td>DAY</td>
<td>Day Data was Loaded</td>
</tr>
<tr>
<td>20</td>
<td>YEAR</td>
<td>Year Data was Loaded</td>
</tr>
<tr>
<td>21-35</td>
<td>TITLE</td>
<td>60 Character Load Case Title</td>
</tr>
</tbody>
</table>

Note: Repeat words 4 to 35 for up to 61 times until all load cases are stored. This header record is composed of two 990 word physical records.

### TABLE 11

**POST MASTER RECORD**

<table>
<thead>
<tr>
<th>WORD</th>
<th>NAME</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NEMAST</td>
<td>Number of Entries in Master Record</td>
</tr>
<tr>
<td>2-4</td>
<td>Blank</td>
<td>Not Used</td>
</tr>
<tr>
<td>5</td>
<td>NCOND</td>
<td>Condition Number</td>
</tr>
<tr>
<td>6</td>
<td>NODE</td>
<td>Largest Node in Node Data Record</td>
</tr>
<tr>
<td>7</td>
<td>NREC</td>
<td>Pointer to Node Data Record</td>
</tr>
<tr>
<td>8</td>
<td>NGRIDS</td>
<td>Number of Sets of Data in Record</td>
</tr>
</tbody>
</table>

Note: Repeat words 5 to 8 for up to 246 times until all load cases are stored and node data records are pointed to by this master record.

### TABLE 11

<table>
<thead>
<tr>
<th>WORD</th>
<th>NAME</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NEMAST</td>
<td>Number of Entries in Master Record</td>
</tr>
<tr>
<td>2-4</td>
<td>Blank</td>
<td>Not Used</td>
</tr>
<tr>
<td>5</td>
<td>NCOND</td>
<td>Condition Number</td>
</tr>
<tr>
<td>6</td>
<td>ETYP'E</td>
<td>Element Type for Data Record</td>
</tr>
<tr>
<td>7</td>
<td>NREC</td>
<td>Pointer to Node Analysis Data Record</td>
</tr>
<tr>
<td>8</td>
<td>NEL</td>
<td>Number of Sets of Data in Record</td>
</tr>
</tbody>
</table>

Note: Repeat words 5 to 8 for up to 246 times until all load cases are stored and element data records are pointed to by this master record.
### TABLE 12
POST NODE DATA RECORDS

<table>
<thead>
<tr>
<th>WORD</th>
<th>NAME</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ID</td>
<td>Node Identification Number</td>
</tr>
<tr>
<td>2</td>
<td>TX</td>
<td>Translational - X Value</td>
</tr>
<tr>
<td>3</td>
<td>TY</td>
<td>Translational - Y Value</td>
</tr>
<tr>
<td>4</td>
<td>TZ</td>
<td>Translational - Z Value</td>
</tr>
<tr>
<td>5</td>
<td>RX</td>
<td>Rotational - X Value</td>
</tr>
<tr>
<td>6</td>
<td>RY</td>
<td>Rotational - Y Value</td>
</tr>
<tr>
<td>7</td>
<td>RZ</td>
<td>Rotational - Z Value</td>
</tr>
</tbody>
</table>

Note: Repeat words 1 to 7 for up to 141 times per data record until all of the values are stored for each node. The values may be either displacement or eigenvectors.

### TABLE 13
POST ELEMENT DATA RECORDS

<table>
<thead>
<tr>
<th>WORD</th>
<th>NAME</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ID</td>
<td>Element Identification Number</td>
</tr>
<tr>
<td>2-n</td>
<td>VALUES</td>
<td>Element Component Values</td>
</tr>
</tbody>
</table>

Note: Repeat words 1 to n for each element for each case. The VALUES are the element stress or force components as listed in Tables 6-8 of the Volume II, "User's Guide." They are stored in the same order, for each element type, as listed in those tables.
5.0 SUBROUTINE DESCRIPTIONS

Each of the subroutines or functions developed for the CADS program is described in this section. The DI-3000 graphics routines and system-supplied routines are not included. The routine descriptions include an outline of its purpose and approach, routine inputs and outputs, error messages, external calls, the argument list, a key variable list, and a list of common blocks used in the routine.

The subroutines are listed in alphabetical order which is the order provided in the source code and program listing tape files. The naming convention used for the CADS subroutines attempts to group those routines making up a certain module into a single segment. For example, routines dealing with reading data start with RD; natural generation routines start with NAT; output routines with OUT; plotting routines with PLT, and utility routines with UTL. The characters following the leading characters are generally abbreviations of the command or keyword the routine processes.

The first group of routines includes the Executive Monitor and Block Data routines of CADS. These are followed by the BOX and CHANGE subroutines.
**Program:**

CADS

**Algorithm:**

This is the main controller for the CADS program. It sets initial switches, opens the geometric and post data base files and calls the routines which are the high level controllers for the various CADS functional modules.

**Input/Output:**

Opens terminal message unit 7.

Unit 7 - terminal output for messages

Unit 5 - terminal input

RDCARD - Free read command input

**Error Messages:**

**** MAIN OPTION COMMAND ____ NOT FOUND

**External Calls:**

OUTPUT  PLOTCN  RDCARD  RDCONT  START  EDITCT

**Argument List:**

None

**Important Variables:**

NBLANK - number of words in BLANK (scratch) common

**Common Blocks:**

<table>
<thead>
<tr>
<th>READ</th>
<th>PLOTCN</th>
<th>SCALAR</th>
<th>MAT12</th>
<th>TYPEN</th>
<th>TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLOTC</td>
<td>PLOTEL</td>
<td>HEDG</td>
<td>CHAR</td>
<td>SOLIDS</td>
<td>PLTITL</td>
</tr>
<tr>
<td>NASTRN</td>
<td>READN</td>
<td>D1TOKD</td>
<td>G03</td>
<td>DIBAUD</td>
<td>MOHEAD</td>
</tr>
<tr>
<td>DAVE</td>
<td>PINFLA</td>
<td>PLOTBD</td>
<td>PLOTB2</td>
<td>TEMP</td>
<td>DBREC</td>
</tr>
<tr>
<td>TKTRNX</td>
<td>NOHEAD</td>
<td>MATL</td>
<td>ELHEAD</td>
<td>HEADPP</td>
<td>SYSTEM</td>
</tr>
<tr>
<td>MATPRO</td>
<td>READCM</td>
<td>NATDSP</td>
<td>BLANK</td>
<td>TRACK1</td>
<td>PERM</td>
</tr>
<tr>
<td>PLOT</td>
<td>OPTIND</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

30
Block Data: MATPRO/TYPEN/

Algorithm: This block data initializes the MATPRO and TYPEN labeled common blocks.

Input/Output: None

Error Messages: None

External Calls: None

Argument List: None

Important Variables: DMAT1 - defines material commands for material types MAT1, MAT2, MAT4, MAT5
DMAT2 - property definition command array
NUMEL - element type numbers for NASTRAN elements
NUPEL - number of property values per element
NUTEL - NASTRAN numbers for material and property cards
NUVEL - number of values/property type

Common Blocks: MATPRO
TYPEN
**Block Data:**
CHAR/NASTRN/SOLIDS

**Algorithm:**
Block data initializes arrays for the NASTRAN elements.

**Input/Output:**
None

**Error Messages:**
None

**External Calls:**
None

**Argument List:**
None

**Important Variables:**
- **FORM** - NASTRAN bulk data cards supported
- **NTERM** - number of items per NASTRAN card
- **NSTOR** - number of terms to be stored/element type
- **NNODE** - number of nodes per element
- **GRIDAX** - grid type data cards
- **NGTERM** - number of terms per grid type cards
- **NU** - character array
- **ITA** - array contains the number of lines defining a higher order element
- **IT20** - number of line segments for the S020 element
- **J6A** - array of line endpoints for the TM6 element
- **J8A** - array of line endpoints for the QM8 element
- **I4A** - array of line endpoints for the S04 element
- **I6A** - array of line endpoints for the S06 element
- **I8A** - array of line endpoints for the S08 element
- **I20A** - array of line endpoints for the S020 element

**Common Blocks:**
CHAR
NASTRN
SOLIDS
Block Data: PLOTBD

Algorithm: Block Data procedure used to initialize the PLOTBD and PLOTB2 common block variables for the DISPLAY module. The valid analysis output type, element pointer, and output component name arrays are initialized.

Input/Output: None

Error Messages: None

External Calls: None

Argument List: None

Important Variables:
- OUTNAM - character array with valid analysis output type names
- KEY1 - pointer array into K2 and K3 based on the element type
- K2 - array with number of valid components per data type
- K3 - array with offsets into the component name arrays
- MASKDF - displacement component names
- MASKST - stress component names
- MASKFO - force component names

Common Blocks: PLOTBD
                 PLOTB2
Subroutine: BOX

Algorithm: This routine determines which nodes are inside a given geometric shape. It works with the SET module on the BOX command and, through the SPHERE, CYLNDR, and SLAB entry points, processes those commands. For each command the routine cycles through all previously defined nodes and checks their coordinates to determine if they are inside the given geometric shape using standard geometric procedures.

Input/Output: Unit 7 - terminal output for messages

Error Messages: NO NODES LOCATED WITHIN BOX SET __ DEFINED BY, XL = __ XU = __ YL = __ YU = __ ZL = __ ZU = __
VECTOR NODES FOR CYLINDER OPTION FOR SET __ UNDEFINED
NODE KEYS WERE __ __ __ __
CENTER NODE FOR SPHERE OPTION FOR SET __ UNDEFINED
NODE KEYS WERE __ __
VECTOR NODES FOR SLAB OPTION FOR SET __ UNDEFINED
NODE KEYS WERE __ __ __ __ __ __

External Calls: IFINDN

Argument List: XL,XU,YL,YU,ZL,ZU - lower and upper ranges on the box axes
KOUNT - number of nodes found
LNOC - node number of nodes found
NAME - set name of node set
KPR - error message switch
Argument List (Continued)

<table>
<thead>
<tr>
<th>ENTRY</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CYLNR ENTRY</td>
<td>N1,N2 - nodes defining the cylinder center line</td>
</tr>
<tr>
<td></td>
<td>R       - radius</td>
</tr>
<tr>
<td></td>
<td>KOUNT, LNOC, NAME, KPR - same as BOX</td>
</tr>
<tr>
<td>SPHERE ENTRY</td>
<td>N1     - sphere center point</td>
</tr>
<tr>
<td></td>
<td>R       - radius</td>
</tr>
<tr>
<td></td>
<td>KOUNT, LNOC, NAME, KPR - same as BOX</td>
</tr>
<tr>
<td>SLAB ENTRY</td>
<td>N1,N2,N3 - nodes defining the SLAB plane</td>
</tr>
<tr>
<td></td>
<td>R       - distance (thickness) in the positive normal direction</td>
</tr>
<tr>
<td></td>
<td>TT      - distance (thickness) in the negative normal direction</td>
</tr>
<tr>
<td></td>
<td>KOUNT, LNOC, NAME, KPR - same as BOX</td>
</tr>
</tbody>
</table>

Important Variables: Same as argument list

Common Blocks: BLANK, PERM
Subroutine: CHANGE

Algorithm: This routine takes a series of eight characters and converts them from the free format input to a real number.

Input/Output: Unit 5 - terminal input
Unit 7 - terminal output for messages

Error Messages: **ERROR** ___ IS AN ILLEGAL REAL NUMBER RE-ENTER:

External Calls: None

Argument List: A - array of real numbers returned from CHANGE
LHOLD - 8 by NWORD character array of input characters
NWORD - number of sets of characters to convert

Important Variables: Same as argument list.

Common Blocks: SYSTEM
CHAR
The following block of subroutines makes up the EDIT module of CADS. These routines are used to perform the editing functions of CADS thus allowing changes to the GEOMETRY data base. The routines in this block are:

EDITCA
EDITC1
EDITC2
EDITCT
EDITEL
EDITE1
EDITE2
EDITE3
EDITE4
EDITE5
EDITE6
EDITE7
EDITE8
EDITE9
EDITMA
EDITND
EDITPR
EDIT10
EDIT11
Subroutine: EDITCA

Algorithm: This routine edits the NASTRAN, ANALYZE, and OPTSTAT executive and case control decks at the terminal. It provides delete, list, replace and add functions for editing. These commands are processed on a line by line basis and the updated information is stored back to the data base.

Input/Output: Unit 7 - terminal output for messages
RDCARD - free read command input
IOPAC - packed data base I/O routine
NDBUNT - geometry data base output

Error Messages: ** ERROR OPTION _____ NOT FOUND **

** NO CASE CONTROL CARDS FOR _____ **

INCORRECT LIST REQUEST CARD IGNORED

INSUFFICIENT TERMS ENTERED FOR REPLACE/INSERT OPTION

External Calls: EDITC1 NUMBER
EDITC2 RDCARD
IOPAC UTLLTG

Argument List: None

Important Variables: None

Common Blocks: READ DBREC
NOHEAD MATL
READCM
Subroutine: EDITC1

Algorithm: This routine deletes and inserts cards in the case control files. It first processes inserts by placing the new cards into the correct location in the file. Next it deletes the user specified case control cards out of the case file.

Input/Output: None

Error Messages: None

External Calls: None

Argument List: IG switch is used to define delete or insert processing
NL the number of case control cards to be inserted
K2 the number of entries in LIST
LIST array which holds the card numbers for processing

Important Variables: Same as argument list.

Common Blocks: READ
MATL
**Subroutine:**

EDITC2

**Algorithm:**

This routine reads in new case control cards for insertion or addition to the existing case control file.

**Input/Output:**

Unit 7 - terminal output for messages
NUNIT - user input unit (usually 5)

**Error Messages:**

None

**External Calls:**

None

**Argument List:**

PROMPT - the prompt string for the user
NUNIT - input unit for the user commands
HOLD - buffer array to hold commands

**Important Variables:**

Same as argument list.

**Common Blocks:**

None
Subroutine: EDITCT

Algorithm: Routine controls the edit routine for changing nodes, elements, materials, and properties. Opens a save file, reads the user commands, sets switches, and calls in the processing routines.

Input/Output: Unit 5 - terminal input
              Unit 7 - terminal output for messages
              IOHEAD - data base header record
              RDCARD - free read command input

Error Messages: *** ERROR ** OPEN ERROR ON UNIT ___ STATUS ___

*** MODULE: ____ CARD: ____

EDIT CONTROL OPTION ____ NOT VALID

EDIT PROCESSOR ____ DOES NOT EXIST

EDIT PROCESSOR HAS NOT BEEN ENTERED CORRECTLY

*** ERROR ** OPEN ERROR ON UNIT ___ STATUS ___
FILE WAS NOT SAVED

External Calls: EDITEL  EDITPR  OUTGRD
                EDITMA  IOHEAD  RDCARD
                EDITND  NUMBER  EDITCA

Argument List: None

Important Variables: None

Common Blocks: READ  MOHEAD  PERM
                BLANK  SYSTEM
                READCM  DBREC
Subroutine: EDITEL

Algorithm: Routine gets the element connectivity tables for editing. Processes the element edit commands and calls in the routine to act on those commands. Acts as the edit element control routine.

Input/Output: Unit 7 - terminal output for messages
RDCARD - free read command input

Error Messages: ** EDIT OPTION _____ NOT FOUND **
** INSUFFICIENT TERMS ENTERED FOR DELETE OPTION **

External Calls: EDITE1 EDITE6
EDITE3 LIGRNO
EDITE5 RDCARD

Argument List: None

Important Variables: None

Common Blocks: READ PERM
BLANK SYSTEM
Subroutine: EDITE1

Algorithm: This routine lists the element data for the edit element LIST command. It decodes the user commands, retrieves the element group tables and sets switches for groups and elements to be processed. Finally, it lists the requested element data to the terminal.

Input/Output: IOPAC - geometric data base read/write
Unit 7 - terminal output for messages

Error Messages:

*** INCORRECT SYNTAX ELEMENT LIST, CARD IGNORED __ __ __

*** GROUP NUMBER ______ DOES NOT EXIST, CARD IGNORED

*** INCORRECT LIST REQUEST, CARD IGNORED

*** FOLLOWING ELEMENTS DO NOT EXIST, LIST IGNORED

External Calls:

EDITE2
IOPAC UTLLTG
NUMBER ZRAYI

Argument List: None

Important Variables: NGR - group numbers for the list
LIST - array with the list of elements
LICH - array with non-existent elements

Common Blocks: READ ELHEAD MATL SYSTEM
TYPE BLANK NASTRN
CHAR DBREC PERM
Subroutine: EDITE2

Algorithm: This routine is called by EDITE1 to list the specific element connectivities and element numbers. It simply cycles through the element list outputting information until the list is exhausted.

Input/Output: Unit 7 - terminal output for messages

Error Messages: None

External Calls: None

Argument List: IC    - element connectivity array
NSS   - number of values/element
NSN   - number of nodes/element
NELMT - number of elements to be listed
LIST  - list of requested elements
IID   - element numbers array
NEL   - maximum number of elements in the group

Important Variables: Same as argument list

Common Blocks: None
Subroutine: EDITE3

Algorithm: This routine deletes a group or list of elements in a group. It checks the command syntax, finds the group, and gets the user element list when needed. Then it retrieves the element connectivities and calls EDITE4 to process the actual delete operation. Finally, it writes the element data back to the data base.

Input/Output: Unit 7 - terminal output for messages
IOPAC - packed array I/O to geometry data base

Error Messages:
** INCORRECT, DELETE SYNTAX, CARD IGNORED **
** GROUP NUMBER ___ DOES NOT EXIST, CARD IGNORED **
** INCORRECT, DELETE LIST REQUEST, CARD IGNORED **
** FOLLOWING ELEMENTS DO NOT EXIST, DELETE IGNORED **

External Calls: EDITE4 UTLLTG
IOPAC ZRAYI NUMBER

Argument List: None

Important Variables: NELGRH - group/element pointer array
NGR - group number
LIST - list of elements being processed
D - scratch array with element data

Common Blocks: CHAR BLANK NASTRN
READ DBREC PERM
ELHEAD MATL SYSTEM
Subroutine: EDITE4

Algorithm: This routine deletes elements from a group and compresses the group of elements. It cycles through the list of elements, zeros out their values, and compresses the group array.

Input/Output: None

Error Messages: None

External Calls: None

Argument List: IC - element connectivity array
NSS - number of connectivity values per element
NELD - number of elements to be deleted
NELMT - total number of elements in the group
LIST - list of requested elements to be deleted
IID - element ID array
IP - element property array
NP - number of property values per element

Important Variables: Same as argument list

Common Blocks: None
Subroutine: EDITE5

Algorithm: This routine sets up a list of elements to be changed or added in EDITE7. It decodes user input commands, sets appropriate switches and pointers and gets the element list. Then it calls the EDITE7 routine and finally updates the element connectivities on the geometry data base.

Input/Output: Unit 7 - terminal output for messages
RDCARD - free read input
IOPAC - packed array I/O to geometry data base

Error Messages:
** INCORRECT ADD OR CHANGE SYNTAX, CARD IGNORED

** GROUP NUMBER _____ DOES NOT EXIST, CARD IGNORED

** INCORRECT ADD OR CHANGE LIST REQUEST, CARD IGNORED

** NODE NUMBERS DON'T MATCH LIST OF ELEMENTS, CARD IGNORED

** NODE NUMBER _____ DOES NOT EXIST, CARD IGNORED

[FOLLOWING ELEMENTS DO NOT EXIST DELETE IGNORED.]

External Calls: EDITE7 NUMBER UTLLTG
IFINDN OUTGRD ZRAYI
IOPAC RDCARD

Argument List: None

Important Variables: NELGRH - group/element data pointers array
LIST - list of elements to be processed
NVREC - geometry data base record number for elements
KAC - switch for add/change
<table>
<thead>
<tr>
<th>Common Blocks:</th>
<th>CHAR</th>
<th>ELHEAD</th>
<th>MOHEAD</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ</td>
<td>DBREC</td>
<td></td>
<td>PERM</td>
</tr>
<tr>
<td>BLANK</td>
<td>NASTRN</td>
<td></td>
<td>MATL</td>
</tr>
</tbody>
</table>
Subroutine: EDITE6

Algorithm: This routine lists the valid keywords and commands for the element, property, and material editors. Provides a shorthand help function for the edit routines.

Input/Output: Unit 7 - terminal output for messages

Error Messages: ** INCORRECT SYNTAX FOR HELP USE: HELP EL **

** GROUP NO. _____ DOES NOT EXIST, CARD IGNORED**

** ELEMENT TYPE _____ NOT SUPPORTED, CARD IGNORED**

External Calls: NUMBER

Argument List: None

Important Variables: NUMEL - element type to be supported
NUPEL - property type to be supported
NUTEL - type in the material table
NUVEL - number of values per property type
MP - definition array for keywords
MAP - material keyword array

Common Blocks: READ ELHEAD TYPEN
MATPRO NASTRN
CHAR PERM
Subroutine: EDITE7

Algorithm: Routine adds or changes elements for a previously defined group based upon element number or offset values. Cycles through the element list updating node connectivity tables. These are then passed back for storage to the geometry data base.

Input/Output: None

Error Messages: None

External Calls: None

Argument List:
- IC - element connectivity array
- NSS - number of values per element
- NSN - number of nodes per element
- NELD - number of elements to be added or changed
- NELMT - total number of elements in the group
- LIST - list of requested elements to be added or changed
- LISTN - node list for requested elements
- IID - element ID array
- KSW - 1 element numbers are inputted; 2 element offsets are inputted
- KAC - 1 change element connectivity request; 2 add element connectivity request
- K - group number

Important Variables: Same as argument list

Common Blocks: None
Subroutine: EDITE8

Algorithm: This routine is used to decode property values passed from EDITPR. First it checks the keywords against the valid list; decodes the values and stores them in appropriate positions in the data array for the given element. Finally it gets the element ID list from the data base and compares it for matches to the requested element list.

Input/Output: Unit 7 - terminal output for messages  
IOPAC - packed geometry data base I/O

Error Messages: ** INCORRECT CHANGE REQUEST, CARD IGNORED **

** DESCRIPTION ___ NOT VALID FOR PROPERTY TYPE, CARD IGNORED **

** FOLLOWING ELEMENTS DO NOT EXIST, DELETE IGNORED **

External Calls: CHANGE UTLLTG
IOPAC ZRAYI
NUMBER

Argument List: DMP - array of valid property keywords
I1 - number of integers in DMP
NIN - locations of the integers in DMP
* - alternate return

Important Variables: Same as argument list

Common Blocks: READ DBREC
PERM MATL
TEMP
Subroutine: EDITE9

Algorithm: Routine changes material table pointers for a given list of elements. Determines type of element list; generates list of numbers and updates pointer array with new material keys.

Input/Output: Unit 7 - terminal output for messages

Error Messages: ** INCORRECT CHANGE REQUEST, CARD IGNORED **
** FOLLOWING ELEMENTS DO NOT EXIST, CHANGE IGNORED **
** REQUESTED MAT ID ___ DOES NOT EXIST, CHANGE IGNORED **

External Calls: UTLITG
ZRAYI

Argument List:
IMAT - material data array
LV - number of variables (components) in IMAT
NMAT - number of materials (rows) in IMAT
KEY - key pointer into the material table

Important Variables: Same as argument list

Common Blocks: READ MATL
PERM
TEMP
Subroutine:  EDITMA

Algorithm:  This routine edits the material property table as defined by the user. First, it reads and decodes the input command line. Next, it passes control to the LIST, HELP, CHANGE and ADD processing sections. The routine finishes by updating the geometry data base using IOPAC. It retrieves the appropriate material table based on the requested data types, updates the table and packs it on the geometry data base.

Input/Output:  RDCARD - free read terminal input
               Unit 7 - terminal output for messages
               IOPAC - packed geometry data base I/O
               IODB - performs geometry data base I/O

Error Messages:
** EDIT OPTION _____ NOT FOUND **

** MATERIAL TABLE _____ DOES NOT EXIST **

** INSUFFICIENT TERMS ENTERED FOR CHANGE OPTION **

** GROUP _____ CANNOT BE LOCATED; USE LI GR TO LIST EXISTING GROUPS **

External Calls:  EDITE1  EDIT10  NUMBER
                 EDITE6  IOPAC  RDCARD
                 EDITE9  LIGRNO  IODB

Argument List:  None

Important Variables:  LIST - array of values to be processed
                     NELGRH - element/group pointer array
                     MID   - material table number
                     DMATP - array of material values
                     LV    - number of variables per element type
                     NVREC - record being processed from geometry data base
Common Blocks:

<table>
<thead>
<tr>
<th>MATPRO</th>
<th>ELHEAD</th>
<th>TEMP</th>
<th>MATL</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ</td>
<td>READCM</td>
<td>MOHEAD</td>
<td>TYPEN</td>
</tr>
<tr>
<td>BLANK</td>
<td>DBREC</td>
<td>SYSTEM</td>
<td>PERM</td>
</tr>
</tbody>
</table>
Subroutine: EDITNDD

Algorithm: Routine edits the node values as defined by the user. Will LIST, DELETE, CHANGE, or ADD nodes. Routine transfers to the appropriate command execution array and decodes the appropriate parameters. Makes up a list of nodes to be operated on and then performs the requested operation.

Input/Output: RDCARD - free read from the terminal
Unit 7 - terminal output for messages

Error Messages:
** EDIT OPTION ____ NOT FOUND
** INCORRECT SYNTAX FOR THE LIST GENERATION **
** NODE NUMBER ____ DOES NOT EXIST **
** INSUFFICIENT TERMS ENTERED FOR DELETE OPTION **
** INCORRECT SYNTAX FOR CHANGE COMMAND
** NODE NUMBER ____ DOES NOT EXIST, CHANGE IGNORED

External Calls: CHANGE NUMBER RDCARD ZRAYI
IFINDN OUTGRD RDN Pac
NATNOD UTLLTG

Argument List: None

Important Variables: NCXYZ - array of node data
LIST - list of nodes to be processed
NONODE - number of nodes
KZ - number of nodes in the LIST array
Common Blocks:

- READ
- BLANK
- DBREC
- NATDSP
- NOHEAD
- PERM
- MATL
- MOHEAD
- SYSTEM
Subroutine: EDITPR

Algorithm: Routine is used to edit the property values for user defined elements. Will list valid property values through the HELP command as well as group/elements through LIST. Decodes the user command, transfers to the appropriate command section and processes the command. The CHANGE section retrieves the property block, updates the values as defined by the user, and restores them to the data base.

Input/Output: RDCARD - free read terminal input
Unit 7 - terminal output for messages
IOPAC - packed array I/O to geometry data base

Error Messages: ** EDIT OPTION ____ NOT FOUND
** INSUFFICIENT TERMS ENTERED FOR CHANGE OPTION
** GROUP ____ CANNOT BE LOCATED, USE LI GR TO LIST EXISTING GROUPS
** NO PROPERTY TABLE FOR ELEMENT TYPE ____

External Calls: EDIT11 IOPAC RDCARD
EDIT6 LIGRN0 ZRAYB
EDIT8 PRSTR3 NUMBER

Argument List: None

Important Variables: NELGRH - array of pointers to element/group data
LOC - location of data in the property array
LV - number of values per type

Common Blocks: TYPE BLANK TEMP MATL PERM
MATPRO ELHEAD DBREC SYSTEM TYPEN
READ READCM MOHEAD NASTRN

57
Subroutine: EDIT10

Algorithm: This routine is used to decode the material property values. It checks the input keyword against the valid names and then places the new value in the appropriate data array.

Input/Output: Unit 7 - terminal output for messages

Error Messages: DESCRIPTION ___ NOT VALID FOR MATERIAL TYPE, CARD IGNORED

External Calls: CHANGE

Argument List: DMP - character array of valid keywords
                LV - number of valid keywords
                IMAT - material pointer array
                NMAT - number of materials

Important Variables: same as argument list

Common Blocks: READ MATL
                PERM
                TEMP
Subroutine: EDIT11

Algorithm: This routine decodes the property values for EDITPR. It checks the element list and gets the valid elements from the user defined list. It then gets the properties for that element list and prints them out to the terminal as a list of element numbers and property values.

Input/Output: Unit 7 - terminal output for messages

Error Messages:
** INCORRECT LIST PROPERTY REQUEST, CARD IGNORED **
** FOLLOWING ELEMENTS DO NOT EXIST LIST IGNORED **

External Calls: EDITE6 ULLTG
IOPAC ZRAYI

Argument List: D real array for the property values
ND integer array for the property values
NST number of components (rows) in the property arrays
DMP valid property types for requested element
I1 number of property types which have integer values
NIN pointer array to property types which are integer

Important Variables: Same as argument list.

Common Blocks: READ DBREC
PERM MATL
TEMP
The following block of subroutines and functions are general routines used for storing data to the data base, packing data types, and finding grid numbers. The routines in this block are:

FREPCK
FRESTR
GETMAT
GROUPS
IFINDN
IODB
IOHEAD
IOPAC
IOROUT
ISIMEQ
JFINDG
JFINDN
LIGRNO
LINPTS
MAIDCH
MASTR1
MATPCK
MPSTR1
Subroutine: FREPCK

Algorithm: This routine packs the node suppressions array ISWS into the word IPCK for packed storage to the data base. The entry FREUCK unpacks the word into an array. If the particular freedom switch in ISWS is set, i.e. ISWS(I) = 1 for I = 1,6, IPCK is multiplied by 10 and the appropriate I value is added to IPCK. The FREUCK entry point reverses the process by dividing by 10.

Input/Output: None

Error Messages: None

External Calls: None

Argument List:
- ISWS - valued array for suppressions
- IPCK - packed word of suppressions (output)

FREUCK ENTRY
- JSWS - array of suppressions (output)
- JPCK - packed word of suppressions

Important Variables: Same as argument list.

Common Blocks: None
Subroutine: FRESTR

Algorithm: This routine stores the grid suppressions for a given node to the required grid array position. Entry FREFRE frees the suppression values previously defined. This routine calls FREUCK and FREPCK to unpack and pack words as needed to save the suppressions information.

Input/Output: None

Error Messages: None

External Calls: FREPCK
FREUCK

Argument List: N5 - node location in the grid array for a given suppression
KPCK - packed suppressions
ISWS - unpacked suppressions
IC - code for the coordinate system type

FREFRE ENTRY N5,ISWS,IC - same as FRESTR

Important Variables: Same as argument list.

Common Blocks: None
Subroutine: GETMAT

Algorithm: Routine gets material properties from the geometric data base. It is a lower level routine which uses the number of materials and values per material to determine the location of the material data and obtain the pointers to the correct data base location.

Input/Output: Calls on IOPAC routine to access the geometric data base.

Error Messages: None

External Calls: ZRAYI, IOPAC

Argument List: NSET - storage array for material properties
N1 - beginning location in NSET for each material type
IA - offset location in the header record per material
IROW - number of values per material
NOMAT - array of record numbers and offsets per data set
II - number of input data sets

Important Variables: Same as argument list.

Common Blocks: ELHEAD, DBREC
Subroutine: GROUPS

Algorithm: Routine develops group values as elements are generated or read onto the geometric data base. Values are placed in an in-core matrix and are then used to pack group data for output to the geometry data base. GROUPS calls IOPAC to pack the element group data for the data base.

Input/Output: Unit 7 - terminal output for messages

Error Messages: *** THE NUMBER OF GROUPS EXCEEDS INCORE CAPACITY ***

External Calls: IOPAC

Argument List: MATRIX - array of information with element connectivities
N1 - number of elements in the group
IPROP - property values for the group (new)
N2 - number of properties
IOLD - old properties
N3 - number of previous values

Important Variables: Same as argument list.

Common Blocks: DBREC
ELHEAD
PERM
TEMP
Function: IFINDN

Algorithm: Function checks to see if a node has been previously defined by using a binary search. If it does exist, its location is returned. If it does not exist, a flag is set and the position at which it would be inserted into the node table is returned.

Input/Output: None

Error Messages: None

External Calls: None

Argument List: NODE - node number to search on
               ILL - location for the new node

Important Variables: Same as the argument list.

Common Blocks: BLANK
               PERM
Subroutine: IODB

Algorithm: Routine reads from and writes to the geometric data base on a direct access record basis.

Input/Output: Direct access READ of a given record from a given unit. Direct access WRITE to a given record on a given unit. Unit 7 - Terminal output for messages

Error Messages: REQUEST ERROR FROM SUBROUTINE

External Calls: None

Argument List: IG - switch for read or write: 1 = read; 2 = write
A - buffer array for I/O
N - number of values in array A
IREC - record number of the data
NUNIT - unit number of the data
SUBNAM - character name of the calling routine for the error message.

Important Variables: Same as argument list.

Common Blocks: None
Subroutine: IOHEAD

Algorithm: Routine saves or retrieves the header records on the geometry data base. It calls the IODB routine for these functions. It will also perform a straight copy from one unit to another of the entire data base.

Input/Output: IODB - geometry data base I/O
READ - record by record read for the copy function
WRITE - record by record write for the copy function

Error Messages: None

External Calls: IODB

Argument List: IG - switch for the command 1 = retrieve;
               2 = save; 3 = copy
N1 - unit to read from for copy
N2 - unit to write to for copy
SUBNAM - character name of calling routine

Important Variables: NVRLA - next record in the GEOM data base

Common Blocks: NOHEAD PERM
               ELHEAD MOHEAD
               DBREC
**Subroutine:** IOPAC

**Algorithm:** This routine packs an array of information for sending to the data base. It blocks the data to the direct access record size of the data base and calls IODB to perform the actual I/O function.

**Input/Output:** None

**Error Messages:** None

**External Calls:** IODB

**Argument List:**
- ARRAY - data array
- NWRD - number of words to be stored or retrieved
- IG - switch to read or write data: 1 = read; 2 = write
- NDBUNT - data base unit number
- SUBNAM - name of the calling routine for the error message in IODB

**Important Variables:** Same as argument list.

**Common Blocks:** DBREC
Subroutine: IOROUT

Algorithm: Performs record I/O for temporary files. Reads or writes blocks or buffers of data and stores their locations and pointers in the MAT array for later processing. Uses IODB routine to perform the actual I/O.

Input/Output: None

Error Messages: None

External Calls: IODB

Argument List:
- IO - reads data if =1; writes data if =2
- NAM - four character matrix name for data to be stored
- NROW - number of rows in the matrix
- NCOL - number of columns in the matrix
- IDATE - calendar date
- MATRIX - matrix of information

Important Variables: Same as argument

Common Blocks: TRACK1
**Function:** ISIMEQ

**Algorithm:** Solves simultaneous linear equations using the determinant and pivot method.

**Input/Output:** None

**Error Messages:** None

**External Calls:** None

**Argument List:**
- DSM - size of the coefficient matrix
- NE - actual number of equations for this call
- NC - number of columns in the constant matrix
- A - coefficient matrix
- B - constant matrix
- DET - input: scale factor, output: factor times the determinant value of the coefficient matrix
- C - temporary storage matrix

**Important Variables:** Same as argument list.

**Common Blocks:** None
Function: JFINDG

Algorithm: Similar to function IFINDN except that it searches a table passed to it for the requested node. JFINDG uses a binary search procedure to determine a node's position in the given table. If it is not found, the location at which the node should be inserted is returned.

Input/Output: None

Error Messages: None

External Calls: None

Argument List: NODE - node number to be found
NUM - number of nodes in the input table
NODES - node table with NUM nodes each with N values
N - number of values per node in the table
M - position in node values at which to look for NODE

Important Variables: Same as argument list

Common Blocks: None
Function: JFINDN

Algorithm: Searches a one-dimensional input array for the location of a given integer value. Uses a binary search to determine the position of a given value in the array.

Input/Output: None

Error Messages: None

External Calls: None

Argument List: NODE - given integer value to be searched for
NUM - number of entries in the NODES array
NODES - array to be searched for NODE

Important Variables: Same as argument list.

Common Blocks: PERM
Subroutine: LIGRNO

Algorithm: This routine controls the listing of groups or nodes in the SET, PRINT, and NATURAL modules. First, it decodes the input command line to determine the list of nodes or groups to be retrieved for printing. Then it cycles through the node or group tables and lists out the appropriate information.

Input/Output: Unit 7 - terminal output for messages

Error Messages: ** CONTROL OPTION ___ FOR LIST GENERATOR IS INCORRECT
** INCORRECT SYNTAX FOR THE LIST GENERATION **

External Calls: IFINDN PAGMOD UTLLTG

Argument List: NVAR - number of variables on the command line
LIST - list of nodes or groups to be printed

Important Variables: Same as argument list.

Common Blocks: READ ELHEAD PERM
TYPE BLANK SYSTEM
CHAR NASTRN
Subroutine: LINPTS

Algorithm: Generates a list of nodes and coordinates along a line in space. Determines the number of points to generate and then uses NATSTR to store the points based upon the delta values used to move along the line. If the ALIGNMENT parameter was used on the NODE command, this routine determines the distance to move along the given line to generate the next node. The routine then cycles until the node list to be defined is exhausted.

Input/Output: Unit 7 - terminal output for messages

Error Messages:

**** MODULE: ____ CARD: ____
LINEAR SPACING USED FOR NODES ____ RATHER THAN PERCENTAGE FUNCTION

**** MODULE: ____ CARD: ____
DELTA INTERVAL BAD BETWEEN NODES ____

External Calls: NATSTR
PLTRO1

Argument List: None

Important Variables: None

Common Blocks: BLANK NATDSP MATL
READCM TEMP
**Subroutine:** MAIDCH

**Algorithm:**
Routine is used to change duplicate material numbers if they occur, when multiple data sets are read in to form a single model. This routine uses a step-by-step procedure to check each currently stored material number against the new ones for duplicates. If duplicates are found the second is incremented by a value greater than the last one stored and processing continues.

**Input/Output:** None

**Error Messages:** None

**External Calls:** IOPAC

**Argument List:** None

**Important Variables:** None

**Common Blocks:** ELHEAD  PERM  MAT12
BLANK  DBREC
Subroutine: MASTR1

Algorithm: Stores material table number for a list of elements and stores material values for the elements. Routine loads the IOLD array based upon the material and element counts and a list of input values. Entry MASTR2 loads the property values.

Input/Output: None

Error Messages: None

External Calls: None

Argument List: IMAT - material numbers
IOLD - storage array for the material numbers
LIST - list of element positions at which to load the values
K3 - number of elements in LIST

MASTR2 ENTRY MAT - material values
IOLD, LIST, K3 - same as MASTR1
K5 - start position to be used in the IOLD list

Important Variables: Same as argument list.

Common Blocks: None
Subroutine: MATPCK

Algorithm: Routine controls the material property table storage to the geometry data base. It reads into core the material data from a temporary file and then calls in RDNMAT to sort and store the tables.

Input/Output: ND2 - scratch file read to get material tables  
Unit 7 - terminal output for messages

Error Messages: *** INSUFFICIENT MATERIAL WORKING ROOM (MAT1)

External Calls: RDNMAT

Argument List: None

Important Variables: None

Common Blocks: BLANK TEMP NASTRN MOHEAD
PERM READN DBREC TRACK1
Subroutine: MPSTRI

Algorithm: Routine decodes and stores the element geometric property values for the DIRECT property generator. The CHANGE and NUMBER functions are used to convert parameter variables to values which are stored as properties. The values are decoded based upon the allowable size parameters for a specific element type.

Input/Output: Unit 7 - terminal output for messages

Error Messages: *** MODULE: ___ CARD: ___
DESCRIPTION ___ NOT VALID FOR MATERIAL OR PROPERTY ID. _____

External Calls: CHANGE ZRAYB
NUMBER RDNOUT

Argument List: MP - element type number to define the possible number of values
DMP - array of character keywords for the given element type
Il - number of values in the NIN array
NIN - array of switches identifying integer valued parameters for the given element properties

Important Variables: Same as argument list.

Common Blocks: READ TEMP NASTRN
READCM BLANK TYPEN
The following block of subroutines and functions makes up the NATURAL generation processor of CADS. These routines provide the node, element, constraint, load, and attribute definition generation functions for CADS. The routines in this block are:

NATANS  NATLOA
NATAN1  NATLO1
NATAN2  NATNCT
NATAN3  NATNIT
NATAN4  NATNOD
NATAN5  NATPRP
NATDIR  NATSHP
NATDOP  NATSTR
NATDO1  NATTMS
NATELM  NATTRF
NATFRE  NUMBER
Subroutine: NATANS

Algorithm: Routine is used to define the composite layer data for user specified elements. It starts by initializing counters and begins decoding inputs for the routine. The BASIS command sets the zero degree material direction; CID is decoded to provide the uniaxial properties; and finally, the PLIES and GROUP commands are processed to give the laminate layups on the individual elements.

Input/Output: IODB - direct access read of material data from the GEOM data base
RDCAR1 - free read input of user commands
Unit 7 - terminal output for messages

Error Messages:
*** CONTROL OPTION _____ FOR COMPOSITE PROCESSOR IS NOT CORRECT, CARD IGNORED
*** PLY KEYWORD _____ IS NOT VALID, RECORD IGNORED
*** CID _____ NOT IN MATERIAL TABLE, RECORD IGNORED

External Calls: CHANGE NATAN2 NATAN5 RDCAR1 ZRAYB
IODB NATAN3 NUMBER UTLBAS
NATAN1 NATAN4 OUTGRD UTLLTG

Argument List: None

Important Variables: OPTION - array with valid command options
NMID - number of the composite material

Common Blocks: READ DBREC MOHEAD ELHEAD
BLANK MATL PERM SYSTEM TEMP
Subroutine: NATAN1

Algorithm: This routine actually decodes the uniaxial composite material properties and places them in the appropriate location in the composite material data array.

Input/Output: Unit 7 - terminal output for messages

Error Messages: *** CONTROL OPTION ___ FOR COMPOSITE MATERIALS IS NOT CORRECT, CARD IGNORED

External Calls: CHANGE NUMBER

Argument List: NMID - number of the material in the composite table

Important Variables: MID - composite uniaxial data table for integer values CMID - real array for composite values; equivalenced to MID

Common Blocks: READ MATL PERM
Subroutine:     NATAN2

Algorithm:      This routine is used to break up the scratch core area for the working arrays needed for the composite material processing. Using the number of elements, nodes, and property values the working storage requirement is determined.

Input/Output:   Unit 7 - terminal output for messages

Error Messages: *** NATAN2 NET ___ GROUP ___ CANNOT BE LOCATED

                 *** INSUFFICIENT WORKING AREA FOR COMPOSITE PROCESSING
                 INCREASE NBLANK AND DIMENSION D IN MAIN BY ___

External Calls: ZRAYB

Argument List:  None

Important Variables: NELMT - number of elements in the group
                   NSP  - number of property values for the element type
                   NONODE - number of nodes in the model
                   NBLANK - maximum blank working storage area
                   N3    - number of words required = NONODE*5+22+NELMT* (30+NSP)

Common Blocks:  BLANK    MOHEAD    TEMP    DBREC
                 ELHEAD    PERM    NASTRN


82
Subroutine: NATAN3

Algorithm: This routine is used to incorporate the beta angle for each element. This angle is formed by the basis vector and the element x-axis. It is computed by determining the difference between the basis angle and the element x-axis from the global or base X-axis.

Input/Output: IOPAC - packs the beta angle table for output to the data base

Error Messages: None

External Calls: IFINDN
IOPAC

Argument List: IBUFF - buffer array for the element node connectivities
BUFF - buffer array for the beta angle element value
NROW - number of rows in IBUFF and BUFF
R1 - basis vector angle
LIST - list of composite elements for beta angle generation

Important Variables: Same as argument list

Common Blocks: BLANK MOHEAD DBREC
ELHEAD TEMP

83
Subroutine: NATAN4

Algorithm: This routine stores the ply orientations and numbers of plies for the defined elements. It uses a list of elements to be processed and the NNPLY array of ply values.

Input/Output: None

Error Messages: None

External Calls: None

Argument List: NNPLY - array of ply values
LIST - list of elements for processing
N - number of elements in LIST

Important Variables: Same as argument list

Common Blocks: TEMP
Subroutine: NATAN5

Algorithm: This routine computes the composite element thicknesses and packs the layer property information for storage to the geometry data base. First, it reads the property values, beta angles, and ply values. These arrays are then packed into storage records for output to the data base. Finally the element pointer arrays are updated to point to the new composite values.

Input/Output: IOPAC - packed data I/O for GEOM data base

Error Messages: None

External Calls: IOPAC ZRAYB

Argument List: NNPLY - array with the ply layer data
IOLD - array with the element pointer data
PROP - array with the composite property information
NROW - number of rows in PROP
LIST - list of composite elements
LC - switch array for the composite elements

Important Variables: Same as argument list

Common Blocks: ELHEAD MOHEAD MATL
DBREC TEMP
Subroutine: NATDIR

Algorithm: Routine is the main DIRECT property generation processor. It decodes the command lines; sets up stacks of material and property tables; and processes the various DIRECT generation commands.

Input/Output: Unit 7 - terminal output for messages
Unit 2 - scratch data base unit
RDCAR1 - free read command lines.
IOPAC - packed data I/O for GEOM data base

Error Messages:

*** MODULE____ CARD____
   CONTROL OPTION ____ FOR DIRECT PROPERTY INPUT
   IS NOT CORRECT

   NET ____ GROUP ____ CANNOT BE LOCATED

   NET ____ GROUP ____ KEYWORD ____ IS NOT VALID,
   DEFAULT USED

   _____ D ____ NOT LOCATED FOR NET ____ GROUP ____

   INVALID CHANGE SYNTAX ENCOUNTERED, CHANGES IGNORED

   SIZE KEYWORD NOT LOCATED FOR PGEN CARD IN NET ____ GROUP ____

   NUMBER OF ELID'S DO NOT EQUAL NUMBER OF VALUES ON PGEN
   CARD FOR NET ____ GROUP ____

External Calls: CHANGE MATPCK PRSTR1 MASTR1
IOPAC MPSTR1 PRSTR2 RDCAR1
MASTR2 NUMBER UTLLTG

86
**Argument List**  
None

**Important Variables:**  
None

**Common Blocks:**  
<table>
<thead>
<tr>
<th>BLANK</th>
<th>ELHEAD</th>
<th>TYPEN</th>
<th>PERM</th>
<th>READCM</th>
</tr>
</thead>
<tbody>
<tr>
<td>MATPRO</td>
<td>DBREC</td>
<td>MATL</td>
<td>READ</td>
<td>TEMP</td>
</tr>
<tr>
<td>MOHEAD</td>
<td>NASTRN</td>
<td>SYSTEM</td>
<td>READN</td>
<td></td>
</tr>
</tbody>
</table>
Subroutine: NATDOP

Algorithm: Routine closes DI-3000 and resets the window and viewport before re-opening DI-3000 for additional graphics output.

Input/Output: None

Error Messages: None

External Calls: JCLOSE JVPORT
                JFRAME JWINDO
                JOPEN

Argument List: None

Important Variables: None

Common Blocks: DITOKD
Subroutine: NATDO1

Algorithm: Routine resets the window and viewport before opening DI-3000 for graphics output. Routine is used in the display of element generator functions.

Input/Output: None

Error Messages: None

External Calls: JWINDO
JVPORT
JOPEN

Argument List: None

Important Variables: None

Common Blocks: D1TOKD
Subroutine: NATELM

Algorithm: This routine generates elements based upon the user requests. The routine first decodes the support commands (i.e. ROTATE, LIST, etc.) and then decodes the actual element definition command. It establishes the element type and connectivity list and loads the data in the group currently being processed.

Input/Output: Unit 7 - terminal output for messages

Error Messages: ** ROTATE KEYWORD _____ IS NOT VALID RE-ENTER **

*** GROUP NUMBER MUST BE SUPPLIED BEFORE ELEMENT INPUT

*** MODULE: _____ CARD: _____
ELEMENT TYPE _____ NOT SUPPORTED FOR NATURAL MODE INPUT

ALL ELEMENTS OF A GROUP MUST BE THE SAME TYPE
INCORRECT NUMBER OF NODES SUPPLIED FOR ELEMENT TYPE _____

INCORRECT NUMBER OF INCREMENTS SUPPLIED FOR ELEMENT TYPE _____

INCORRECT NUMBER OF CLOSING ELEMENT NODES SUPPLIED FOR ELEMENT TYPE _____

FOR ZERO NODE INCREMENT VALUES, STARTING AND ENDING NODES MUST BE THE SAME

ALL NODE DELTA INCREMENTS BETWEEN START TO FINISH MUST BE THE SAME

REPEAT CARD DOES NOT MATCH PREVIOUS NODE CARD FOR NUMBER OF VARIABLES REQUIRED
Error Message:
(Continued)

GROUP ___ HAS INVALID DUPE KEYWORD

INVALID LIST GENERATED FOR SUB-ELEMENT GENERATION

GROUP ___ DUPE COMMAND HAS NOT REFERENCED A
PREVIOUSLY STORED GROUP

External Calls:
CHANGE  JCLOSE  NATDO1  PLTNAT  PLTRVH  ZRAYB
GROUPS  JMOVE  NATDOP  PLTNAT  RDCAR1  ZRAYI
IOPAC  LIGRNO  NUMBER  PLTPLS  UTLLTG  JFRAME

Argument List:
H  - horizontal position of the node on the screen
V  - vertical position of the node on the screen
Z  - z position of the node
IBUFF  - buffer array for scratch storage
MATRIX  - storage array for element connectivity
IOLD  - array to track the element group numbers

Important Variables:
Same as argument list.

Common Blocks:
READ  MOHEAD  NASTRN
TYPEN  BLANK  TEMP
TYPE  READCM  PERM
ELHEAD  SYSTEM  MATL
PLOTCM  DBREC  NATDSP
Subroutine: NATFRE

Algorithm: Routine controls the specification of node degrees of freedom in the NATURAL generation module. It begins by decoding the command line and then gets the nodes to be processed and their constraints. Finally, it calls the FRESTR routine to actually store the constraints in the data base.

Input/Output: RDCAR1 - performs free read command line input
Unit 7 - terminal output for messages

Error Messages:
*** FREEDOM CONTROL OPTION ___ NOT CORRECT, CARD IGNORED

*** INVALID KEYWORD ___ ENCOUNTERED, EITHER MISSING OR MISSPELLED

*** THE ALL KEYWORD WAS ASSUMED

*** NODE SET ___ IS NOT ON DATA BASE

*** NO NODES ENCOUNTERED WITHIN INPUT LIST

*** NO DISPLACEMENT COMPONENTS WERE INDICATED FOR OPERATIONS

External Calls: FREFRE IFINDN UTLLTG RDCAR1
FREPCK IOROUT FRESTR

Argument List: None

Important Variables: None

Common Blocks: BLANK SYSTEM PERM
**Subroutine:** NATLOA

**Algorithm:** Routine is used to read in node point external load data as translational forces or rotational moments. Routine decodes user options, branches to set the appropriate factor, case, and other switches, and obtains the list of affected nodes. Finally, it places the load factors on the nodes and calls in a packing array.

**Input/Output:**
- RDCAR1 - free read input for user commands
- Unit 7 - terminal output for messages

**Error Messages:**
- *** CONTROL OPTION ____ FOR LOAD PROCESSOR IS NOT CORRECT, CARD IGNORED
- *** NODE KEYWORD ____ IS NOT VALID, RECORD IGNORED
- *** CARD ____ IS NOT VALID, RECORD IGNORED

**External Calls:**
- CHANGE RDCAR1 NUMBER
- NATLO1 UTLLTG ZRAYB

**Argument List:**
- LOAD - array to hold the applied loads values

**Important Variables:**
- FACTL - force factor
- FACTM - moment factor
- NCA - case number

**Common Blocks:**
- READ DBREC PERM MATL
- ELHEAD SYSTEM NOHEAD MOHEAD
Subroutine: NATLO1

Algorithm: Routine takes the applied load matrices and packs them for storage on the GEOM data base. Routine stores the key pointer, case number, and number of loads.

Input/Output: IOPAC - pack routine for loading the GEOM data base

Error Message None

External Calls: IOPAC

Argument List: LOMO - input loads or moment array
LL - key to load or moment position

Important Variables: Same as argument list

Common Blocks: NOHEAD PERM
MOHEAD DBREC
Subroutine: NATNCT

Algorithm: This routine controls the node generation functions. It decodes a command line, sets appropriate switches, and finally calls in the correct routine for processing the command.

Input/Output: RDCAR1 - free read command input
Unit 7 - terminal output for messages

Error Messages:
**** NODES CONTROL OPERAND ____ NOT CORRECT, CARD IGNORED

**** NODES PROCESSOR MODULE ____ NOT SUPPORTED

**** NODE PROCESSOR MODULE NAME HAS NOT BEEN ENTERED

External Calls: NATFRE NATTRF RDPAC NATLOA
NATNOD OUTGRD NATSHP RDCAR1

Argument List: None

Important Variables: None

Common Blocks: READ MOHEAD NATDSP SYSTEM
BLANK NOHEAD PERM DBREC
Subroutine: NATNIT

Algorithm: Routine interprets the node string command input. Checks to see if a start node exists; interprets coordinate locations; and cycles through the input list storing newly generated nodes.

Input/Output: Unit 7 - terminal output for messages

Error Message: *** MODULE:_____ CARD:_____ NODE_____ DOES NOT PREVIOUSLY EXIST

External Calls: IFINDN
               NATSTR
               PLTRO1

Argument List: None

Important Variables: None

Common Blocks: BLANK PERM TEMP
               READCM NATDSP MATL
Subroutine: NATNOD

Algorithm: Generates the node coordinate locations in the DIRECT submodule of the NATURAL generation module. Interprets the node commands and sets the switches as needed. Next, the routine sets up the node and coordinates from the command line for the generation of the actual list of nodes. The REPEAT command is then processed and the nodes are stored to the data base.

Input/Output: RDCAR1 - free read command input.
Unit 7 - terminal output for messages

Error Messages: *** MODULE: _________ CARD: _________
                   NODE CONTROL OPTION _____ NOT CORRECT CARD IGNORED

*** _____ IS A BAD KEYWORD RE-ENTER

*** COORDINATE DIRECTION IS NOT CORRECT

INVALID NODE GENERATION KEYWORD

REPEAT CARD DOES NOT MATCH PREVIOUS NODE CARD FOR NUMBER OF VARIABLES REQUIRED

CANNOT FIND NODE _____ FOR REPEAT

LENGTH MUST EQUAL SUM OF INDIVIDUAL TERMS

EQUATE LIST HAS INCORRECT NUMBER OF TERMS

NODE _____ DOES NOT PREVIOUSLY EXIST, NODE _____ NOT ASSIGNED

NODE _____ EXISTS AND CANNOT BE EQUATED
External Calls:

CHANGE JMOVE NATD01 NATSTR RDCAR1
IFINDN LIGRNO NATDOP NUMBER JFRAME
JCLOSE LINPTS NATNIT PLTRO1

Argument List:
None

Important Variables:

INTP - stores node or coordinate value from the command line

KIND - keeps the type of value for the equivalent command line available

Common Blocks:
READ READCM SYSTEM NATDSP
CHAR PLOTCM PERM
BLANK TEMP MATL
Subroutine: NATPRP

Algorithm: Routine decodes the property generation commands to determine which generator to call in. Calls in the DIRECT or COMPOSITE generator.

Input/Output: RDCAR1 - free read command input
Unit 7 - terminal output for messages

Error Messages: 
**** PROPERTY CONTROL OPERAND ___ NOT CORRECT CARD IGNORED

**** PROPERTY PROCESSOR ___ IS NOT SUPPORTED

**** PROPERTY PROCESSOR NAME HAS NOT BEEN ENTERED

External Calls: NATANS
NATDIR
RDCAR1

Argument List: None

Important Variables: None

Common Blocks: BLANK
PERM
SYSTEM
Subroutine: NATSHP

Algorithm: Generates nodes along a section of a circle, ellipse, or parabola. First initializes the variables, next decodes the command input, and finally processes the command. Nodes are generated and stored for the current command based upon the type of curve being used to define the shape.

Input/Output: RDCAR1 - free read command input
                Unit 7 - terminal output for messages

Error Messages:
*** SHAPES CONTROL OPERAND ______ NOT CORRECT, CARD IGNORED

*** SHAPES KEYWORD ______ IS NOT CORRECT

*** NO NODES ENCOUNTERED WITHIN THE INPUT LIST

*** INCORRECT SYNTAX FOR THE SHAPE OPERATION

*** INCORRECT LENGTH SPECIFIED FOR SHAPE, LENGTH MUST BE LESS THAN ______ DEGREES

*** INCORRECT NUMBER OF INTERVALS SPECIFIED FOR NODE SPACING

*** LENGTH MUST BE EQUAL TO SUMMATION OF TERMS

*** ______ IS A BAD KEYWORD RE-ENTER ***

External Calls: CHANGE LIGRNO NUMBER JFRAME
                IFINDN NATD01 PLTRO1
                JCLOSE NATDOP RDCAR1
                JMOVE NATSTR UTLLTG
Argument List: None

Important Variables: VALUES - array of degree interval values for the new nodes
CL - array of coordinates for the new node

Common Blocks: TEMP SYSTEM PERM
READ PLOTCM BLANK NATDSP
Subroutine: NATSTR

Algorithm: Routine stores the actual node and coordinates to the permanent array and data base. Will generate the mirror and center nodes if required.

Input/Output: Unit 7 - terminal output for messages

Error Messages: *** NODE _____ _____ _____ ALREADY EXISTS,
                 WAS REPLACED

External Calls IFINDN

Argument List: NEWPNT - array with the new node and its coordinates
               ISLOT - position in the permanent array for the node

Important Variables: Same as argument list

Common Blocks: BLANK
               PERM
               TEMP
Subroutine: NAITMS

Algorithm: Routine resolves the coordinate system transformation matrices for rectangular, cylindrical, and spherical reference systems. Determines the transformation matrix based on the given input vectors for the specified reference system.

Input/Output: Unit 7 - terminal output for messages

Error Messages: 
*** COORDINATE SYSTEM _______ WAS DUPLICATED

*** COORDINATE SYSTEM _______ APPEARS COLINEAR

External Calls: UTLCRS

Argument List: ISYS - coordinate system number
NSYS - number of different systems
TA - three value vector defining point A
TB - three value vector defining point B
TC - three value vector defining point C

Important Variables: Same as argument list

Common Blocks: PERM
MATL
Subroutine: NATTRF

Algorithm: Routine transforms a set of nodes from the currently defined coordinates to a new user specified system. Routine decodes the transformation commands, calculates the required transformation matrix and then processes the defined set of nodes for their transformation.

Input/Output: RDCAR1 - free read command input
Unit 7 - terminal output for messages

Error Messages: *** ERROR IN TRANSFORMATION ROUTINE, SECTION: ___
 *** COMMAND WORD BEING ANALYZED WAS: ___
 *** COMMAND IGNORED

External Calls: CHANGE ISIMEQ U PLLTG RDCAR1
IFINDN NUMBER IOROUT

Argument List: None

Important Variables: VALUE - array of decoded input values
TRANS - transformation array

Common Blocks: MATL PERM
BLANK SYSTEM
Function: NUMBER

Algorithm: Function converts input characters to internal integer numbers.

Input/Output: Unit 5 - terminal input of new variable after error message
Unit 7 - terminal output for messages

Error Messages: *** _____ IS AN ILLEGAL INTEGER NUMBER RE-ENTER

External Calls: None

Argument List: LHOLD - character array input to be converted

Important Variables: Same as argument list

Common Blocks: SYSTEM
CHAR
The following block of subroutines makes up the OUTPUT module and processors of CADS. These routines provide the output translators from the GEOMETRY data base to the ANALYZE, NASTRAN, or OPTSTAT formats. The routines in this block are:

- OUTANA
- OUTAN1
- OUTAOG
- OUTAOL
- OUTBAR
- OUTCAL
- OUTCEL
- OUTCQA
- OUTCQT
- OUTGRD
- OUTNAL
- OUTNAT
- OUTNAX
- OUTNB2
- OUTNCD
- OUTNHX
- OUTNLM
- OUTNQT
- OUTNRO
- OUTNR2
- OUTNT6
- OUTNWT
- OUTOPT
- OUTOP1
- OUTOP2
- OUTPPK
- OUTPUT
- PAGMOD
Subroutine: OUTANA

Algorithm: Routine is used to output the model information in the ANALYZE program format. Routine first retrieves the material, connectivity, and coordinate information. The control words, material properties, and connectivity are then output. Finally the grids and applied loading data are output.

Input/Output: KUNIT - output file unit for ANALYZE data
IODB - direct access routine for data base I/O

Error Messages: None

External Calls: IODB OUTAOL OUTAOG
OUTAN1 ZRAYB

Argument List: None

Important Variables: MATI - material table array
PAT1 - element sizes array
IBND - boundary condition array
XYZ - coordinate position array
MEMBS - number of elements
JOINTS - number of nodes

Common Blocks: NOHEAD MOHEAD BLANK
ELHEAD PERM DBREC MATL
Subroutine: OUTANI

Algorithm: Routine retrieves the element connectivity and material data for output in the ANALYZE format. It cycles through the pointer table and retrieves the connectivity for each element type after having retrieved the entire material table.

Input/Output: IODB - performs I/O directly to GEOM data base
IOPAC - data pack/unpack routine for data base I/O
Unit 7 - terminal output for messages

Error Messages: *** ERROR *** TOO MANY ANALYZE ELEMENTS; USED= ____
LIMIT FOR ANALYZE IS 295

External Calls: IOPAC JFINDG
IODB OUTOP2

Argument List: MEMBS - number of elements in the model

Important Variables: NNODES - array with the number of nodes per element
NELMT - number of elements per type
MA, MB, MC, MD - connectivity arrays for the elements

Common Blocks: NOHEAD MOHEAD BLANK MATL
ELHEAD PERM DBREC
Subroutine: OUTAOG

Algorithm: Subroutine outputs the grid point coordinate information for ANALYZE and OPTSTAT. This routine retrieves the coordinate and suppression information from the database and then unpacks the data into the coordinate and boundary arrays. These are returned to OUTANA or OUTOPT for output in the correct format.

Input/Output: IODB - data base I/O routine

Error Messages: None

External Calls: IODB SORTQ

Argument List: XYZ - coordinate value array
IB - boundary condition array
MM - switch for 2-D or 3-D problem
JOINTS - number of nodes
IBND - counter for the number of boundary points

Important Variables: Same as argument list

Common Blocks: NOHEAD PERM
MOHEAD DBREC
Subroutine: OUTAOL

Algorithm: Subroutine is used to retrieve and pack the applied load data for ANALYZE and OPTSTAT. Gets the pointers to the loads and retrieves them from the data base. This data is then packed by node, component, and case for output in OUTANA or OUTOPT.

Input/Output: IODB - performs I/O directly to GEOM data base
IOPAC - data base pack/unpack routine for I/O

Error Messages: None

External Calls: IOPAC IODB ZRAYB JFINDG

Argument List: NJLODS - array with the number of values per load case
LOADS - number of load cases

Important Variables: JM - array with nodes loaded for each value and load case
IM - array of load component directions for each JM node
TFR - array of actual load values for JM nodes

Common Blocks: MOHEAD PERM DBREC
NOHEAD BLANK MATL
Subroutine: OUTBAR

Algorithm: Routine outputs the bending beam property data in NASTRAN format as a PBAR card. It determines the number of values to be output and then uses the correct format to output the required PBAR card along with any necessary continuation cards.

Input/Output: KUNIT - writes bulk data cards to a specified output unit

Error Messages: None

External Calls: None

Argument List: ICON - counter for the number of continuation cards output
JBAR - integer array of PBAR values
BAR - real array of PBAR values
KB - counter for the number of PBAR cards required

Important Variables: Same as argument list

Common Blocks: PERM
Subroutine: OUTCAL

Algorithm: Routine outputs layered composite elements in NASTRAN format. Gets the layer data; sets up the element format; and then outputs the connectivity, property, and material cards.

Input/Output: IOPAC - packed array I/O to geometry data base
KUNIT - writes bulk data cards to the given output unit

Error Messages: None

External Calls: IOPAC
OUTCQA

Argument List: IT - location of the group in the header array
IBUFF - integer connectivity array
BUFF - real connectivity array (equivalenced IBUFF)
N1 - number of values per element in the connectivity array
NPROP - integer property array
PROP - real property array (equivalenced IPROP)
N2 - number of values per element in the property array
ID - array containing element id, number of orientation angles, composite material table key and layer offset information

Important Variables: Same as argument list

Common Blocks: CHAR MATL PERM
ELHEAD TEMP
DBREC NASTRN
Subroutine: OUTCEL

Algorithm: Routine retrieves data for layered composite elements for output in NASTRAN format. Gets the size and pointer values for each data block (element, property, material) and calls IOPAC to retrieve the appropriate data.

Input/Output: IOPAC - packed array I/O to geometry data base

Error Messages: None

External Calls: IOPAC
                OUTCAL
                OUTCQT

Argument List: None

Important Variables: NELGRH - group[element pointer array
                    NGROUP - number of groups
                    D      - scratch array for element data
                    N      - position in D of the element data
                    N1     - position in D of the property data
                    N2     - position in D of the material information

Common Blocks:  BLANK      MATL      PERM
                ELHEAD    NASTRN
                DBREC     TEMP
Subroutine: OUTCQA

Algorithm: Routine computes the element thickness and material matrix (Q) for a single layer orientation. Passes the information back for output as NASTRAN property and MAT2 bulk data cards. Uses standard lamination theory to generate the material matrix.

Input/Output: Unit 7 - terminal output for messages

Error Messages: ** MATERIAL ID _____ DOES NOT EXIST IN COMPOSITE MATERIAL TABLE FOR ELEMENT NO. _____

External Calls: JFINDG
                ZRAYB

Argument List: A - output Q matrix, density, and allowable stresses
                T2 - output thickness of the element
                ID - input array containing the element id, material id, layer offsets, and material key
                * - alternate return

Important Variables: Same as argument list

Common Blocks: MATL
                TEMP
Subroutine: OUTCQT

Algorithm: Routine outputs layer composite data for the CQUAD4 and CTRIA3 elements. First it retrieves the layer data and ply information for the elements. Routine then calls OUTCQA to get the material matrix and cycles through the elements outputting the appropriate connectivity, property, and material cards.

Input/Output: KUNIT - NASTRAN bulk data output unit
IOPAC - packed array I/O to geometry data base

Error Messages: None

External Calls: IOPAC
               OUTCQA

Argument List: IT - location of the group in the header array
               IBUFF - integer connectivity array
               BUFF - real connectivity array
               N1 - number of values per element in IBUFF
               NPROP - integer property array
               PROP - real property array
               N2 - number of values per element in IPROP
               ID - array containing element id, material id, layer offset, and material key

Important Variables: Same as argument list

Common Blocks: CHAR     MATL     PERM
               ELHEAD    TEMP
               DBREC     NASTRN
Subroutine: OUTGRD

Algorithm: Outputs the model coordinate information as NASTRAN GRID bulk data cards. First, it reads in the grid data records from the data base; codes the information into the correct format, and finally prints out the correct GRID or RINGAX card.

Input/Output: KUNIT - outputs card image file for NASTRAN bulk data

Error Messages: None

External Calls: IODB OUTNCD ZRAYB

Argument List: IG - switch for the type of output
CXYZ - real array of node values
NCXYZ - integer array of node values

Important Variables: Same as argument list

Common Blocks: BLANK PERM NOHEAD
DBREC SYSTEM MATL
Subroutine: OUTNAL

Algorithm: Routine outputs the standard three and four corner elements, sets the switches, decodes the element type, sets the correct format, and writes the connectivity and property cards.

Input/Output: KUNIT - output file for NASTRAN data

Error Messages: None

External Calls: OUTPPK

Argument List: IT - group number for the elements to be output
IBUFF - integer number array for the element data
BUFF - real number array for the element data
N1 - number of rows in the IBUFF array
NPROP - integer number array for the properties
PROP - real number array for the properties
N2 - number of rows in the PROP and PMAT arrays
IOLD - element number position array
N3 - number of rows in the IOLD array
NPMAT - integer number array for materials
PMAT - real number array for materials

Important Variables: Same as argument list

Common Blocks: CHAR  ELHEAD  NASTRN
              TEMP  PERM  SYSTEM
Subroutine: OUTNAT

Algorithm: Routine outputs the model in NASTRAN format by calling in special output routines; it establishes the data to be passed to those routines; and finally, writes out the material cards. It starts by initializing switches and counters; then the element data blocks are retrieved; and subroutines are called to output the data. Finally, the NASTRAN MAT cards are output.

Input/Output: KUNIT - output file for NASTRAN bulk data deck

Error Messages: None

External Calls: IOPAC OUTNAL OUTNHX OUTNR2 OUTNWT MAIDCH OUTNAX OUTNLMAUTNR0 OUTCEL OUTGRD OUTNB2 OUTNQT OUTNT6

Argument List: NR1 - sets the number of digits in the grid coordinate values

Important Variables: NELMT - number of elements in the group NTYPE - element type NVREC - current record being accessed from the data base

Common Blocks: BLANK DBREC MATL PERM ELHEAD NASTRN NOHEAD
Subroutine: OUTNAX

Algorithm: Routine outputs the axisymmetric elements for NASTRAN. Follows the same process as the OUTNAL routine with minor changes for the CTRAPAX and CTRIAAX requirements.

Input/Output: KUNIT - output unit for bulk data deck

Error Messages: None

External Calls: OUTPPK

Argument List:
- IT - group number for the elements to be output
- IBUFF - integer number array for the element data
- BUFF - real number array for the element data
- N1 - number of rows in the IBUFF array
- NPROP - integer number array for properties
- PROP - real number array for properties
- N2 - number of rows in the PROP and PMAT arrays
- IOLD - element number position array
- N3 - number of rows in the IOLD array
- NPMAT - integer number array for materials
- PMAT - real number array for materials

Important Variables: Same as argument list.

Common Blocks:
- CHAR
- NASTRN
- TEMP
- ELHEAD
- PERM
- SYSTEM
Subroutine: OUTNB2

Algorithm: Routine outputs the beam element as a NASTRAN CBAR element using the same procedure set up for the general NASTRAN element output routine OUTNAL.

Input/Output: KUNIT - output file for bulk data deck.

Error Messages: None

External Calls: IFINDN OUTBAR OUTNCD

Argument List: IT - position in NELGRH array of CBAR group
IBUFF - array contains element identification data
N1 - number of rows in the IBUFF array
NPROP - integer numbers in property array
PROP - real numbers in property array
N2 - number of property table rows
IOLD - element identification array
N3 - number of element identifications in IOLD
JBAR - integer property card values
BAR - real property card values
IBAR - count of the number of CBAR output cards

Important Variables: Same as argument list

Common Blocks: BLANK DBREC PINFLA SYSTEM
CHAR PERM ELHEAD

120
Subroutine: OUTNCD

Algorithm: Codes a real value into an 8-character output array for output to a file. Processes three numbers at a time—keeping the most significant digits.

Input/Output: None

Error Messages: None

External Calls: None

Argument List: R4 - three element array of real numbers
R8 - three element array of 8-character output

Important Variables: Same as argument list

Common Blocks: CHAR
Subroutine: OUTNHX

Algorithm: Output routine for the solid isoparametric elements. It follows the same format as OUTNAL, the general NASTRAN element output routine.

Input/Output: KUNIT - output file for bulk data deck

Error Messages: None

External Calls: OUTPPK

Argument List: IT - group number for the elements to be output
IBUFF - integer number array for the element data
BUFF - real number array for the element data
N1 - number of rows in the IBUFF array
NPROP - integer number array for properties
PROP - real number array for properties
N2 - number of rows in the PROP and PMAT arrays
IOLD - element number position array
N3 - number of rows in the IOLD array
NPMAT - integer number array for materials
PMAT - real number array for materials

Important Variables: Same as argument list

Common Blocks: CHAR NASTRN TEMP
ELHEAD PERM SYSTEM
Subroutine: OUTNLM

Algorithm: Routine outputs the forces or moments in NASTRAN format. Cycles through the pointer array to retrieve the force or moment data from the geometry data base. The data is then output as NASTRAN bulk data cards.

Input/Output: KUNIT - output file for the bulk data deck
IOPAC - retrieves force/moment data from geometry data base

Error Messages: None

External Calls: IOPAC

Argument List: None

Important Variables: MF - array with node number, load number, and coordinate data for the nodes
FM - force or moment data array

Common Blocks: BLANK
MOHEAD
NOHEAD
PERM
DBREC
Subroutine: OUTNQT

Algorithm: Routine outputs the higher order bending elements in NASTRAN CQUAD4 and CTRIA3 format. It uses the same procedure as the general element output routine OUTNAL.

Input/Output: KUNIT - output file for bulk data deck

Error Messages: None

External Calls: OUTPPK

Argument List:
IT - group number for the elements to be output
IBUFF - integer number array for the element data
BUFF - real number array for the element data
N1 - number of rows in the IBUFF array
NPROP - integer number array for properties
PROP - real number array for properties
N2 - number of rows in the PROP and PMAT arrays
IOLD - element number position array
N3 - number of rows in the IOLD array
NPMAT - integer number array for materials
PMAT - real number array for materials

Important Variables: Same as argument list

Common Blocks: CHAR NASTRN TEMP
ELHEAD PERM SYSTEM
Subroutine: OUTNRO

Algorithm: This routine outputs the axial rod elements as NASTRAN CROD elements using the general OUTNAL procedure.

Input/Output: KUNIT - output unit for bulk data deck

Error Messages: None

External Calls: None

Argument List:
- IT - group number for the elements to be output
- IBUFF - integer number array for the element data
- BUFF - real number array for the element data
- N1 - number of rows in the IBUFF array
- NPROP - integer number array for properties
- PROP - real number array for properties
- N2 - number of rows in the PROP and PMAT arrays
- IOLD - element number position array
- N3 - number of rows in the IOLD array
- NPMAT - integer number array for materials
- PMAT - real number array for materials
- IP - array containing the compressed property identifications

Important Variables: Same as argument list

Common Blocks:
- CHAR
- PERM
- ELHEAD
- NASTRN
- SYSTEM
Subroutine: OUTNER2

Algorithm: Routine outputs the CONROD and CELAS2 elements in NASTRAN format. First it determines the element type and data record pointers and then cycles through the data array formatting and writing the correct bulk data deck cards.

Input/Output: KUNIT - output unit for bulk data deck

Error Messages: None

External Calls: None

Argument List: IT - group number for the elements to be output
IBUFF - integer number array for the element data
BUFF - real number array for the element data
N1 - number of rows in the IBUFF array
NPROP - integer number array for properties
PROP - real number array for properties
N2 - number of rows in the PROP array
IOLD - element number position array
N3 - number of rows in the IOLD array

Important Variables: Same as argument list

Common Blocks: CHAR SYSTEM
ELHEAD PERM
Subroutine: OUTNT6

Algorithm: Routine outputs the higher order triangular membrane element in NASTRAN CTRIM6 format. It follows the general element output procedure used in OUTNAL.

Input/Output: KUNIT - output unit for bulk data deck

Error Messages: None

External Calls: OUTPPK

Argument List: IT - group number for the elements to be output
IBUFF - integer number array for the element data
BUFF - real number array for the element data
N1 - number of rows in the IBUFF array
NPROP - integer number array for properties
PROP - real number array for properties
N2 - number of rows in the PROP and PMAT arrays
IOLD - element number position array
N3 - number of rows in the IOLD array
NPMAT - integer number array for materials
PMAT - real number array for materials

Important Variables: Same as argument list

Common Blocks: CHAR NASTRN TEMP
ELHEAD PERM SYSTEM
Subroutine: OUTNWT

Algorithm: Routine outputs the wedge and tetrahedron elements in NASTRAN CWEDGE and CTETRA formats. It first sets the type and pointer variables and then processes the data arrays. Next, the connectivities and element card data are formatted; and finally, the data is written to the KUNIT file.

Input/Output: KUNIT - output file for bulk data deck

Error Messages: None

External Calls: None

Argument List: IT - index into the pointer arrays for a particular group
IBUFF - element connectivity data
N1 - number of rows in the IBUFF array
IOLD - property card pointers
N3 - number of rows in the IOLD array

Important Variables: Same as argument list

Common Blocks: CHAR PERM ELHEAD
NASTRN SYSTEM
Subroutine: OUTOPT

Algorithm: This routine outputs the model data in OPTSTAT format. It starts by retrieving the model data, such as the coordinate, connectivity, and material information for storage in the appropriate arrays. This information is then output to KUNIT using a series of WRITE statements to correctly format the arrays of data.

Input/Output: KUNIT - output unit for the data deck

Error Messages: None

External Calls: OUTAOG ZRAYB IOPAC OUTAOL OUTOP1

Argument List: None

Important Variables: ISOTRN - isotropic material counter NISOTR - composite material counter XYZ - node coordinate array

Common Blocks: NOHEAD BLANK OPTIND DBREC PERM MOHEAD MATL
Subroutine: OUTOP1

Algorithm: Subroutine retrieves the element connectivity and material data for output to OPTSTAT. The isotropic and composite material tables are read from the data base. The element connectivity and material pointers are then read and the routine cycles through the element list. The element information is placed in the output arrays based upon the element type and material data.

Input/Output: IODB - direct access I/O from the GEOM data base
IOPAC - packs/unpacks tables from the data base
Unit 7 - terminal output for messages

Error Messages: *** ERROR *** NUMBER OF OPTSTAT ELEMENTS WERE = ___
MAXIMUM ALLOWED IS 160

External Calls: IODB JFINDG
IOPAC OUTOP2

Argument List: None

Important Variables: MEMBS - number of members (elements) to be output
MA,MB,MC,MD - arrays holding nodes 1-4 of the various elements
NNODES - array of element type numbers

Common Blocks: NOHEAD MOHEAD BLANK DBREC
ELHEAD PERM OPTIND MATL
Subroutine: OUTOP2

Algorithm: This routine records the ANALYZE or OPTSTAT element connectivity outputs. It cycles through the output elements checking the current element number against the corresponding input element number. When necessary it changes the numbers to match the input number.

Input/Output: None

Error Messages: None

External Calls: None

Argument List:
- NZ - array of reordered element numbers
- NW - array of current input element numbers
- LAM - array of pointers to composite elements
- MEMBS - the number of members (elements) in the model
- ISIZE - limit on the number of the elements in the model
- NISOTR - number of composite materials

Important Variables: Same as argument list

Common Blocks: None
Subroutine: OUTPPK

Algorithm: Routine processes the property card identifications for a group of elements into a compressed table of required data. Basically it checks new values against the previously stored tables to see if there are any differences in values. If there are, the property is added to the table list, otherwise the property number is changed to the previously stored value.

Input/Output: None

Error Messages: None

External Calls: None

Argument List: PMAT - compressed table of real property values
NPMAT - compressed table of property identifications
PROP - property values to be checked against the table
N2 - number of rows in the PMAT, NPMAT, and PROP arrays

Important Variables: Same as argument list

Common Blocks: TEMP
**Subroutine:** OUTPUT

**Algorithm:** Routine is used to control the output subroutines. It decodes the command line input; sets appropriate switches; and finally, calls in the particular translator control routine.

**Input/Output:**
- RDCARD - free read command input
- UNIT 7 - terminal output for messages

**Error Messages:**
- *** OUTPUT CONTROL OPTION _______ NOT VALID
- *** OUTPUT PROGRAM TYPE _______ NOT SUPPORTED
- *** ERROR *** OUTPUT UNIT NUMBER _______ MUST BE GREATER THAN 20; REENTER
- *** ERROR *** PROBLEM _______ OPENING UNIT _______ FOR OUTPUT OF FILE _______
- *** THE PROGRAM KEYWORD IS MISSING FOR THE OUTPUT CONTROL OPTION

**External Calls:**
- NUMBER
- OUTNAT
- OUTANA
- OUTOPT
- RDCARD

**Argument List:** None

**Important Variables:**
- ISW - switch for the output translator type
- NR1 - number of digits to be output for real values

**Common Blocks:**
- BLANK
- DBREC
- PERM
- SYSTEM

133
Subroutine: PAGMOD

Algorithm: Routine monitors the number of lines on the screen so that it can be erased. It also queries the user to permit a user controlled abort of a command which is listing data at the terminal.

Input/Output: Unit 5 - free read terminal I/O
Unit 7 - terminal output for messages

Error Messages: None

External Calls: JFRAME  JCLOSE  JIQERR

Argument List: KKSYSM - terminal type
ISTOP - switch used to skip the user prompt code
* - alternate return

Important Variables: Same as argument list

Common Blocks: None
The following block of subroutines makes up the DISPLAY Module of CADS. These routines provide the display and plot functions for CADS. The routines in this block are:

- PLOTGN
- PLOTNO
- PLOTNR
- PLOTS
- PLTARW
- PLTBEG
- PLTBOF
- PLTBOT
- PLTBO1
- PLTBO2
- PLTBO3
- PLTBO4
- PLTCMA
- PLTCFA
- PLTCFI
- PLTCRT
- PLTCLV
- PLTDCL
- PLTDIS
- PLTDMP
- PLTDOP
- PLTEID
- PLTELH
- PLTELM
- PLTESW
- PLTES1
- PLTEXP
- PLTGNO
- PLTHBR
- PLTHCF
- PLTHCK
- PLTHID
- PLTHLN
- PLTHMN
- PLTHPL
- PLTHSA
- PLTHSK
- PLTHST
- PLTHSV
- PLTHVR
- PLTHV1
- PLTHV2
- PLTLAY
- PLTMAG
- PLTMAT
- PLTMA1
- PLTMXN
- PLTNAT
- PLTMOD
- PLTPLS
- PLTPRP
- PLTPRO
- PLTPRO1
- PLTRVH
- PLTSAF
- PLTSAI
- PLTSCL
- PLTSYM
- PLTWID
- PRSTR1
- PRSTR2
Subroutine: PLOTGN

Algorithm: This routine calls in the SETGEN subroutine for generating sets and then it sets up for the display routines. It begins by initializing the variables and setting up the scratch arrays for the number of nodes and elements. It then calls in the SET and DISPLAY module command routines.

Input/Output: Unit 7 - terminal output for messages

Error Messages: *** MODEL TOO LARGE FOR PLOTTING
INCREASE NBLANK AND DIMENSION D IN MAIN BY ___

External Calls: OUTGRD SETUP
PLOTS SETGEN

Argument List: None

Important Variables: NBLANK - size of blank common
NONODE - number of nodes in the model

Common Blocks: BLANK PLOTCM
ELHEAD PERM
Subroutine: PLOTNO

Algorithm: Routine outputs integer numbers to the terminal screen. It converts the number to characters; strips off the blanks; positions the margin; and finally, outputs the text string.

Input/Output: Internal character file READ/WRITE

Error Messages: None

External Calls: J1STRG
                JCONVW
                JMARGN

Argument List: AIH - horizontal screen position for the number
               AIV - vertical screen position for the number
               LV  - not used
               ISW  - switch for the type of number
               IPT1 - first number to be output
               IPT2 - second number to be output

Important Variables: Same as argument list

Common Blocks: PLOT
                SYSTEM
Subroutine: PLOTNR

Algorithm: Converts and outputs real numbers to the screen. This routine sets the output format; converts the number, and strips off the blanks before positioning and outputting the text string.

Input/Output: Internal character file READ/WRITE

Error Messages: None

External Calls: J1STRG

Argument List: AIH - horizontal screen position of the number
AIV - vertical screen position of the number
LV - not used
ISW - number of characters to the right of the decimal point
PT1 - number to be output
IPT2 - not used
K - number of blanks to precede the output number string

Important Variables: Same as argument list

Common Blocks: None
**Subroutine:**
PLOTS

**Algorithm:**
Subroutine decodes the DISPLAY module commands and controls the output of display information to the terminal. First it sets a series of switches and then decodes the plot commands by setting the specific switches for that command. Next it calls the required routines depending on the previous commands. After calling PLTRED it draws the display lines at the screen and outputs any requested data values to the terminal. Next it does the end of frame and view box processing before finishing up with calls to the contour plot routines.

**Input/Output:**
Unit 7 - terminal output for messages
Unit 5 - terminal input for user inputs
Internal character file READ/WRITE

**Error Messages:**
*** PLOT OPTION NOT FOUND
*** MAT2 DATA BLOCK DOES NOT EXIST

**External Calls:**
CHANGE OUTGRD JEND JVPORT PLTCSI PLTELM PLTNOD
RDCARD JFRAME JWINDO PLTCTR PLTESW PLTRED IOROUT
JBEAM JLSTYL PLTBEG PLTDCL PLTROT LIGRNO JCLOSE
JMOVE PLTBOT PLTDMP PLTEXP PLTSAF NUMBER JDRAW
JOPEN PLTCMA PLTDIS PLTHID PLTSYM EDITCT JKEYBD
PLTLAY PLTCSF PLTDOP PLTMAG PLTWID XYGRAF EDITE6
PLTCTE SETS SETUP SETETN

**Argument List:**
IRET - error return switch
H - horizontal node position array
V - vertical node position array
IDS - array of nodes in the display list
LINES - array of line connectivities in the display list
Z - z coordinate array
IISW - return switch for multiple set processing
Important Variables: Same as argument list

Common Blocks:

<table>
<thead>
<tr>
<th>CHAR</th>
<th>DIBAUD</th>
<th>NOHEAD</th>
<th>PINFLA</th>
<th>PLTITL</th>
<th>PLOT</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ</td>
<td>PLOTBD</td>
<td>ELHEAD</td>
<td>DBREC</td>
<td>TEMP</td>
<td>PLOTCM</td>
</tr>
<tr>
<td>PLOTEL</td>
<td>BLANK</td>
<td>SCALAR</td>
<td>PERM</td>
<td>TKTRNX</td>
<td>DITOKD</td>
</tr>
</tbody>
</table>
Subroutine: PLTARW

Algorithm: Routine plots arrowheads for vectors. Draws two sets of three vectors, one inside the other, in order to fill in the arrowhead.

Input/Output: None

Error Messages: None

External Calls: JDRAW JMOVE

Argument List:
- AIH - start horizontal position for the arrow
- AIV - start vertical position for the arrow
- ICODE - switch array for the number of arrows to be drawn
- H - horizontal offset for the arrow
- V - vertical offset for the arrow

Important Variables: Same as argument list

Common Blocks: None
<table>
<thead>
<tr>
<th>Subroutine:</th>
<th>PLTBEG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithm:</td>
<td>Calls all the initialization routines to start DI-3000 before the plots are started.</td>
</tr>
<tr>
<td>Input/Output:</td>
<td>None</td>
</tr>
<tr>
<td>Error Messages:</td>
<td>None</td>
</tr>
</tbody>
</table>
| External Calls:  | JBEAM  JDINIT  JIENAB  JVSPAC  
                           JBEGIN  JEND  JSETER  JWCLIP  
                           JDEVON  JFILES  JVPORT  JWINDO  
                           JSETDB  JDSIZE |
| Argument List:   | ISW - not used |
| Important Variables: | None        |
| Common Blocks:   | None         |
Subroutine: PLTBCF

Algorithm: Routine outputs the CBAR offset values to the terminal. It checks the display list for CBAR elements and then determines the components to be output. These component values are then displayed at the respective element centroids.

Input/Output: IODB - Data base input and output
IOPAC - Packed I/O to data base

Error Messages: None

External Calls: IODB JCONVW JMOVE
IOPAC JMARGN PLOTNU
PLOTNR

Argument List: N2 - number of elements in AIS
AIS - list of element centroids and numbers

Important Variables: Same as argument list

Common Blocks: ELHEAD DBREC
DITCKD PLTITL
MATL
Subroutine: PLTB01

Algorithm: Routine superimposes the deformed on the undeformed shape plots. It begins by saving the deformed coordinates and then determines the minimum and maximum scales and computes new screen positions for the model coordinates. Finally, it recalls the deformed coordinates, computes their locations and plots the deformed model to the screen.

Input/Output: None

Error Messages: None

External Calls: IOROUT JMOVE PLTB02 PLTDMP JDRAW OUTGRD PLTB03 PLTNOD JLSTYL PLTB01 PLTB04

Argument List: H - horizontal screen positions V - vertical screen positions LINES - display line connectivity array Z - dummy array of Z coordinate values

Important Variables: Same as argument list

Common Blocks: BLANK DITOKD PERM
Subroutine: PLTB01

Algorithm: Routine computes the minimum and maximum sizes for the superimposed deformed or undeformed plots. It rotates the coordinates to the desired orientation and calls PLTMXN to get the minimum and maximum values.

Input/Output: None

Error Messages: None

External Calls: PLTMXN

Argument List: XX - array of x coordinates
YY - array of y coordinates
ZZ - array of z coordinates
NSET1 - set of nodes matching the coordinates
N1 - number of nodes in NSET1
TT - array of minimums and maximums returned

Important Variables: Same as argument list

Common Blocks: PLOTCM
BLANK
Subroutine: PLTB02

Algorithm: Routine converts the model coordinates to horizontal and vertical screen positions. It uses the minimum and maximum values to scale the model coordinates to the maximum screen size.

Input/Output: None

Error Messages: None

External Calls: None

Argument List: H - horizontal screen position returned
XX - x coordinate values
V - vertical screen position returned
YY - y coordinate values
ZZ - z coordinate values
NSET1 - set array of nodes matching the coordinates
N1 - number of nodes in NSET1
TT - array of minimums and maximums

Important Variables: Same as argument list

Common Blocks: PLOTCM
Subroutine: PLTB03

Algorithm: This routine is a subset of the PLTROT subroutine. It contains the code used to determine which lines are common. PLTB03 determines which lines are within the plot and sets them up for plotting at the terminal.

Input/Output: None

Error Messages: None

External Calls: IOROUT

Argument List: 
H - horizontal screen position of the coordinates 
V - vertical screen position of the coordinates 
LINES - display line connectivity array 
J1 - number of output vectors 
I - switch for the line table name

Important Variables: Same as argument list

Common Blocks: None
Subroutine: PLTB04

Algorithm: PLTB04 determines the minimum and maximum coordinate values for the deformed or the undeformed plots. It checks the deformed min/max array against the undeformed minimum/maximum array and returns the final array.

Input/Output: None

Error Messages: None

External Calls: None

Argument List: T1 - input minimum/maximum array for the deformed plot
T2 - input minimum/maximum array for the undeformed plot
T3 - output minimum/maximum array

Important Variables: Same as argument list

Common Blocks: None
Subroutine: PLTCMA

Algorithm: Routine sets the contour level values for the element material properties. First, it retrieves the material data and element material pointers. Next the routine cycles through the element list, loads the material values at the element nodes, and finally calls the PLTCVL routine to average the node material values to determine the contour line position.

Input/Output: Unit 7 - terminal output for messages

Error Messages: *** CONTOUR DATA BLOCK DOES NOT EXIST
*** MAT2 DATA BLOCK DOES NOT EXIST
*** THERE ARE ISOTROPIC MATERIAL PROPERTIES IN ELEMENT SET

External Calls: GETMAT IOROUT IOPAC
IFINDN PLTCVL

Argument List: KFLAG - error return flag
CNTR - array with material values at the nodes
ICNTR - array with the number of values at each node
NSET - set of display list element pointers
VALUES - material values record
IOLD - element numbers

Important Variables: Same as argument list

Common Blocks: MATL TEMP NASTRN DITOKD
ELHEAD DBREC PERM MAT12
Subroutine: PLTCSF

Algorithm: Routine determines the contour level values for the element stresses or forces. First, it retrieves the element pointers and stress or force data block. Next, the routine cycles through the element list retrieving the appropriate data components and storing the data values at the element's nodes. Finally, it calls PLTCVL to average the stress or force data values at the nodes.

Input/Output: Unit 7 - terminal output for messages

Error Messages: *** LOAD CASE ____ DOES NOT EXIST FOR ____ DATA
*** ____ DATA BLOCK DOES NOT EXIST
*** NO INPUT ON CADS DATABASE FOR ELEMENT TYPE ____
*** ELEMENT TYPE ____ HAS NO INFORMATION ON MASTER FILE
*** ELEMENT NO. ____ DOES NOT HAVE OUTPUT DATA
*** OPTSTAT ELEMENT ____ DOES NOT HAVE LAYER DATA
*** OPTSTAT ELEMENT ____ HAS LAYER DATA: NO STRESSES

External Calls: IFINDN IOROUT UTLD&P
IODB PLTCVL IOPAC

Argument List: KFLAG - error return flag
CNTR  - array of data values at the nodes
ICNTR - array of the number of values at each node
NSET  - set of element display list pointers
IOLD  - array of actual element numbers
IBUFF - buffer record for data base I/O

Important Variables: Same as argument list

150
<table>
<thead>
<tr>
<th>Common Blocks:</th>
<th>MATL</th>
<th>PLOTB1</th>
<th>DITOKU</th>
<th>PLOT</th>
<th>PERM</th>
<th>HEADPP</th>
<th>PLOTBD</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ELHEAD</td>
<td>SYSTEM</td>
<td>TEMP</td>
<td>PLOTB2</td>
<td>CBREC</td>
<td>NASTRN</td>
<td>PLOTCM</td>
</tr>
<tr>
<td></td>
<td>TYPE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

151
Subroutine: PLTCSI

Algorithm: Routine is used to determine contour values for the element size data. It begins by retrieving the property data and element pointers to that data. Next, PLTCSI cycles through the elements checking for the correct property components and places those values at the element nodes. Finally, it calls PLTCVL to average the node values.

Input/Output: Unit 7 - terminal output for messages

Error Messages: *** SIZE CONTOUR DATA BLOCK DOES NOT EXIST
*** THERE ARE MIXED PROPERTY TYPES IN SET

External Calls: IFINDN PLTCVL IOPAC IOROUT

Argument List: KFLAG - error return flag
CNTR - array of nodes with component values
ICNTR - array of the number of values at each node
NSET - element display list pointers
VALUES - property data record

Important Variables: Same as argument list

Common Blocks: MATL DBREC TEMP PERM
ELHEAD NASTKN DITOKD
Subroutine: PLTCTE

Algorithm: Routine processes the element display list and determines the centroid of each element in screen coordinates. It cycles through the element list; and retrieves the element nodes and their screen positions. These node positions are then averaged to determine the element centroid. If the centroid is outside the screen window the element is not displayed.

Input/Output: None

Error Messages: None

External Calls: IFIHDN JCONWV IOPAC IOROUT

Argument List: N2 - number of element centroids (output) IS - element display list pointer array H - array of node horizontal screen positions V - array of node vertical screen positions NSET1 - element display set list AIS - centroids of the display elements

Important Variables: Same as argument list

Common Blocks: BLANK NOHEAD TEMP MATL PERM ELHEAD NASTRN TKTRNX DBREC TYPE
Subroutine: PLTCTN

Algorithm: Routine processes the node display list to check for placement on the screen. If off the screen, then the node is not processed. Routine is used to place the node number and other data at the node without screen wraparound.

Input/Output: None

Error Messages: None

External Calls: IOROUT
              JCONWV

Argument List: N2  - number of nodes returned
               IS  - node centroid pointer array
               H   - node horizontal screen position array
               V   - node vertical screen position array
               NSETI - node set display list
               AIS  - node centroid array

Important Variables: Same as argument list

Common Blocks: TKTRNX
               PERM
               TEMP
Subroutine: PLTCTR

Algorithm: Routine outputs the contour lines to the screen. It begins by plotting the boundary lines of the display and then retrieves the element connectivity and contour data blocks. Linear interpolation between the element nodes is used to contour the value lines and these are then output. Finally the margin information is output for the plot.

Input/Output: None

Error Messages: None

External Calls
IFINDN  JDRAW  IOROUT  J1IGET  PLTDOP
IOPAC  JMOVE  J1STRG  PLTDCL

Argument List:
H - node horizontal screen position array
V - node vertical screen position array
VALUES - contour values array
LINES - connectivity array for the display
NSET - set of elements array for the display

Important Variables: Same as argument list

Common Blocks:
PLOT     TEMP     D1TOKD   TKTRNX
ELHEAD   DBREC    NASTRN   MATL   PLOTCM
Subroutine: PLTCVL

Algorithm: Routine determines the contour level values at the node locations for the various element values. It cycles through the contour block averaging the values at each node and then calls PLTSCL to obtain the scaled levels for the contour values.

Input/Output: Unit 7 - terminal output for messages

Error Messages: *** ERROR OCCURRED IN CONTOUR LEVEL SCALING ROUTINE

External Calls: IOROUT PLTSCL

Argument List: KFLAG - error switch flag
FLAG - scaling routine flag switch
CNTR - real array of contour values at the nodes
ICNTR - integer array of the number of values at each node

Important Variables: Same as argument list

Common Blocks: PERM TEMP DITOKD
**Subroutine:** PLTDCL

**Algorithm:** Routine closes the current plot display, resets the screen window and finally reopens the terminal for output.

**Input/Output:** None

**Error Messages:** None

**External Calls:** JCLOSE JWINDO JOPEN JVPORT

**Argument List:** None

**Important Variables:** None

**Common Blocks:** D1TOKD
Subroutine: PLTDIS

Algorithm: Routine retrieves the node displacements or the eigenvectors and outputs them as values at the nodes. First it retrieves the correct data block and then cycles through the node display list. It searches for the requested node value components and outputs them to the screen. Finally it outputs the appropriate titles to the display margin.

Input/Output: Unit 7 - terminal output for messages

Error Messages: *** NO LOAD CASES STORED FOR ___ DATA

*** ERROR LOAD CASE ___ DOES NOT EXIST FOR ___ DATA

External Calls: IODB JMARGN PLTNR PLTDO
IOPAC JMOVE PLTCTN UTLD
JCONVW JISTRG PLTD

Argument List: H - horizontal screen position for the nodes
V - vertical screen position for the nodes
IS - array of node pointers for AIS
AIS - array of node centroids
NSET1 - set of nodes in the display list

Important Variables: VALUE - node value array
LDCASE - subcase number
MASTER - master pointer record

Common Blocks: MATL DITOKD HEADPP PLOTCM PLOT
BLANK TEMP DBREC PLOTB PLOTITL
SYSTEM TYPE PERM PLOTEL
Subroutine: PLTDMP

Algorithm: Routine stores or retrieves the H and V arrays to provide additional scratch array area for plotting.

Input/Output: IOROUT - dumps array to scratch file

Error Messages: None

External Calls: IOROUT

Argument List:

H - horizontal position array
V - vertical position array
IG - a switch: =1 will retrieve the H and V tables; =2 will store the H and V tables.
N1 - number of values in the H and V arrays
NAMEH - character variable for the name of the H array
NAMEV - character variable for the name of the V array

Important Variables: Same as argument list

Common Blocks: None
Subroutine: PLTDOP

Algorithm: Resets DI-3000 with an open and close for the screen window.

Input/Output: None

Error Messages: None

External Calls: JCLOSE JVPORT
                 JOPEN JWINDO

Argument List: None

Important Variables: None

Common Blocks: D1TOKD
Subroutine: PLTEID

Algorithm: Routine retrieves the element identification numbers for the element display list. These are then stored in the NSET1 array for output to the terminal screen.

Input/Output: None

Error Messages: None

External Calls: IOPAC

Argument List: N2 - number of elements in the display list
               IS - pointer array to the elements in the display list
               NSET1 - output array with element numbers

Important Variables: Same as argument list

Common Blocks: ELHEAD
               DBREC
               MATL
Subroutine: PLTELM

Algorithm: Routine outputs the element information to the screen based upon the switches set in PLTRED by the user. First, it outputs the element labels as text strings, next it outputs the element numbers, or the element group-offset number, and finally calls PLTMAT or PLTPRP for the material and size data.

Input/Output: None

Error Messages: None

External Calls: GETMAT JMOVE PLTCTE PLTEID PLTMA1 JCONVW J1STRG PLTDCL PLTMAT JMARGIN PLOTNO PLTDOP PLTPRP

Argument List: H - horizontal node position array V - vertical node position array NSET1 - element set for the display list ASET1 - dummy array passed to PLTPRP IS - element centroid pointer into AIS AIS - array of element centroids

Important Variables: ISWS - switch array for specifying outputs

Common Blocks: PLOT D1TOKD MAT12 TEMP TYPE DBREC PERM PLOTCM ELHEAD NASTRN PLTITL
Subroutine: PLTESW

Algorithm: Routine selects the stress or force components to be output. First, it decodes the user command and transfers it to an appropriate area. Next it determines the switches to be set on each element and packs the switches into a word. The routine then processes the DISPLAY CLEAR command for previously set switches and the DISPLAY MODE commands which are used to set the specific output mode.

Input/Output: IOPAC - performs packed I/O to the GEOM data base
RDCARD - free read terminal command input
Unit 7 - terminal output for messages

Error Messages:
*** COMPONENT ____ IS NOT VALID FOR ELEMENT TYPE ____
HELP ELEMENT TYPE WILL DISPLAY VALID COMPONENTS

*** INSUFFICIENT TERMS FOR CLEAR OPERATION

*** ____ IS INVALID ELEMENT TYPE

*** ____ IS AN INVALID OUTPUT MODE

*** ERROR *** TYPE ____ IS NOT A VALID ANALYSIS PROGRAM; REENTER

*** ERROR *** PROGRAM TYPE ____ DOES NOT MATCH PROGRAM TYPE ____ ON POST DATA BASE; REENTER

External Calls: RDCARD
UTLSLS
PLTES1
IOPAC

Argument List: None
Important Variables:

- NBTYPE - output mode type
- NELSW - array of component switches
- ELTYPE - array of valid element types

Common Blocks:

- READ
- PLOTEL
- PERM
- TEMP
- PLOTBD
- HEADPP
- SYSTEM
- D1TOKD
- PLOTB2
- TYPE
- DBREC
Subroutine: PLTES1

Algorithm: This routine decodes the element type and component keywords for stress and force output. First, it checks the mode and valid element type. Next the routine searches the component names for valid names and outputs them for the HELP command.

Input/Output: Unit 7 - terminal output for messages

Error Messages:

*** OUTPUT MODE AND PROGRAM MUST BE SET BEFORE ELEMENT COMPONENT SELECTION

*** FORMAT FOR HELP IS HELP ELEMENT TYPE REENTER

*** ____ IS INVALID ELEMENT TYPE

*** ELEMENT TYPE ____ CANNOT OUTPUT INDICATED INFORMATION

External Calls: None

Argument List:

MODE - switch for the type of output
IG - switch for help: 1=no help, 2=help
NELMTP - element type number
NW - number of possible components for the element
NF - offset into the component name array for the element
* - alternate return

Important Variables: Same as argument list

Common Blocks: READ PLOTEL PERM PLOTB2
TYPE PLOTBD SYSTEM
Subroutine: PLTEXP

Algorithm: Routine plots elements as exploded displays. It cycles through the element display list retrieving the corner point node positions. New corner points are determined so that the element is shrunken about its centroid. The element lines are then drawn on the screen in the current line style. At the end of the routine the element local axis is determined and output if requested.

Input/Output: None

Error Messages: None

External Calls: IFINDN JDRAW IOROUT JPINDEX JPOLGN IOPAC JLSTYL JMOVE JPINTR

Argument List: H - horizontal node screen position array
V - vertical node screen position array
NSET1 - element set display list
ICOL - switch for color processing 0=no color; 1=color
ICOLS - array with element types to be colored

Important Variables: IBUFF - element connectivity array buffer
NT - element type number

Common Blocks: D1TOKD NASTRN TKTRNX NTL SOLIDS
DBREC PERM TYPE PINFLA
ELHEAD TEMP TYPEN PLOT
Subroutine: PLTGN0

Algorithm: Routine plots the node numbers for elements generated during the NATURAL generation mode. Cycles through the node list, converts the binary number to a character string and outputs the string to the terminal.

Input/Output: None

Error Messages: None

External Calls: J1STRG
                JMOVE

Argument List: H - horizontal node screen position array
                V - vertical node screen position array

Important Variables: Same as argument list

Common Blocks: MATL
               PERM
Subroutine: PLTHBR

Algorithm: Routine generates new grid locations for hidden line plots of exploded views. It cycles through the coordinates for the hidden lines and scales them for exploded views.

Input/Output: None

Error Messages: None

External Calls: None

Argument List: NTRP - number of corners to the element
NEL - pointer into coordinates for the element
T1 - total x axis value to be averaged
T2 - total y axis value to be averaged
T3 - total z axis value to be averaged
RSCL - scale factor for the exploded views
XYZ - array for the coordinates of the element

Important Variables: Same as argument list

Common Blocks: None
**Subroutine:** PLTHCF

**Algorithm:** This routine is the COEF routine for COSMIC's hidden line package. It determines the equations of lines and planes. It looks for matching coordinates and then determines the line segment and plane equations.

**Input/Output:** None

**Error Messages:** None

**External Calls:** None

**Argument List:**
- X - array of x coordinates for the points
- Y - array of y coordinates for the points
- Z - array of z coordinates for the points
- XXX - storage array for the plane equations
- JXX - offset factor for the pointer into the CCC and XXX arrays
- NC - not referenced
- NS - number of points being processed
- CCC - storage array for the line segment equations
- LZ - offset factor for determining the point position in the CCC array

**Important Variables:** Same as argument list

**Common Blocks:** GO3
Subroutine: PLTHCK

Algorithm: Routine is the CHECK routine from the COSMIC hidden line package. It solves for the points of intersection of the lines of the jth element with other relevant lines and planes.

Input/Output: None

Error Messages: None

External Calls: None

Argument List:
- XXX - array from PLTHCF with the equations of the planes
- CCC - array from PLTHCF with the equations of the lines
- NNO - array with the counters for the elements to be checked
- J - not referenced
- II - counter for the number of lines to be checked
- NC - not referenced
- XI - storage array for the x coordinate of the intersections
- YI - storage array for the y coordinate of the intersections
- NGX - counter array for number of intersections
- ZM - maximum z value for checking the point
- ZMI - minimum z value for checking the point
- RV - maximum y value for checking the point
- RVI - minimum y value for checking the point
- TGM - minimum x value for checking the point
- TGI - maximum x value for checking the point
- ZI - storage array for z coordinate of intersection
- LZ - used to get offset into CCC array

Important Variables: Same as argument list
Common Blocks:  DAVE
               HEDG
               GO3
Subroutine: PLTHID

Algorithm: Routine sets up element data for processing through the hidden line routines. It finds the maximum diagonal of the display to adjust the screen area. Next the routine cycles through the element display list retrieving the node coordinates and storing them in the correct arrays for the hidden line processors based upon the element type. Finally it cycles through the element list calling the hidden line processing routines.

Input/Output: Unit 7 - terminal output for messages

Error Messages: *** ERROR MORE THAN ___ FACES IN SET ___ NUMBER WAS ___

*** ERROR _____ ELEMENTS ARE NOT YET SUPPORTED FOR HIDDEN LINE PLOTS WILL CONTINUE PROCESSING

External Calls: IFINDN JCLOSE JVPORT PLTDOP
IOPAC PLTHSK JWINDO PLTHBR
IOROUT JOPEN PLTDCL PLTHMN

Argument List: H - horizontal node positions used for the x coordinate
V - vertical node position used for the y coordinate
U - scratch array used for the z coordinate
NSET1 - element set for the display list
NSET2 - node set for the display list
IBUFF - element connectivity buffer record

Important Variables: XYZ - coordinate array of element faces for hidden line processing
NEL - number of faces

Common Blocks: TEMP PLOTCM TYPE TYPEN PERM SCALAR
BLANK ELHEAD DBREC D1TOKD NASTRN

172
Subroutine: PLTHLN

Algorithm: This is the LIN routine from the COSMIC hidden line package. It performs the executive functions for the hidden line processing. It first sets the counters and offsets into the working arrays and determines the Euler rotation angles. Next it stores the coordinate and pen positions and 3-D transformations for the points and calculates the line and plane equations for the grid points and relevant elements. Next PLTHLN sorts the coordinate points and relevant and polygon projections and intersected line segments before finally cycling through the intersection and line tables to actually plot the lines.

Input/Output: None

Error Messages: None

External Calls: PLTHCF PLTST PLTHV1 PLTHVR PLTHCK PLTHSV PLTHPL

Argument List: X - x coordinates of the face points
Y - y coordinates of the face points
Z - z coordinates of the face points
NP - number of points in the face
NC - switch for the last face: 0=more faces, 1=last face

Important Variables: Same as argument list.

Common Blocks: BLANK SCALAR
G03 HEDG

173
Subroutine: PLTHMN

Algorithm: Determines the pointers to the maximum and minimum nodes for a coordinate axis. It cycles through a given array searching for the minimum and maximum values.

Input/Output: None

Error Messages: None

External Calls: None

Argument List:
- IXMIN - minimum pointer returned
- IXMAX - maximum pointer returned
- ARRAY - array containing the real values to be searched
- IARRAY - array containing the pointers into ARRAY
- NUMB - number of values to be searched

Important Variables: Same as argument list

Common Blocks: None
Subroutine: PLTHPL

Algorithm: This is the PLT routine from the COSMIC hidden line package. It plots points governed by the IM and IJ switches. It determines the X and Y values for the current point and moves or draws to it.

Input/Output: None

Error Messages: None

External Calls: JDRAW
                JMOVE

Argument List: X1 - x coordinate array for defining the current point
                Y1 - y coordinate array for defining the current point
                IJ - switch for a draw or move
                IM - switch to reset IJ

Important Variables: Same as argument list

Common Blocks: None
Subroutine: PLTHSA

Algorithm: This is the STAT routine from the COSMIC hidden line package. It takes points of intersection from PLTHSV and picks the maximum and minimum x coordinates of the points. First it determines the projection of the point and then the minimum and maximum values.

Input/Output: None

Error Messages: None

External Calls: None

Argument List: MT - number of points to be processed
NIT - counter for an added line
IXR - used to get the offset into the X21, Y21, Z21 arrays
X21 - storage array for the maximum and minimum x coordinates
Y21 - storage array for the corresponding y coordinates
Z21 - storage array for the corresponding z coordinates
IIA - storage array for the move/draw switches for PLTHPL
IV - array of values used to determine offsets into the CCC and XXX arrays
A,B,C - coefficients for the line being studied
IK - used to get the offset into the XXX array
XA - x coordinate of the intersection points
YA - y coordinate of the intersection points
ZA - z coordinate of the intersection points
CCC - array from PLTHCF used for the line segment equations
XXX - array used to get the offsets into CCC; it came from PLTHCF
NC - not referenced
LZ - used to get the offset into the CCC array
Important Variables: Same as argument list

Common Blocks: DAVE
              GO3
Subroutine: PLTHSK

Algorithm: This is the SKETCH routine from the COSMIC hidden line package. It sets up the move or draw motion detectors. First it initializes internal variables and then searches for matching coordinates and stores the matches. Finally it sets the move switch and returns.

Input/Output: None

Error Messages: None

External Calls: PLTHLN

Argument List: X - x coordinates for the face being processed
Y - y coordinates for the face being processed
Z - z coordinates for the face being processed
NP - number of corner points for the face
NC - switch: 0=more faces in the plot; 1=the last face

Important Variables: Same as argument list

Common Blocks: SCALAR
Subroutine: PLTHST

Algorithm: This is the STATUS routine from the COSMIC hidden line package. It determines the visibility of a point by drawing a line from the point in question to infinity and counting the number of times it crosses the boundaries of a relevant element.

Input/Output: None

Error Messages: None

External Calls: None

Argument List:
- OJ - x value of the point being processed
- TMJ - y value of the point being processed
- XXX - array from PLTHCF with the plane equations
- TGM - minimum x storage array for the boundary check
- RV - maximum y storage array for the boundary check
- RVI - minimum y storage array for the boundary check
- TGI - maximum x storage array for the boundary check
- ZM - maximum z storage array for the boundary check
- NNO - array with offset pointers into the CCC array
- II - counter for the number of relevant elements to be checked
- H - not referenced
- IM - switch for the move or draw of a line
- JXT - not referenced
- ZJ - z value of the point being processed
- NC - not referenced
- ZMI - not referenced
- CCC - array from PLTHCF with the line equations
- LZ - factor for the offset into CCC

Important Variables: Same as argument list

Common Blocks: G03
Subroutine: PLTHSV

Algorithm: This is the SOLVE routine from the COSMIC hidden line package. It solves for the intersection lines resulting from the intersection of the Jth element with other relevant elements. First it checks if the element is to be considered, determines the line equations and checks the boundary. Finally it determines the lines of intersection across the elements and stores the corresponding values.

Input/Output: None

Error Messages: None

External Calls: PLTHSA

Argument List: IXR - passed to PLTHSA
J - used for the offset into the XXX array
XXX - array from PLTHCF used to get the offsets into CCC
CCC - array from PLTHCF with the equations of the line segments
II - counter for the number of relevant elements to check
NNO - array with counters for the elements to be checked
NIT - counter for added lines
X21 - dummy array for PLTHSA
Y21 - dummy array for PLTHSA
Z21 - dummy array for PLTHSA
IIA - dummy array for PLTHSA
NC - passed to PLTHSA
ZM,ZML - arrays with switch checks for determining the relevant elements
LZ - factor used in getting the offsets into the CCC array
Important Variables: Same as argument list

Common Blocks: DAVE
              GO3
Subroutine: I'LHVR

Algorithm: This is the VSRTR routine supplied with the COSMIC hidden line package. It sorts an array of values for the hidden line package. Routine is an IMSL routine.

Input/Output: None

Error Messages: None

External Calls: None

Argument List: A - array to be sorted
LA - number of elements in A to be sorted
IR - vector of length LA

Important Variables: Same as argument list

Common Blocks: None
Subroutine: PLTHV1

Algorithm: This is the VSRT1 routine from the COSMIC hidden line package. It performs a partial sort for the hidden line package.

Input/Output: None

Error Messages: None

External Calls: None

Argument List: A - array to be sorted
LA - length of the elements to be sorted
IR - scratch array of length LA

Important Variables: Same as argument list

Common Blocks: None
Subroutine: PLILAY

Algorithm: Routine outputs composite layer data to the terminal. First finds out which actual elements were picked by the user in PLTWID and then retrieves the composite data for the elements. The selected data is then processed and output to the terminal as detailed composite element layer data.

Input/Output: IOPAC - performs packed data I/O to the GEOM data base

Error Messages: None

External Calls: IOPAC JRPLGN PLTDCL JCLOSE
JMOVE J1STRG PLTDOP JFRAME
JRMOVE PLTCTE PLTEID JOPEN

Argument List: H - array of horizontal grid positions
V - array of vertical grid positions
NSET1 - element set for display
NP34 - integer array with composite layer data values
P34 - real array with composite layer data values
IS - array of element centroid pointers
AIS - array of element centroid positions

Important Variables: Same as argument list

Common Blocks: MATL ELHEAD
DBREC TEMP
DITOKD
Subroutine: PLTMAG

Algorithm: Routine plots the margin information and lines after a plot is completed. It steps through the switches and outputs the margin text as required. Next it draws the separator lines at the correct locations and finally outputs the model orientation as a small axis system.

Input/Output: IODB - performs I/O directly to the GEOM data base
IOPAC - performs packed data I/O to the GEOM data base

Error Messages: None

External Calls: JISTRG PLTDCL JMOVE IOPAC JPINTR
JDRAW PLTDP IODB JPINDEX JRPLGN

Argument List: H - array of horizontal coordinate positions
V - array of vertical coordinate positions
ICOL - switch for color processing 0=no color, 1=color
ICOLS - array with element types to be colored

Important Variables: Same as argument list

Common Blocks: D1TOKD TYPE PLOT PLTTL PLOTCM
TEMP PLOTCM SYSTEM PERM NASTRN
HEADPP DBREC
Subroutine: PLTMAI

Algorithm: Routine outputs the element material information to the screen. It checks for the value types to be output and then cycles through the element list. For each element it retrieves the appropriate material values, positions the values around the element centroid and calls PLOTNR to output the actual value. It continues searching the material switches to output all requested properties for that element.

Input/Output: None

Error Messages: None

External Calls: IOPAC JMARG PLTDCL JCONVW J1STRG JMOVE PLTDOP PLOTNR

Argument List: N2 - number of elements in the set
IS - array of the element pointers
AIS - array of element centroids
H - node point horizontal screen positions
V - node point vertical screen positions
MAT1 - material array for the isotropic elements
M1 - number of rows in MAT1
MAT2 - anisotropic material array
M2 - number of rows in MAT2

Important Variables: Same as argument list

Common Blocks: PLOT ELHEAD D1TOKD MAT12 PLTITL PLOTCM
MATL TYPE DBREC PERM NASTRN TEMP
Subroutine: PLTMAI

Algorithm: This routine outputs the composite lamina properties to the terminal based upon the user requests. First it retrieves the components to be output. Next it sets up the material values for output before finally cycling through the element list to write out the values to the terminal.

Input/Output: Unit 7 - terminal output for messages
IODB - data base direct access I/O

Error Messages: ** NO COMPOSITE MATERIAL PROPERTIES **

External Calls: IODB JCONVW PLOTNO PLTDOP
IOPAC JMARGIN PLOTNR
J1STRG JMOVE PLTDCL

Argument List: N2 - number of elements in IS and AIS
IS - array with the element list to be plotted
AIS - array with the centroids of the elements
H - array with the horizontal screen positions of the nodes
V - array with the vertical screen positions of the nodes
NPLY - integer array for the composite material values
PLY - real array for the composite material values

Important Variables: Same as argument list.

Common Blocks: PLOT D1TOKD PLTITL
MATL DBREC PLOTCM
ELHEAD PERM TEMP
Subroutine: PLTMXN

Algorithm: Routine searches an array for its minimum and maximum values.

Input/Output: None

Error Messages: None

External Calls: None

Argument List: XMIN - minimum value of the array
XMAX - maximum value of the array
ARRAY - array of values to be searched
IARRAY - list of positions in ARRAY to be checked.
NUMB - number of values in array IARRAY

Important Variables: Same as argument list

Common Blocks: None
Subroutine: PLTNAT

Algorithm: Routine plots elements as they are generated in the NATURAL generation routines. It gets the element's nodes; moves the node coordinates into a temporary array; and finally outputs the element line segments.

Input/Output: None

Error Messages: None

External Calls: IFINDN, JDRAW, JMOVE

Argument List: H - array of horizontal node positions
V - array of vertical node positions
IBUFF - buffer of node connectivity defining the elements
NST - number of rows in IBUFF
NTYPE - the element type number
KS - start position in IBUFF for the element to be displayed
KL - end position in IBUFF for the element to be displayed

Important Variables: Same as argument list

Common Blocks: NASTRN, MATL, PERM, SOLIDS
Subroutine: PLTNOD

Algorithm: Routine outputs the node numbers and coordinates to the screen. First it puts the node numbers at the nodes using the plot set information. The routine then checks for the coordinate value output request and cycles through those values if necessary. Next, it outputs the suppression data if requested and finally outputs the external force and moment data as appropriate.

Input/Output: None

Error Messages: None

External Calls: JISTRG JMOVE PLTCTN FREUCK IOPAC JFINDG JCONVW PLOTNO PLTDCL JMARGN PLOTNR PLTDOP FREPCK PLTARW

Argument List: 
H - horizontal node position
V - vertical node position
IS - array of node pointers for the plot
NSET1 - set of nodes for plotting
AIS - centroids of the nodes on the plot
ICASE - external load case number to be output

Important Variables: 
CXYZ - coordinates of the nodes in the model
ISWS - switch array for the type of output information

Common Blocks: BLANK TKTRNX PLTITL PERM TEMP DBREC D1TOKD PLOTCM PINFLA PLOT MATL NOHEAD
Subroutine: PLTPLS

Algorithm: Routine outputs a plus sign (+) at the node locations for the node only plots. It also writes out the node number as a character string.

Input/Output: None

Error Messages: None

External Calls: J1STRG JMOVE
                JCMARK JMARK

Argument List: H - horizontal positions of the nodes
               V - vertical positions of the nodes
               N - number of nodes to be output

Important Variables: NCXYZ - array of node numbers and their coordinates

Common Blocks: BLANK
Subroutine: PLTPRP

Algorithm: Routine outputs the element size information to the screen based upon the user commands. First it outputs the correct titles to the plot margin and then cycles through the element list. It checks for the correct property table and then outputs the appropriate sizes to the terminal.

Input/Output: IOPAC - performs packed data I/O to the data base

Error Messages: None

External Calls: IOPAC JMARGIN PLTDCL JCONVW PLTBOF J1STRG JMOVE PLTDOP PLOTNR

Argument List: N2 - number of elements in the display list IS - element pointers for the display list AIS - array of element centroid positions H - node horizontal screen positions V - node vertical screen positions

Important Variables: GEDA - size value NELGRH - array of element group/offset pointers to the property data PM - property size record from the data base

Common Blocks: ELHEAD DITOKD TEMP PLTITL DBREC NASTRN TYPE PLOTCM MATL PERM PLOT
Subroutine PLTRED

Algorithm:
Routine decodes the user keywords from the PLOT command. It clears the display switches and checks the input keywords against the valid options. It sets the output switch for the command. Finally, it checks the virtual and screen window sizes before returning to PLOTS.

Input/Output:
Unit 7 - terminal output for messages

Error Messages:
*** INVALID KEYWORD _______ FOUND FOR PLOT OPTION

*** PLOT WINDOW _______ IS UNDEFINED

*** NUMBER OF CONTOURS REQUESTED EXCEEDS MAXIMUM

*** ELEMENT TYPE _______ IS NOT VALID FOR CURRENT PROGRAM MODE

*** ELEMENT OUTPUT INDICATORS DO NOT MATCH REQUESTED TYPE

*** ______ IS NOT A VALID MATERIAL CONTOUR REQUEST

*** INVALID CONTOUR REQUESTED

External Calls: CHANGE
NUMBER

Argument List:
* - alternate error return
ICASE - external load case number for data display

Important Variables: None

Common Blocks:
READ TYPE BLANK PERM TEMP
PLOTEL PLOTBD TKTRNX PLITIL PLOTCM
TYPEN SYSTEM PLOT DI TOKD
Subroutine: PLTRNS

Algorithm: Reverses two values. It simply switches I1 to I2 and I2 to I1.

Input/Output: None

Error Messages: None

External Messages: None

Argument List: I1 - first input value to be switched
I2 - second input value to be switched

Important Variables: Same as argument list

Common Blocks: None
Subroutine: PLTRO1

Algorithm: Routine performs coordinate rotations based upon the user requests. First it decodes the ROTATE command and retrieves the undeformed coordinates. Then it checks for a deformed shape request and selects the deformations if needed. Next it calls PLTRVH to rotate the coordinates and define the H and V arrays. Finally the routine goes through the line connectivity array optimizing for plotting in the given view.

Input/Output: Unit 7 - terminal output for messages

Error Messages:

*** ROTATE KEYWORD ______ IS NOT VALID

*** LOAD CASE HAS NOT BEEN SET CORRECTLY, CASE = _____

*** MODE MUST BE SET TO DISP OR EIGE BEFORE ELEMENT COMPONENT SELECTED

*** NO LOAD CASES STORED FOR DISPLACEMENTS OR EIGENVECTORS

*** NODE NO. _____ DOES NOT HAVE OUTPUT DATA

External Calls: CHANGE IOPAC OUTGRD
IODB IOROUT PLTRVH

Argument List: IPASS - switch to mark the first pass through the routine
H - horizontal screen positions for the nodes
XX - x coordinate values of the nodes
V - vertical screen positions of the nodes
YY - y coordinate values of the nodes
ZZ - z coordinate values of the nodes
LINES - node connectivity of the display lines
NSET1 - pointer to the node positions in the node data arrays
Important Variables: Same as argument list

Common Blocks:

| READ | PLOTEL | DBREC | TEMP | HEADPP |
| MATL | D1TOKD | PLOTCM | BLANK | PERM   |

196
Subroutine: PLTRO1

Algorithm: This routine computes the node screen location during the node generation and outputs a plus sign at that point. It uses the current rotation and screen size factors to compute the rotated and screen positions for the node. Finally, it determines the node number and outputs the plus sign to the screen.

Input/Output: None

Error Messages: None

External Calls: JISTRG
                JCMARK
                JMOVE

Argument List: K1 - node position in the coordinate array

Important Variables: CXYZ - node coordinate array

Common Blocks: BLANK   NATDSP
                DITOKD   PLOTCM
Subroutine: PLTRVH

Algorithm: Routine converts the real node coordinate to the screen coordinate based upon the requested rotation. First it establishes the rotation angles and cycles through the node coordinates making up the rotated coordinate arrays. Next it determines the minimum and maximum values of the coordinates and finally factors the coordinates to the screen size.

Input/Output: None

Error Messages: None

External Calls: PLTMXN

Argument List: H - horizontal screen position of the nodes
XX - rotated x coordinate of the nodes
V - vertical screen position of the nodes
YY - rotated y coordinate of the nodes
ZZ - rotated z coordinate of the nodes
NSET1 - set of display node pointers
N1 - number of values in NSET1

Important Variables: CL - array of rotation angles

Common Blocks: BLANK PLOTCM
D1TOKD PLOTCM
Subroutine: PLTSAF

Algorithm: Routine outputs the element stress and force components for the display list of elements. It checks for the correct data blocks and pointer records for the requested data type. Next it cycles through the element list and retrieves the components based on the requested data type. Finally it outputs the values to the screen.

Input/Output: Unit 7 - terminal output for messages

Error Messages: 
*** LOAD CASE _____ DOES NOT EXIST FOR _____ DATA 
*** _____ DATA BLOCK DOES NOT EXIST 
*** NO INPUT ON CADS DATABASE FOR ELEMENT TYPE _____ 
*** ELEMENT TYPE ____ HAS NO INFORMATION ON MASTER FILE 
*** ELEMENT NO. ____ DOES NOT HAVE OUTPUT DATA 

External Calls: IODB JCONVW PLOTNR PLTDOP 
IOPAC JMARGN PLCTE UTLDBP 
J1STRG JMOVE PLTDCL 

Argument List: H - horizontal node position 
V - vertical node position 
IS - array of element list pointers 
AIS - array of element centroids 
NSET - set of elements to be displayed 
IOLD - array of element numbers 

Important Variables: LDCASE - load case 
MASTER - master pointer to display data 
VALUE - array of output values
<table>
<thead>
<tr>
<th>Common Blocks:</th>
<th>MATL</th>
<th>SYSTEM</th>
<th>PLOT</th>
<th>DBREC</th>
<th>PLOTBD</th>
<th>PLTITL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ELHEAD</td>
<td>D1TOKD</td>
<td>PLOTB2</td>
<td>HEADPP</td>
<td>PLOTCM</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PLOTEL</td>
<td>TYPE</td>
<td>PERM</td>
<td>NASTRN</td>
<td>TEMP</td>
<td></td>
</tr>
</tbody>
</table>
Subroutine: PLTS1

Algorithm: Routine retrieves a subset of the material property values for contour plots. First it gets E, G, and Poisson's ratio from the isotropic arrays and then retrieves the modulus arrays from the MAT2 data records.

Input/Output: None

Error Messages: None

External Calls: IOPAC

Argument List: MAT1 - isotropic material array
               NV1 - number of isotropic materials
               NV1R - data record for the isotropic materials
               MAT2 - anisotropic material array
               NV2 - number of anisotropic materials
               NV2R - record number for the anisotropic materials

Important Variables: Same as argument list

Common Blocks: MATL
               DBREC
Subroutine: PLTSCL

Algorithm: Routine scales the default contour levels to "nice" values of 1.0, 2.0, 2.5, or 5.0. It uses the minimum and maximum values to be scaled, determines the power of ten to be used, and then cycles through 14 contour levels to determine their respective values.

Input/Output: None

Error Messages: None

External Calls: None

Argument List: CMAX - maximum value
               CMIN - minimum value
               FLAG - error return flag

Important Variables: CLEVEL - array of contour level values

Common Blocks: D1TOKD
Subroutine: PLTSYM

Algorithm: This routine outputs a symbol at a specified screen location using the JMARK routine.

Input/Output: None

Error Messages: None

External Calls: JCMARK
              JMARK

Argument List: NOPT - option switch for the symbol type
               AIH - horizontal position for the symbol
               AIV - vertical position for the symbol

Important Variables: Same as argument list

Common Blocks: TKTRNX
Subroutine: PLTWID

Algorithm: Routine does the window processing for the VIEW, PORT, and WINDOW end of plot commands. First it positions the VIEW processing by retrieving corner points, using the cursor, of boxes to be expanded. Next it does PORT and WINDOW processing by retrieving the port or window numbers, title, and locations. This information is stored for up to 9 windows or ports. Finally it checks for additional values to be displayed on the screen and decodes those parameters as needed.

Input/Output: None

Error Messages: None

External Calls: JISTRG JKEYBD JPOLGN PLTRNS JMOVE JCLOSE JLOCAT PLTDCL JCONVW PLTDOP

Argument List: None

Important Variables: JTT - character switch for processing the command

SWH, SWV - arrays with the corner points of the window boxes

UWH, UWV - arrays with the virtual corner points of the boxes

Common Blocks: TEMP PLTITL PLOTCM

TKTRNX D1TOKD
Subroutine: PRSTR1

Algorithm: Routine stores the element property (size) values based upon the current group being processed. It brings in a data block from the temporary scratch file and updates the pointer values based upon that block. It then cycles through the block loading the property tables for permanent storage based upon the element type and allowed values.

Input/Output: ND2 - scratch unit for direct access I/O
Unit 7 - terminal output for messages

Error Messages:
*** NO PROPERTY BLOCK FOR GROUP ____ TYPE ____
*** ____ IS A BAD PROPERTY CODE FOR GROUP ____

External Calls: None

Argument List: IPARY - scratch area for the property blocks
IPAREA - input array with the property table numbers for the group
IPN - number of tables in IPAREA
IPROP - output array of group properties
MA2 - number of rows in IPROP
NUV - not used
JNDEX - index into the pointer table for the scratch unit
NUMEL - number of elements in the group
INJ - property id being processed
LIST - list of property ids for the elements
KS - last used position
IG - switch for adding values

Important Variables: Same as argument list
<table>
<thead>
<tr>
<th>Common Blocks:</th>
<th>PERM</th>
<th>TRACK1</th>
<th>SYSTEM</th>
</tr>
</thead>
<tbody>
<tr>
<td>TEMP</td>
<td>READN</td>
<td>NASTRN</td>
<td></td>
</tr>
</tbody>
</table>
Subroutine: PRSTR2

Algorithm: Routine stores a list of property values for a given list of elements.

Input/Output: None

Error Messages: None

External Calls: None

Argument List: 
- PROP - array of real property values
- NR - number of rows in PROP
- RLIST - list of values to be placed in PROP
- LIST - list of element numbers for values placed in PROP
- K3 - number of values in LIST
- LOC - row location of values in PROP

PRSTR3 ENTRY: PROP, NR, RLIST, LIST, K3, - same as PRSTR2

LOCP - array with location of input values for storage in PROP

NP - number of values in LOCP array

Important Variables: Same as argument list

Common Blocks: None
The following block of subroutines makes up the READ module of CADS. These routines provide the translation functions for CADS to read existing model data into the GEOMETRY data base for processing. The routines in this block are:

ROANAL  RDNAS6
RDAOGS  RDNATR
RDAOLS  RDNBAR
RDCARD  RDNCRD
RDCAR1  RDNGRD
RDCMIS  RDNGR1
RDCONT  RDNMAT
RDNASS  RDNOUT
RDNAST  RDNPAC
RDNAS1  RDNSHF
RDNAS2  RDOPTS
RDNAS3  RDOPT1
RDNAS4  RDOPT2
RDNAS5
Subroutine: RDANAL

Algorithm: Routine reads the ANALYZE program input data and sets it up for storage to the data base. First it clears the scratch arrays and reads the ANALYZE control cards. Next it reads in the property data, member connectivity data, and finally the coordinate data. RDAOGS is called to save the coordinate data. Finally, the applied loads are read in and stored by RDAOLS, and the element data is stored by RDCMIS.

Input/Output: NUNIT - input unit for ANALYZE data

Error Messages: None

External Calls: RDAOGS ZRAYB
               RDAOLS RDCMIS

Argument List: None

Important Variables: PAT1 - array of property values
                     XYZ - array of the coordinate values
                     NNODES - array with the number of nodes for each element
                     MA,MB,MC,MD - arrays with the node connectivities for the elements

Common Blocks: BLANK MOHEAD DBREC
               NOHEAD PERM MATL
Subroutine: RDAOGS

Algorithm: Routine converts the ANALYZE and OPTSTAT coordinates for storing on the GEOM data base. It cycles through the number of joints (nodes) for the model placing the coordinates, node number and boundary condition data into records for output to the data base.

Input/Output: IODB - routine for actual output to the data base.

Error Messages: None

External Calls: IODB

Argument List: XYZ - array of coordinates for the nodes
IB - boundary conditions of the nodes
MM - dimension of the problem (2 or 3)
JOINTS - number of joints or nodes in the model

Important Variables: Same as argument list

Common Blocks: BLANK DBREC MOHEAD
NOHEAD PERM
Subroutine: RDAOLS

Algorithm: Routine is used to pack the ANALYZE and OPTSTAT external load data for storing on the GEOM database. It cycles through the input load array and places the values into appropriate positions in a load record. These records are then stored to the database.

Input/Output: IODB - routine for direct access database I/O

Error Messages: None

External Calls: IODB ZRAYB

Argument List: FIN - array with the input loads LL - load case number

Important Variables: Same as argument list

Common Blocks: NOHEAD PERM MOHEAD DBREC
Subroutine: RDCARD

Algorithm: RDCARD performs the free read processing of user input commands. It reads a record into the CARD working buffer and then cycles through the characters filling the HOLD array with the character strings making up the individual variables.

Input/Output: NUNIT - input control record unit
Unit 7 - terminal output for messages

Error Messages: *** TRUNCATION WILL OCCUR ON THE FOLLOWING CARD ***

*** MAXIMUM WORDS FOR ARRAY EXCEEDED, WORDS INPUT ___

External Calls: NUMBER

Argument List: PROMPT - prompt string, usually the name of the calling routine
MENU - not used
* - alternate routine for error
NUNIT - input unit for the command record
HOLD - variable array returned as CHARACTER*8 words
LHOLD - variable array returned as CHARACTER*1 words
NVAR - number of variables in HOLD

Important Variables: Same as argument list

Common Blocks: CHAR
SYSTEM
READCM
Subroutine: RDCAR1

Algorithm: This routine is essentially the same as the RDCARD routine except that it is used in the NATURAL generation modules. RDCAR1 will echo the user input commands to unit 3 for future use and/or modification of a steering file.

Input/Output: NUNIT - input unit for control cards
Unit 7 - terminal output for messages
Unit 3 - control card echo file

Error Messages: *** TRUNCATION WILL OCCUR ON THE FOLLOWING CARD ___
*** MAXIMUM WORDS FOR ARRAY EXCEEDED, WORDS INPUT ___

External Calls: NUMBER

Argument List: PROMPT - prompt string, usually the name of the calling routine
MENU - not used
* - alternate routine for error
NUNIT - input unit for the command record
HOLD - variable array returned as CHARACTER*8 words
LHOLD - variable array returned as CHARACTER*1 words
NVAR - number of variables in HOLD

Important Variables: Same as argument list

Common Block: CHAR
SYSTEM
READCM
Subroutine: RDCMIS

Algorithm: Routine stores the element connectivities, sizes, and material data for ANALYZE and OPTSTAT to the data base. It loads arrays for the elements with their input connectivity data. These are then stored to the data base as groups of elements. The material property table is stored by IODB since it requires no rearrangement before being processed.

Input/Output: IODB - performs data base direct access I/O

Error Messages: None

External Calls: GROUPS
IODB
ZRAYB

Argument List: NMAT - number of material properties
MEMBS - number of elements (members)

Important Variables: ICAO - element correspondence table
I2C, I3C, I4C, I5C - arrays with the element connectivities
I2P, I3P, I4P, I5P - arrays with the element sizes

Common Blocks: NOHEAD DBREC MATL TEMP
ELHEAD MOHEAD BLANK PERM
Algorithm:
Routine controls the selection of the READ translator subroutines. It starts by setting switches, reading and decoding the command input, and finally calls in the correct translator module.

Input/Output:
RDCARD - free read command inputs
Unit 7 - terminal output for messages

Error Messages:
*** READ CONTROL OPTION ____ NOT VALID
*** READ PROGRAM TYPE ____ NOT SUPPORTED
*** ERROR *** INPUT UNIT NUMBER (_) MUST BE GREATER THAN 20; REENTER
*** YOU SHOULD HAVE SIX VALUES IN THE ABOVE ORDER - REENTER
*** ERROR *** FILE NAME ____ DOES NOT EXIST REENTER NAME
*** ERROR *** FILE NAME ____ ON UNIT ____ CANNOT BE OPENED; PROBLEM: ____
*** THE PROGRAM KEYWORD IS MISSING FOR THE READ CONTROL OPTION

External Calls:
CHANGE NUMBER RDCARD RDNATR UTLLTG
IOHEAD PLTBEGB RDNAST RDOPTS ZRAYB
LIGRNO RDANAL RDNASS START1

Argument List:
None
Important Variables:

ISW  - switch for the program type being processed
NVREC - next valid record

Common Blocks:

READ   PLOTCM  MOHEAD  NATDSP  PINFLA
TYPE    ELHEAD  BLANK   PERM
CHAR    DBREC   NASTRN  SYSTEM
Subroutine: RDNASS

Algorithm: Routine controls the translation and storage of NASTRAN bulk data decks to the geometric data base. First it resets the pointers and then cycles through the types of data cards calling in the appropriate routine to process the data.

Input/Output: ND2 - direct access unit to read the initial link record
Unit 7 - terminal output for messages

Error Messages: *** INSUFFICIENT MATERIAL WORKING ROOM (MAT1)
*** INCREASE BLANK COMMON BY _________

INSUFFICIENT WORKING AREA FOR _________ CONRODS

External Calls: RDNAS1 RDNAS4 RDNMAT RDNAS6
RDNAS2 RDNAS5 RDNAS3

Argument List: NS - switch for sorting; if NS=0, bulk data was already sorted

Important Variables: LOCATE - array which points to the data card types to be processed
D - scratch array for in-core processing
NTERM - number of data values per card type

Common Blocks: TRACK1 PERM NASTRN ELHEAD
BLANK TEMP DBREC READN
Subroutine: RDNAST

Algorithm: This is the first routine to process a NASTRAN bulk data deck. It reads in a card; determines the type; decodes the data; and stores it on a temporary scratch file.

Input/Output: NUNIT - input unit for the NASTRAN data deck
Unit 10 - output scratch unit
Unit 7 - terminal output for messages

Error Messages:
*** CONTINUATION CARD FOR CIHEX ELEMENT _____ IS MISPLACED OR MISSING: ELEMENT WILL NOT BE PROCESSED

*** COMBINATION OF AXISYMMETRIC AND STANDARD ELEMENTS IS NOT ALLOWED

*** BULK DATA INPUT TERMINATED ABNORMALLY BEGIN BULK CARD COULD BE MISSING

*** CONTINUATION CARD - IS MISPLACED OR MISSING

External Calls: IODB   RDNGRD   ZRAYB   RDNBAR
IOPAC   RDNOUT   ZRAYI   RDNSHF

Argument List: ISPC - switch for processing SPC cards

Important Variables: BUFFER - buffer for processing a NASTRAN card
NUMBA - array to track the number of cards of each data type
NTERM - number of terms on a bulk data deck card

Common Blocks: BLANK   MATL   CHAR   PINFLA   TRACK1
NOHEAD   SYSTEM   DBREC   PERM
MOHEAD   TEMPl   READN   NASTRN
Subroutine: RDNAS1

Algorithm: Routine controls the sorting of NASTRAN input data decks. It brings in the data type; calls in the sort routine; and saves the sorted data on a scratch file. It cycles through the various element types until completed.

Input/Output: ND2 - scratch file used for temporary NASTRAN storage
                   Unit 7 - terminal output for messages

Error Messages: *** INSUFFICIENT IN CORE SORT AREA AVAILABLE

External Calls: SORTQ

Argument List: INDEX - pointer into the LOCATE array for data type position
                   IROW - number of variables per data card

Important Variables: LOCATE - array with locations of the data types on a scratch file
                              AREA - in-core sort array filled from the scratch file

Common Blocks: TRACK1
                   BLANK
                   READN
Subroutine: RDNAS2

Algorithm: This routine sets up and stores CONROD information to the geometric database. It reads sorted elements, resolves the material identifications; places elements in groups, and writes data to the database. In resolving material information it checks that the material number of the CONROD matches a previously defined MAT1 card.

Input/Output: ND2 - scratch NASTRAN data file
Unit 7 - terminal output for messages

Error Messages: *** CONROD ELEMENT _____ HAS A BAD MATERIAL CODE
*** READ ERROR IN SUBROUTINE RDNAS2

External Calls: GROUPS

Argument List: INARY - buffer array for a record of information
INAREA - array used to store element data by card
NW - number of variables per CONROD
MATRIX - compressed array passed to GROUPS for output to the database
M1 - number of rows in MATRIX
IPROP - array of property values for each element
M2 - number of rows in IPROP
IOLD - array with pointers to the material tables for the elements
M3 - number of rows in IOLD
NCO - number of elements in the group

Important Variables: Same as argument list

Common Blocks: TRACK1 PERM
BLANK TEMP
READN SYSTEM

220
Subroutine: RDNAS3

Algorithm: Routine stores NASTRAN elements to the data base. It retrieves a property data block for resolution between the element connectivity and property tables. It then outputs the connectivity for the elements and resolves their property and material callouts. Finally it compresses the data tables and establishes the pointer links needed to associate the properties and materials with the element group.

Input/Output: ND2 - scratch data file with sorted NASTRAN data
Unit 7 - terminal output for messages

Error Messages: *** _____ ELEMENT _____ HAS A BAD PROPERTY CODE
*** _____ ELEMENT _____ HAS A BAD MATERIAL CODE

External Calls: GROUPS

Argument List: INDEX - pointer into the LOCATE array for a scratch file position
INARY - buffer for the data base record
INAREA - array for the element connectivity
NROW - rows in INAREA
IPARY - buffer for the property table records
IPAREA - array for the property values
IPN - number of rows in IPAREA
MATRIX - array passed to GROUPS for output to the data base
MA1 - number of rows in MATRIX
IPROP - compressed property table for the group being processed
Argument List: (continued)
- MA2 - number of rows in IPROP
- IOLD - pointer links from element groups to property tables
- MA3 - number of rows in IOLD
- NUMEL - number of elements per element type

Important Variables: Same as argument list

Common Blocks:
- TRACK1
- TEMP
- NASTRN
- PERM
- BLANK
- SYSTEM
- CHAR
- READN
Subroutine: RDNAS4

Algorithm: Routine stores the CELAS1 NASTRAN element. It follows the same procedures as the RDNAS2 routine for the CONROD elements with some additional property table processing for the CELAS1 element.

Input/Output: ND2 - scratch file input
Unit 7 - terminal output for messages

Error Messages: *** ____ ELEMENT _____ HAS A BAD PROPERTY CODE

External Calls: GROUPS

Argument List: INDEX - pointer into the LOCATE array for a scratch file position
INARY - buffer for the data base record
INAREA - array for the element connectivity
NROW - rows in INAREA
IPARY - buffer for property table records
IPAREA - array for property values
IPN - number of rows in IPAREA
MATRIX - array passed to GROUPS for output to the data base
MA1 - number of rows in MATRIX
IPROP - compressed property table for the group being processed
MA2 - number of rows in IPROP
IOLD - pointer links from element groups to property tables
MA3 - number of rows in IOLD
NUMEL - number of elements per element type

Important Variables: Same as argument list

Common Blocks: TRACK1 TEMP NASTRN PERM
BLANK SYSTEM CHAR READN

223
Subroutine: RDNAS5

Algorithm: Routine stores CELAS2 NASTRAN elements to the data base. It follows the procedures used for CELAS1 in RDNAS4 but does not require any special property card processing.

Input/Output: ND2 - scratch file for sorted NASTRAN data
Unit 7 - terminal output for messages

Error Messages: None

External Calls: GROUPS

Argument List: INDEX - pointer into the LOCATE array for a scratch file position
INARY - buffer array for a record of information
INAREA - array used to store element data by card
NROW - number of rows in INAREA
MATRIX - compressed array passed to GROUPS for output to the data base
MA1 - number of rows in MATRIX
IPROP - array of property values for the elements
MA2 - number of rows in IPROP
IOLD - array with pointers to the material tables for the elements
MA3 - number of rows in IOLD
NUMEL - number of elements in the group

Important Variables: Same as argument list

Common Blocks: TRACK1 TEMP NASTRN PERM
BLANK SYSTEM CHAR READN
Subroutine: RDNAS6

Algorithm: Routine stores the solid CTETRA and CWEDGE NASTRAN elements. It follows the same procedures used to save the CONROD data, however, these elements do not use property size data so that processing is not required.

Input/Output: ND2 - scratch file of NASTRAN sorted data
Unit 7 - terminal output for messages

Error Messages: **__ELEMENT ___ HAS A BAD MATERIAL CODE

External Calls: GROUPS

Argument List: INDEX - pointer to the LOCATE array for a scratch file position
INARY - buffer for the connectivity record for the data base
INAREA - array of connectivity data by element
NROW - number of rows in INAREA
MATRIX - compressed array for output to GROUPS
MA1 - number of rows in MATRIX
IOLD - link array of pointers from the elements to the materials
MA3 - number of rows in IOLD
NUMEL - number of elements in the group

Important Variables: Same as argument list

Common Blocks: TRACK1  TEMP  NASTRN  PERM
BLANK  SYSTEM  CHAR  READN
Subroutine: RDNAIR

Algorithm: This routine is the read controller for the NATURAL generation module. It starts by decoding the high level generation options and then calls in the routines needed to perform the specific option.

Input/Output: RDCARD - free read command input routine
Unit 7 - terminal output for messages

Error Messages:
*** CONTROL OPTION ______ NOT CORRECT FOR NATURAL MODE INPUT

*** NATURAL PROCESSOR MODULE NOT CORRECT ______

*** MODULE NAME HAS NOT BEEN ENTERED

*** MODULE: _______ CARD: _______
EQUATE LIST HAS INCORRECT NUMBER OF TERMS
NODE ______ DOES NOT PREVIOUSLY EXIST, NODE ______
NOT ASSIGNED

NODE ______ EXISTS AND CANNOT BE EQUATED
MAXIMUM OF 200 EQUATED NODES ALLOWED AT ONE TIME
USE ANOTHER EQUATE COMMAND FOR REST

External Calls: IFINDN NATELM NUMBER SETGEN JEND
IOPAC NATNCT OUTGRD SORTQ
JFINDG NATPRP RDCARD JCLOSE

Argument List: None

Important Variables: None

Common Blocks: BLANK ELHEAD NATDSP SYSTEM
READ MOHEAD NASTRN READCM
MATL DBREC PERM

226
Subroutine: RDNBAR

Algorithm: Routine decodes and stores the local reference axis information for NASTRAN beams. It checks the reference flag, and interprets the BAROR card (if supplied). Next either the reference vector is read and the data is stored as a dummy node with a node number greater than 90000000 or the grid point defining the reference axis is read.

Input/Output: Unit 7 - terminal output for messages

Error Messages:
*** BAROR CARD MUST PRECEDE ALL CBAR CARDS
*** REFERENCE PLANE NOT SET UP BY BAROR CARD

External Calls: None

Argument List:
ICODE - code for the axis flag: 1, by vector; 2 by nodes
DATA - data array to be decoded
IGRID - real or dummy node number of reference axis
LOP - BAROR card check for reference values

Important Variables: Same as argument list

Common Blocks: MATL
PERM
TEMP
Subroutine: RDNCRD

Subroutine: Routine packs and stores the NASTRAN coordinate system data from the COORD cards. It stores the current coordinate data in an array and updates a counter. Once the counter reaches 82, it sends the entire coordinate array to the data base.

Input/Output: IODB - performs data I/O directly to the GEOM data base

Error Messages: None

External Calls: IODB ZRAYI

Argument List:

IG - switch used to store data; 1=record not full continue; 2=record full so store
NCORD - count on the number of different coordinate systems
M - number of data values to be stored in the coordinate array
NSYS - switch for the type of data - grid point or values
JUNK - array with actual values for the reference system

Important Variables: NDBREC - packed coordinate data array

Common Blocks: DBREC NOHEAD
Subroutine: RDNGRD

Algorithm: First pass routine for the NASTRAN data. It reads the bulk data deck and sorts storable card to a scratch file for additional processing. The routine sets counters, checks the data card against acceptable formats, and then begins processing to decode and store GRID, BAROR, COORD, SPC, and RINGAX data cards.

Input/Output: NUNIT - input unit for the NASTRAN bulk data
Unit 10 - scratch unit for non-processed cards from this routine
Unit 7 - terminal output for messages

Error Messages:
*** ONLY ONE BAROR CARD MAY APPEAR IN THE BULK DATA DECK

*** ALL COORDINATE POINTS MUST REFERENCE BASE SYSTEM

*** ALL CHILDREN MUST FOLLOW PARENT CARDS

*** GRID POINT _____ IS UNDEFINED FOR COORDINATE SYSTEM _____

External Calls:
IFINDN OUTGRD RDNPAC
NATSTR RDNCARD RDNGR1
NATTMS RDNSHF ZRAYI

Argument List: None

Important Variables: BUFFER - input buffer for a bulk data card
GRIDAX - array of processed data types
NT - type number for card being processed

Common Blocks: BLANK MOHEAD CHAR PERM
NOHEAD SYSTEM DBREC NASTRAN
ELHEAD MATL TEMP TRACK1
Subroutine: RDNGRI

Algorithm: Routine reads in and processes the NASTRAN force and moment cards. Loads a data array with given values from the input card and packs the array to the geometry data base as it is filled. After the data is read in, it is retrieved, sorted, and saved on the data base.

Input/Output: IOPAC - performs packed geometry data base input/output
               IODB - performs direct I/O to the geometry data base

Error Messages: None

External Calls: IODB
               IOPAC
               SORTQ

Argument List: IST - key array for the scratch file (unit 2)
               JUNK - decoded array of forces/moments
               NLFM - force or moment array for storage
               NLM - number of forces/moments
               IREC - number of records on the scratch file
               IG - 1= save NLFM to the scratch file
                  2= save NLFM to the permanent data base
               MS - key to the forces/moments pointers

Important Variables: Same as argument list

Common Blocks: MATL TRACK1 NASTRN
               BLANK DBREC
               NOHEAD PERM
Subroutine: RDNMAT

Algorithm: Routine stores the material property data tables to the geometric data base. It takes the input array of values MAT1 and stores them. Next it places pointer information in the MAT2 array for use in resolving the element connectivities, sizes, and materials for the model.

Input/Output: None

Error Messages: None

External Calls: IOPAC

Argument List:
- MAT1 - input array of material values
- N1 - number of rows in MAT1
- MAT2 - output array of pointer data from material storage
- N2 - number of rows in MAT2
- NMAT - number of materials being transferred
- NN - index into the data base header for the material data record
- MM - index in MAT2 for the total number of materials stored
- NDBREC - data base record for data

Important Variables: Same as argument list

Common Blocks: DBREC
- ELHEAD
Subroutine: RDNOUT

Algorithm: Stores sorted information to the direct access scratch file for data processing. It is used by the element store routines to retrieve material, property, and connectivity data for resolution of the NASTRAN bulk data deck information.

Input/Output: ND2 - scratch direct access file
Unit 7 - terminal output for messages

Error Messages: *** SCRATCH FILE SPACE EXCEEDED

External Calls: None

Argument List: IREQ - the number of the NASTRAN data type being processed
NWORD - number of words in the output string
IOUT - output buffer array

Important Variables: NEXT - pointer array for the next record of data for a given type
IREC - record being read or written

Common Blocks: TRACK1
BLANK
READN
Subroutine: RDNPAC

Algorithm: Routine packs the node data to the data base. It sets up pointer arrays and transforms the coordinates using UTLTRN as needed.

Input/Output: IODB - routine for I/O to the data base

Error Messages: None

External Calls: IODB UTLTRN

Argument List: COORBA - scratch array for the transformed coordinates
COORIN - scratch array for the original coordinates

Important Variables: Same as argument list

Common Blocks: BLANK PERM DBREC NOHEAD MATL
Subroutine: RDNSHF

Algorithm: Shifts data on a NASTRAN bulk data card so that it is right justified within an eight character field.

Input/Output: None

Error Messages: None

External Calls: None

Argument List: II - number of values on the card
DATA - scratch array of 8 character words
LDATA - scratch array of 1 character words overlaid on DATA
BUFFER - input card of 8 character words
LBUFF - 1 character words overlaid on BUFFER

Important Variables: Same as argument list

Common Blocks: None
Subroutine: RDOPTS

Algorithm: Routine reads in the OPTSTAT data decks for storage on the data base. First it reads the control cards followed by the material properties. These are saved in isotropic or composite material arrays as needed. Next, the element material numbers, connectivity and similar information is read. The node coordinate and boundary condition data is then read in and passed to RDAOGS. Finally the applied load information is read and passed to RDAOLS.

Input/Output: NUNIT - input unit for the OPTSTAT data IOPAC - performs packed data I/O to the GEOM data base

Error Messages: None

External Calls: RDAOLS RDOPT2 RDOPT1 IOPAC RDAOGS ZRAYB RDCMIS

Argument List: None

Important Variables: MAT1 - isotropic material array MATC - composite material array XYZ - node coordinate data array IB - boundary condition data array MA, MB, MC, MD - element connectivity arrays NNODES - number of nodes per element

Common Blocks: BLANK MOHEAD OPTIND MATL NOHEAD PERM DBREC
**Subroutine:** IPTI

**Algorithm:** Routine computes the beta angle for composite material OPTSTAT elements. It cycles through the elements and if they are composite determines the element x direction from the connectivity. The material direction is then compared to this angle and the appropriate element beta angle is defined.

**input/Output:** None

**Error Messages:** None

**External Calls:** None

**Argument List:** XYZ - node coordinate array

**Important Variables:** NNODES - array with the number of nodes per element NA,MB,MC,MD - arrays with the node connectivity for the elements

**Common Blocks:** MATL OPTIHO
**Subroutine:** RDOPT2

**Algorithm:** Routine stores the element data to the data base. First it saves the material properties to the data base and then cycles through the elements placing their connectivity and property data into arrays based upon the element type. These arrays are then packed and output to the data base.

**Input/Output:**
- IODB - routine performs direct access I/O to the data base
- IOPAC - performs packed data I/O to the data base

**Error Messages:** None

**External Calls:**
- IODB
- ZRAYB
- IOPAC
- GROUPS

**Argument List:** None

**important Variables:**
- I2C, I3C, I4C, I5C - arrays for the element connectivity data
- I2P, I3P, I4P, I5P - arrays for the element size data

**Common Blocks:**
- NOHEAD
- DBREC
- MATL
- TEMP
- OPTIND
- ELHEAD
- MOHEAD
- BLANK
- PERM
The following block of subroutines makes up the SET module of CADS. These routines provide the SET processing functions for the definition of element and node sets for plotting. These routines are followed by several general routines for starting CADS, sorting arrays, and moving character blocks. The routines in this block are:

SETELM
SETETN
SETGEN
SETNOC
SETS
SETUP
SETUPI
SHIFT
SORTD
SORTQ
START
START1
Subroutine: SETELM

Algorithm: This routine generates element sets based upon user commands. It checks the command and transfers to the appropriate processing area. The first step is to get a list of elements based on the ID command. Next the keyword ALL is processed to get all of the elements in the model. The third section performs the union, intersection or exclusion of two previously defined sets. These sets are brought into core and operated on as required. Finally, the GROUP and TYPE keywords are processed by bringing element groups or types into core based on the requested GROUP number pointers and/or element TYPE pointers.

Input/Output: Unit 7 - terminal output for messages

Error Messages: *** MODEL TOO LARGE TO BE PLOTTED BY ALL

*** SET ______ DOES NOT PREVIOUSLY EXIST

*** INVALID OPERATION SPECIFIED FOR ELEMENT SET

*** ELEMENT TYPE ________ NOT VALID

*** SET ________ IS A NULL SET

External Calls: IFINDW, NUMBER, UTLMVW, UTLLTG
IOPAC, SETS, IOROUT

Argument List: NSET1 - element set returned from the routine
NSET2 - working array for making the set
NSET3 - working array for input list and set checks
Argument List:
NR - number of 2 noded elements times the number of words required for the definition of the elements.
NT - number of other elements times the number of words used to define them.

Important Variables: Same as argument list

Common Blocks:
BLANK    DBREC    PERM    SYSTEM    MATL
READ     ELHEAD   TEMP    TYPE
CHAR     NOHEAD   NASTRN  TYPEN
Subroutine: SETETN

Algorithm: Subroutine obtains the node numbers associated with a previously defined element set. It retrieves the given element set, uses IFINDN to locate the nodes defined for the element connectivities, and places them into the output set.

Input/Output: IOPAC - performs packed data I/O to the data base

Error Messages: None

External Calls: IFINDN

Argument List: NSET1 - output set of node numbers
N1 - the number of values in NSET1
NSET2 - input set of elements
N2 - the number of values in NSET2
ICODE - switch for NSET1 data: 0 gets the node numbers; 1 gets the pointers to the nodes

Important Variables: Same as argument list

Common Blocks: DBREC BLANK TYPE NASTRN
ELHEAD PERM MATL SYSTEM
Subroutine: SETGEN

Algorithm: Routine controls the generation of sets. First it decodes the input command, sets the appropriate switches and then calls in the routines to process the command. Finally it sets up for the DISPLAY module and passes control to either the executive or display routines based upon the user input.

Input/Output: Unit 6 - terminal output for the list and print commands
Unit 7 - terminal output for messages

Error Messages: *** SET NAME ______ IS NOT A PROPER NAME
*** SET ______ DOES NOT EXIST
*** NO RECORDS STORED TO DATA BASE
*** INCORRECT NUMBER OF ARGUMENTS USED FOR PLOT
*** SET ______ DOES NOT EXIST FOR PLOTTING

External Calls: IFINDN SETELM SETETN LIGRNO IOROUT RDCARD SETNOD

Argument List: IRET - not used
NSET1 - array of requested set values
NSET2 - first temporary scratch array for set processing
NSET3 - second temporary array for set processing
NR - number of 2 noded element data values
NT - number of other element data values
NO - number of nodes returned from SETNOD
ISW - switch to check if an automatic return to the DISPLAY module occurs

Important Variables: Same as argument list.
<table>
<thead>
<tr>
<th>Common Blocks:</th>
<th>BLANK</th>
<th>PERM</th>
<th>TRACK1</th>
<th>PINFLA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CHAR</td>
<td>TEMP</td>
<td>PLOTCH</td>
<td>MATL</td>
</tr>
<tr>
<td></td>
<td>READ</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Subroutine: SETNOD

Algorithm: Routine generates node sets. It processes each valid command separately. First it obtains all the nodes if the ALL command was used. Next it processes the nodes based on a list of node numbers. The routines to do the CYLINDER, SPHERE, SLAB, and BOX commands are then called. Finally, the SETS routine is called to process the union, intersection or exclusion commands and the definition of nodes by suppression type is performed.

Input/Output: Unit 7 - terminal output for messages

Error Messages:

*** COORDINATE AXIS ________ NOT FOUND FOR BOX

*** SET ________ DOES NOT PREVIOUSLY EXIST

*** INVALID OPERATION SPECIFIED FOR NODE SET

*** INVALID OPERATION FOR NODE SETS

*** FREEDOM INDICATOR ________ IS NOT VALID

*** SET ________ IS A NULL SET

External Calls: BOX FREUCK NUMBER SLAB UTLMVW
CHANGE IFINDN SETETN SPHERE
CYLNDR IOROUT SETS UTLLTG

Argument List: NSET1 - array containing the given node set
NSET2 - first scratch array for set processing
NSET3 - second scratch array for set processing
NO - size of the NSET1 array returned
Important Variables: Same as argument list

Common Blocks: PERM  BLANK  READ
           CHAR  TEMP
Subroutine: SETS

Algorithm: Routine performs the set algebra functions. An input switch is used to transfer to the union, exclusion, or intersection processing area as requested by the user. In all cases sets being processed are brought into core and cycled saving the output as a new set.

Input/Output: Unit 7 - terminal output for messages

Error Messages: *** UNION OF SET _______ WITH SET _______ GREATER THAN ALLOWABLE CAPACITY

External Calls: None

Argument List: ICODE - command type: union, intersection, exclusion, complement
NAME1 - character name of the first set
K1 - array containing the first set
N1 - number of values in K1
NAME2 - name of the second input set
K2 - array for the second set
N2 - number of values in the second set
NAME3 - not used
K3 - output set of values
N3 - number of values in array K3
KPR - error return code

Important Variables: Same as argument list

Common Blocks: None
Subroutine:  SETUP

Algorithm:  This routine processes element sets before they are sent to the DISPLAY module. It determines the element types and loads their connectivities into solid and dashed line tables. The solid element types are processed separately due to the three-dimensional nature of those elements.

Input/Output:  Unit 7 - terminal output for messages

Error Messages:  *** THE FOLLOWING NODES HAVE NOT BEEN INPUT A LOCATION

External Calls:  IFINDN  IOROUT  SETUP1  IOPAC

Argument List:  IERR - error return switch
                NSET1 - input element set to be processed
                IDDAS - dashed line table
                IDSOL - solid line table
                IDDBL - boundary line table for contour plots

Important Variables:  Same as argument list

Common Blocks:  BLANK  NASTRN  MATL  PINFLA
                 ELHEAD  PERM  TYPE  SOLIDS
                 DBREC  TEMP  TRACK1
Subroutine: SETUP

Algorithm: Routine is called by SETUP to complete the line definition processing for plots. It compresses the solid and dashed line tables by removing duplicate line segments. Finally the boundary line table is re-ordered to optimize its drawing.

Input/Output: None

Error Messages: None

External Calls: IOROUT
SORTD

Argument List: IDOL - solid line table
IDAS - dashed line table
IDBL - boundary line table
NAMES - character name array for the tables

Important Variables: Same as argument list

Common Blocks: PERM
TEMP
Subroutine: SHIFT

Algorithm: Routine shifts NVAR command character strings of NBYTE each so that they are left justified. It strips leading blanks and then packs the remainder of the string.

Input/Output: None

Error Messages: None

External Calls: None

Argument List: LCOMD - CHARACTER*1 array of commands to be shifted
NVAR - number of strings to be shifted
NBYTE - number of characters in each command

Important Variables: Same as argument list

Common Blocks: None
Subroutine: SORTD

Algorithm: Routine performs an in-core sort of a two dimensional array. It starts by reformatting the array so that it can be sorted on the first column only. The routine then cycles through the array placing values in ascending order.

Input/Output: None

Error Messages: None

External Calls: None

Argument List: JD - 2-dimensional array to be sorted
N - number of values in array JD

Important Variables: Same as argument list

Common Blocks: None
Subroutine: SORTQ

Algorithm: Routine is used to perform an in-core sort of various sized arrays. It sorts an array into ascending order by working through each entry and placing it in position. It then moves the rest of the entry's columns into position.

Input/Output: None

Error Messages: None

External Calls: None

Argument List: ID - array to be sorted
N  - number of rows in ID
NC - number of columns in ID
IER - error switch, not used
NS - column of ID on which to sort
NL - column length, not used
FF - scratch array of length N, not used
A  - not used

Important Variables: Same as argument list

Common Blocks: None
Subroutine: START

Algorithm: Routine is used to initialize the main program variables and internal counters. It also performs the initial user queries.

Input/Output: Unit 7 - terminal output for message
RDCARD - free read input routine
IOHEAD - gets the data base header record

Error Messages:
*** INCORRECT TERMINAL TYPE, RE-ENTER
*** PROGRAM TYPE _____ NOT SUPPORTED
*** ERROR *** FILE: _____ DOES NOT EXIST; REENTER
*** ERROR *** OPEN ERROR ON UNIT: _____ STATUS: ____
*** ERROR *** FILE: _____ ALREADY EXISTS: ENTER NEW NAME

External Calls:
RDCARD   IOHEAD
ZRAYI   START1

Argument List: None

Important Variables: NWPB - number of words per data base block
NUMBEL - number of element types

Common Blocks:
READ   TYPEN   ELHEAD   TRACK1   HEADPP   PLOTCM
TYPE   PLOT   NOHEAD   MOHEAD   PERM
DIBAUD   PLOTEL   BLANK   DBREC   SYSTEM
Subroutine: START1

Algorithm: Routine is called by START to set the element label names. They are used to call out the elements and are loaded into the TYPES array based upon the communication mode.

Input/Output: None

Error Messages: None

External Calls: None

Argument List: None

Important Variables: TYPES - array of element names being loaded
MODE - communications mode
NUMBEL - number of element types
ELMS - array containing all the different element names

Common Blocks: SYSTEM
TYPE
The following block of subroutines are utility routines used throughout the code. The routines in this block are:

UTLBAS
UTLCRS
UTLLTG
UTLMVW
UTLSLS
UTLTRN
Subroutine: UTLBAS

Algorithm: Utility routine used to determine vectors for establishing basis directions. It converts the given user nodes or coordinates into a vector.

Input/Output: Unit 7 - terminal output for messages

Error Messages:

*** KEYWORD ______ NOT VALID FOR BASIS COMMAND

*** NODE ______ DOES NOT EXIST FOR BASIS COMMAND

*** INSUFFICIENT TERMS SUPPLIED TO BASIS TO EXECUTE THE VECTOR OPTION

External Calls: CHANGE
IFINDN
NUMBER

Argument List:

NC - number of coordinate directions for the vector

* - alternate return

Important Variables: Same as argument list

Common Blocks: READ   PERM
SYSTEM   BLANK
Subroutine: UTLCRS

Algorithm: Utility routine used to perform cross product operations for three element vectors, A, B, and C, where $A = B \times C$.

Input/Output: None

Error Messages: None

External Calls: None

Argument List: A - cross product resultant vector
B - first input vector
C - second input vector

Important Variables: Same as argument list

Common Blocks: None
Subroutine: UTLLTG

Algorithm: Utility list generator routine. It decodes the standard TO/BY lists of numbers used throughout the program. It begins by setting counters, checking for the TO and BY keywords before decoding the actual numbers supplied for the list. Once the list is generated the ICODE switch is used to check if each generated list number is a previously defined node number.

Input/Output: Unit 7 - terminal output for messages

Error Messages: *** BAD DATA LIST FOR SET _____ OR NUMBER GENERATION

External Calls: IFINDN NUMBER

Argument List: HOLD - array of 8 character elements with the list for decoding
NVAR - number of elements in HOLD
K3 - integer array with the decoded list
N3 - number of elements in K3
NAME1 - set name for error messages
KPR - error switch
ICODE - switch for a final node existence check against the list

Important Variables: Same as argument list

Common Blocks: PERM
Subroutine: UTLMVW

Algorithm: Utility subroutine used to move words from one matrix to another.

Input/Output: None

Error Messages: None

External Calls: None

Argument List: K2 - number of words to be transferred
                MATRIX - input array
                IBUFF - output array

Important Variables: Same as argument list

Common Blocks: None
Subroutine: UTLSLS

Algorithm: A utility routine which is used to set an array of one character elements as logical switches. It sets an element defined by the ISWS array to the character 1. The entry point UTLDBP unpacks the ELSWS character array back into the ISWS array.

Input/Output: None

Error Messages: None

External Calls: None

Argument List: ELSWS - CHARACTER*1 array of switches
N - number of switches to be set
ISWS - packed input array of switches

UTLDBP Entry: ELSWS, N, ISWS - same as UTLSLS

Important Variables: Same as argument list

Common Blocks: None
Subroutine: UTLTRN

Algorithm: Utility routine used to perform coordinate transformations from an input system to the base system. It uses the input coordinate system number to search for a previously defined transformation table to define the required transformation matrix. This matrix then operates on the input values to transform them to the base system.

Input/Output: Unit 7 - terminal output for messages

Error Messages: *** COORDINATE SYSTEM ________ NOT FOUND

External Calls: None

Argument List: ISYS - input coordinate system number
               C - values to be transformed

Important Variables: Same as argument list

Common Blocks: PERM
               MATL
The following block of subroutines are used to perform the X-Y graph functions of the CADS DISPLAY module. They retrieve data, establish the scales and grids, and perform the required output to the terminal. The routines in this block are:

XYDISP
XYERR
XYGRAF
XYGRID
XYHLBL
XYLGND
XYLINE
XYLSYM
XYMXN
XYSCL
XYSYM
XYTERM
XYTICL
XYTIME
XYTLBL
XYVLBL
ZRAYB
Subroutine: XYDISP

Algorithm: Routine retrieves the requested displacement or eigenvector data based on the component number, case, and node list for a particular x-y plot. Retrieval data is stored in the VALUES array and passed back to XYGRAF for the actual display.

Input/Output: Unit 7 - terminal output for messages
IODB - post data base reads
IOPAC - packed data I/O to the data base

Error Messages: *** NO LOAD CASES STORED FOR _____ DATA **
*** ERROR *** LOAD CASE _____ DOES NOT EXIST FOR _____ DATA

External Calls: IODB
IOPAC

Argument List: NL - number of nodes in NLIST
NLIST - list of nodes to be retrieved
ICRV - curve number
ISW - type of values, 1 = x; 2 = y
IWant - component number of data to be retrieved
LCASE - case number
NBTYPE - displacement or eigenvector type of data
VALUES - array to hold retrieval data values

Important Variables: Same as argument list

Common Blocks: MATL HEADPP
D1TOKD DBREC
PLOTBD
Subroutine: XYERR

Algorithm: Routine prints error messages prior to x-y graph outputs. It checks the error number and prints out the appropriate message to unit 7. This helps to flag input or setup errors.

Input/Output: Unit 7 - terminal output for messages

Error Messages: *** ERROR: DIFF. BETWEEN MIN. AND MAX. < 1.0E-13 ***

External Calls: None

Argument List: IERR - error number switch

Important Variables: Same as argument list

Common Blocks: None
Subroutine: XYGRAF

Algorithm: Routine plots an x-y graph with up to five different curves. First it prompts for user inputs to define the number of curves, data points, and labels. Next it determines minimum and maximum values and sets up the terminal for output. Finally it cycles through special purpose routines to output the titles and curves.

Input/Output: Unit 5 (INDEV) - terminal input unit for user commands
Unit 6 (OUTDEV) - terminal output unit for program prompts
Unit 7 - terminal output for messages
IODEB - performs I/O directly to the data base
IOPAC - performs packed data I/O to the data base

Error Messages:

*** ERROR *** OPTION _____ NOT VALID: REENTER

*** ERROR *** NUMBER OF CURVES _____ IS GREATER THAN 5

*** WARNING * NUMBER OF CASES ____ WAS MORE THAN 60

*** WARNING * NUMBER OF CASES ____ DOES NOT EQUAL NUMBER OF CURVES ____

*** ERROR *** NUMBER OF VALUES DEFINED ____ DOES NOT EQUAL NUMBER OF CURVES ____

*** ERROR *** KEYWORD _____ NOT VALID: REENTER LINE

*** ERROR *** NODE SET ____ IS EMPTY: REENTER LINE

*** ERROR *** EITHER TIME STEPS HAVE NOT BEEN SET OR THERE IS NO TIME HISTORY DATA ON THE POST DATA BASE
External Calls: JBEAM JVPORT JWINDOW XYLGRID XYSCL JLSTYLE XYHLBL
XYSYM JCLOSE JDRAW JMOVE XYGRID XMXN XYTLBL
RDCARD NUMBER UTLTG XTERMIN PLTBEG JOPEN XYVBL
PLTDOP JKEYBD JFRAME IOROUT CHANGE IOPAC IFINDN
I0DB XYTIME XYDISP XYLE

Argument List: None

Important Variables: CDATA - array with curve values
NUMCRV - number of curves
NUMPTS - number of points per curve
CRVNUM - curve number being output

Common Blocks: DITOKD PERM
BLANK DBREC
READ HEADPP
Subroutine: XYGRID

Algorithm: Routine plots a grid line for each of the x and y axis tic marks. First it plots the border line and then cycle through the tic mark arrays and draws the grid lines.

Input/Output: None

Error Messages: None

External Calls: JDRAW
               JLSTYL
               JMOVE

Argument List: XTICS - array of x tic values
               YTICS - array of y tic values
               NXTICS - number of x tics
               NYTICS - number of y tics
               XMIN - the minimum of the x values
               XMAX - the maximum of the x values
               YMIN - the minimum of the y values
               YMAX - the maximum of the y values

Important Variables: Same as argument list

Common Blocks: None
Subroutine: XYHLBL

Algorithm: Routine outputs a horizontal label text string. Moves to the desired position and outputs an input string based upon the number of characters to be output.

Input/Output: None

Error Messages: None

External Calls: JMOVE
               JISTRG

Argument List: NHCHAR - number of characters in the label
               ILABEL - array with the label characters (72 characters)
               Ihlabl - array with up to 48 characters
               IXPOS - x start position on the screen
               IYPOS - y start position on the screen

Important Variables: Same as argument list

Common Blocks: None
Subroutine: XYLGNDC

Algorithm: Routine outputs the legend block for the x-y plot. It draws the border, plots the heading labels and finally outputs the individual curve legends.

Input/Output: None

Error Messages: None

External Calls: JDRAW XLYSYM
               JMOVE
               J1STRG

Argument List: NUMCRV - number of curves to be plotted

Important Variables: Same as argument list

Common Blocks: None
Subroutine: XYLINE

Algorithm: This routine sets the dashed line style for each of the curves to be plotted. It cycles through the curves, setting the line type.

Input/Output: None

Error Messages: None

External Calls: None

Argument List: CRVNUM - curve number
LINTYP - line type for the curve

Important Variables: Same as argument list

Common Blocks: None
Subroutine: XYLSYM

Algorithm: This routine places the curve symbol at a desired location. It uses the curve number to determine the current symbol. It is used for the symbols in the legend box.

Input/Output: None

Error Messages: None

External Calls: JCMARK
                JMARK

Argument List: CRVNUM - curve number
                IX    - x screen position for the mark
                IY    - y screen position for the mark

Important Variables: Same as argument list

Common Blocks: None
Subroutine: XYMNX

Algorithm: Routine determines the minimum and maximum values in a one dimensional array. It uses a straight search through the array values to determine these values.

Input/Output: None

Error Messages: None

External Calls: None

Argument List:
XMIN - minimum array value
XMAX - maximum array value
ARRAY - array of values
NUMB - number of values in ARRAY

Important Variables: Same as argument list

Common Blocks: None
Subroutine:  XYSCL

Algorithm:  Routine scales the curve values so that even increments based upon 1.0, 2.0, 2.5, or 5.0 to powers of 10 are obtained. First it scales the x-values based upon the minimum and maximum x's using common logs and powers of 10 to divide the x-distance. The final scale value is used to determine the corresponding tic marks. This process is then repeated for the y-axis values.

Input/Output:  Unit 7 - terminal output for messages

Error Messages:  *** ERROR: DIFF. BETWEEN MIN. AND MAX.
<1.0E-13***

External Calls:  XYERR

Argument List:  XMIN - minimum x axis value
XMAX - maximum x axis value
YMIN - minimum y axis value
YMAX - maximum y axis value
XTICS - x tic mark value array
YTICS - y tic mark value array
FLAG - error flag switch
NXTICS - number of x tic marks
NYTICS - number of y tic marks

Important Variables:  Same as argument list

Common Blocks:  None
**Subroutine:** XSYM

**Algorithm:** Routine outputs a curve symbol along the curve as it is plotted. It sets the correct mark and outputs the symbol using the curve number to get the current mark.

**Input/Output:** None

**Error Messages:** None

**External Calls:** JCMARK

**Argument List:**
- CRVNUM - curve number
- X - x screen position of the mark
- Y - y screen position of the mark

**Important Variables:** Same as argument list

**Common Blocks:** None
Subroutine: XYTERM

Algorithm: This routine performs x-y graphing for user supplied sets of x and y values. It prompts the user for the curve titles and the sets of x and y values to be graphed. It then determines the correct tic marks and grid lines before finally cycling through the values to plot the required curves.

Input/Output: RDCARD - free read terminal input
Unit 5 - terminal input
Unit 6 - terminal output
Unit 7 - terminal output for messages

Error Messages: *** ERROR *** NUMBER OF X VALUES ______ MUST EQUAL NUMBER OF Y ______

External Calls: CHANGE JDRAW JLSTYL JVPORT PLTDOP XYHLBL
JBEAM JFRAME JMOVE JWINDO RDCARD XYLGN
JCLOSE JKEYBD JOPEN PLTBEG XYGRID XYLINE
XYMXN XYSCL XSYM XTYLBL XYVLBL

Argument List: None

Important Variables: NUMCRV - number of curves to be processed
CRVNUM - number of the particular curve being processed
CDATA - array with the x,y values to be graphed
JDATA - pointer array describing the size of each curve

Common Blocks: None
Subroutine: XYTICL

Algorithm: Outputs the tic mark labels along the x and y axes. Checks for the axis direction, converts the value at the mark into characters and then outputs the characters.

Input/Output: None

Error Messages: None

External Calls: JMOVE
               J1STRG

Argument List: X - x screen position of the mark
                Y - y screen position of the mark
                VALUE - tic mark value

Important Variables: Same as argument list

Common Blocks: None
Subroutine: XYTIME

Algorithm: This routine retrieves the node displacements or eigenvectors by time step number for x-y graphs. First it finds the correct data table headers and then retrieves the pointer records for the input node list. Finally the routine cycles through the data values placing the appropriate values into the output array.

Input/Output: IOPAC - packed data base I/O
Unit 7 - terminal output for messages

Error Messages: *** NO LOAD CASES STORED FOR ____ DATA ***
*** ERROR *** LOAD CASE ____ DOES NOT EXIST FOR ____ DATA

External Calls: IODB
IOPAC

Argument List: NLIST - node number to define data to be retrieved
ICRV - curve number being processed
ISW - switch for values: 1, x values; 2, y values
IWANT - the data component to be retrieved (i.e., TX, TY, etc)
NCASE - the load case number
NCASES - list of the load case numbers
NBTYPE - switch for displacement or eigenvector data
VALUES - output array with values to be graphed

Important Variables: Same as argument list.

Common Blocks: MATL HEADPP
D1TOKD OBREC
PLOTBD
Subroutine: XYTLBL

Algorithm: Routine determines the tic mark line positions and converts them to output coordinate locations.

Input/Output: None

Error Messages: None

External Calls: XYTICL

Argument List: XTICS - x tic mark value array
NXTICS - number of x tic marks
YTICS - y tic mark value array
NYTICS - number of y tic marks

Important Variables: Same as argument list

Common Blocks: None
Subroutine: XYVLBL

Algorithm: Routine outputs the vertical axis label. It moves to the start position and outputs the title down the screen.

Input/Output: None

Error Messages: None

External Calls: JMOVE J1STRG

Argument List: IVLBL - array of characters in the label

Important Variables: Same as argument list

Common Blocks: None
Subroutine: ZRAYB

Algorithm: This routine is used to zero out input matrices. The entry point ZRAYI zeroes out integer matrices.

Input/Output: None

Error Messages: None

External Calls: None

Argument List: AA - real matrix to be zeroed
N - number of values to be reset

ZRAYI ENTRY
NA - integer matrix to be zeroed
N - same as ZRAYB

Important Variables: Same as argument list

Common Blocks: None
The following block of subroutines makes up the CADSPP software used to read analysis output to the POST data base for use by the CADS software. The routines in this block are:

POST
TYPEVA, HEADTI
IODB
IOPAC
RDANAL
RDOPTS
RDPCHT
RDPCH1
RDPCH2
RDPCH3
RDSTDE
RSTD1
RSTD2
RDSTFS
RSTF1
RSTF2
RSTF3
RSTF4
SHPPVA
STARTP
VARRAY
WRMAST
Main: POST

Algorithm: This is the main program module for the CADSPP POST data base loading program. It opens the message file on unit 7 and then calls in the operational routines.

Input/Output: Unit - 7 for terminal output messages

Error Messages: None

External Calls: RDPCHT  RDPCH1  RDOPTS
               STARTP  RDANAL

Argument List: None

Important Variables: NPROG - type of analysis data to be loaded

Common Blocks: READ  DBREC  MASTER
               TYPEVA  HEADER  TITLES
               HEADTI  BLANK  NODEL
Algorithm: This block data module initializes the TYPEVA and HEADTI common blocks. The NCFS array is initialized with the NASTRAN element type, CADS element type, number of forces, and number of stresses per element type. The TYPE array is initialized with the key command type words for NASTRAN analysis decks and the OPTION array contains the names of the valid analysis types supported by CADSPP.

Input/Output: None

Error Messages: None

External Calls: None

Argument List: None

Important Variables: NCFS - 4 X 25 array for storing counters
TYPE - type values for decoding the input cards
OPTION - array containing the supported data types

Common Blocks: TYPEVA
HEADTI
Subroutine: IODB

Algorithm: This subroutine performs the direct access I/O to the POST data base. The input switch IG defines the operation to be performed. The buffer array A and record number IREC are then used to retrieve or store the given information.

Input/Output: NUNIT - unit used for direct access I/O
Unit 7 - terminal output for messages

Error Messages: *** ____ REQUEST ERROR FROM SUB. ____ ON UNIT ____

External Calls: None

Argument List: IG - read/write switch for the subroutine
A - array of data for data base I/O
N - number of elements in A
IREC - direct access record number for I/O
NUNIT - direct access unit number
SUBNAM - name of the calling subroutine

Important Variables: Same as argument list

Common Blocks: None
Subroutine: IOPAC

Algorithm: This routine packs an array of information for sending to the data base. It blocks the data to the record size of the data base and calls IODB to perform the actual I/O function.

Input/Output: IODB - actual data base I/O

Error Messages: None

External Calls: IODB

Argument List: ARRAY - array with data for I/O
NWRD - number of words to be stored or retrieved
IG - switch to read or write data: 1 read; 2 write
NDBUNT - data base unit number
SUBNAM - name of the calling routine; passed to IODB

Important Variables: Same as argument list.

Common Blocks: DBREC
Subroutine: RDANAL

Algorithm: This routine reads and stores the analysis results from the ANALYZE program. It decodes the command cards; reads in all of the displacement values, and then outputs those values to the POST data base. The stress data is then read and stored. The results are stored to the POST data base so that they look similar to NASTRAN results.

Input/Output: Unit 7 - terminal output for messages
IODB - data base I/O
IOPAC - packed data base I/O
NASTPU - input unit for analysis data

Error Messages: ** ERROR ** E-O-F ON UNIT _____ SEARCHING _____ DATA

External Calls: IODB VARRAY
IOPAC WRMAST

Argument List: None

Important Variables: NCASES - number of load cases in analysis results
MASTER - master header record array
NELEM - number of elements in the model
DISP - data array of displacements
STRES - data array of stresses

Common Blocks: READ BLANK DBREC
TYPEVA TITLES MASTER
HEADTI HEADER
Subroutine: RDOPTS

Algorithm: This routine reads and stores the analysis results from the OPTSTAT program. It is very similar to the RDANAL routine with some minor differences for the OPTSTAT result formats. The displacement data is stored followed by the stress results.

Input/Output: Unit 7 - terminal output for messages
IODB - data base I/O
IOPAC - packed data base I/O
NASTPU - input unit for analysis data

Error Messages: ** ERROR ** E-O-F ON UNIT ______ SEARCHING ______ DATA

External Calls: IODB VARRAY
IOPAC WRMAST

Argument List: None

Important Variables: NCASES - number of load cases in analysis results
MASTER - master header record array
NELEM - number of elements in the model
DISP - data array of displacements
STRES - data array of stresses

Common Blocks: READ BLANK DBREC
TYPEVA TITLES MASTER
HEADTI HEADER
Subroutine: RDPCHT

Algorithm: This routine reads the title and header cards from the NASTRAN output punch file for decoding and processing by other routines. Once a valid data type is found a decoding routine is called and the appropriate node or element data is stored.

Input/Output: NASTPU - unit with NASTRAN analysis results data
Unit 7 - terminal output for messages

Error Messages: ELEMENT TYPE __________ IS NOT SUPPORTED BY THE PROGRAM

External Calls: IOPAC WRMAST
RDSTDE RDSTFS

Argument List: None

Important Variables: NHEAD - header array for POST data base
NOCOND - number of load cases
NELTYP - element type number being processed

Common Blocks: READ BLANK DBREC
TYPEVA TITLES MASTER
HEADTI HEADER
Subroutine: RDPCH1

Algorithm: This routine calls in the element or node NASTRAN output read routine for dynamic data. It switches between processing element or node data and then it calls in the appropriate decoding routine. Finally, the routine saves the time increments and header records to the POST data base.

Input/Output: IOPAC - packed data base I/O
NPPUNT - POST data base write of new record

Error Messages: None

External Calls: IOPAC    RDSTD1    RDSTF1
              RDPCH2    RDSTD2    RDSTF2

Argument List: None

Important Variables: None

Common Blocks: NODEL    MASTER
               HEADER
               DBREC
Subroutine: RDPCH2

Algorithm: This routine provides dynamic NASTRAN analysis result processing for CADSPP. It decodes the NASTRAN cards for valid card types and sets up the pointers for storing the result values based upon those cards and the user supplied requests.

Input/Output: INPUNT - input unit for NASTRAN analysis data
Unit 7 - terminal output for messages
NPPUNT - POST data base write

Error Messages: ELEMENT TYPE _____ IS NOT SUPPORTED BY THE PROGRAM

External Calls: RDPCH3

Argument List: None

Important Variables: TYPE - array of valid analysis output card types
IREQ - array with user requested data types
NCFS - array with the valid element types and their pointers

Common Blocks: READ TITLES DBREC
TYPEVA NODEL
HEADTI HEADER
Subroutine: RDPCH3

Algorithm: This routine reads in the actual NASTRAN analysis data values for a particular node or element. It determines the number of data cards to be read and then places the data values into the TIME output array.

Input/Output: INPUNT - card input of NASTRAN results
Unit 7 - terminal output for messages

Error Messages: ** ERROR ELEMENT OR POINT ID IS NOT ON INPUT FILE **

External Calls: None

Argument List: NV - number of data values to be read

Important Variables: Same as argument list

Common Blocks: READ DBREC HEADTI NODEL
Subroutine: RDSTDE

Algorithm: This routine reads and stores the node displacement and eigenvector data from NASTRAN. First it checks that the required subcase is found and then it checks for the correct data type. Next it reads the values card by card and compresses them into an output record for storage in the POST data base. Finally the routine updates the header record.

Input/Output: IODB - routine performs actual direct access I/O
NASTPU - NASTRAN output information unit
Unit 7 - terminal output for messages

Error Messages: *** ERROR SUBCASE FOR DISP OR EIGEN IS NOT ON TAPE ***
*** ERROR EIGENVALUE IS NOT ON INPUT TAPE ***
*** ERROR INCOMPLETE INPUT FOR NODE = _____ COND
   NO = _____
*** ERROR END OF FILE ON INPUT WHEN ATTEMPTING TO READ
   DISPLACEMENTS OR EIGENVECTORS ***

External Calls: IODB
VARRAY
WRMAST

Argument List: IDSP - array holds the integer values for the node outputs
DSP - array holds the real values for the node outputs
IG - switch for the data type being processed
IEND - end of file indicator

Important Variables: Same as argument list
<table>
<thead>
<tr>
<th>Common Blocks</th>
<th>READ</th>
<th>DBREC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>HEADTI</td>
<td>HEADER</td>
</tr>
<tr>
<td></td>
<td>TITLES</td>
<td>MASTER</td>
</tr>
</tbody>
</table>
Subroutine: RDSTD1

Algorithm: This routine reads a set of dynamic displacement values for one node and decodes the values from characters into the appropriate program variables.

Input/Output: INPUNT - input unit with NASTRAN results
Unit 7 - terminal output for messages
INPT - scratch output unit

Error Messages: *** ERROR POINT ID IS NOT ON INPUT TAPE ***
** ERROR INCOMPLETE INPUT FOR NODE =
*** ERROR E-O-F ON UNIT ___ WHEN READING DISPLACEMENTS ***

External Calls: None

Argument List: IG - data type to be processed
INPT - scratch file

Important Variables: Same as argument list

Common Blocks: READ
HEADTI
DBREC
Subroutine: RDSTD2

Algorithm: This routine reads the dynamic displacements or eigenvectors from the scratch file and stores them to the POST data base. It reads and packs the data values into records, updates the pointer records based on the time step, and finally stores the data values to the POST data base.

Input/Output: IODB - direct access data base I/O
IT1 - input scratch unit
IT2 - output scratch unit

Error Messages: None

External Calls: IODB
WRMAST

Argument List: IG - type of data to be stored
IT1 - input scratch file
IT2 - output scratch file

Important Variables: Same as argument list

Common Blocks: NODEL HEADER
BLANK MASTER
DBREC
Subroutine: RDSTFS

Algorithm: This routine processes the element stress and force data. It reads in the data values and stores them in a buffer array before calling IOPAC for output to the POST data base. Finally it updates the header record and returns.

Input/Output: IOPAC - packs data array for output to direct access file
NASTPU - unit with NASTRAN output file
Unit 7 - terminal output for messages

Error Messages: *** ERROR INCOMPLETE INPUT FOR TYPE = _____ COND NO = _____ EL. ID = _____

External Calls: IOPAC

Argument List: IDSP - buffer array for integer values
DSP - buffer array for real values
N - number of values in DSP or IDSP
NV - number of values per element and data type
NCTYPE - CADS program element type number
IG - switch for the type of data being processed
ILND - end of file switch

Important Variables: Same as argument list

Common Blocks: READ TITLES MASTER
TYPEVA DBREC
HEADTI HEADER
Subroutine: RDSTF1

Algorithm: This routine decodes the dynamic stress or force data. First it checks the NASTRAN analysis cards for validity and then determines the number of data components for the particular analysis data type and element type being processed. Finally it reads the actual data values and writes them to the scratch file.

Input/Output: INPUNT - NASTRAN analysis card input
INPT - scratch file unit
Unit 7 - terminal output for messages

Error Messages:
** ERROR ELEMENT TYPE NOT ON INPUT FILE **
** ERROR ELEMENT ID IS NOT ON INPUT FILE **
** ERROR INCOMPLETE INPUT FOR TYPE = ___ ID = ____ **

External Calls: None

Argument List: IG - type of data block being processed, 1=Forces,
2=Stresses
INPT - scratch file

Important Variables: Same as argument list

Common Blocks: READ DBREC
TYPEVA
HEADTI
Subroutine: RDSTF2

Algorithm: This routine reads in the data values for dynamic stresses or forces from a scratch unit and formats them for output to the POST data base. It cycles through the data values, sets up the counters for them, and finally updates the pointer records for the element data.

Input/Output: IT1 - input scratch unit
IT2 - output scratch unit

Error Messages: None

External Calls: RDSTF3
RDSTF4
WRMAST

Argument List: IG - type of data being processed
IT1 - input scratch array
IT2 - output scratch array

Important Variables: Same as argument list

Common Blocks: NODEL HEADER
BLANK MASTER
DBREC
Subroutine: RDSTF3

Algorithm: This routine packs a column of a two dimensional array with a one dimensional vector of values.

Input/Output: None

Error Messages: None

External Calls: None

Argument List:
- IDSP - two dimensional output array
- NVP1 - number of rows in IDSP
- IVAL - one dimensional vector to be placed in IDSP
- ITOT - column of IDSP to be filled by IVAL

Important Variables: Same as argument list

Common Blocks: None
Subroutine: RDSTF4

Algorithm: This routine stores the dynamic stress or force information directly to the POST data base. It updates the master pointer record and then calls IOPAC to store the data value array to the data base.

Input/Output: IOPAC - packed data base I/O

Error Messages: None

External Calls: IOPAC

Argument List: IDSP - stress or force data value array
NVP1 - number of components per element type
ITOT - NASTRAN type number for the element
NCTYPE - CADS type number for the element
M - record number
IG - type of input data: 1=forces; 2=stresses

Important Variables: Same as argument list

Common Blocks: DBREC
MASTER
Subroutine: SHPPVA

Algorithm: This routine packs a user input card for processing in the STARTP routine. Basically it removes blanks from between variables and places the compressed values back into the HOLD array.

Input/Output: None

Error Messages: None

External Calls: None

Argument List: HOLD - buffer of 8 character input variables
LHOLD - overlaid on HOLD but stored as 8, 1 character values
NVAR - number of variables in HOLD

Important Variables: Same as argument list

Common Blocks: None
Subroutine: STARTP

Algorithm: This routine is the initialization and control routine used to process the analysis outputs for storage on the POST data base. It initializes switches and counters; prompts for input commands; and returns to the main control routine for later processing of the actual values.

Input/Output: Unit 7 - terminal output for messages
Unit 5 - input unit for user

Error Messages:
*** ERROR *** FILE ____ DOES NOT EXIST: REENTER

*** ERROR *** OPTION ____ IS NOT VALID; REENTER

*** ERROR *** FILE ____ AND EXISTENCE STATUS DO NOT MATCH; REENTER

*** ERROR *** _____ TYPE DATA ALREADY ON POST DATA BASE DOES NOT MATCH REQUESTED _____ DATA TYPE

** NO SPACE IN THE HEADER RECORD COND. LIMIT IS 61 **

*** ERROR INDICATE INPUT TYPE STATIC OR DYNAMIC ***

*** ERROR *** END OF DATA STATUS= _____

*** ERROR *** OPEN ERROR ____ ON FILE: _____

External Calls: IOPAC
SHPPVA
VARRAY

Argument List: IPASS - switch for multiple input data sets
Important Variables: None

Common Blocks: HEADTI HEADER DBREC
TITLES MASTER
Subroutine:  VARRAY

Algorithm:  This routine is used to initialize an array to a given value.

Input/Output:  None

Error Messages:  None

External Calls:  None

Argument List:  AA - array to be initialized
N - number of array elements to be initialized
V - value to be used for initialization

Important Variables:  Same as argument list

Common Blocks:  None
**Subroutine:** WRMAST

**Algorithm:** This routine is used to store the master header record to the POST data base. It stores the master record pointers into the header record and calls IODB to write the record to the data base.

**Input/Output:** IODB - direct access file I/O.

**Error Messages:** None

**External Calls:** IODB

**Argument List:**
- MCOND - number of entries in the master record
- MASTER - master record
- NTYPE - master record data type

**Important Variables:** Same as above

**Common Blocks:**
- TITLES
- DBREC
- HEADER
6.0 ERROR MESSAGES

The CADS software will attempt to recover from input or processing errors in one of several different ways. The type and severity of the error will define the error handling procedure to be used by CADS.

The most common errors are generally mistypings of command words, options, or parameters. In these cases, CADS will say that option or command was not found or is not valid and ask that the entire command line be re-entered. The user should then enter the entire line with the correct spellings and options.

CADS will check parameter numeric values for real or integer numbers as required. If an incorrect or mistyped numeric value is entered, CADS will echo the character string and ask that a real or integer number be entered. In these cases the user should enter the numeric values only and not the entire command line.

Finally, the DI-3000 graphics package may issue a warning or error message based upon some series of actions it is taking. The level at which errors will be printed out and the unit on which they will be printed can be changed by the CADS software maintenance personnel. The JSETER and JFILES routines control the DI-3000 error messages. Typically, DI-3000 will continue processing after an error message through its own internal routines. The CADS command may have to be re-entered and/or modified to obtain a correct display after a DI-3000 message since DI-3000 may not have taken the appropriate action in processing the given error.
7.0 REFERENCES


END
12-86
DTIC