

Programmer's Manual for Mechanical APDL



ANSYS, Inc. Southpointe 275 Technology Drive Canonsburg, PA 15317 ansysinfo@ansys.com http://www.ansys.com (T) 724-746-3304 (F) 724-514-9494

Release 12.0 April 2009

ANSYS, Inc. is certified to ISO 9001:2008.

Copyright and Trademark Information

© 2009 SAS IP, Inc. All rights reserved. Unauthorized use, distribution or duplication is prohibited.

ANSYS, ANSYS Workbench, Ansoft, AUTODYN, EKM, Engineering Knowledge Manager, CFX, FLUENT, HFSS and any and all ANSYS, Inc. brand, product, service and feature names, logos and slogans are registered trademarks or trademarks of ANSYS, Inc. or its subsidiaries in the United States or other countries. ICEM CFD is a trademark used by ANSYS, Inc. under license. CFX is a trademark of Sony Corporation in Japan. All other brand, product, service and feature names or trademarks are the property of their respective owners.

Disclaimer Notice

THIS ANSYS SOFTWARE PRODUCT AND PROGRAM DOCUMENTATION INCLUDE TRADE SECRETS AND ARE CONFIDENTIAL AND PROPRIETARY PRODUCTS OF ANSYS, INC., ITS SUBSIDIARIES, OR LICENSORS. The software products and documentation are furnished by ANSYS, Inc., its subsidiaries, or affiliates under a software license agreement that contains provisions concerning non-disclosure, copying, length and nature of use, compliance with exporting laws, warranties, disclaimers, limitations of liability, and remedies, and other provisions. The software products and documentation may be used, disclosed, transferred, or copied only in accordance with the terms and conditions of that software license agreement.

ANSYS, Inc. is certified to ISO 9001:2008.

U.S. Government Rights

For U.S. Government users, except as specifically granted by the ANSYS, Inc. software license agreement, the use, duplication, or disclosure by the United States Government is subject to restrictions stated in the ANSYS, Inc. software license agreement and FAR 12.212 (for non-DOD licenses).

Third-Party Software

See the legal information in the product help files for the complete Legal Notice for ANSYS proprietary software and third-party software. If you are unable to access the Legal Notice, please contact ANSYS, Inc.

Published in the U.S.A.

Table of Contents

Preface	xv
. Guide to Interfacing with ANSYS	1
1. Format of Binary Data Files	3
1.1. Understanding ANSYS Binary Files	3
1.1.1. Conventions Used to Describe Binary Files	3
1.1.2. The Standard Header for ANSYS Binary Files	4
1.2. Description of the Results File	5
1.2.1. Nomenclature	5
1.2.2. Standard ANSYS File Header	5
1.2.3. Results File Format	5
1.3. Description of the Reduced Displacement File	22
1.3.1. Standard ANSYS File Header	22
1.3.2. RDSP File Format	
1.4. Description of the Reduced Complex Displacement File	
1.4.1. Standard ANSYS File Header	26
1.4.2. RFRQ File Format	
1.5. Description of the Modal Results File	
1.5.1. Standard ANSYS File Header	
1.5.2. MODE File Format	30
1.6. Description of the Element Matrices File	
1.6.1. Standard ANSYS File Header	
1.6.2. EMAT File Format	
1.7. Description of the Substructure Matrices File	
1.7.1. Standard ANSYS File Header	
1.7.2. SUB File Format	
1.8. Description of the Component Mode Synthesis Matrices (CMS) File	
1.8.1. Standard ANSYS File Header	
1.8.2. CMS File Format	
1.9. Description of the Full Stiffness-Mass File	
1.9.1. Standard ANSYS File Header	
1.9.2. FULL File Format	
2. Accessing Binary Data Files	
2.1. Accessing ANSYS Binary Files	
2.1.1. Access Routines to Results, Substructure, and Matrix Files	
2.1.2. Characteristics of ANSYS Binary Files	
2.1.3. Viewing Binary File Contents	
2.1.4. Abbreviations	
2.1.5. binini (Initializing Buffered Binary I/O Systems)	
2.1.6. Function sysiqr (Retrieving the Status of a File)	
2.1.7. Function biniqr8 (Retrieving System-Dependent Parameters)	
2.1.8. Function binset (Opening a Blocked Binary File or Initializing Paging Space)	
2.1.9. Subroutine bintfo (Defining Data for a Standard ANSYS File Header)	
2.1.10. Subroutine binhed (Writing the Standard ANSYS File Header)	
2.1.12. Subroutine binirus (Writing Data from a Buffered File)	
2.1.12. Subroutine binwrto (Writing Data to a Bullered File)	
2.1.14. Subroutine exinc4 (Decoding an integer String into a Character String)	
2.1.14. Subroutine linexc4 (Coding a Character String into an integer String)	
2.1.16. Subroutine largeIntGet (Converting Two Integers into a Pointer)	
2.2. Demonstration Routines	

2.2.1. Program bintst (Demonstrates Dumping a Binary File and Copying It for Co	=
poses)	
2.2.2. Subroutine bintrd (Demonstrates Printing a Dump of File Contents)	
2.2.3. Subroutine binting (Demonstrates Copying Binary File Contents)	
2.2.4. Program wrtsub (Demonstrates Writing an ANSYS Substructure File)	
2.2.5. Program rdsubs (Demonstrates Reading a Substructure File)	64
2.2.6. Program rdfull (Demonstrates Reading and Reformatting the .FULL File)	64
2.2.7. Program ResRdDemo (Demonstrates Reading a Results File)	
2.2.8. Program ResWrDemo (Demonstrates Writing a Results File)	
3. Results File Access Routines	
2.3.1. Overview of the Routines	
2.3.2. ResRdBegin (Opening the File and Retrieving Global Information)	
2.3.3. ResRdGeomBegin (Retrieving Global Geometry Information)	
2.3.4. ResRdType (Retrieving Element Types)	
2.3.5. ResRdReal (Retrieving Real Constants)	
2.3.6. ResRdCsys (Retrieving Coordinate Systems)	
2.3.7. ResRdNode (Retrieving Nodal Coordinates)	
2.3.8. ResRdElem (Retrieving Elements)	
2.3.9. ResRdSolBegin (Retrieving Result Set Location)	
2.3.10. ResRdDisp (Retrieving Nodal Solution)	
2.3.11. ResRdRfor (Retrieving Reaction Solution)	
2.3.12. ResRdFix (Retrieving Applied Nodal Constraints)	
2.3.13. ResRdForc (Retrieving Applied Nodal Loads Solution)	
2.3.14. ResRdEstr (Retrieving Element Solutions)	
e CDWRITE (CDB) File Format	71
.1. Using the CDWRITE Command	
3.1.1. Customizing Degree of Freedom Labels: the /DFLAB Command	
.2. Coded Database File Commands	72
3.2.1. CE Command	73
3.2.2. CP Command	
3.2.3. CMBLOCK Command	
3.2.4. EBLOCK Command	
3.2.5. EDCADAPT Command	
3.2.6. EDCGEN Command	76
3.2.7. EDCURVE Command	
3.2.8. EDDRELAX Command	
3.2.9. EDLCS Command	
3.2.10. EDLOAD Command	
3.2.11. EDPREAD Command	
3.2.12. EDWELD Command	
3.2.13. EN Command	
3.2.14. LOCAL Command	
3.2.15. M Command	
3.2.16. MPDATA Command	
3.2.17. MPTEMP Command	
3.2.18. N Command	
3.2.19. NBLOCK Command	
3.2.20. R Command	
3.2.21. RLBLOCK Command	
3.2.22. SECBLOCK Command	
3.2.23. SFBEAM Command	85

3.2.24. SFE Command	86
4. ANSYS Graphics File Format	87
4.1. Pixmap Format for Graphic Display Files	87
4.2. Neutral Graphics File Format	88
4.2.1. Characters the Graphics File Uses	88
4.2.2. Graphics File Directives	89
4.2.2.1. Parameter Types for Graphics File Directives	89
4.2.2.2. Directive Descriptions	90
4.2.2.3. Color Specification	92
4.3. Decoding a Graphics File: an Example	93
4.3.1.The Example Command Stream	93
4.3.2. Example Graphics File Contents	
I. Guide to ANSYS User Programmable Features	
5. Using User Programmable Features (UPFs)	
5.1. What Are UPFs?	
5.2. What You Should Know Before Using UPFs	
5.3. Planning Your UPFs	
5.4. Studying the ANSYS User Routines	
5.5. Programming in Languages Other than Fortran	
5.6. Developing UPFs: a Suggested Strategy	
5.7. Include Decks	
5.8. Linking User Routines	
5.9. Compiling and Linking UPFs on UNIX/Linux Systems	
5.10. Compiling and Linking UPFs on Windows Systems	
5.11. Activating UPFs	
5.12. Running Your Custom Executable	
5.13. Verifying Your Routines	
5.14.1 Tracking the Poth of Program Lagis	
5.14.1.Tracking the Path of Program Logic	
5.14.2. Debugging Elements and Solutions	
5.14.2.1. Solution Debug Format	
5.14.2.3. General Debug Format	
5.14.2.5. General Debug Format	
5.16. Generating Output	
5.17. Reading Large Data Files More Rapidly	
6. UPF Subroutines and Functions	
6.1. Creating a New Element	
6.1.1. Input and Output Abbreviations	
6.1.2. Creating a New Element via the User-Defined Element API	
6.1.2.1. Subroutine UserElem (Writing Your Own Elements)	
6.1.2.2. Subroutine ElemGetMat (Calling the ANSYS Standard Structural Material Library)	
6.1.3. Creating a New Element by Directly Accessing the ANSYS Database	
6.1.3.1. User Subroutines	
6.1.3.2. Subroutine uec100 (Defining Characteristics of the usr100 Subroutine)	
6.1.3.2.1. Subroutines uec101 through uec105	
6.1.3.3. Subroutine uex100 (Overriding Element Characteristic Defaults)	
6.1.3.3.1. Subroutines uex101 through uex105	
6.1.3.4. Subroutine uel 100 (Computing Element Matrices, Load Vectors, and Results)	
6.1.3.4.1. Subroutines uel101 through uel105	
6.1.3.5. Subroutine uep100 (Printing Output for User Elements in POST1 via	
PRESOL,ELEM)	131

	6.1.3.5.1. Subroutines uep101 through uep105	
	6.1.3.6. Subroutine usertr (Adjusting the Nodal Orientation Matrix)	
	6.1.3.7. Subroutine userac (Accessing Element Information)	. 132
6.2	. Supporting Subroutines for Element Creation	
	6.2.1. Subroutine nminfo (Returning Element Reference Names)	133
	6.2.2. Subroutine sygidx (Fetching the Index for Saved Variables)	
	6.2.3. Subroutine syrget (Fetching Saved Variable Data for an Element)	
	6.2.4. Subroutine syrput (Writing an Element's Saved Variable Set)	. 134
	6.2.5. Subroutine svpidx (Writing the Saved Variable Element Index to a File)	. 135
	6.2.6. Subroutine mreuse (Determining Which Element Matrices Can Be Reused)	. 135
	6.2.7. Subroutine subrd (Reading Element Load Data for a Substructure Generation Run)	. 136
	6.2.8. Subroutine subwrt (Writing an Element Load Vector to a File for a Substructure Generation	1
	Run)	
	6.2.9. Subroutine rvrget (Fetching Real Constants for an Element)	. 137
	6.2.10. Subroutine propev (Evaluating a Group of Material Properties)	. 138
	6.2.11. Subroutine prope1 (Evaluating One Material Property)	
	6.2.12. Subroutine pstev1 (Evaluating Material Properties for 1-D Elements)	. 139
	6.2.13. Subroutine tbuser (Retrieving User Table Data)	. 139
	6.2.14. Subroutine plast1 (Updating an Element's Plastic History)	
	6.2.15. Subroutine plast3 (Updating an Element's Plastic History, 4 or 6 components)	. 140
	6.2.16. Subroutine creep1 (Updating an Element's Creep History)	
	6.2.17. Subroutine creep3 (Updating an Element's Creep History, 3-D Elements)	. 142
	6.2.18. Subroutine swell1 (Updating an Element's Swelling History)	
	6.2.19. Subroutine swell3 (Updating an Element's Swelling History, 3-D Elements)	
	6.2.20. Function elLenPsvrBuf (Determining additional ESAV Record for Plasticity)	
	6.2.21. Function nlget (Retrieving Material Nonlinear Property Information)	
	6.2.22. Subroutine usereo (Storing Data in the nmisc Record)	
	6.2.23. Subroutine eldwrtL (Writing Element Data to a File)	
	6.2.24. Subroutine eldwrnL (Writing Element Nonsummable Miscellaneous Data to the Results	
	File)	
	6.2.25. Subroutine trrot (Computing the Rotation Vector)	
	6.2.26. Subroutine rottr (Computing the Transformation Matrix)	
	6.2.27. Subroutine xyzup3 (Updating an Element's 3-D Nodal Coordinates)	
	6.2.28. Subroutine updrot (Updating the Rotation Pseudovector)	
	6.2.29. Subroutine tmpget (Defining Current Temperature Loads)	
	6.2.30. Subroutine prsget (Defining Current Pressure Loads)	
	6.2.31. Subroutine cnyget (Defining Current Convection Loads)	
	6.2.32. Subroutine hgnget (Defining Current Heat Generation Loads)	
	6.2.33. Subroutine prinst (Computing principal stress and stress intensity)	
6.3	Subroutines for Modifying and Monitoring Existing Elements	
	6.3.1. Subroutine userou (Storing User-Supplied Element Output)	
	6.3.2. Subroutine useran (Modifying Orientation of Material Properties)	. 151
	6.3.3. Subroutine userrc (Performing User Operations on COMBIN7 and COMBIN37 Paramet-	4-4
	ers)	
	6.3.4. Subroutine UEIMatx (Accessing Element Matrices and Load Vectors)	
	6.3.5. Subroutine UTHICK (Getting User-defined Initial Thickness)	
	6.3.6. Subroutine UsrFictive (Providing User-defined Fictive Temperature Relationship)	
	6.3.7. Subroutine Us_Surf_Str (Captures surface stresses)	
	6.3.8. Subroutine uflex (Computes flexibility factors for PIPE288 and PIPE289)	. 154
	6.3.9. Subroutine usflex (Computes the flexibility factor for PIPE16, PIPE17, PIPE18, and	1 -
	PIPE60)	
	v.s. iv. suproutine usishin ivaiculates oseudonne nine increment	. 10

6.4	. Subroutines for Customizing Material Behavior	
	6.4.1. Subroutine UserMat (Creating Your Own Material Model)	
	6.4.1.1. UserMat Overview	
	6.4.1.2. Stress, Strain, and Material Jacobian Matrix	
	6.4.1.3.The UserMat API	
	6.4.1.4. UserMat Variables	
	6.4.1.5. Table (TB) Commands for UserMat	
	6.4.1.6. Material Constitutive Integration with UserMat	
	6.4.1.7. UserMat Restrictions	
	6.4.1.8. Accessing Material and Element Data for UserMat	
	6.4.1.9. Utility Functions for UserMat	
	6.4.2. Subroutine UserHyper (Writing Your Own Hyperelasticity Laws)	
	6.4.3. Subroutines UserCreep and UserCr (Defining Viscoplastic/Creep Material Behavior)	165
	6.4.3.1. Creep Subroutine UserCreep	165
	6.4.3.2. Creep Subroutine UserCr	
	6.4.4. Subroutine user_tbelastic (Defining Material Linear Elastic Properties)	168
	6.4.4.1. Overview of the user_tbelastic Subroutine	168
	6.4.4.2. Data Types Supported by user_tbelastic	168
	6.4.4.3. Table (TB) Command for user_tbelastic	
	6.4.4.4. User Interface for user_tbelastic	169
	6.4.4.5.The user_tbelastic API	
	6.4.4.6. Usage Example for user_tbelastic	170
	6.4.5. Subroutine UserPL (Writing Your Own Plasticity Laws)	
	6.4.6. Subroutine usersw (Writing Your Own Swelling Laws)	
	6.4.7. Subroutine userck (Checking User-Defined Material Data)	
	6.4.8. Subroutine UserVisLaw (Defining Viscosity Laws)	174
	6.4.9. Subroutine userfric (Writing Your Own Friction Laws)	175
	6.4.10. Supporting Function egen	
6.5	. Subroutines for Customizing Loads	
	6.5.1. Subroutine usrefl (Changing Scalar Fields to User-Defined Values)	
	6.5.2. Subroutine userpr (Changing Element Pressure Information)	
	6.5.3. Subroutine usercv (Changing Element Face Convection Surface Information)	
	6.5.4. Subroutine userfx (Changing Element Face Heat Flux Surface Information)	181
	6.5.5. Subroutine userch (Changing Element Face Charge Density Surface Information)	
	6.5.6. Subroutine userfd (Computing the Complex Load Vector for Frequency Domain Logic)	182
	6.5.7. Function userpe (Calculating Rotation Caused by Internal Pressure)	
	6.5.8. Subroutine UsrViscEl (Performs Viscoelastic Computation)	184
	6.5.9. Subroutine usrsurf116 (Modifying SURF151 and SURF152 Film Coefficients and Bulk Tem-	
	peratures)	
	6.5.10. Subroutine User116Cond (Computing the conductance coefficient for FLUID116)	
	6.5.11. Subroutine User116Hf (Computing the film coefficient for FLUID116)	
	6.5.12. Subroutine userPartVelAcc (Computing Particle Velocities and Accelerations)	187
	6.5.12.1. Subroutine userPartVelAccSetup (Initializing Data for Use by the userPartVelAcc	
	Subroutine)	
	6.5.12.2. Subroutine userWavHt	
	6.5.12.3. Subroutine wvhybl	
	6.5.12.4. Subroutine wvargu	
	6.5.12.5. Subroutine wavHtSec	
	Running ANSYS as a Subroutine	
6.7	Defining Your Own Commands	
	6.7.1. Function user01	
	6.7.2. Function user02 (Demonstrates Offsetting Selected Nodes)	195

	6.7.3. Function user03 (Demonstrates Using ANSYS Memory)	196
	6.7.4. Function user04	199
	6.7.5. Functions user05 through user10	200
	6.8. Supporting Subroutines	200
	6.8.1. Function GetRForce (Getting Nodal Reaction Force values)	200
	6.8.2. Function GetStackDisp (Getting Current Displacement Values)	201
	6.8.3. Subroutine ElResultStrt (Getting Load Data from Analysis Results)	202
	6.8.4. Subroutine ElResultGet (Getting Results Values at Selected Points)	202
	6.8.5. Subroutine ElInterp (Finding Element Coordinates)	202
	6.9. Access at the Beginning and End of Various Operations	203
	6.10. Creating Your Own Optimization Subroutine	204
	6.10.1. Linking a Custom Optimization Program to ANSYS	
	6.10.2. Subroutine userop (Defining a Custom Optimization Subroutine)	
	6.10.3. Structuring Your Input	
	6.11. Memory Management Subroutines	
	6.11.1. Using the Memory-Management Subroutines	
	6.11.2. Function fAnsMemAlloc (Allocating Space and Returning a Pointer)	
	6.11.3. Subroutine fAnsMemFree (Deallocating Space)	
	6.12. Parameter-Processing Subroutines	
	6.12.1. Subroutine pardim (Creating a Dimensioned Parameter)	
	6.12.2. Function parevl (Finding and Evaluating a Parameter)	
	6.12.3. Subroutine pardef (Adding a Parameter)	
	6.13. Miscellaneous Useful Functions	
	6.13.1. Using Function RunCommand	
	6.13.2. Using the / UNDO Command	
	6.13.3. Using the /HOLD command	
7.	Accessing the ANSYS Database	
	7.1. Routines for Selecting and Retrieving Nodes and Elements	
	7.1.1. ndnext Function (Getting the Next Node Number)	
	7.1.2. ndprev Function (Getting the Number of the Previous Selected Node)	
	7.1.3. ndnxdf Function (Getting the Number of the Next Defined Node)	
	7.1.4. ndsel Function (Selecting, Unselecting, Deleting, or Inverting a Node)	
	7.1.5. elnext Function (Getting the Number of the Next Element)	
	7.1.6. elprev Function (Getting the Number of the Previous Selected Element)	
	7.1.7. elnxdf Function (Getting the Number of the Next Defined Element)	
	7.1.8. elsel Subroutine (Selecting, Unselecting, Deleting, or Inverting an Element)	
	7.2. Node Information Routines	
	7.2.1. ndingr Function (Getting Information About a Node)	218
	7.2.2. getnod Function (Getting a Nodal Point)	
	7.2.3. putnod Function (Storing a Node)	
	7.2.4. ndgall Function (Getting the XYZ/Rotation Coordinates Vector for a Node)	
	7.2.5. ndspgt Subroutine (Getting the Nodal Solution for a Node of an Element)	
	7.3. Element Attribute Routines	
	7.3.1. elmiqr Function (Getting Information About an Element)	221
	7.3.2. elmget Function (Getting an Element's Attributes and Nodes)	
	7.3.3. elmput Subroutine (Storing an Element)	
	7.3.4. etyiqr Function (Getting a Data Item About an Element Type)	
	7.3.5. etyget Function (Getting Information About an Element Type)	
	7.3.6. etyput Subroutine (Storing Element Type Data)	
	7.3.7. echrtr Subroutine (Getting Information About Element Characteristics)	
	7.3.8. etysel Subroutine (Selecting, Unselecting, Deleting, or Inverting an Element Type)	
	7.3.9. mpingr Function (Getting Information About a Material Property)	

	7.3.10. mpget Function (Getting a Material Property Table)	226
	7.3.11. mpput Subroutine (Storing a Material Property Table)	
	7.3.12. mpdel Subroutine (Deleting a Material Property Table)	
	7.3.13. rlingr Function (Getting Information About a Real Constant Set)	227
	7.3.14. rlget Function (Getting Real Constant Data)	
	7.3.15. rlsel Subroutine (Selecting or Deleting a Real Constant Set)	
	7.3.16. csyiqr Function (Getting Information About a Coordinate System)	
	7.3.17. csyget Function (Getting a Coordinate System)	229
	7.3.18. csyput Subroutine (Storing a Coordinate System)	
	7.3.19. csydel Subroutine (Deleting a Coordinate System)	
	7.3.20. userac Subroutine (Demonstrates Use of Element Attribute Routines)	
7.4	Coupling and Constraint Routines	
	7.4.1. cpinqr Function (Getting Information About a Coupled Set)	
	7.4.2. cpget Function (Getting a Coupled Set)	
	7.4.3. cpput Subroutine (Storing a Coupled Set)	
	7.4.4. cpsel Subroutine (Selecting or Deleting a Coupled Set)	
	7.4.5. ceinqr Function (Getting Information About a Constraint Equation Set)	232
	7.4.6. ceget Function (Getting an Constraint Equation)	232
	7.4.7. ceput Subroutine (Storing a Constraint Equation)	233
	7.4.8. cesel Subroutine (Deleting or Selecting a Constraint Equation)	233
7.5	Nodal Loading Routines	
	7.5.1. disiqr Function (Getting a Information About Constraints)	233
	7.5.2. disget Function (Getting a Constraint from the Database)	234
	7.5.3. disput Subroutine (Storing a Constraint at a Node)	234
	7.5.4. disdel Subroutine (Deleting a Constraint at a Node)	235
	7.5.5. foriqr Function (Getting Information About Nodal Loads)	235
	7.5.6. forget Function (Getting a Constraint from the Database)	235
	7.5.7. forput Subroutine (Storing a Nodal Load at a Node)	236
	7.5.8. fordel Subroutine (Deleting a Nodal Load at a Node)	236
	7.5.9. ntpiqr Function (Getting Information About a Nodal Temperature)	
	7.5.10. ntpget Function (Getting a Specified Nodal Temperature)	
	7.5.11. ntpput Subroutine (Storing a Nodal Temperature)	
	7.5.12. ntpdel Subroutine (Deleting a Nodal Temperature)	
	7.5.13. nhgiqr Function (Getting Information About Nodal Heat Generations)	
	7.5.14. nhgget Function (Getting a Nodal Heat Generation)	238
	7.5.15. nhgput Subroutine (Storing Nodal Heat Generation)	
	7.5.16. nhgdel Subroutine (Deleting a Nodal Heat Generation)	
	7.5.17. nfuiqr Function (Getting Information About Nodal Fluences)	239
	7.5.18. nfuget Function (Getting a Nodal Fluence)	239
	7.5.19. nfuput Subroutine (Storing a Nodal Fluence)	240
	7.5.20. nfudel Subroutine (Deleting a Nodal Fluence)	240
	7.5.21. ndciqr Function (Getting Information About Nodal Current Densities)	240
	7.5.22. ndcget Function (Getting a Nodal Current Density)	241
	7.5.23. ndcput Subroutine (Storing a Nodal Current Density)	241
	7.5.24. ndcdel Subroutine (Deleting a Nodal Current Density)	241
	7.5.25. nvdiqr Function (Getting Information About Nodal Magnetic Virtual Displacements)	241
	7.5.26. nvdget Function (Getting a Nodal Magnetic Virtual Displacement)	242
	7.5.27. nvdput Subroutine (Storing a Nodal Virtual Displacement)	242
	7.5.28. nvddel Subroutine (Deleting a Nodal Virtual Displacement)	242
7.6	Element Loading Routines	243
	7.6.1. epriqr Function (Getting Information About Element Pressure/Convection)	243
	7.6.2. eprget Function (Getting an Element Face Pressure)	243

	7.6.3. eprput Subroutine (Storing an Element Face Pressure)	. 244
	7.6.4. eprdel Subroutine (Deleting an Element Pressure/Convection)	. 244
	7.6.5. ecviqr Function (Getting Information About Element Convections)	. 244
	7.6.6. ecvget Function (Getting an Element Face Convection)	. 245
	7.6.7. ecvput Subroutine (Storing an Element Face Convection)	. 245
	7.6.8. ecvdel Subroutine (Deleting a Convection on an Element)	
	7.6.9. etpiqr Function (Getting Information About Element Temperatures)	. 246
	7.6.10. etpget Function (Getting an Element Temperature)	
	7.6.11. etpput Subroutine (Storing an Element Temperature)	
	7.6.12. etpdel Subroutine (Deleting an Element Temperature)	
	7.6.13. ehgigr Function (Getting Information About Element Heat Generation)	
	7.6.14. ehgget Function (Getting an Element Heat Generation)	
	7.6.15. ehgput Subroutine (Storing an Element Heat Generation)	
	7.6.16. ehgdel Subroutine (Deleting an Element Heat Generation)	. 249
	7.6.17. efuigr Function (Getting Information About Element Fluences)	
	7.6.18. efuget Function (Getting an Element Fluence)	
	7.6.19. efuput Subroutine (Storing an Element Fluence)	
	7.6.20. efudel Subroutine (Deleting an Element Fluence)	
	7.6.21. edciqr Function (Getting Information About Element Current Densities)	
	7.6.22. edcget Function (Getting Element Current Densities)	
	7.6.23. edcput Subroutine (Storing an Element Current Density)	. 251
	7.6.24. edcdel Subroutine (Deleting an Element Current Density)	. 252
	7.6.25. evdiqr Function (Getting Information About Element Virtual Displacements)	. 252
	7.6.26. evdget Function (Getting an Element Virtual Displacement)	. 252
	7.6.27. evdput Subroutine (Storing an Element Virtual Displacement)	. 253
	7.6.28. eimigr Function (Getting Information About Element Impedances)	. 253
	7.6.29. eimget Function (Getting an Element Face Impedance)	
	7.6.30. eimput Subroutine (Storing an Element Impedance)	. 254
	7.6.31. eimdel Subroutine (Deleting an Element Impedance)	. 254
	7.6.32. esfiqr Function (Getting Information About Element Surface Stress Data)	. 255
	7.6.33. esfget Function (Getting Element Surface Stress Data)	. 255
	7.6.34. esfput Subroutine (Storing Element Surface Stress Data)	. 255
	7.6.35. esfdel Subroutine (Deleting an Element's Surface Stress Data)	. 256
	7.6.36. efsdel Subroutine (Deleting a Flagged Surface on an Element)	. 256
	7.6.37. efsget function (Getting Element Face Flagged Surfaces)	. 256
	7.6.38. efsiqr function (Getting Information About Flagged Surfaces)	. 256
	7.6.39. efsput Subroutine (Storing an Element Face Flagged Surface)	. 257
7.7	. Results Information Routines	. 257
	7.7.1. dspiqr Function (Getting Information About Nodal Results)	. 257
	7.7.2. dspget Function (Getting a Nodal Result from the Database)	. 258
	7.7.3. dspput Subroutine (Storing a Result at a Node)	. 258
	7.7.4. dspdel Subroutine (Deleting a Result at a Node)	. 258
	7.7.5. emsiqr Function (Getting Information About an Element's Miscellaneous Summable	
	Data)	
	7.7.6. emsget Function (Getting an Element's Miscellaneous Summable Data)	. 259
	7.7.7. emsput Subroutine (Storing an Element's Miscellaneous Summable Data)	. 259
	7.7.8. emsdel Subroutine (Deleting an Element's Miscellaneous Summable Data)	. 260
	7.7.9. enfiqr Function (Getting Information About Element Nodal Forces)	
	7.7.10. enfget Function (Getting an Element's Nodal Forces)	
	7.7.11. enfput Subroutine (Storing an Element's Nodal Forces)	
	7.7.12. enfdel Subroutine (Deleting an Element's Nodal Forces)	
	7.7.13 ensign Function (Getting Information About an Flement's Nodal Stresses)	261

7.7.14. ensget Function (Getting an Element's Nodal Stresses)	
7.7.15. ensput Subroutine (Storing Nodal Stresses at an Element)	
7.7.16. ensdel Subroutine (Deleting an Element's Nodal Stresses)	263
7.7.17. engiqr Function (Getting Information About an Element's Energies)	
7.7.18. engget Function (Getting an Element's Energies)	
7.7.19. engput Subroutine (Storing an Element's Energies and Volume)	
7.7.20. engdel Subroutine (Deleting an Element's Energies)	
7.7.21. egriqr Function (Getting Information About an Element's Nodal Gradients)	
7.7.22. egrget Function (Getting an Element's Nodal Gradients)	
7.7.23. egrput Subroutine (Storing an Element's Nodal Gradients)	
7.7.24. egrdel Subroutine (Deleting an Element's Nodal Gradients)	
7.7.25. eeliqr Function (Getting Information About an Element's Nodal Elastic Strains)	
7.7.26. eelget Function (Getting an Element's Nodal Elastic Strains)	
7.7.27. eelput Subroutine (Storing an Element's Nodal Elastic Strains)	
7.7.28. eeldel Subroutine (Deleting an Element's Nodal Elastic Strains)	
7.7.29. epliqr Function (Getting Information About an Element's Nodal Plastic Strains)	
7.7.30. eplget Function (Getting an Element's Nodal Plastic Strains)	
7.7.31. eplput Subroutine (Storing an Element's Nodal Plastic Strains)	269
7.7.32. epidel Subroutine (Deleting an Element's Nodal Plastic Strains)	
7.7.33. ecriqr Function (Getting Information About an Element's Nodal Creep Strains)	270
7.7.34. ecrget Function (Getting an Element's Nodal Creep Strains)	
7.7.35. ecrput Subroutine (Storing an Element's Nodal Creep Strains)	
7.7.36. ecrdel Subroutine (Deleting an Element's Nodal Creep Strains)	272
7.7.37. ethiqr Function (Getting Information About an Element's Nodal Thermal Strains)	272
7.7.38. ethget Function (Getting an Element's Nodal Thermal Stresses)	
7.7.39. ethput Subroutine (Storing an Element's Nodal Thermal Stresses)	
7.7.40. ethdel Subroutine (Deleting an Element's Thermal, Initial, and Swelling Strains)	
7.7.41. euliqr Function (Getting Information About an Element's Euler Angles)	
7.7.42. eulget Function (Getting an Element's Nodal Euler Angles)	
7.7.43. eulput Subroutine (Storing an Element's Euler Angles)	
7.7.44. euldel Subroutine (Deleting an Element's Euler Angles)	
7.7.45. efxiqr Function (Getting Information About Element Fluxes)	
7.7.46. efxget Function (Getting an Element Flux)	
7.7.47. efxput Subroutine (Storing an Element's Fluxes)	
7.7.48. efxdel Subroutine (Deleting Element Fluxes)	
7.7.49. elfiqr Function (Getting Information About Element Local Forces)	
7.7.50. elfget Function (Getting an Element Local Force)	
7.7.51. elfput Subroutine (Storing an Element's Local Forces)	
7.7.52. elfdel Subroutine (Deleting Element Local Forces)	278
7.7.53. emniqr Function (Getting Information About Element Miscellaneous Non-summable	
Data)	
7.7.54. emnget Function (Getting an Element's Miscellaneous Non-summable Data)	279
7.7.55. emnput Subroutine (Storing an Element's Miscellaneous Non-summable Data)	
7.7.56. emndel Subroutine (Deleting an Element's Miscellaneous Non-summable Data)	279
7.7.57. ecdiqr Function (Getting Information About Element Current Densities)	280
7.7.58. ecdget Function (Getting an Element Current Density)	
7.7.59. ecdput Subroutine (Storing an Element's Current Densities)	
7.7.60. ecddel Subroutine (Deleting Element Current Densities)	
7.7.61. enliqr Function (Getting Information About Element Nonlinear Tables)	
7.7.62. enlget Function (Getting Element Nonlinear Tables)	
7.7.63. enlput Subroutine (Storing an Element's Nonlinear Tables)	282
7.7.64. enIdel Subroutine (Deleting Element Nonlinear Tables)	282

7.7.65. ehciqr Function (Getting Information About Calculated Element Heat Generations)	282
7.7.66. ehcget Function (Getting a Calculated Element Heat Generation)	283
7.7.67. ehcput Subroutine (Storing an Element's Calculated Heat Generations)	283
7.7.68. ehcdel Subroutine (Deleting Element Calculated Heat Generations)	283
8. Subroutines for Users' Convenience	285
8.1. Input and Output Abbreviations	285
8.2. General Subroutines	285
8.2.1. dptoch Subroutine (Retrieve Eight Characters From a Double Precision Variable)	285
8.2.2. wringr Function (Obtain Information About Output)	
8.2.3. eringr Subroutine (Obtaining Information from the Errors Common)	
8.2.4. TrackBegin Subroutine (Beginning Tracking for a Subroutine Call)	
8.2.5. TrackEnd Subroutine (Ending Tracking for a Subroutine Call)	
8.2.6. erhandler Subroutine (Displaying ANSYS Errors)	
8.2.7. intrp Subroutine (Doing Single Interpolation)	
8.2.8. tranx3 Subroutine (Processing Geometry for 3-D Line Elements)	
8.2.9. systop Subroutine (Stopping an ANSYS Program Run)	
8.3. Vector Functions	
8.3.1. vdot Function (Computing the Dot Product of Two Vectors)	
8.3.2. vidot Function (Computing the Dot Product of Two Vectors with Increments)	
8.3.3. vsum Function (Summing Vector Components)	
8.3.4. vmax Function (Retrieving the Maximum Vector Value at a Given Location)	
8.3.5. lasty Function (Retrieving the Position of the Last Nonzero Term in a Double Precision	
Vector)	. 292
8.3.6. izero Function (Setting an Integer Vector to Zero)	
8.3.7. imove Function (Assigning Equal Values to Two Integer Vectors)	
8.3.8. vzero Subroutine (Initializing a Vector to Zero)	
8.3.9. vmove Subroutine (Moving One Vector into Another)	
8.3.10. vimove Subroutine (Moving One Vector into Another Incrementally)	
8.3.11. vinit Subroutine (Assigning a Scalar Constant to a Vector)	
8.3.12. viinit Subroutine (Assigning a Scalar Constant to a Vector Incrementally)	
8.3.13. vapb Subroutine (Setting a Vector to Sum of Two Vectors)	
8.3.14. vapb1 Subroutine (Combining Two Vectors in One)	
8.3.15. vapcb1 Subroutine (Multiplying a Vector to a Constant)	
8.3.16. vamb Subroutine (Gets a Third Vector by Subtracting One Vector from Another)	
8.3.17. vamb1 Subroutine (Subtracting One Vector from Another)	
8.3.18. vmult Subroutine (Multiplying a Vector by a Constant)	
8.3.19. vmult1 Subroutine (Multiplying a Vector by a Constant)	
8.3.20. vcross Subroutine (Defining a Vector via a Cross Product)	
8.3.21. vnorme Subroutine (Normalizing a Three-Component Vector)	
8.3.22. vnorm Subroutine (Normalizing a Vector to Unit Length)	
8.3.23. ndgxyz Function (Getting the X,Y,Z Vector for a Node)	
8.3.24. ndpxyz Subroutine (Storing X,Y,Z for a Node)	
8.4. Matrix Subroutines	
8.4.1. maxv Subroutine (Multiplying a Vector by a Matrix)	
8.4.2. maxv1 Subroutine (Multiplying a Vector by a Matrix)	
8.4.3. matxv Subroutine (Multiplying a Vector by a Full Transposed Matrix)	
8.4.4. matxv1 Subroutine (Multiplying a Vector by a Full Transposed Matrix)	
8.4.5. matxb Subroutine (Transposing a matrix)	
8.4.6. maat Subroutine (Changing a Matrix Value via Addition, Multiplication, and Transposi-	
tion)	299
8.4.7. matba Subroutine (Updating Matrix Value via Transposition, Multiplications, and Addi-	
tion)	300

8.4.8. matsym Subroutine (Filling the Upper Triangle from the Lower Triangle)	300
8.4.9. mctac Subroutine (Transposing a symmetric matrix)	300
8.4.10. tran Subroutine (Transposing a matrix)	301
8.4.11. symeqn Subroutine (Solving Simultaneous Linear Equations)	301
A. Creating External Commands in UNIX	303
A.1. Tasks in Creating an External Command	303
A.1.1. Creating Compatible Code	303
A.1.2. Creating a Shared Library	304
A.1.3. Creating an External Table File	
A.1.4. Setting the ANSYS_EXTERNAL_PATH Environment Variable	
A.1.5. Using External Commands	306
A.1.6. Checking External Command Status	306
A.1.7. Resetting External Commands	306
B. Creating External Commands in Windows	
B.1. Tasks in Creating an External Command	309
B.1.1. Creating Compatible Code	309
B.1.2. Creating a New Project	310
B.1.3. Creating an External Definition File	310
B.1.4. Creating a Shared Library	310
B.1.5. Creating an External Table File	311
B.1.6. Setting the ANSYS_EXTERNAL_PATH Environment Variable	311
B.1.7. Using External Commands	312
B.1.8. Checking External Command Status	312
B.1.9. Resetting External Commands	312
C. User Material (UserMat) Subroutine Example	313
C.1. UserMat Example Description	
C.2. UserMat Example Input Data	
C.3. UserMat Example POST26 Output Results	
C.4. USERMAT.F List File for This Example	
C.5. Accessing Solution and Material Data	
Index	323
List of Figures	
4.1 Divides Format for 7 boffered Combine	0.0
4.1. Display Format for Z-buffered Graphics	
4.2. Example Display of a Graphics File	93
List of Tables	
6.1. ANSYS Exit Codes	193

Release 12.0 - © 2009 SAS IP, Inc. All rights reserved Contains proprietary and confidential information
nercase 12.0 9 2003 3/13/1/mes in rights reserved. Contains proprietary and confidential information
of ANSYS. Inc. and its subsidiaries and affiliates.

Preface

About the Programmer's Manual for ANSYS

The *Programmer's Manual for ANSYS* provides information about the various programming interfaces available to customers. This manual assumes that you have at least a basic knowledge of programming (a working knowledge of Fortran would be very helpful). The two part manual includes:

Part I - Guide to Interfacing with ANSYS

This guide describes a group of utilities as well as a set of Fortran routines that you can use to directly access the ANSYS database. You can also use these capabilities to access data in any of the binary files that ANSYS writes or uses.

Part II - Guide to ANSYS User Programmable Features

ANSYS provides a set of Fortran functions and routines that are available to extend or modify the program's capabilities. Using these routines requires relinking the ANSYS program, resulting in a custom version of ANSYS. ANSYS provides an external commands capability which you can use to create shared libraries available to ANSYS (either from ANSI standard C or Fortran). You can use this feature to add custom extensions to ANSYS without the need to rebuild the ANSYS executable.

In addition, you can find the ANSYS Parametric Design Language Guide as part of the ANSYS online documentation. This guide was designed for ANSYS users that have some programming skills and wish to tap the power of the ANSYS Parametric Design Language (APDL) to increase the productivity. APDL is a scripting language that is very similar to Fortran. The guide describes how to define parameters (variables), how to create macro programs using APDL, how to use APDL for simple user interaction, how to encrypt an APDL macro, and how to debug an APDL macro.

Note

The *Programmer's Manual for ANSYS* is offered solely as an aid, and does not undergo the same rigorous verification as the ANSYS product documentation set. Therefore, the *Programmer's Manual for ANSYS* is not considered to be part of the formal program specification as stated in your license agreement.

Release 12.0 - © 2009 SAS IP, Inc. All rights reserved Contains proprietary and confidential information
nercase 12.0 9 2003 3/13/1/mes in rights reserved. Contains proprietary and confidential information
of ANSYS. Inc. and its subsidiaries and affiliates.

Part l	l, Guide to I	nterfacin	g with Al	NSYS

Chapter 1: Format of Binary Data Files

The ANSYS program writes several binary files to store data during an analysis. These files are named Jobname.ext, where Jobname is the name of the analysis that caused the file to be generated and .ext is an extension indicating the type of data in the file.

The following Binary Data File topics are available in this chapter:

- 1.1. Understanding ANSYS Binary Files
- 1.2. Description of the Results File
- 1.3. Description of the Reduced Displacement File
- 1.4. Description of the Reduced Complex Displacement File
- 1.5. Description of the Modal Results File
- 1.6. Description of the Element Matrices File
- 1.7. Description of the Substructure Matrices File
- 1.8. Description of the Component Mode Synthesis Matrices (CMS) File
- 1.9. Description of the Full Stiffness-Mass File

1.1. Understanding ANSYS Binary Files

ANSYS-written binary files include the following:

- The following results files, in which the ANSYS program stores the results of solving finite element analysis problems:
 - Jobname . RST A structural or coupled-field analysis
 - Jobname . RTH A thermal analysis
 - Jobname . RMG A magnetic analysis
 - Jobname.RFL A FLOTRAN analysis
- The Jobname . MODE file, storing data related to a modal analysis
- The Jobname . RDSP file, storing data related to a reduced or mode-superposition transient analysis.
- The Jobname . RFRO file, storing data related to a reduced or mode-superposition harmonic analysis
- The Jobname. EMAT file, storing data related to element matrices
- The Jobname . SUB file, storing data related to substructure matrices
- The Jobname . FULL file, storing the full stiffness-mass matrix

The files listed above cover almost all users' needs, although there are others. For more information, see the *Basic Analysis Guide*.

1.1.1. Conventions Used to Describe Binary Files

In the information describing the binary file formats:

 Record ID is the identifier for this record. Not all records will have identifiers; they're indicated only for records whose record pointers are stored in a header.

- Type indicates what kind of information this record stores.
- Number of records indicates how many records of this description are found here.
- Record length indicates the number of items stored in the record.

In some record descriptions, actual variable names used may appear in the record contents area.

1.1.2. The Standard Header for ANSYS Binary Files

Each of the ANSYS program's binary files contains a standard, 100-integer file header that describes the file contents. The header contains the items listed below, always in the order shown:

Item 1	The file number
Item 2	The file format. This item has a value of 1 if the file is small format, -1 if large format.
Item 3	The time, in compact form (i.e., 130619 is 13:06:19)
Item 4	The date, in compact form (i.e., 20041023 is 10/23/2004)
Item 5	The units of measurement used. The value of this item is as follows:
	0 for user-defined units
	1 for SI units
	2 for CSG units
	3 for U. S. Customary units (feet)
	4 for U. S. Customary units (inches)
	5 for MKS units
	6 for MPA units
	• 7 for µMKS units
Item 10	The ANSYS release level in integer form ("X.X" in character form)
Item 11	The date of the ANSYS release
Items 12-14	The machine identifier in integer form (three four-character strings)
Items 15-16	The Jobname in integer form (two four-character strings)
Items 17-18	The ANSYS product name in integer form (two four-character strings)
Item 19	The ANSYS special version label in integer form (one four-character string)
Items 20-22	The user name in integer form (three four-character strings)
Items 23-25	The machine identifier in integer form (three four-character strings)
Item 26	The system record size
Item 27	The maximum file length
Item 28	The maximum record number
Items 31-38	The Jobname (eight four-character strings)
Items 41-60	The main analysis title in integer form (20 four-character strings)
Items 61-80	The first subtitle in integer form (20 four-character strings)
Item 95	The split point of the file
Items 97-98	LONGINT of the maximum file length

1.2. Description of the Results File

The next few pages describe the format of the ANSYS results file. (In the following tables, records with a record ID containing an asterisk (*) are those you can read and store into the ANSYS database via the **LDREAD** command.)

Note: The pointers in the solution data headers are relative, not absolute pointers. For example, the 12th item in the solution data header will be relative to a position in the Data Set Index (ptrESL = DSI(i) + ptrESL).

This section explains the contents of the results file; that is, those files with the following extensions:

```
.rfl .brfl
.rmg .brmg
.rst .brst
.rth .brth
.lnn
```

1.2.1. Nomenclature

A load case contains the results for an instance in an analysis. A load case is defined by a load step number and a substep number. A load case is also categorized by a cumulative iteration number and time (or frequency) values. A load case is identified by all three methods in the results file.

The results file does not have to contain all the load cases of an analysis.

A data set is used in this chapter to designate a load case.

For a complex analysis, there will be two data sets for each load case. The first data set contain the real solution and the second contains the imaginary solution.

1.2.2. Standard ANSYS File Header

See *The Standard Header for ANSYS Binary Files* (p. 4) for a description of this set. File number (Item 1) is 12.

1.2.3. Results File Format

```
*comdeck,fdresu
c *** copyright(c) 2008 SAS IP, Inc. All rights reserved.
c *** ansys, inc
      ******* description of results file *******
С
C
      --- used for the following files:
                .brfl
       .rfl
С
       .rma
                .brma
С
        .rst
                .brst
С
        .rth
                .brth
        .lnn(lxx)
     LONGINT
                     resufpL
                     resubk, resuut, resuLong, resuSpare(3)
     common /fdresu/ resufpL, resubk, resuut, resuLong, resuSpare
    ****** common variable descriptions *******
                file position on file resu
co resufpL
```

```
block number for file resu (usually 6)
CO
   resubk
                 file unit for file resu
   resuut
                                               (0 if not open) FUN12
CO
   resuLong
                 0, old 32 bit integer form
                                              1, 64 bit form (8.1)
    See fddesc for documentation of how binary files are stored.
С
      ****** file format *******
С
         recid tells the identifier for this record. Not all records will have
C
              identifiers -- they are only indicated for those records whose
C
              record pointers are stored in a header.
С
         type tells what kind of information is stored in this record:
С
С
              i - integer
              dp - double precision
C
              cmp - complex
С
         nrec tells how many records of this description are found here
С
         lrec tells how long the records are (how many items are stored)
C
c recid
                   nrec
                            lrec
                                     contents
           type
                                     standard ANSYS file header (see binhed for
С
                    1
                            100
                                     details of header contents)
C
            i
                    1
                             40
                                     .RST FILE HEADER
                                      fun12, maxn, nnod, resmax, numdof,
С
С
                                       maxe, nelm,
                                                     kan, nsets,ptrend,
                                     ptrDSI,ptrTIM,ptrLSP,ptrELM,ptrNOD,
С
С
                                     ptrGEO, ptrCYC, CMSflg, csEls, units,
C
                                    nSector,csCord,ptrEnd8,ptrEnd8,fsiflag,
                                      pmeth, noffst, eoffst, nTrans, ptrTRAN,
C
С
                                      kLong, csNds,cpxrst,extopt,nlgeom,
                                  AvailData, mmass,
                                                        0,
                                                                0,
C
                                    each item in header is described below:
С
                                     fun12 - unit number (resu file is 12)
C
                                            - maximum node number of the model
С
C
                                     nnod
                                            - the actual number of nodes used in
С
                                              the solution phase
                                     resmax - the maximum number of data sets
C
                                              allowed on the file (defaults to
C
                                              1000; minimum allowed is 10)
С
С
                                     numdof - number of DOFs per node
                                           - maximum element number of the
С
                                     maxe
                                              finite element model
С
                                           - number of finite elements
С
                                     nelm
                                            - analysis type
                                     kan
C
                                     nsets - number of data sets on the file
                                     ptrend - pointer to the end of the file
С
С
                                               (see ptrEnd8 in 23,24)
                                     ptrDSI - pointer to the data steps index
С
C
                                              table
                                     ptrTIM - pointer to the table of time values
С
С
                                              for a load step
                                     ptrLSP - pointer to the table of load step,
С
С
                                              substep, and cumulative iteration
С
                                              numbers
                                     ptrELM - pointer to the element equivalence
C
                                              table
                                     ptrNOD - pointer to the nodal equivalence
C
С
                                              table
                                     ptrGEO - pointer to the beginning of
С
                                              geometry information
C
С
                                     ptrCYC - pointer to the table of cyc sym
С
                                              nodal-diameters at each load step
                                     CMSflg - CMS results flag: 0-non cms, >0-cms
C
                                     csEls - Cyclic sym # eles in master sector
units - unit system used
С
C
```

```
= 0 - user defined units
C
С
                                              = 1 - SI (MKS)
С
                                              = 2 - CSG
                                              = 3 - U.S. Customary, using feet
С
С
                                              = 4 - U.S. Customary, using inches
С
                                              = 6 - MPA
                                              = 7 - uMKS
C
C
                                     nSector - number of sectors for cyclic sym
                                     csCord - Cyclic symmetry coordinate system
C
С
                                     ptrEnd8 23,24 64 bit file length
                                     fsiflag - FSI analyis flag
С
                                     pmeth - p-method analyis flag
C
                                     noffst - node offset used in writing file
C
С
                                     eoffst - elem offset used in writing file
                                     nTrans - number of SE transformation vects
C
                                     ptrTRAN - pointer to SE transformation vects
С
                                            - 1, 64 bit integer form
C
                                     kLong
                                             - Cyclic sym # nds in master sector
C
С
                                     cpxrst - complex results flag (0-no, 1-yes)
                                     extopt - mode extraction option
C
                                     nlgeom
                                             - NLGEOM key
С
С
                                  AvailData
                                             - bits indicating available data any
                                               where on the file; see resucm.inc
C
С
                                     mmass
                                             - number of missing mass resp. present
             i
                     1
                           numdof
                                     Degrees of freedom per node
C
С
                                     DOF reference numbers are:
          UX = 1, UY = 2, UZ = 3, ROTX = 4, ROTY = 5, ROTZ = 6, AX = 7, AY = 8
С
          AZ = 9, VX = 10, VY = 11, VZ = 12 *** 13-15 are spares ***, WARP=16
C
          HPEX=17, HDSP=18, PRES=19, TEMP=20, VOLT=21, MAG =22, ENKE=23, ENDS=24
С
          EMF =25, CURR=26, SP01=27, SP02=28, SP03=29, SP04=30, SP05=31, SP06=32
С
С
          TBOT=33, TE2 =34, TE3 =35, TE4 =36, TE5 =37, TE6 =38, TE7 =39, TE8 =40
          TE9 =41, TE10=42, TE11=43, TE12=44, TE13=45, TE14=46, TE15=47, TE16=48
C
          TE17=49, TE18=50, TE19=51, TE20=52, TE21=53, TE22=54, TE23=55, TE24=56
C
С
          TE25=57, TE26=58, TE27=59, TE28=60, TE29=61, TE30=62, TE31=63, TTOP=64
           **** 65-128 are spares ***
С
                                      (curdof(i),i=1,numdof)
C
С
    NOD
                     1
                            nnod
                                     Nodal equivalence table. This table equates
                                     the number used for storage to the actual
C
                                     node number
С
C
                                      (Back(i), i=1, nnod)
С
    ELM
                     1
                            nelm
                                     Element equivalence table. The ANSYS program
                                     stores all element data in the numerical
C
С
                                     order that the SOLUTION processor solves the
С
                                     elements. This table equates the order
                                     number used to the actual element number
C
    DSI
С
             i
                           2*resmax
                                     Data sets index table. This record contains
                                     the record pointers for the beginning of
C
                                     each data set. The first resmax records are
C
                                     the first 32 bits of the index, the second
C
С
                                     resmax records are the second 32 bits. To
С
                                     create the 64 bit pointer, use:
                                       LONGPTR = largeIntGet (first,second)
C
                                     Read the solution data header as follows:
С
С
                                       call bioBasePut (nblk,LONGPTR)
                                       loc = bioigr (nblk,12)
С
С
                                       call biord (nblk, loc,...
С
                                     The rest of the file reading continues to use
                                     the ptrXXX's that are in the headers.
C
    TIM
                                     Time/freq table. This record contains the
C
            dр
                     1
                           resmax
                                     time (or frequency) values for each data
С
С
                                     set.
С
    LSP
                           3*resmax
                                     Data set identifiers. This record contains
С
                                     the load step, substep, and cumulative
                                     iteration numbers for each data set.
C
             i
C
    CYC
                     1
                            resmax
                                     Cyclic symmetry harmonic index
```

```
25
                                     Substructure transformation vectors
    TRAN
             dр
                   nTran
C
    GEO
             i
                     1
                             40
                                    Geometry data header(was 20 in 32 bit vers)
C
С
                                          0, maxety, maxrl, ndnod, nelm,
                                    maxcsy,ptrETY,ptrREL,ptrNOD,ptrCSY,
С
C
                                    ptrEID.
                                              0, 0, 0, 0,
                                    ptrMAS, csysiz, elmsiz, etysiz, rlsiz,
C
С
                                          ptrETYL,
                                                       ptrRELL,
                                     ptrCSYL,
                                               ptrNODL,
                                                               ptrEIDL,
С
                                          ptrMASL, 0,
                                                               0,
C
                                                                  0,
                                                0,
                                                        0,
                                                               0,
C
                                   each item in header is described below:
C
С
                                           - position not used
                                    maxety - the maximum element type reference
C
С
                                             number in the model
                                    maxrl - the maximum real constant reference
C
                                             number in the model
С
                                    ndnod - the number of defined nodes in the
С
                                             model
C
                                           - the number of defined elements in
С
                                    nelm
С
                                              the model
                                    maxcsy - the maximum coordinate system
C
С
                                              reference number in the model
                                    ptrETY - pointer to the element type index
C
                                             table
C
                                    ptrREL - pointer to the real constant
С
С
                                             index table
С
                                    ptrNOD - pointer to the nodal point
С
                                              locations
                                    ptrCSY - pointer to the local coordinate
C
С
                                             system index table
С
                                    ptrEID - pointer to the element index
                                              table
С
                                    ptrMAS - pointer to the diagonal mass matrix
С
                                     csysiz - the number of items describing a
С
                                              local coordinate system (usually
C
С
                                     elmsiz - the maximum number of nodes that a
C
С
                                             defined element may have
                                     etysiz - the number of items describing an
С
                                           element type(=IELCSZ from echprm.inc)
C
                                    rlsiz - the maximum number of items
С
С
                                              defining a real constant (0, if no
                                             real constants are defined)
С
                                    ptrETYPL - 64 bit pointer to TYPE
С
                                    ptrRELL - 64 bit pointer to REAL
С
                                    ptrCSYL - 64 bit pointer to CSYS
C
                                    ptrNODL - 64 bit pointer to NODES
C
                                    ptrEIDL - 64 bit pointer to ELEMENTS
C
С
    ETY
             i
                     1
                           maxety
                                    The element types index table. This record
C
                                    contains record pointers for each element
С
                                     type description. (Relative to ptrETYPL
С
                                     for 64 bit version)
С
             i
                   numety etysiz
                                    Element type description. Each of these
С
                                    records is pointed to by a record pointer
С
                                    given in the record labeled ETY. See
                                    routines echprm and elccmt for a complete
C
                                    description of the items stored here.
C
                                    These items are typically stored into the
С
                                    IELC array, and are used to determine the
C
С
                                    element type characteristics at runtime.
С
                                    The following items are typically of
C
                                     interest:
С
                                     * Item 1
                                                  - element type reference number
                                     * Item 2
                                                  - element routine number
C
```

0 0 0 0 0 0 0 0 0 0 0 0 0					* Items 3-14 - element type option keys
C C C	REL	i	1	maxrl	Real constants index table. The record contains record pointers for each real constant set. (Relative to ptrRELL for 64 bit version)
0 0 0 0 0 0 0 0		dp	numrl	varies	Element real constant data. These records contain real constant data used for the elements. (See the ANSYS Elements Reference manual for values for a specific element.) Each of these records is pointed to by a record pointer given in the record labeled REL. The length of these records varies for each element type (actual length is returned from routine BINRD8).
0 0 0 0 0 0 0 0 0	CSY	i	1	maxcsy	Coordinate systems index table. This record contains the record pointers for each coordinate system set. The ANSYS program writes coordinate systems only if local coordinate systems were defined. If a local system was defined, the predefined global systems 1 to 2 also will be written. The global Cartesian system 0 will never be written. (Relative to ptrCSYSL for 64 bit version)
0 0 0 0		dp	numcsy	csysiz	Coordinate system description. These records contain coordinate system data for each coordinate system defined. Each of these records is pointed to by a record pointer given in the record labeled SYS.
C					The items stored in each record: * Items 1-9 are the transformation matrix.
					* Items 10-12 are the coordinate system origin (XC,YC,ZC). * Items 13-14 are the coordinate system parameters (PAR1, PAR2). * Items 16-18 are the angles used to define the coordinate system. * Items 19-20 are theta and phi singularity keys. * Item 21 is the coordinate system type (0, 1, 2, or 3). * Item 22 is the coordinate system reference number.
с с с	NOD	dp	1	7*ndnod	This group contains the node number and coordinates (in the order Node,X,Y,Z,THXY,THYZ,THZX) for each node. (32 bit version)
с с	NOD	dp	ndnod	7	(64 bit version) Node,X,Y,Z,THXY,THYZ,THZX for each node Nodes are in node number order

```
Element descriptions index table. This
С
    EID
                           nelm
С
                                     record contains the record pointers for each
                                     element description. (LONGINT (2*nelm) for
С
                                     64 bit version, relative to ptrEIDL).
С
C
                                     The order of the elements is the same as
                                     the order in the element equivalence table.
С
             i
                   nelm 10+nodelm Element descriptions. Each of these records
C
С
                                     is pointed to by a record pointer given in
                                     the record labeled EID. The length of these
С
                                     records varies for each element (actual
C
                                     length is returned from routine BINRD8).
С
С
                                     nodelm shown here is the number of nodes for
                                     this element. Its value is defined in the
C
                                     element type description record.
С
                                     The items stored in each record:
С
С
                                        mat, type, real, secnum, esys,
                                      death, solidm, shape, elnum,
C
                                      NODES
С
                                    each item is described below:
C
С
                                             - material reference number
                                      mat
                                             - element type number
C
                                      type
С
                                             - real constant reference number
                                      secnum - section number
C
                                           - element coordinate system
                                      esys
C
                                      death - death flag
С
                                               = 0 - alive
С
                                               = 1 - dead
С
С
                                      solidm - solid model reference
                                      shape - coded shape key
C
                                      elnum - element number
С
С
                                      NODES - node numbers defining the element.
                                               (See the ANSYS Elements Reference
C
                                               for nodal order of an element).
С
                     1 nnod*numdof Diagonal mass matrix.
C
    MAS
            dp
         The solution information is stored starting at this point in the file.
C
         The remaining records on the file are repeated as a group nsets times
C
         (once for each data set). Item nsets is defined in the file header.
С
         Each set of data is pointed to by a record pointer given in the record
C
         labeled DSI.
С
                             200
                                      Solution data header. (was 100 in 32 bit)
C
                                     pv3num, nelm, nnod, mask, itime,
C
С
                                       iter, ncumit,
                                                     nrf,cs_LSC, nmast,
С
                                     ptrNSL,ptrESL, ptrRF,ptrMST, ptrBC,
C
                                     rxtrap, mode, isym, kcmplx, numdof,
С
                                     positions 51-70 - title,
С
                                     positions 71-90 - stitle1,
С
С
                                     dbmtim, dbmdat, dbfncl, soltim, soldat,
С
                                     ptrOND,ptrOEL,nfldof,ptrEXA,ptrEXT
С
                                     101-102 ptrEXA (was in 99)
                                     103-104 ptrEXT (was in 100)
                                     105-106 ptrNSL (was in 11)
C
С
                                     107-108
                                              ptrRF (was in 13)
                                     109-110
                                              ptrMST (was in 14)
С
                                     111-112 ptrBC (was in 15)
C
                                     113-114 ptrTRF (was in EXT 125)
С
С
                                     115-116 ptrOND (was in 96)
                                     117-118
                                              ptrOEL (was in 97)
C
С
                                     119-120
                                              ptrESL (was in 12)
                                     131-132 ptrVSL (was in EXT 196)
C
```

С	133-134	ptrASL (was in EXT 197)
С		numRotCmp
С	141-142	
C		nNodStr
С		ptrNODSTR
C	147	AvailData
С	each item	in header is described below:
С	pv3num -	current solu set number
C		number of elements
С		number of nodes
C	mask -	bitmask for the existence of several records. If a bit is set
c		here, it indicates that the
c		corresponding record exists on the
C		file.
С		The items in the bitmask that
C		correspond to each record are shown
C		in the record descriptions below.
С		loadstep
C		iteration number cumulative iteration number
c c		number of reaction forces
c		cyclic symmetry count of the
C		load step for this SOLVE
С	nmast -	number of masters
С	_	pointer to nodal solution
C	_	pointer to element solution
С	-	pointer to reaction forces
C	_	pointer to the masters
C C	_	pointer to the boundary conditions key to extrapolate integration
c	IXCIAP	point results to nodes
C		= 0 - move
С		= 1 - extrapolate unless active
C		non-linear
C	_	= 2 - extrapolate always
C	mode -	mode number of harmonic loading
C		<pre>(for cyclic symmetry: this is cs_LSF = first load step for this SOLVE)</pre>
c c	isym -	symmetry for harmonic loading
C	-	(for cyclic symmetry: this is cs_LSL
С		= last load step for this SOLVE)
C	kcmplx -	complex key
C		= 0 - real
C		= 1 - imaginary
C	numaoi -	number of DOFs/nodes for this data set
c	DOFS -	DOF/node reference numbers (numdof
C		values)
С	title -	main title (in integer form)
C		1st subtitle (in integer form)
С	dbmtim -	time (in compact form) when the
C	dbred - +	database was last modified
C	abiliat -	date (in compact form) when the database was last modified
c	dbfncl -	number of times that the database
C		was modified
С	soltim -	time (in compact form) when the
С		solution for this data set was done
С	soldat -	date (in compact form) when the
C	n+ vOM	solution for this data set was done
C C	brionn -	pointer to the ordered node list (load case files only)
c	ptrOEL -	pointer to the ordered element list
c	r	(load case files only)
С	nfldof -	number of extra Flotran DOFs/nodes
С		for this data set
С	ptrEXA -	pointer to header extension for
C		FLOTRAN DOF/extra DOF list.
С	hrtfyl -	pointer to header extension

```
numRotCmp - number of rotating components
С
С
                                    ptrRCM - pointer to RCM
С
                                    nNodStr - 0, no nodal component stresses
                                               1, one set (TOP for shells)
С
                                               2, two sets (TOP,BOT for shells)
С
С
                                               3, three sets (TOP,BOT,MID)
                                    ptrNODSTR - pointer to nodal component str
С
                                     AvailData - bits indicating available data
                                                 in this data set; see resucm.inc
C
С
                                     Note: ptrXXX are relative to ptrDSI
С
                                    Solution header - double precision data
С
            dр
                                     timfrq,lfacto,lfactn,cptime, tref,
C
С
                                      tunif, tbulk, volbase, tstep, 0.0,
C
                                      velocity-acceleration-center of gravity
                                      terms (positions 11-28)
C
                                      if pmeth=0: load data (positions 51-100)
                                      if pmeth=1: p convergence values
C
                                       (positions 31-100)
С
                                    each item is described below:
C
С
                                      timfrq - time value (or frequency value,
                                               for a modal or harmonic analysis)
C
С
                                      lfacto - the "old" load factor (used in
                                               ramping a load between old and new
C
                                               values)
C
                                      lfactn - the "new" load factor
С
                                      cptime - elapsed cpu time (in seconds)
С
С
                                             - the reference temperature
                                      tunif - the uniform temperature
С
                                      tbulk - Bulk temp for FLOTRAN film coefs.
C
С
                                      VolBase - Initial total volume for VOF
С
                                      tstep - Time Step size for FLOTRAN analysis
                                             - position not used
C
С
                                      positions 11-13 -Linear acceleration terms
                                     positions 14-16 - Angular velocity
C
                                     positions 17-19 - Angular acceleration
C
C
                                     positions 20-22 - Angular velocity about
С
                                                        the center of gravity
С
                                      positions 23-25 - Angular acceleration
                                                        about the center of
C
С
                                                        gravity
С
                                     positions 26-28 - Center of gravity
                                                        location
C
                                      if pmeth=1:
С
C
                                         positions 31-100 - P convergence values
                                      if pmeth=0:
C
                                         positions 51-100 - Load data
C
С
                                         position 53 - Convergence key (if 1,
С
                                                       substep converged)
С
    EXA
             i
                     1
                               64
                                     Header extension (if ptrEXA=ptrEXT, then
                                                        ptrEXA is unused.)
С
С
                                      positions
                                                  1-32
                                                        - current extra Flotran
С
                                                          DOFs for this set
С
                                      positions 33-64 - current extra Flotran
                                                          DOF labels for this set
                                     Extra Flotran DOF reference numbers are:
С
          DENS= 1, VISC= 2, EVIS= 3, COND= 4, ECON= 5, LMD1= 6, LMD2= 7, LMD3= 8
С
C
          LMD4= 9, LMD5=10, LMD6=11, EMD1=12, EMD2=13, EMD3=14, EMD4=15, EMD5=16
          EMD6=17, PTOT=18, TTOT=19, PCOE=20, MACH=21, STRM=22, HFLU=23, HFLM=24
C
С
          YPLU=25, TAUW=26, SPHT=27, CMUV=28
                   ********** 29-32 are spares ****************
C
                     1
                              200
C
    EXT
             i
                                     Header extension
```

С	positions 1-32 - current DOF for this
c c	result set positions 33-64 - current DOF labels for
С	this result set
c c	positions 65-84 - The third title, in integer form
c	positions 85-104 - The fourth title, in integer form
С	positions 105-124 - The fifth title, in integer form
c c	position 125 - ptrTRF- pointer to FLOTRAN
C	previous time step DOF vals
c c	position 126 - trnvar- #dof in FLOTRAN prev time st DOF vals.
С	(Note 2 old steps saved,
С	thus #DP is 2*trnvar*nNode)
C C	position 127 - numvdof, number of velocity items per node (ANSYS
C	transient)
С	position 128 - numadof, number of
C	acceleration items per
C C	node (ANSYS transient) position 131-133 - position of velocity
C	in DOF record
С	(ANSYS transient)
C	position 134-136 - position of acceleration
C C	in DOF record (ANSYS transient)
C	position 137-142 - velocity and
С	acceleration labels
c c	(ANSYS transient) position 143 - number of stress items
c	(6 or 11); a -11 indicates
С	to use principles directly
С	and not recompute (for PSD)
C C	position 144-146 - position of rotational velocity in DOF record
C	(ANSYS transient)
С	position 147-149 - position of rotational
С	accel. in DOF record
C C	(ANSYS transient) position 150-155 - rotational velocity and
C	acceleration labels
С	(ANSYS transient)
С	position 160 - ptrDMI (J Integral results)
c c	position 161 - nContours if pmeth=1:
C	positions 164-200 - p convergence specs
t NOT	of the DOT relation for each and in the mid-
c * NSL dp 1 nnod*Sumd	of The DOF solution for each node in the nodal coordinate system. The DOF order is the
С	same as shown above in the DOF number
С	reference table. The nodal order is the
C	same order given above in the nodal
C C	equivalence table. If a DOF for a node isn't valid, a value of 2.0**100 is used.
c	Note 1: Sumdof = numdof + nfldof.
C	Note 2: If, upon reading of this record,
c c	there is less than nnod*Sumdof items in the record, then only a selected set of nodes
c	were output. Another record follows
С	(integer, less than nnod long) which
C	contains the list of nodes for which DOF
C C	solutions are available. (bit 10 (PDBN) in mask)
_	of The velocity solution for each node in the
C C	nodal coordinate system. The description for the DOF solution above also applies
C	here.
С	ANSYS transient. (bit 27 (PDVEL) in mask)

0 0 0 0	ASL	dp	1 nr	nod*numadof	The acceleration solution for each node in the nodal coordinate system. The description for the DOF solution above also applies here. ANSYS transient. (bit 28 (PDACC) in mask)
0 0 0 0 0	RF	i	1	nrf	Reaction force DOFs. This index is calculated as (N-1)*numdof+DOF, where N is the position number of the node in the nodal equivalence table, and DOF is the DOF reference number. (bit 11 (PDBR) in mask)
c *	*	dp	1	nrf	Reaction forces. The force values are ordered according to the DOF order shown above in the DOF number reference table. (bit 11 (PDBR) in mask)
0 0 0 0 0	MST	i	1	nmast	Master DOF list. This index is calculated as (N-1)*numdof+DOF, where N is the position number of the node in the nodal equivalence table, and DOF is the DOF reference number. (bit 4 in mask)
	BC	i	1	40	Boundary condition index table. (bit 23 (PDBBC) in mask) numdis,ptrDIX,ptrDIS,numfor,ptrFIX, ptrFOR,format, 0
0 0 0 0 0 0 0	DIX	i	1	numdis	if format == 0> Nodal constraint DOF. This index is calculated as N*32+DOF, where N is the node number and DOF is the DOF reference number. Values are in the same order as the DOF number reference table. if format == 1> Nodal constraint node numbers.
0 0 0		i	1	numdis	<pre>if format == 0> does not exist. if format == 1> Nodal constraint DOF. Values are in the same order as the DOF number reference table.</pre>
C C C	DIS	dp	1	4*numdis	Nodal constraints. This record contains present and previous values (real and imaginary) of the nodal constraints at each DOF.

0 0 0 0 0 0	FIX	i	1	numfor	if format == 0> Nodal input force DOFs. This index is calculated as N*32+DOF, where N is the node number and DOF is the DOF reference number. Values are in the same order as the DOF number reference table.
с с					<pre>if format == 1> Nodal input force node numbers.</pre>
С		i	1	numfor	if format == 0> does not exist.
с с с					if format == 1> Nodal input force DOF. Values are in the same order as the DOF number reference table.
с с	FOR	dp	1	4*numfor	Nodal forces. This record contains present and previous values (real and imaginary) of the nodal input force loadings at each DOF.
С С	TRF	dp	1	28*nnod	Two displacement result sets for transient solution in FLOTRAN (bit 24 (PDTRFL) in mask)
C C	OND	i	1	nnod	Ordered node list. This record exists for a load case file only.
C C	OEL	i	1	nelm	Ordered element list. This record exists for a load case file only.
0 0 0 0	ESL	i	1	2*nelm	Element solutions index table. This record contains pointers to each element solution. The order of the elements is the same as the order in the element equivalence table. (bit 12 (PDBE) in mask)
C C	RCM	dp	1	6*numRotCmp	Angular velocities (3) and angular accelerations (3) of components.
C C	DMI	dp	1	3+nContours	Crack ID, Contour ID, TipNode, J Integral values
c c c		at this prepeated	oint i as a g	n the file.	r each individual element is stored starting The next 23 records on the file are mes (once for each element). Item nelm is
С		i	1	25	Individual element index table.
С					ptrEMS,ptrENF,ptrENS,ptrENG,ptrEGR,
С					ptrEEL,ptrEPL,ptrECR,ptrETH,ptrEUL,
C					<pre>ptrEFX,ptrELF,ptrEMN,ptrECD,ptrENL,</pre>
С					ptrEHC,ptrEPT,ptrESF, 0,ptrETB,
C					<pre>ptrECT,ptrEXY,ptrEBA,ptrESV,ptrMNL (Relative to ptrESL for 64 bit version)</pre>
С				е	ach item is described below:
С					ptrEMS - pointer to misc. data
С					ptrENF - pointer to nodal forces
С					ptrENS - pointer to nodal stresses
C					ptrENG - pointer to volume and energies
C					ptrEGR - pointer to nodal gradients
C C					ptrEEL - pointer to elastic strains ptrEPL - pointer to plastic strains
C					ptrECR - pointer to creep strains
C					ptrETH - pointer to thermal strains
С					ptrEUL - pointer to euler angles
С					ptrEFX - pointer to nodal fluxes
С					ptrELF - pointer to local forces

С					ptrEMN - pointer to misc. non-sum values
C					ptrECD - pointer to element current densities
С					ptrENL - pointer to nodal nonlinear data
С					ptrEHC - pointer to calculated heat
C					generations
С					ptrEPT - pointer to element temperatures
C C					ptrESF - pointer to element surface stresses
C					ptrETB - pointer to ETABLE items(post1 only
C					ptrECT - pointer to contact data
C					ptrEXY - pointer to integration point
C					locations
C					ptrEBA - pointer to back stresses ptrESV - pointer to state variables
C C					ptrMNL - pointer to material nonlinear record
Ü					Former to material nonringar record
C					Note! If ptrXXX is negative, then all
C					ptrXXX items are zero and are not on
С					the file.
С	EMS	dр	1	varies	Element summable miscellaneous data. The
С	END	αр	_	varies	contents and number of data items is
C					element-dependent. For a list of what's
C					available, see the SMISC item in the
C					description of the ETABLE command in the
С					ANSYS Commands Reference.
С	ENF	dр	1	varies	Element nodal forces. This record contains
С		-			the forces at each node, in the same DOF
С					order as the DOF number reference table.
С					For static, damping, and inertia forces, a
C C					set of forces will be repeated (as appropriate). Number of data items stored
С					in this record can be calculated as
C					follows: nodfor*NDOF*M, where NDOF is the
С					number of DOFs/node for this element,
С					nodfor is the number of nodes per element
C					having nodal forces (defined in element
C					type description record), and M may be 1, 2, or 3. For a static analysis, M=1 only.
С					For a transient analysis, M can be 1, 2,
С					or 3.
		,	_		
C C	ENS	dp	1	varies	Element nodal component stresses. This record contains the stresses at each corner
C					node, in the order SX,SY,SZ,SXY,SYZ,SXZ,S1,
C					S2,S3,SI,SIGE. Nodal order corresponds to
С					the connectivity defined in the element
С					description. Stresses can be nodal values
C C					extrapolated from the integration points or values at the integration points moved to
C					the nodes. If an element is nonlinear,
C					integration point values always will be
С					written. (See item rxtrap in the solution
С					header for the setting.) An element is
C					considered nonlinear when either plastic,
C C					creep, or swelling strains are present.
C					Definition of common terms referred here
С					and in subsequent EEL, EPL, ECR, ETH,
С					ENL, EUL and EPT sections:
C					nodatr - number of nodes per element
C					nodstr - number of nodes per element having stresses, strains, etc.
C					For higher-order elements, nodstr
C					equals to the number of corner
С					nodes (e.g., for 20-noded SOLID186,
C					nodstr = 8).
~					nodfor - number of modes nor allower
C					nodfor - number of nodes per element
с с					nodfor - number of nodes per element having nodal forces, etc. ncomp - number of solution items per node

```
ncomp = 11 for ENS record
C
С
                                                          7 for EEL record
С
                                                          7 for EPL record
                                                          7 for ECR record
С
                                                          8 for ETH record
С
                                                         10 for ENL record
С
                                              - number of layers in layered
С
C
                                                elements
C
С
                                       * For solid elements or layered solid elements
                                        with KEYOPT(8)=0, the record contains
С
                                        stresses at each corner node, and
C
                                        the number of items in this record is
С
С
                                        nodstr*ncomp.
                                       For shell elements or layered shell elements
C
                                        with KEYOPT(8)=0, the record contains
С
C
                                        stresses at each corner node (first
                                        at the bottom shell surface, then the top
C
С
                                        surface), and the number of items in this
                                        record is 2*nodstr*ncomp.
C
                                        For layered elements SHELL91, SHELL99,
С
С
                                        SOLID46, and SOLID191 with KEYOPT(8) = 0,
                                        if failure criteria were used, the record
C
С
                                        contains additional stresses at each corner
С
                                        nodes (first the bottom surface, then the
                                        top surface) of the layer with the largest
C
С
                                        failure criteria. Therefore, the total number
                                        of items is 4*nodstr*ncomp for SHELL91 and
C
                                        SHELL99, and 2*nodstr*ncomp for SOLID46 and
С
                                        SOLID191.
С
                                      * For layered elements (with KEYOPT(8)=1),
C
С
                                        stresses for each layer are at each
C
                                        corner node (first at the bottom surface, then
                                        at the top surface), and the number of
C
С
                                        items in this record is NL*2*nodstr*ncomp for
                                        layered shells and NL*nodstr*ncomp for
C
                                        layered solid elements.
C
                                       * For layered shell elements with KEYOPT(8)=2,
С
С
                                        the record contains stresses for each layer
                                        at each corner node (first at the bottom
C
                                        surface, then the top, and finally the middle
С
C
                                        surface). Therefore, the number of items
С
                                        in this record is NL*3*nodstr*ncomp.
                                      * For layered membrane elements (SHELL181,
C
                                        SHELL281, SHELL208, and SHELL209 with
C
                                        KEYOPT(1)=1 and KEYOPT(8)=1), the record
С
С
                                        contains stresses for each layer at each
                                        corner node, and the number of items in
C
                                        this record is NL*nodstr*ncomp.
С
                                       * For beam elements, the contents and number
С
C
                                        of data items is element-dependent. See
                                        the Output Data section for the particular
C
                                        element in the ANSYS Elements Reference.
C
С
C
    ENG
            dp
                     1
                             11
                                      Element volume and energies.
С
                                      volume, senergy, aenergy, kenergy, coenergy,
С
                                      incenergy, 0.0, 0.0, thenergy, 0.0, 0.0
С
                                    each item is described below:
c
                                      volume
                                               - element volume
                                      senergy - element energy associated with
C
                                                 the stiffness matrix
C
                                      aenergy - artificial hourglass energy
С
                                      kenergy - kinetic energy
С
                                      coenergy - co-energy (magnetics)
C
                                      incenergy- incremental energy (magnetics)
С
С
                                      0.0
                                               - position not used
                                      0.0
                                               - position not used
C
С
                                      thenergy - thermal dissipation energy
C
                                                (see ThermMat, shell131/132 only)
```

C C					0.0 - position not used 0.0 - position not used
	EGR	dp	1	varies	Element nodal field gradients. This record contains the gradients at each corner node in the order X,Y,Z. Nodal order corresponds to the connectivity defined in the element description. If this is a coupled-field analysis, the data is stored in the following order (as available): fluid, thermal (TEMP), electric (VOLT), and magnetic (AZ). Gradients can be nodal values extrapolated from the integration points or values at the integration points moved to the nodes. See item rxtrap in the solution header for the setting. The number of items in this record is nodstr*3*N, where N can be 1, 2, 3, or 4 (depending on the coupled-field conditions). NOTE: nodstr is defined in the element type description record.
	EEL	dp	1	varies	Element nodal component elastic strains. This record contains strains in the order X,Y,Z,XY,YZ,XZ,EQV. Elastic strains can be can be nodal values extrapolated from the integration points or values at the integration points moved to the nodes. If an element is nonlinear, integration point values always will be written. See item rxtrap in the solution header for the setting. An element is considered nonlinear when either plastic, creep, or swelling strains are present. For beam elements, see item LEPEL in the description in the Output Data section for the particular element in the ANSYS Elements Reference. NOTE: See ENS record section for more details on record content and length.
000000000000	EPL	dp	1	varies	Element nodal component plastic strains. This record contains strains in the order X,Y,Z,XY,YZ,XZ,EQV. Plastic strains are always values at the integration points moved to the nodes. For beam elements, see item LEPPL in the Output Data section for the particular element in the ANSYS Elements Reference. NOTE: See ENS record section for more details on record content and length.
0 0 0 0 0 0 0 0 0 0	ECR	dp	1	varies	Element nodal component creep strains. This record contains strains in the order X,Y,Z,XY,YZ,XZ,EQV. Creep strains are always values at the integration points moved to the nodes. For beam elements, see item LEPCR in the Output Data section for the particular element in the ANSYS Elements Reference. NOTE: See ENS record section for more details on record content and length.
С	ETH	dp	1	varies	Element nodal component thermal strains.

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0					This record contains strains in the order X,Y,Z,XY,YZ,XZ,EQV plus the element swelling strain. Thermal strains can be nodal values extrapolated from the integration points or values at the integration points moved to the nodes. If the element in nonlinear, integration point data always will be written. (An element is considered nonlinear when either plastic, creep, or swelling strains are present.) See item rxtrap in the solution header for the setting. For beam elements, see item LEPTH in the description of the Output Data section for the particular element in the ANSYS Elements Reference. NOTE: See ENS record section for more details on record content and length.
C C	EUL	dp	1	varies	Element Euler angles. This record contains the Euler angles (THXY,THYZ,THZX).
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0					<pre>* For lower-order elements, angles are at the centroid and the number of items in this record is 3. * For higher-order elements, angles are at each corner node and the number of items in this record is nodstr*3. * For layered shells, higher-order layered solid elements, and layered SOLSH190 and SOLID185, angles are at each corner node, plus the layer orientation angle for each layer. The number of items in this record is (nodstr*3)+NL. * For other lower-order layered solid elements, Euler angles are at the centroid, plus the layer orientation angle for each layer. Therefore, the number of items in this record is 3 + NL. NOTE: See ENS record section for definition of terms NL and nodstr.</pre>
	EFX	dp	1	varies	Element nodal field fluxes. This record contains the fluxes at each corner node in the order X,Y,Z. If this is a coupled-field analysis, the flux data is stored in the following order: thermal, electric, magnetic. Nodal order corresponds to the connectivity defined in the element description. Fluxes can be nodal values extrapolated from the integration points or values at the integration points moved to the nodes. See item rxtrap in the solution header for the setting. The number of items in this record is nodstr*3*N, where N can be 1, 2, or 3 depending on the coupled-field conditions. NOTE: nodstr is defined in the element type description record.
C * C C C C C	ELF	dp	1	varies	Element nodal coupled-field forces. This record lists the forces at each node in the order X,Y,Z. For most elements, the number of items in this record is nodfor*3. However, for the PLANE53 element, the number of items in this record is either nodfor*3 or nodstr*3. (See the description of KEYOPT(7) for PLANE53 in the ANSYS

С С С				Elements Reference.) NOTE: nodfor and nodstr are defined in the element type description record.
c c				NOTE: nodstr is defined in the element type description record.
C EMN C C C	dp	1	varies	Element nonsummable miscellaneous data. The contents and number data items for this record is element-dependent. See the description for item NMISC of the ETABLE command in the ANSYS Commands Reference.
c * ECD c	dp	1	3	Element current densities. This record contains the calculated current densities in the order X,Y,Z.
C ENL C C C	dp	1	varies	Element nodal nonlinear data. This record stores nonlinear data at each corner node in the order SEPL, SRAT, HPRES, EPEQ, PSV or CREQ, PLWK, CRWK, and ELENG followed by 2 spares.
				each item is described below: SEPL - equivalent stress parameter SRAT - stress ratio HPRES - hydrostatic pressure EPEQ - accumulated equivalent plastic strain PSV - plastic state variable CREQ - accumulated equivalent creep strain. Applies to current technology element types 180,181,182,183,185,186, 187,188,189,208,209,265, 281,288,289,290 PLWK - plastic strain energy density(work) CRWK - creep strain energy density (work) ELENG - elestic strain energy density * See ENS record section for details on solid and shell elements. * For beam elements, the contents and number of data items in this record is element-dependent. See the description of item NLIN in the Output Data section for the particular element in the ANSYS Elements Reference.
c * EHC	dp	1	1	Element heat generation. This record stores the calculated heat generation.
c EPT	dp	1	varies	Element structural nodal temperatures.
				* For solid elements and SHELL41, the record contains nodal temperatures at each node and the number of items in this record is nodfor. * For shell elements, except SHELL41 and SHELL91, the record contains nodal temperatures at each corner node for the top surface and the bottom surface. The number of items in this record is nodstr*2. * For SHELL91 and SOLID191, the record contains nodal temperatures at each corner node for the bottom of the bottom layer, and each succeeding interlayer surface up to the top of the top layer. The number of items in this record is (NL+1)*nodstr. * For layered shell elements SHELL181,

```
SHELL281, SHELL208, SHELL209, and layered
C
                                        solid elements SOLID185, SOLID186,
С
С
                                        and SOLSH190, the record contains
                                        temperatures for each layer at each
С
                                        corner node (first at the bottom layer
С
                                        surface, then the top). Therefore, the number
С
                                        of items in this record is NL*2*nodstr for
C
C
                                        layered shells and NL*nodstr for layered
                                        solid elements.
C
С
                                       For layered membrane elements (SHELL181,
                                        SHELL281, SHELL208, and SHELL209 with
С
                                        KEYOPT(1)=1), the record contains
C
                                        temperatures for each layer at each
С
С
                                        corner node. Therefore, the number of items
C
                                        in this record is NL*nodstr.
                                      * For beam elements, the contents and
С
C
                                       number of data items in this record is
                                        element-dependent. See the description
C
C
                                        of item LBFE in the Output Data section
                                        for the particular element in the ANSYS
C
                                        Elements Reference.
С
                                     NOTE: See ENS record section for definition
C
                                            of terms NL, nodstr, and nodfor.
С
    ESF
                          nsurf*19
C
            αb
                     1
                                     Element surface stresses. The
С
                                      length of this record is nsurf*19 where
                                     nsurf is the number of surfaces that have
С
                                     surface stress information. The stress
С
                                      information is simply repeated in the
C
                                      format shown below for each surface.
С
С
                                      * For 2d elements:
С
                                      facenm, area, temp, press, eppar,
С
                                              epz, 0.0d0, spar, sper,
                                          sz, 0.0d0, 0.0d0, 0.0d0,
C
                                          s2,
                                                 s3, sint, seqv
С
                                      * For 3d elements:
C
C
                                      facenm, area, temp, press,
                                                                     epx,
C
                                         epv,
                                                epz, epxy,
                                                               sx,
                                                                      sv,
С
                                          sz,
                                                sxy, 0.0d0, 0.0d0,
                                                                      s1,
                                                s3, sint, seqv
                                          s2,
C
С
                                      * For axisymmetric elements:
                                      facenm, area, temp, press, eppar,
С
                                       epper,
                                              epz, epsh, spar,
                                                                   sper,
                                          sz, 0.0d0, 0.0d0,
                                                             ssh.
                                                                      s1.
C
                                          s2,
                                                s3, sint, seqv
С
                                    each item is described below:
                                     facenm - face number
C
                                     area - face area
С
С
                                     temp
                                             - face temperature
                                     press - face pressure
С
С
                                     epx
                                             - strain parallel to face
                                             - strain parallel to face
С
                                      еру
                                             - strain perpendicular to face
C
                                     epz
                                             - shear strain
C
                                      eppar - strain parallel to face
C
                                            - strain perpendicular to face
С
                                     epper
                                             - torsion shear strain
С
                                      epsh
                                             - stress parallel to face
C
                                     sx
С
                                             - stress parallel to face
                                     sy
С
                                             - stress perpendicular to face
                                      SZ
C
                                             - shear stress
                                      SXY
С
                                      spar
                                             - stress parallel to face
                                             - stress perpendicular to face
C
                                      sper
```

```
- torsion shear stress
С
                                      ssh
С
                                             - S(1)
С
                                      s2
                                             - S(2)
                                             - S(3)
С
                                      s3
                                             - S(INT)
С
                                      sint
                                             - S(EQV)
C
                                      seqv
                                      0.0d0 - position not used
C
С
    EXY
            dр
                          varies
                                      Element integration point coordinates
                                      The length of the record is numint*3, where
С
                                      numint is the number of integration points.
C
С
                                      Even two-dimensional elements use the 3.
С
                                      They are output only if requested with the
                                      OUTRES, loci command.
C
С
                                      Applicable only to legacy element types
C
                                      2,42,45,82,92,95, and current technology
                                      element types 180,181,182,183,185,186,187,
C
С
                                      188,189,208,209,265,281,288,289,290
    EBA
                          varies
                                      Element structural nodal back stresses
            dр
C
С
                                      Record has the same form as the plastic
                                      strains. They are output if the form of
C
С
                                      plasticity is kinematic hardening and the
С
                                      plastic strains are requested.
С
                                      Applicable only to legacy element types
С
                                      2,42,45,82,92,95, and current technology
C
                                      element types 180,181,182,183,185,186,187,
                                      188,189,208,209,265,281,288,289,290
C
С
    ESV
            dp
                          varies
                                      Element state variable record. Exists only
                                      if written by user in usermat or usercreep.
С
    MNL
                     1
                                      Material nonlinear record.
                           varies
C
    records marked with * to the left of the record id can be read and stored
C
    into database with "ldread" command.
C
c *** Nodal Component Stresses (unused)
C
  NODSTR
           ďρ
                     1
                          6*nnod
                                      Nodal component stresses (TOP for shells)
С
                                       (nNodStr > 0)
            dр
                          6*nnod
                                      BOT nodal component stresses for shells
C
C
                                       (nNodStr > 1)
            dр
                           6*nnod
                                      MID nodal component stresses for shells
С
                                       (nNodStr > 2)
```

1.3. Description of the Reduced Displacement File

This section explains the content of the reduced displacement file (jobname.rdsp).

1.3.1. Standard ANSYS File Header

See *The Standard Header for ANSYS Binary Files* (p. 4) for a description of this set. File number (Item 1) is 10

1.3.2. RDSP File Format

```
character*8 RDSPNM
      parameter (RDSPNM='rdsp
      LONGINT
                      rdspfpL, rdspfp
      integer rdspbk, rdsput
common /fdrdsp/ rdspfpL, rdspbk, rdsput
      equivalence (rdspfp,rdspfpL)
c write: lnfrcl,lnfrin,lnfrwr
   write: rdtrcl,rdtrin,rdtrwr
   read:
           rdtrs
    ****** common variable descriptions *******
                 file position on file rdsp
co rdspfpL
    rdspbk
                 block number for file rdsp
CO
                 file unit for file rdsp
    rdsput
    See fddesc for documentation of how binary files are stored.
C
C
      ****** file format ******
C
         recid tells the identifier for this record. Not all records will have
С
              identifiers -- they are only indicated for those records whose
C
              record pointers are stored in the second file header.
C
         type tells what kind of information is stored in this record:
C
C
              i - integer
              dp - double precision
C
              cmp - complex
С
         nrec tells how many records of this description are found here
C
         lrec tells how long the records are (how many items are stored)
c recid
           type
                   nrec
                           lrec
                                     contents
С
            i
                    1
                           100
                                     standard ANSYS file header (see binhed for
                                     details of header contents)
C
            i
                    1
                            40
                                   .RDSP FILE HEADER
С
C
                                    fun10, nmrow, nmatrx, nmode, numdof,
С
                                    maxn, wfmax, lenbac, ngaps, ncumit,
С
                                     kan,
                                           nres, ndva, naload, DSPfmt,
                                                              0,
                                       0,
                                              0,modlstp,
                                                                        0.
C
                                  ptrDOF, ptrDNC, ptrSTF, ptrMAS, ptrDMP,
С
С
                                  ptrFRQ, ptrDSP,ptrSTFh,ptrMASh,ptrDMPh,
                                  ptrFRQh,ptrDSPh, ptrDVA,ptrDVAh,
                                                                        0,
С
                                        Ο,
                                                Ο,
                                                                        0
С
                                                        0,
C
                                  each item in header is described below:
                                  fun10 - unit number (rdsp file is 10)
C
                                  nmrow - number of rows/columns in matrices
С
                                  nmatrx - number of reduced matrices on the
C
C
                                           file
                                  nmode - number of modes extracted during
С
С
                                           modal analysis (or nmrow if reduced
                                           method)
С
С
                                  numdof - number of dofs per node
                                         - maximum node number
С
                                  maxn
C
                                  wfmax - maximum wavefront
                                  lenbac - number of nodes
C
                                  ngaps - number of gaps
C
С
                                  ncumit - total number of iterations done
                                           during analysis
С
                                          - analysis type
C
                                  kan
                                           = 5 for reduced transient analysis
С
С
                                  nres
                                         - number of residual vectors used
C
                                  modlstp- multiple load step key
                                         - length of DVA
С
                                  ndva
                                  naload - number of available load vectors
C
```

```
DSPfmt - 0,physical disps .ne.0,modal coords
С
                                  ptrDOF - pointer to degree of freedom set
С
С
                                  ptrDNC - pointer to nodal constraints
                                  ptrSTF - pointer to the reduced stiffness
С
                                  ptrMAS - pointer to the reduced mass matrix
С
С
                                   ptrDMP - pointer to the reduced damping
                                           matrix or mode shapes
С
                                   ptrFRQ - pointer to the frequencies
                                  ptrDSP - pointer to the calculated
C
С
                                           displacements
                                   ptrSTFh- High part of reduced stiffness ptr
С
                                  ptrMASh- High part of reduced mass ptr
C
С
                                  ptrDMPh- High part of reduced damping ptr
С
                                  ptrFRQh- High part of frequency ptr
C
                                  ptrDSPh- High part of displacement ptr
                                   ptrDVA - pointer to modal disp, velo and acc
С
C
                                  ptrDVAh- High part of modal disp, velo and acc
                                         - position not used
C
                          numdof Degrees of freedom per node
            i
C
                                    (curdof(i),i=1,numdof)
C
С
                                  dof reference numbers are:
          UX = 1, UY = 2, UZ = 3, ROTX = 4, ROTY = 5, ROTZ = 6, AX = 7, AY = 8
C
             = 9, VX =10, VY =11, VZ =12 ***** 13-18 are spares ********
С
          ******** PRES=19, TEMP=20, VOLT=21, MAG =22, ENKE=23, ENDS=24
С
                            ****** 27-32 are spares **********
C
          EMF =25, CURR=26
                          lenbac This table equates the actual node number to
С
                                   the number used for storage.
C
                                    (Back(i), i=1, lenbac)
C
            dp
                            10
                                  Time information:
С
C
                                   dtime,
                                              0.0,
                                                      0.0,
                                                              0.0,
                                                                      0.0,
                                                              0.0, timend
                                      0.0.
                                              0.0.
                                                      0.0.
C
                                  each item is described below:
C
C
                                   dtime - the time increment
С
                                   timend - the final time of the analysis
С
                                   0.0
                                           - position not used
C
                                   Degree of freedom set used
C
   DOF
            i
                    1
                          nmrow
С
                                   The DOFs are calculated as (N-1)*numdof+DOF,
С
                                   where N is the position number of the node in
                                   the nodal equivalence table and DOF is the
C
                                   DOF reference number given above.
С
                                   If the analysis uses the reduced method, the
C
                                   original DOF order (see next record) is
С
                                   rearranged so that DOFs having nodal
С
C
                                   constraints are listed first.
                                   If the analysis uses the mode superposition
C
С
                                   method (using the reduced mode extraction
С
                                   technique), the DOF order is the same as the
                                   original order (see next record).
C
                                     (l(i),i=1,nmrow)
С
            i
                                   Original reduced set of DOFs used.
С
                          nmrow+1
С
                                   The DOFs are calculated as (N-1)*numdof+DOF,
С
                                   where N is the position number of the node in
                                   the nodal equivalence table and DOF is the
C
                                   DOF reference number given above.
                                   If the analysis uses the reduced method, the
С
                                   original DOF order, plus the number of nodal
С
C
                                   constraints (nbcdsp), is stored.
С
                                   If the analysis uses the mode superposition
                                   method (using the reduced mode extraction
C
С
                                   technique), this record matches the previous
                                   record. The nmrow+1 entry will be zero.
C
```

С					(lorig(i),i=1,nmrow),nbcdsp
	DNC	i	1	nbcdsp	This record is present only if the analysis uses the reduced method and nbcdsp > 0 (see record at ptrDOF). These numbers are the positions in the previous record of dofs with a nodal constraint. These are nodal constraints only on nodes that also are masters. (na(i),i=1,nbcdsp)
	STF	dp	nmrow	nmrow	Reduced stiffness matrix. Each row of the matrix is stored as a record. The matrix is present only if nmatrx > 0 and analysis is not using mode superposition method Row order is the same as the DOF order in record at ptrDOF. (ak(i,j),i=1,nmrow)
0 0 0 0	MAS	dp	nmrow	nmrow	Reduced mass matrix. Each row of the matrix is stored as a record. The matrix is present only if nmatrx > 1 and analysis is not using mode superposition method. Row order is the same as the DOF order in record at ptrDOF. (am(i,j),i=1,nmrow)
С	DMP	dp	varies	varies	Reduced damping matrix or mode shapes.
0 0 0 0					If the analysis uses the reduced method, each record will be nmrow items in length. The reduced damping matrix is present only if nmatrx > 2. There will be nmrow records of this type stored here. Row order is the same as the DOF order in record at ptrDOF.
0 0 0 0 0 0 0 0 0 0 0					If the analysis uses the mode superposition method (using the reduced mode extraction technique), each record will be nmode items in length. These records contain mode shapes (eigenvectors) of the frequencies (eigenvalues) actually used in the harmonic analysis. There will be nmode records of this type stored here, with the first N records containing the mode shapes and the other records containing zeros, where N is the number of modes actually used in the harmonic analysis. Order corresponds to the DOF order given in record at ptrDOF.
с с с					<pre>If the analysis uses the mode superposition method, this record will not be present. (psi(i,j),i=1,nmrow) (or ac)</pre>
C C C C C C	FRQ	dp	1	nmrow	Frequencies extracted from the modal analysis. This record is present only if the analysis uses the mode superposition method. The first nmode values are the frequencies extracted from the modal analysis. The remaining values have no meaning. (freq(i),i=1,nmrow)
c c c	*** equa	als the	value o	f timend.	eated (as a pair) until the time value The number of iterations is stored as that deal with time)
	DSP	dp	1	nmrow+6	Calculated displacements The first nmrow entries are the displacements in the same order as the original set of DOFs (see record AFTER ptrDOF). If DSPfmt=0, these are physical displacements, If DSPftm!=0, these are the nmode modal coordinates.

```
For the last six entries:
С
С
                                     1. Time for these displacements
С
                                     2. Load step number
С
                                     3. Substep number
С
                                     4. Cumulative iteration number
C
                                     5. Scale factor (zero if the analysis uses
                                        the reduced method).
C
                                     6. numdeflys - number of scale factors
C
С
                                    Note: If, upon reading of this record, there
                                    is less than nmrow+5 items in the record,
С
                                    then only a selected set of nodes were
C
С
                                    output. Another record follows (integer, less
С
                                    than lenbac long) which contains the list of
                                    nodes for which DOF solutions are available.
C
С
            i
                    1 numdeflys
                                    lvscal table scale factor IDs
                                     (ilvscID(i), i=1, numdeflvs)
C
                                    lvscal table scale factor values
C
                    1 numdeflvs
                                     (dlvscVal(i), i=1, numdeflvs)
C
            dр
                                    Gap restoring forces. The order of these
C
                          ngaps
С
                                    forces corresponds to the node position order
С
                                    given in record at ptrDNC. This record is
                                    present only if ngaps > 0.
C
                                     (fgaps(i), i=1, ngaps)
c *** The next 3 records are kept for possible restart using mode superposition
c *** method. They are overwritten upon restarting. They are written once (last
c *** loadstep).
С
   DVA
                          ndva+6 Calculated modal displacements
                                    The first ndva entries are the modal
C
С
                                    displacements. For the last six entries:
                                     1. Time for these displacements
C
                                     2. Load step number
C
С
                                     3. Substep number
С
                                     4. Cumulative iteration number
                                     5. Scale factor (zero if the analysis uses
C
                                        the reduced method).
С
C
                                     6. numdeflys - number of scale factors
            dр
                          ndva
                                   Calculated modal velocities
            dр
                          ndva
                                   Calculated modal accelerations
```

1.4. Description of the Reduced Complex Displacement File

This section explains the content of the reduced complex displacement file (jobname.rfrq).

1.4.1. Standard ANSYS File Header

See *The Standard Header for ANSYS Binary Files* (p. 4) for a description of this set. File number (Item 1) is 10.

1.4.2. RFRQ File Format

```
parameter (RFRQNM='rfrq
     LONGINT
                      rfrqfpL, rfrqfp
      integer
                     rfrqbk, rfrqut
      common /fdrfrq/ rfrqfpL, rfrqbk, rfrqut
      equivalence (rfrqfp,rfrqfpL)
c write: harmcl, harmin, harmwr
c write: hrfrcl,hrfreq
c read:
          harstr
    ****** common variable descriptions *******
С
                file position on file rfrq
co rfrqfpL
co rfrqbk
                block number for file rfrq
  rfraut.
                file unit for file rfrq
CO
    See fddesc for documentation of how binary files are stored.
С
C
      ****** file format ******
C
        recid tells the identifier for this record. Not all records will have
C
С
              identifiers -- they are only indicated for those records whose
              record pointers are stored in the second file header.
C
С
         type tells what kind of information is stored in this record:
              i - integer
C
C
              dp - double precision
              cmp - complex
C
         nrec tells how many records of this description are found here
         lrec tells how long the records are (how many items are stored)
c recid
                           lrec
           type
                   nrec
                                    contents
С
  ___
            i
                   1
                           100
                                    standard ANSYS file header (see binhed for
                                    details of header contents)
С
                                  .RFRO FILE HEADER
C
            i
                    1
                            40
C
                                     fun10, nmrow, nmatrx, nmode, numdof,
С
C
                                      maxn, wfmax, lenbac, 0, ncumit,
С
                                       kan,
                                                0, nmUsed, nvect, DSPfmt,
С
                                    minmod,
                                                0,modlstp,
                                                               Ο,
                                    ptrDOF, ptrDNC, ptrSTF, ptrMAS, ptrDMP,
C
                                    ptrFRQ, ptrDSP,ptrSTFh,ptrMASh,ptrDMPh,
С
                                                                         Ο,
С
                                   ptrFRQh,ptrDSPh,
                                                         Ο,
                                                                 0,
                                         Ο,
                                                 Ο,
                                                         0,
                                                                 0,
C
                                  each item in header is described below:
С
С
                                   fun10 - unit number (rfrq file is 10)
                                   nmrow - number of rows/columns in matrices
C
                                   nmatrx - number of reduced matrices on file
С
С
                                   nmode - number of modes extracted during
                                            modal analysis (or nmrow if reduced
C
С
                                           method)
С
                                   numdof - number of dofs per node
                                   maxn - maximum node number
С
                                   wfmax - maximum wavefront
С
                                   lenbac - number of nodes
С
C
                                   ncumit - total number of iterations done
                                           during analysis
C
                                          - analysis type
C
                                   kan
                                            = 6 - reduced harmonic
С
                                   nmUsed - number of modes used in mode
С
C
                                            superposition
                                   nvect - number of generated loads in .mlv
С
С
                                   DSPfmt - 0,physical disps .ne.0,modal coords
C
                                   minmod - smallest mode number used
                                   modlstp- multiple load step key
С
                                   ptrDOF - pointer to degree of freedom set
C
```

_					
C					used in model
С					ptrDNC - pointer to nodal constraints
С					ptrSTF - pointer to the reduced stiffness
С					matrix
C					ptrMAS - pointer to the reduced mass matrix
C					ptrDMP - pointer to the reduced damping
C					matrix or mode shapes
C					ptrFRQ - pointer to the frequencies
C					ptrDSP - pointer to the calculated
С					displacements
C					ptrSTFh- High part of STF pointer
С					ptrMASh- High part of MAS pointer
С					ptrDMPh- High part of DMP pointer
С					ptrFRQh- High part of FRQ pointer
C					ptrDSPh- High part of DSP pointer
C					0 - position not used
C		i	1	numdof	Degrees of freedom per node
С					(curdof(i),i=1,numdof)
C					dof reference numbers are:
		TTT7 1	****	0 117	
С					3, ROTX= 4, ROTY= 5, ROTZ= 6, AX = 7, AY = 8
С					.1, VZ =12 ***** 13-18 are spares *******
C		*****	*****	** PRES=1	.9, TEMP=20, VOLT=21, MAG =22, ENKE=23, ENDS=24
C		EMF = 25	, CURR=	:26 *****	*** 27-32 are spares ***************
С		i	1	lenbac	This table equates the actual node number to
С					the number used for storage.
C					(Back(i),i=1,lenbac)
0					(Bach(I))I-I)Ichbac)
~		dр	1	10	Unused record. contents:
С		uр	_	10	
С					1.0, 0.0, 0.0, 0.0,
C					0.0, 0.0, 0.0, 0.0
C	DOF	i	1	nmrow	Degree of freedom set used
С					The DOFs are calculated as (N-1)*numdof+DOF,
С					where N is the position number of the node in
C					the nodal equivalence table and DOF is the
					the hodar equivalence table and bor is the
					DOE reference number given above
С					DOF reference number given above.
С					If the analysis uses the reduced method, the
					If the analysis uses the reduced method, the original DOF order (see next record) is
С					If the analysis uses the reduced method, the
C C					If the analysis uses the reduced method, the original DOF order (see next record) is
с с					If the analysis uses the reduced method, the original DOF order (see next record) is rearranged so that DOFs having nodal
c c c					If the analysis uses the reduced method, the original DOF order (see next record) is rearranged so that DOFs having nodal constraints are listed first.
0 0 0 0					If the analysis uses the reduced method, the original DOF order (see next record) is rearranged so that DOFs having nodal constraints are listed first. If the analysis uses the mode superposition
					If the analysis uses the reduced method, the original DOF order (see next record) is rearranged so that DOFs having nodal constraints are listed first. If the analysis uses the mode superposition method (using the reduced mode extraction
					If the analysis uses the reduced method, the original DOF order (see next record) is rearranged so that DOFs having nodal constraints are listed first. If the analysis uses the mode superposition method (using the reduced mode extraction technique), the DOF order is the same as the
					If the analysis uses the reduced method, the original DOF order (see next record) is rearranged so that DOFs having nodal constraints are listed first. If the analysis uses the mode superposition method (using the reduced mode extraction technique), the DOF order is the same as the original order (see next record).
					If the analysis uses the reduced method, the original DOF order (see next record) is rearranged so that DOFs having nodal constraints are listed first. If the analysis uses the mode superposition method (using the reduced mode extraction technique), the DOF order is the same as the
					If the analysis uses the reduced method, the original DOF order (see next record) is rearranged so that DOFs having nodal constraints are listed first. If the analysis uses the mode superposition method (using the reduced mode extraction technique), the DOF order is the same as the original order (see next record). (l(i),i=1,nmrow)
		i	1	nmrow+1	If the analysis uses the reduced method, the original DOF order (see next record) is rearranged so that DOFs having nodal constraints are listed first. If the analysis uses the mode superposition method (using the reduced mode extraction technique), the DOF order is the same as the original order (see next record). (l(i),i=1,nmrow) Original reduced set of DOFs used.
		i	1	nmrow+1	If the analysis uses the reduced method, the original DOF order (see next record) is rearranged so that DOFs having nodal constraints are listed first. If the analysis uses the mode superposition method (using the reduced mode extraction technique), the DOF order is the same as the original order (see next record). (l(i),i=1,nmrow)
		i	1	nmrow+1	If the analysis uses the reduced method, the original DOF order (see next record) is rearranged so that DOFs having nodal constraints are listed first. If the analysis uses the mode superposition method (using the reduced mode extraction technique), the DOF order is the same as the original order (see next record). (l(i),i=1,nmrow) Original reduced set of DOFs used.
		i	1	nmrow+1	If the analysis uses the reduced method, the original DOF order (see next record) is rearranged so that DOFs having nodal constraints are listed first. If the analysis uses the mode superposition method (using the reduced mode extraction technique), the DOF order is the same as the original order (see next record). (l(i),i=1,nmrow) Original reduced set of DOFs used. The DOFs are calculated as (N-1)*numdof+DOF, where N is the position number of the node in
		i	1	nmrow+1	If the analysis uses the reduced method, the original DOF order (see next record) is rearranged so that DOFs having nodal constraints are listed first. If the analysis uses the mode superposition method (using the reduced mode extraction technique), the DOF order is the same as the original order (see next record). (l(i),i=1,nmrow) Original reduced set of DOFs used. The DOFs are calculated as (N-1)*numdof+DOF, where N is the position number of the node in the nodal equivalence table and DOF is the
		i	1	nmrow+1	If the analysis uses the reduced method, the original DOF order (see next record) is rearranged so that DOFs having nodal constraints are listed first. If the analysis uses the mode superposition method (using the reduced mode extraction technique), the DOF order is the same as the original order (see next record). (l(i),i=1,nmrow) Original reduced set of DOFs used. The DOFs are calculated as (N-1)*numdof+DOF, where N is the position number of the node in
		i	1	nmrow+1	If the analysis uses the reduced method, the original DOF order (see next record) is rearranged so that DOFs having nodal constraints are listed first. If the analysis uses the mode superposition method (using the reduced mode extraction technique), the DOF order is the same as the original order (see next record). (l(i),i=1,nmrow) Original reduced set of DOFs used. The DOFs are calculated as (N-1)*numdof+DOF, where N is the position number of the node in the nodal equivalence table and DOF is the DOF reference number given above.
		i	1	nmrow+1	If the analysis uses the reduced method, the original DOF order (see next record) is rearranged so that DOFs having nodal constraints are listed first. If the analysis uses the mode superposition method (using the reduced mode extraction technique), the DOF order is the same as the original order (see next record). (l(i),i=1,nmrow) Original reduced set of DOFs used. The DOFs are calculated as (N-1)*numdof+DOF, where N is the position number of the node in the nodal equivalence table and DOF is the DOF reference number given above. If the analysis uses the reduced method, the
		i	1	nmrow+1	If the analysis uses the reduced method, the original DOF order (see next record) is rearranged so that DOFs having nodal constraints are listed first. If the analysis uses the mode superposition method (using the reduced mode extraction technique), the DOF order is the same as the original order (see next record). (l(i),i=1,nmrow) Original reduced set of DOFs used. The DOFs are calculated as (N-1)*numdof+DOF, where N is the position number of the node in the nodal equivalence table and DOF is the DOF reference number given above. If the analysis uses the reduced method, the original DOF order, plus the number of nodal
		i	1	nmrow+1	If the analysis uses the reduced method, the original DOF order (see next record) is rearranged so that DOFs having nodal constraints are listed first. If the analysis uses the mode superposition method (using the reduced mode extraction technique), the DOF order is the same as the original order (see next record). (l(i),i=1,nmrow) Original reduced set of DOFs used. The DOFs are calculated as (N-1)*numdof+DOF, where N is the position number of the node in the nodal equivalence table and DOF is the DOF reference number given above. If the analysis uses the reduced method, the
		i	1	nmrow+1	If the analysis uses the reduced method, the original DOF order (see next record) is rearranged so that DOFs having nodal constraints are listed first. If the analysis uses the mode superposition method (using the reduced mode extraction technique), the DOF order is the same as the original order (see next record). (l(i),i=1,nmrow) Original reduced set of DOFs used. The DOFs are calculated as (N-1)*numdof+DOF, where N is the position number of the node in the nodal equivalence table and DOF is the DOF reference number given above. If the analysis uses the reduced method, the original DOF order, plus the number of nodal constraints (nbcdsp), is stored.
		i	1	nmrow+1	If the analysis uses the reduced method, the original DOF order (see next record) is rearranged so that DOFs having nodal constraints are listed first. If the analysis uses the mode superposition method (using the reduced mode extraction technique), the DOF order is the same as the original order (see next record). (l(i),i=1,nmrow) Original reduced set of DOFs used. The DOFs are calculated as (N-1)*numdof+DOF, where N is the position number of the node in the nodal equivalence table and DOF is the DOF reference number given above. If the analysis uses the reduced method, the original DOF order, plus the number of nodal
		i	1	nmrow+1	If the analysis uses the reduced method, the original DOF order (see next record) is rearranged so that DOFs having nodal constraints are listed first. If the analysis uses the mode superposition method (using the reduced mode extraction technique), the DOF order is the same as the original order (see next record). (l(i),i=1,nmrow) Original reduced set of DOFs used. The DOFs are calculated as (N-1)*numdof+DOF, where N is the position number of the node in the nodal equivalence table and DOF is the DOF reference number given above. If the analysis uses the reduced method, the original DOF order, plus the number of nodal constraints (nbcdsp), is stored.
		i	1	nmrow+1	If the analysis uses the reduced method, the original DOF order (see next record) is rearranged so that DOFs having nodal constraints are listed first. If the analysis uses the mode superposition method (using the reduced mode extraction technique), the DOF order is the same as the original order (see next record). (l(i),i=1,nmrow) Original reduced set of DOFs used. The DOFs are calculated as (N-1)*numdof+DOF, where N is the position number of the node in the nodal equivalence table and DOF is the DOF reference number given above. If the analysis uses the reduced method, the original DOF order, plus the number of nodal constraints (nbcdsp), is stored. If the analysis uses the mode superposition method (using the reduced mode extraction
		i	1	nmrow+1	If the analysis uses the reduced method, the original DOF order (see next record) is rearranged so that DOFs having nodal constraints are listed first. If the analysis uses the mode superposition method (using the reduced mode extraction technique), the DOF order is the same as the original order (see next record). (l(i),i=1,nmrow) Original reduced set of DOFs used. The DOFs are calculated as (N-1)*numdof+DOF, where N is the position number of the node in the nodal equivalence table and DOF is the DOF reference number given above. If the analysis uses the reduced method, the original DOF order, plus the number of nodal constraints (nbcdsp), is stored. If the analysis uses the mode superposition method (using the reduced mode extraction technique), this record matches the previous
		i	1	nmrow+1	If the analysis uses the reduced method, the original DOF order (see next record) is rearranged so that DOFs having nodal constraints are listed first. If the analysis uses the mode superposition method (using the reduced mode extraction technique), the DOF order is the same as the original order (see next record). (l(i),i=1,nmrow) Original reduced set of DOFs used. The DOFs are calculated as (N-1)*numdof+DOF, where N is the position number of the node in the nodal equivalence table and DOF is the DOF reference number given above. If the analysis uses the reduced method, the original DOF order, plus the number of nodal constraints (nbcdsp), is stored. If the analysis uses the mode superposition method (using the reduced mode extraction technique), this record matches the previous record. The nmrow+1 entry will be zero.
		i	1	nmrow+1	If the analysis uses the reduced method, the original DOF order (see next record) is rearranged so that DOFs having nodal constraints are listed first. If the analysis uses the mode superposition method (using the reduced mode extraction technique), the DOF order is the same as the original order (see next record). (l(i),i=1,nmrow) Original reduced set of DOFs used. The DOFs are calculated as (N-1)*numdof+DOF, where N is the position number of the node in the nodal equivalence table and DOF is the DOF reference number given above. If the analysis uses the reduced method, the original DOF order, plus the number of nodal constraints (nbcdsp), is stored. If the analysis uses the mode superposition method (using the reduced mode extraction technique), this record matches the previous
	DNC				If the analysis uses the reduced method, the original DOF order (see next record) is rearranged so that DOFs having nodal constraints are listed first. If the analysis uses the mode superposition method (using the reduced mode extraction technique), the DOF order is the same as the original order (see next record). (l(i),i=1,nmrow) Original reduced set of DOFs used. The DOFs are calculated as (N-1)*numdof+DOF, where N is the position number of the node in the nodal equivalence table and DOF is the DOF reference number given above. If the analysis uses the reduced method, the original DOF order, plus the number of nodal constraints (nbcdsp), is stored. If the analysis uses the mode superposition method (using the reduced mode extraction technique), this record matches the previous record. The nmrow+1 entry will be zero. (lorig(i),i=1,nmrow),nbcdsp
	DNC	i	1	nmrow+1	If the analysis uses the reduced method, the original DOF order (see next record) is rearranged so that DOFs having nodal constraints are listed first. If the analysis uses the mode superposition method (using the reduced mode extraction technique), the DOF order is the same as the original order (see next record). (l(i),i=1,nmrow) Original reduced set of DOFs used. The DOFs are calculated as (N-1)*numdof+DOF, where N is the position number of the node in the nodal equivalence table and DOF is the DOF reference number given above. If the analysis uses the reduced method, the original DOF order, plus the number of nodal constraints (nbcdsp), is stored. If the analysis uses the mode superposition method (using the reduced mode extraction technique), this record matches the previous record. The nmrow+1 entry will be zero. (lorig(i),i=1,nmrow),nbcdsp This record is present only if the analysis
	DNC				If the analysis uses the reduced method, the original DOF order (see next record) is rearranged so that DOFs having nodal constraints are listed first. If the analysis uses the mode superposition method (using the reduced mode extraction technique), the DOF order is the same as the original order (see next record). (l(i),i=1,nmrow) Original reduced set of DOFs used. The DOFs are calculated as (N-1)*numdof+DOF, where N is the position number of the node in the nodal equivalence table and DOF is the DOF reference number given above. If the analysis uses the reduced method, the original DOF order, plus the number of nodal constraints (nbcdsp), is stored. If the analysis uses the mode superposition method (using the reduced mode extraction technique), this record matches the previous record. The nmrow+1 entry will be zero. (lorig(i),i=1,nmrow),nbcdsp This record is present only if the analysis uses the reduced method and nbcdsp > 0 (see
	DNC				If the analysis uses the reduced method, the original DOF order (see next record) is rearranged so that DOFs having nodal constraints are listed first. If the analysis uses the mode superposition method (using the reduced mode extraction technique), the DOF order is the same as the original order (see next record). (l(i),i=1,nmrow) Original reduced set of DOFs used. The DOFs are calculated as (N-1)*numdof+DOF, where N is the position number of the node in the nodal equivalence table and DOF is the DOF reference number given above. If the analysis uses the reduced method, the original DOF order, plus the number of nodal constraints (nbcdsp), is stored. If the analysis uses the mode superposition method (using the reduced mode extraction technique), this record matches the previous record. The nmrow+1 entry will be zero. (lorig(i),i=1,nmrow),nbcdsp This record is present only if the analysis uses the reduced method and nbcdsp > 0 (see record at ptrDOF). These numbers are the
	DNC				If the analysis uses the reduced method, the original DOF order (see next record) is rearranged so that DOFs having nodal constraints are listed first. If the analysis uses the mode superposition method (using the reduced mode extraction technique), the DOF order is the same as the original order (see next record). (l(i),i=1,nmrow) Original reduced set of DOFs used. The DOFs are calculated as (N-1)*numdof+DOF, where N is the position number of the node in the nodal equivalence table and DOF is the DOF reference number given above. If the analysis uses the reduced method, the original DOF order, plus the number of nodal constraints (nbcdsp), is stored. If the analysis uses the mode superposition method (using the reduced mode extraction technique), this record matches the previous record. The nmrow+1 entry will be zero. (lorig(i),i=1,nmrow),nbcdsp This record is present only if the analysis uses the reduced method and nbcdsp > 0 (see record at ptrDOF). These numbers are the positions in the previous record of dofs with
	DNC				If the analysis uses the reduced method, the original DOF order (see next record) is rearranged so that DOFs having nodal constraints are listed first. If the analysis uses the mode superposition method (using the reduced mode extraction technique), the DOF order is the same as the original order (see next record). (l(i),i=1,nmrow) Original reduced set of DOFs used. The DOFs are calculated as (N-1)*numdof+DOF, where N is the position number of the node in the nodal equivalence table and DOF is the DOF reference number given above. If the analysis uses the reduced method, the original DOF order, plus the number of nodal constraints (nbcdsp), is stored. If the analysis uses the mode superposition method (using the reduced mode extraction technique), this record matches the previous record. The nmrow+1 entry will be zero. (lorig(i),i=1,nmrow),nbcdsp This record is present only if the analysis uses the reduced method and nbcdsp > 0 (see record at ptrDOF). These numbers are the positions in the previous record of dofs with a nodal constraint. These are nodal
	DNC				If the analysis uses the reduced method, the original DOF order (see next record) is rearranged so that DOFs having nodal constraints are listed first. If the analysis uses the mode superposition method (using the reduced mode extraction technique), the DOF order is the same as the original order (see next record). (l(i),i=1,nmrow) Original reduced set of DOFs used. The DOFs are calculated as (N-1)*numdof+DOF, where N is the position number of the node in the nodal equivalence table and DOF is the DOF reference number given above. If the analysis uses the reduced method, the original DOF order, plus the number of nodal constraints (nbcdsp), is stored. If the analysis uses the mode superposition method (using the reduced mode extraction technique), this record matches the previous record. The nmrow+1 entry will be zero. (lorig(i),i=1,nmrow),nbcdsp This record is present only if the analysis uses the reduced method and nbcdsp > 0 (see record at ptrDOF). These numbers are the positions in the previous record of dofs with

C C					<pre>masters. (na(i),i=1,nbcdsp)</pre>
0 0 0 0	STF	dp	nmrow	nmrow	Reduced stiffness matrix. Each row of the matrix is stored as a record. The matrix is present only if nmatrx > 0 and analysis is not using mode superposition method Row order is the same as the DOF order in record at ptrDOF. (ak(i,j),i=1,nmrow)
0 0 0 0	MAS	dp	nmrow	nmrow	Reduced mass matrix. Each row of the matrix is stored as a record. The matrix is present only if nmatrx > 1 and analysis is not using mode superposition method. Row order is the same as the DOF order in record at ptrDOF. (am(i,j),i=1,nmrow)
С	DMP	dp	varies	varies	Reduced damping matrix or mode shapes.
0 0 0 0 0					If the analysis uses the reduced method, each record will be nmrow items in length. The reduced damping matrix is present only if nmatrx > 2. There will be nmrow records of this type stored here. Row order is the same as the DOF order in record at ptrDOF.
					If the analysis uses the mode superposition method (using the reduced mode extraction technique), each record will be nmode items in length. These records contain mode shapes (eigenvectors) of the frequencies (eigenvalues) actually used in the harmonic analysis. There will be nmode records of this type stored here, with the first N records containing the mode shapes and the other records containing zeros, where N is the number of modes actually used in the harmonic analysis. Order corresponds to the DOF order given in record at ptrDOF. If the analysis uses the mode superposition
C					<pre>method, this record will not be present. (psi(i,j),i=1,nmrow) (or ac)</pre>
c c c	FRQ	dp	1	nmrow	Frequencies extracted from the modal analysis. This record is present only for analyses using the mode superposition method. (freq(i),i=1,nmrow)
C C				_	eated (as a pair) s stored as ncumit.
	DSP	cmp r	ncumit	nmrow+5	Calculated complex displacements The first nmrow entries are the displacements in the same order as the original set of DOFs (see record AFTER ptrDOF). If DSPfmt=0, these are physical displacements, If DSPftm!=0, For the last five entries: Real part

```
5. numdeflvs
С
С
                                         (cvs(i), i=1, nmrow), (freq, delf),
С
                                         (itime,itter),(ncumit,0.0),(0.0,0.0),
С
                                         (fscale, 0.0)
С
                                   Note: If, upon reading of this record, there
                                   is less than nmrow+5 items in the record,
C
                                   then only a selected set of nodes were
                                   output. Another record follows (integer, less
C
С
                                   than lenbac long) which contains the list of
                                   nodes for which DOF solutions are available.
С
С
                    1 numdeflvs
                                    lvscal table scale factor IDs
С
                                     (ilvscID(i),i=1,numdeflvs)
С
                    1 numdeflvs
                                    lvscal table scale factor values
С
                                      (dlvscVal(i), i=1, numdeflvs)
C
```

1.5. Description of the Modal Results File

This section explains the content of the modal results file (jobname.mode).

1.5.1. Standard ANSYS File Header

See The Standard Header for ANSYS Binary Files (p. 4) for a description of this set. File number (Item 1) is 9.

1.5.2. MODE File Format

```
*comdeck,fdmode
c *** copyright(c) 2008 SAS IP, Inc. All rights reserved.
c *** ansys, inc.
      ****** description of modal result file *******
c *** mpg fdmode < modspc romstr lire_freq_mode lire_nb_mode: mode file desc
     character*8 MODENM
     parameter (MODENM='mode
     *** NOTE: if this variable is changed in the future it should be
C
      ***
               updated in spdefines.h also for symbolic assembly (jrb)
C
                     MODEHDLEN
     integer
     parameter
                    (MODEHDLEN=60)
     LONGINT
                     modefpL, modefp
     integer
                     modebk, modeut
     common /fdmode/ modefpL, modebk, modeut
     equivalence (modefp, modefpL)
   ****** common variable descriptions ********
C
                file position on file mode
   modefpL
CO
                block number for file mode
CO
   modebk
                file unit for file mode
CO
   modeut
   See fddesc for documentation of how binary files are stored.
C
С
      ****** file format ******
С
        recid tells the identifier for this record. Not all records will have
С
С
             identifiers -- they are only indicated for those records whose
             record pointers are stored in the second file header.
C
```

```
type tells what kind of information is stored in this record:
С
              i - integer
С
              dp - double precision
              cmp - complex
С
С
         nrec tells how many records of this description are found here
         lrec tells how long the records are (how many items are stored)
c recid
           type
                   nrec
                           lrec
                                     contents
            i
                    1
                           100
                                     standard ANSYS file header (see binhed for
С
  ---
                                     details of header contents)
С
                    1
                            60
                                     .MODE FILE HEADER
С
C
                                     fun09, nmrow, nmatrx, nmode, numdof,
C
C
                                      maxn, wfmax, lenbac,
                                                                0, nontp,
                                     lumpms, extopt, SvCode,
                                                               kan, ldstep,
C
                                     numitr, expbeg, expend, nspect, nSPdat,
С
С
                                     ptrRDF, ptrFRQ,
                                                        0, ptrSHP, ptrLOD,
                                     ptrSTF, ptrMAS, ptrDMP,
                                                               0, 0,
C
                                    ptrLPM, ptrSP1,ptrSHPh,ptrLODh,ptrSTFh,
С
С
                                    ptrMASh,ptrDMPh,ptrLPMh,ptrSPlh,ptrIRHSl,
C
                                  ptrIRHSh,PowerDyn,ptrRES,ptrRESh,
                                                                          0,
С
                                  KeyStress, ptrELD,ptrELDh,
                                                                  0,
                                                                           0,
C
                                    modlstp, nresi,
                                                         Ο,
                                                                  0,
                                                                           0,
                                          Ο,
                                                  0,
                                                          0,
                                                                  0,
                                                                           0
C
                                    each item in header is described below:
С
С
                                     fun09 - unit number (mode file is 9)
                                     nmrow - number of rows/columns in matrices
C
С
                                              (maxn*numdof). If extopt = 0, nmrow
С
                                              is the number of rows in the
                                              reduced matrices and the number of
C
                                              master degrees of freedom.
С
                                     nmatrx - number of reduced matrices on the
С
                                              file (applies only if extopt=0)
C
                                     nmode - number of modes extracted
С
                                     numdof - number of dof per node
C
С
                                     maxn
                                           - maximum node number (If extopt = 3
С
                                              or 4, the actual number of nodes is
                                              referenced.)
C
                                     wfmax - maximum wavefront (Does not apply
С
С
                                              if extopt = 3 \text{ or } 4.)
                                     lenbac - number of nodes
C
                                            - position not used
С
                                     nontp - number of equations on the .LN22
С
C
                                              file (Does not apply if extopt =
                                              0.)
C
                                     lumpms - lumped mass key
C
С
                                              = 0 - default matrix type
С
                                              = 1 - lumped
C
                                              (Does not apply if extopt = 3 or
С
                                              4.)
С
                                     extopt - mode extraction method
                                              = 0 - reduced
С
С
                                              = 3 - unsymmetric Lanczos
                                              = 4 - damped Lanczos
С
C
                                              = 6 - block Lanczos
                                              = 7 - QR damped
C
                                              = 8 - SuperNode
C
С
                                              = 9 - PCG Lanczos
                                     SvCode - Solver assembly code
С
                                              = 0 Frontal assembly (SV_ANSYS)
C
С
                                              = 1 Symbolic assembly (SV_CASI)
С
                                     kan
                                            - analysis type
C
                                              = 1 - buckling
                                              = 2 - modal
С
                                     ldstep - load step number
C
```

```
numitr - total number of cumulative
С
                                               iterations done during analysis
С
С
                                               (Does not apply if extopt = 3 or
С
                                               4.)
С
                                     expbeg - beginning of the frequency range of
С
                                               interest
                                     expend - end of the frequency range of
С
                                              interest
                                     nspect - number of spectra; if -6, these are
C
С
                                              the 6 default unit spectra
                                     nSPdat - number of data items per spectrum
С
                                     ptrRDF - pointer to reduced degree of
C
С
                                              freedom set used in model
С
                                     ptrFRQ - pointer to the frequencies
                                     ptrSHP - pointer to the mode shapes
C
С
                                              (eigenvectors)
                                     ptrLOD - pointer to the load vectors
С
                                     ptrSTF - pointer to the reduced stiffness
C
С
                                              matrix
                                     \operatorname{ptrMAS} - pointer to the reduced mass matrix
C
                                     ptrDMP - pointer to the reduced damping
С
С
                                              matrix
                                               (if extopt=7: pointer to the modal
C
С
                                                damping matrix)
С
                                     ptrLPM - pointer to the diagonal mass vector
C
                                     ptrSP1 - pointer to the the spectrum data
С
                                 ptrIRHSl,h - pointer to imaginary part of RHS vector
                                   PowerDyn - PowerDynamics key
C
                                              = 0 non-PowerDynamics method
C
                                              = 1 PowerDynamics method
С
                             ptrRES,ptrRESh - pointer to residual vectors
С
С
                                    modlstp - multiple load step key
С
                                      nresi - number of residual vectors in file
                                  KeyStress - key set if mode stresses on file
C
С
                             ptrELD, ptrELDh - pointer to element records
                                     0
                                            - position not used
C
                           numdof
                                     Degrees of freedom per node
С
С
                                     DOF reference numbers are:
          UX = 1, UY = 2, UZ = 3, ROTX = 4, ROTY = 5, ROTZ = 6, AX = 7, AY = 8
C
          AZ = 9, VX = 10, VY = 11, VZ = 12 ***** 13-18 are spares *********
С
                            PRES=19, TEMP=20, VOLT=21, MAG =22, ENKE=23, ENDS=24
C
                             ****** 27-32 are spares ***************
С
          EMF = 25, CURR = 26
                                      (curdof(i), i=1, numdof)
C
            i
                    1
                           lenbac
                                     Nodal equivalence table
С
С
                                     This table equates the number used for
                                     storage to the actual node number.
C
                                      (Back(i), i=1, lenbac)
С
   RDF
            i
                    1
                          nmrow
                                     Reduced set of degrees of freedom used.
C
                                     This record is present only if extopt = 0
C
                                     The DOFs are calculated as (N-1)*NUMDOF+DOF,
C
С
                                     where N is position number of the node in
С
                                     the nodal equivalence table and DOF is the
                                     DOF reference number given above
C
С
                                      (l(i), i=1, nmrow) (if nmatrx>0)
   FRO
            dp
                    1 nmode+nresi
                                    Frequencies (eigenvalues). Frequencies are
C
С
                                     complex if extopt=3 or 4. Numbers stored are
С
                                     the squares of the natural circular
С
                                     frequencies (w^{**2}, where w=radians/time).
                                     You can obtain the natural frequencies, f
C
                                     (in cycles/time), using the equation f=w/2pi
C
                                      (freq(i), i=1, nmode)
С
                                     Mode shapes (eigenvectors). Mode shapes are
C
   SHP
            dρ
                  nmode
                          nmrow
                                     complex if extopt=3 or 4. If extopt=0, the
C
С
                                     mode shape order corresponds to the DOF list
                                     stored at position ptrRDF. If extopt does
C
С
                                     not equal 0, the order corresponds to the
C
                                     nodal equivalence table
```

```
(psi(i,j),i=1,nmrow)
C
                                       residual vectors
С
   RES
             ďρ
                   nresi
                            nmrow
                                       Load vector. This record is present only if
С
   LOD
             dp
                      1
                            nmrow
                                       extopt=0 or 1.
C
                                        (f(i),i=1,nmrow)
С
c IRHS
             dρ
                      1
                            nmrow
                                       Imaginary Load vector. This record is present
                                       only if extopt = 6.
C
   LPM
                      1
                                       Lumped mass vector. This record is present
С
             dp
                            nmrow
                                       only if lumpms=1 and nmatrix=0. It is a
C
                                       vector containing the mass at each node in
С
С
                                       the system.
                                        (mass(i),i=1,nmrow)
C
                                       Reduced stiffness matrix. Each row of the
C
   STF
             dp
                   nmrow
                            nmrow
                                       matrix is stored as a record. The matrix is
C
C
                                       present only if nmatrx > 0. Row order is the
                                       same as the DOF order stored at position
C
                                       ptrRDF.
С
С
                                        (ak(i,j),i=1,nmrow)
                                       Reduced mass matrix. Each row of the matrix
С
   MAS
             dp
                   nmrow
                            nmrow
С
                                       is stored as a record. The matrix is present
                                       only if nmatrx > 1. Row order is the same as
C
C
                                       the DOF order stored at position ptrRDF.
                                        (am(i,j),i=1,nmrow)
C
   DMP
                                       Reduced damping matrix. Each row of the
С
             dp
                   nmrow
                            nmrow
                                       matrix is stored as a record. The matrix is
С
                                       present only if nmatrx > 2. Row order is the
С
С
                                       same as the DOF order stored at position
                                       ptrRDF.
C
C
                                        (ac(i,j),i=1,nmrow)
c for each spectrum (|nspect| records):
    For SPRS it is written by redpcl.F/lanpcl.F/subpcl.F
    For MPRS and PSD it is written by writeSpecInModeFile.F
C
   SP1
             dр
                         nmode+nresi Participation factors for this spectra
С
                      1
                          nmode+nresi Mode coefficients for this spectra
C
             dp
                          nmode+nresi Modal damping values
C
   ___
             ďρ
                     1
С
             dp
                      1
                            613
                                       svcom.inc (freqtb,...)
С
   ___
             dp
                      1
                            20
                                       misc. spectra data
С
   ELD
             int
                      1
                            15
                                       nelm,
                                               mask, nItems, ptrELM, ptrERS,
С
                                     ptrCER, ptrCERh, ptrESL, ptrESLh,
                                      ptrFR, ptrRFh
C
                                      each item in header is described below:
C
С
                                        nelm - number of elements
                                        mask - output mask (OUTRES)
C
                                      nItems - number of element records (7, VOL
С
C
                                                not included)
                                      ptrELM - pointer to element equivalence table
C
                                      ptrERS - pointer to element record sizes
С
С
                                    ptrCER,h - pointer to constant element records
                                    ptrESL,h - pointer to element index
С
                                     \ensuremath{\text{nRF}} - number of reaction forces \ensuremath{\text{ptrRF}}, \ensuremath{\text{h}} - pointer to reaction forces
С
С
С
                         above pointers are relative to ptrELD
             int.
                                      Total size of each element record (LONGINT)
                      1
                          2*nItems
C
   ELM
             int
                      1
                            nelm
                                      Element equivalence table
C
                                       This table equates the order number used to
C
                                       the actual element number
С
   ERS
             int.
                                      Sizes of the nItem element results sets for
C
                   nItems nelm
                                       each element
C
```

```
CER
                                  ptrVOL, ptrEPT, ptrEUL,
                                                                0.
                                                                        0
C
            int.
                    1
С
                       above pointers are relative to ptrCER
      constant element records (do not vary by mode):
C
С
            dр
                    1 nelm*1
                                    Element volume
C
    EPT
            dр
                    1
                      nelm*size
                                    Element structural nodal temperatures
                    1 nelm*size
                                    Element Euler angles
    EUL
            ďρ
C
   ESL
            int.
                    1
                                  ptrENS, ptrEEL, ptrEMS, ptrENF, ptrENG,
С
                          10
С
                                  ptrENSh,ptrEELh,ptrEMSh,ptrENFh,ptrENGh
                       above pointers are relative to ptrESL
С
С
     non-constant element records (do vary by mode):
С
    ENS
                nelm nmode*size Element nodal component stresses
    EEL
                nelm nmode*size
                                    Element nodal component elastic strains
C
            αb
С
            dр
                nelm nmode*size
                                    Element summable miscellaneous data
                nelm nmode*size
С
    ENF
            dр
                                    Element nodal forces
                nelm nmode*3
                                    Element energies
    ENG
           ďρ
C
      see fdresu.inc for more information on the element results
C
```

1.6. Description of the Element Matrices File

This section explains the content of the element matrices file (jobname.emat).

1.6.1. Standard ANSYS File Header

See The Standard Header for ANSYS Binary Files (p. 4) for a description of this set. File number (Item 1) is 2.

1.6.2. EMAT File Format

```
*comdeck,fdemat
c *** copyright(c) 2008 SAS IP, Inc. All rights reserved.
c *** ansys, inc.
      ****** description of element matrix file *******
C
C
С
 *** mpg fdemat.inc < eoelem elostr eofini outelm elfini EmatAssemble sffini
          eqprep sfform elstrt slvstr: emat file description
C
С
     character*8 EMATNM
     parameter (EMATNM='emat
     LONGINT
                     ematfpL, ematfp
                     ematbk, ematut
     common /fdemat/ ematfpL, ematbk, ematut
     equivalence (ematfp,ematfpL)
    ****** common variable descriptions *******
   ematfpL
                file position on file emat
CO
                block number for file emat
CO
                file unit for file emat
CO
   See fddesc for documentation of how binary files are stored.
С
C
      ****** file format *******
С
        recid tells the identifier for this record. Not all records will have
C
             identifiers -- they are only indicated for those records whose
С
С
              record pointers are stored in the second file header.
        type tells what kind of information is stored in this record:
С
С
             i - integer
             dp - double precision
C
C
             cmp - complex
```

```
nrec tells how many records of this description are found here
         lrec tells how long the records are (how many items are stored)
С
c recid
           type
                   nrec
                           lrec
                                     contents
C
            i
                    1
                           100
                                     standard ANSYS file header (see binhed for
                                     details of header contents)
C
С
            i
                    1
                            40
                                     .EMAT FILE HEADER
C
                                              nume, numdof, lenu, lenbac,
С
                                     fun02,
С
                                      maxn,
                                               0, 0, nodref, lumpm,
                                                      kycd, kygss, kygaf,
C
                                     kyast,
                                               kygm,
                                                          Ο,
                                                                 0,
С
                                     kygrf,
                                                 Ο,
C
                                    ptrElmh,ptrFSTh,ptrLSTh,ptrBITh,ptrEHDh,
                                    ptrIDXh, numCE, maxLeng, ptrCEl, ptrCEh,
C
                                    ptrDOF, ptrBAC,ptrELMl,ptrFSTl,ptrLSTl,
                                    ptrBITl,ptrEHDl,ptrIDXl,ptrendH,ptrendL
C
С
                                 each item in header is described below:
                                     fun02 - unit number (emat file is 2)
С
С
                                     nume - number of elements
C
                                     numdof - number of dofs per node
С
                                     lenu - total DOFs of model
                                     lenbac - number of nodes
C
                                     maxn - maximum node number
С
                                           - position not used
С
                                     Λ
С
                                           - position not used
                                     nodref - actual number of nodes referenced
С
С
                                     lumpm - lumped mass key
                                             = 0 - default matrix type
C
С
                                              = 1 - lumped
С
                                     kygst - global stiffness matrix calculate
C
                                              kev
                                              = 0 - do not calculate
С
                                             = 1 - calculate
С
                                            - global mass matrix calculate key
C
                                     kyqm
                                              = 0 - do not calculate
С
C
                                              = 1 - calculate
С
                                     kvcd
                                            - global damping matrix calculate key
С
                                             = 0 - do not calculate
                                              = 1 - calculate
C
                                           - global stress stiffening matrix
С
                                     kyqss
С
                                              calculate key
                                              = 0 - do not calculate
C
                                              = 1 - calculate
С
                                     kygaf - global applied force vector
С
C
                                              calculate kev
                                              = 0 - do not calculate
C
                                              = 1 - calculate
C
С
                                     kygrf - global restoring force vector
С
                                              calculate key (Newton-Raphson only)
                                              = 0 - do not calculate
C
                                              = 1 - calculate
С
С
                                     Λ
                                            - position not used
                                     0
С
                                            - position not used
С
                                     0
                                            - position not used
С
                                            - position not used
С
                                     ptrELMh- Highpointer to element equivalence
C
                                              table
                                     ptrFSTh- High pointer to first element at a
С
                                              DOF table
С
                                     ptrLSTh- High pointer to last element at a
C
С
                                             DOF table
С
                                     ptrBITh- High pointer to dof bits
C
                                     ptrEHDh- High pointer to the start of the
С
                                              element matrices
                                     ptrIDXh- High pointer to element matrices
C
```

```
index table
С
С
                                     numCE - number of internal CEs
                                     maxLeng- maximum length of any internal CE
С
                                     ptrCEl - low pointer to internal CE list
С
                                     ptrCEh - high pointer to internal CE list
С
                                     ptrDOF - pointer to degrees of freedom per
С
C
                                             node used in model
                                     ptrBAC - pointer to nodal equivalence table
C
                                     ptrELMl- Low pointer to element equivalence
С
С
                                              table
С
                                     ptrFSTl- Low pointer to first element at a
С
                                              DOF table
                                     ptrLST1- Low pointer to last element at a
C
С
                                              DOF table
С
                                     ptrBITl- Low pointer to dof bits
                                     ptrEHDl- Low pointer to the start of the
C
С
                                              element matrices
                                     ptrIDX1- Low pointer to element matrices
C
                                              index table
С
                                     ptrendH- High pointer to end of file
C
С
                                     ptrendL- Low pointer to end of file
        Note: the analysis type sets the global calculate keys.
C
                    1
                             20
                                     Time information
С
           dp
C
                                     timval, timinc, frqval, timbeg, timend,
С
                                                0.0,
                                                        0.0,
                                                                0.0,
С
                                        0.0,
                                                                         0.0.
С
                                        0.0,
                                                0.0,
                                                        0.0,
                                                                 0.0,
                                                                         0.0,
С
                                        0.0,
                                                0.0,
                                                        0.0,
                                                                 0.0,
                                                                         0.0,
С
                                    each item is described below:
                                     timval - the current time
С
                                     timinc - the time increment
С
                                     frqval - the current frequency (from a
С
                                              harmonic analysis)
C
                                     timbeg - the start time for the analysis
C
C
                                     timend - the end time for the analysis
                                            - position not used
С
                                     0.0
С
                                     0.0
                                            - position not used
                                            - position not used
                                     0.0
C
                                     0.0
                                            - position not used
С
С
                                     0.0
                                            - position not used
                                     0.0
                                            - position not used
С
                                     0.0
С
                                            - position not used
                                            - position not used
С
                                     0.0
                                     0.0
                                            - position not used
C
                                     0.0
                                            - position not used
                                     0.0
C
                                            - position not used
С
                                     0.0
                                            - position not used
С
                                     0.0
                                            - position not used
                                            - position not used
C
                                     0.0
                                            - position not used
С
                                     0.0
   DOF
                    1
                                     Degrees of freedom per node
                         numdof
C
С
                                     DOF reference numbers are:
          UX = 1, UY = 2, UZ = 3, ROTX = 4, ROTY = 5, ROTZ = 6, AX = 7, AY = 8
С
          AZ = 9, VX = 10, VY = 11, VZ = 12 ***** 13-18 are spares ********
C
          ******** PRES=19, TEMP=20, VOLT=21, MAG =22, ENKE=23, ENDS=24
C
          EMF =25, CURR=26 ******* 27-32 are spares *******************
C
С
                                      (curdof(i),i=1,numdof)
                                     Nodal equivalence table. This table equates
C
   BAC
            i
                         lenbac
                                     the number used for storage to the actual
С
С
                                     node number
                                      (Back(i), i=1, lenbac)
C
                                     Element equivalence table. The ANSYS program
  ELM
            i
                    1
                          nume
```

```
order that the SOLUTION processor solves the
С
С
                                     elements. This table equates the order
                                     number used to the actual element number
С
                                      (Order(i), i=1, nume)
С
С
   FST
                           lenu
                                     First element at a DOF table. This record
C
                                     signifies the first element encountered at a
                                     particular DOF.
C
С
                                      (First(i), i=1, lenu)
            i
                    1
                           lenu
                                     Last element at a DOF table. This record
C
   LST
                                     signifies the last element encountered at a
С
                                     particular DOF.
С
C
                                      (Last(i),i=1,lenu)
C
   BIT
            i
                     1
                           lenu
                                     Bits set at a DOF table. This record
                                     has bits for constraints, forces, etc.
C
С
                                      (DofBits(i), i=1, lenu) (added at 10.0)
   IDX
            i
                    1
                           2*nume
                                     Element index table. This record specifies
C
С
                                     the file location for the beginning of the
                                     data for each element.
C
С
                                      (index(i),i=1,nume) Low part of pointer
С
                                      (index(i),i=1,nume) High part of pointer
   The records at the end of the file store element information and get written
  as a set for each element(nume sets of these records will appear on the file
  at this point) ptrEHD indicates the beginning of the element data.
   If substructure matrices are written to the EMAT file, they are written in a
C
   different format than is shown here. This alternate format is not documented
   at this time, as it is likely to change in the future.
   EHD
                    1
                            10
                                     Element matrix header
C
С
                                                        dkey, sskey, akey,
С
                                      stkey,
                                               mkey,
                                                          Ο,
C
                                      nrkey,
                                               ikey,
                                                                   0, nmrow
                                    each item in header is described below:
C
С
                                     stkey
                                           - stiffness matrix key
                                              = 0 - matrix not present
C
                                              = 1 - matrix present
С
С
                                     mkev
                                            - mass matrix key
                                              = 0 - matirx not present
C
                                              = 1 - matrix present
С
                                             - damping matrix key
С
                                     dkey
C
                                              = 0 - matrix not present
                                              = 1 - matrix present
C
                                           - stress stiffening matrix key
C
                                     sskev
С
                                              = 0 - matrix not present
С
                                              = 1 - matrix present
                                            - applied load vector key
C
                                     akey
                                              = 0 - vector not used
С
С
                                              = 1 - vector used
                                           - newton-raphson(restoring) load
С
                                     nrkey
С
                                              vector key (for nonlinear analyses)
С
                                              = 0 - vector not used
C
                                              = 1 - vector used
                                             - imaginary load vector key
C
                                                 (for complex analyses)
C
С
                                               = 0 - vector not used
                                              = 1 - vector used
С
                                     0
C
                                             - position not used
С
                                             - position not used
С
                                            - numbers/columns in matrices. If the
                                     nmrow
C
                                              number is negative, the matrices
                                              will be written in lower triangular
С
C
                                              form.
```

C

stores all element data in the numerical

```
DOF index table. This record specifies the
С
С
                                   DOF locations of this element matrix in
                                   relation to the global matrix. The index is
С
                                    calculated as (N-1)*NUMDOF+DOF, where N is
С
С
                                    the position number of the node in the nodal
                                    equivalence table and DOF is the DOF
C
                                   reference number given above
С
           dp
                varies varies
                                   Element matrices. This record is repeated
                                    for each stiffness, mass, damping, and
С
                                    stress stiffening matrix. If the matrix is
C
С
                                   diagonal, the length of the records will be
С
                                   nmrow. If the matrix is unsymmetric, the
                                    length of the records will be nmrow*nmrow.
C
С
                                    If the matrix is symmetric, only the lower
C
                                    triangular terms are written and the length
                                   of the records will be (nmrow)*(nmrow+1)/2.
C
                                   Element force vectors. This record contains
C
           dp
                         2*nmrow
                                   both the applied force vector and the
C
С
                                    (restoring or imaginary) load vector.
C
С
      С
      The following records repeat numCE times... one for each internal
C
С
      CE created during solution... these are stored here for the psolve
      command, such as the case of a prestressed nonlinear modal analysis
С
C
   CE
                                   First part is the CE number, the second part is
С
С
                                    the number of terms in this internal CE, and
С
                                    the third part is the external element number
С
                                    of the element that created this internal CE
C
С
                nTerms
                        numCE
                                    integer info (list of node*32 + dof)
C
                                   dp info (list of coefficients including constant term)
C
         dρ
                nTerms
                        numCE
С
С
C
    kyast
                global stiffness matrix calculate key
С
                global mass matrix calculate key
С
    kygd
                global damping matrix calculate key
C
    kvass
                global stress stiffening matrix calculate key
                global applied force matrix calculate key
    kygaf
                global restoring force matrix calculate key
    kygrf
```

1.7. Description of the Substructure Matrices File

This section explains the contents of the substructure matrices file (jobname.sub).

1.7.1. Standard ANSYS File Header

See The Standard Header for ANSYS Binary Files (p. 4) for a description of this set. File number (Item 1) is 8.

1.7.2. SUB File Format

```
*comdeck,fdsub
c
c *** copyright(c) 2008 SAS IP, Inc. All rights reserved.
c *** ansys, inc

c ********* description of substructure matrix file ********
    character*8 SUBNM
    parameter (SUBNM='sub ')
    LONGINT subfpL, lenSubL
    integer subbk, subut
```

```
c write: matout
c read:
    ****** common variable descriptions *******
C
co subfpL
               file position on file sub
                block number for file sub
                file unit for file sub
    subut.
CO
                length of sub file (saved for slvdta.F)
   lenSubL
    See fddesc for documentation of how binary files are stored.
C
С
      ****** file format *******
С
      recid tells the identifier for this record. Not all records will have
С
           identifiers -- they are only indicated for those records whose
C
           record pointers are stored in the second file header.
C
      type tells what kind of information is stored in this record:
C
           i - integer
C
С
           dp - double precision
           cmp - complex
C
С
      nrec tells how many records of this description are found here
      lrec tells how long the records are (how many items are stored)
c recid
                           lrec
                                    contents
           type
                   nrec
            i
                           100
                                    standard ANSYS file header (see binhed
С
                    1
                                    for details of header contents)
С
                            60
                                    .SUB FILE HEADER (FULL MATRICES)
  HED
           i
                    1
C
С
С
                                      8, nmrow, nmatrx, nedge, numdof,
С
                                   maxn, wfmax, lenbac, nnod, kunsym,
                                   kstf, kmass, kdamp,
С
                                                          kss, nvect,
                                 nWorkL, lenU1, sesort, lenlst,ptrLodL,
С
                                 ntrans, ptrMtx, ptrXFM, ptrHED, name1,
C
                                            0,
                                                     0, name3, name4,
С
                                 ptrDOF, ptrDST, ptrBAC, ptrTIT, ptrNOD,
C
С
                                 ptrXYZ, ptrEDG, ptrGDF, thsubs, ptrPOS,
С
                                 ptrORG, stfmax,ptrLodH, nmodes, keydim,
                              cmsMethod, name5, name6, name7, name8,
C
                                nvnodes,ptrCTXM, nWorkH, ptrCG,
                                                                      0.
С
C
                                     0,
                                             0,
                                                     0,
                                                              0,
                                                                      Ω
   HED
            i
                   1
                            60
                                    .SUB FILE HEADER (SPARSE MATRICES)
С
С
C
                                           nEqn, nmatrx,
                                                               . numdof.
                                          , lenbac, nnod, kunsym,
C
                                   maxn,
                                   kstf, kmass, kdamp, , nvect,
C
                                 nTermL,
С
                                                               ,ptrLodL,
                                                  ,
                                     ,ptrMtxL,
                                                  , ptrHED, name1,
, name3, name4,
С
C
                                  name2, ,
                                               , ptrBAC, ptrTIT, ptrNOD,
С
                                 ptrDOF,
С
                                 ptrXYZ,
                                                     , thsubs,
                                      , stfmax,ptrLodH,
С
С
                                       , name5, name6, name7, name8,
С
                                              , nTermH,ptrMtxH,ptrColL,
                                ptrColH,ptrCofL,ptrCofH,
C
                                each item in header is described below:
C
                                 fun08 - unit number (full sub file is 8)
C
                                         (sparse substructure file is 9)
C
С
                                 nmrow - number of rows in matrices (also
С
                                          number of dofs in substructure)
                                 nmatrx - number of matrices on file
C
                                 nedge - number of edges for outline
numdof - number of dofs per node
С
C
```

common /fdsub/ subfpL, lenSubL, subbk, subut

c	maxn - maximum node number of complete
c c	<pre>model presently in database wfmax - maximum wavefront of substruct.</pre>
C	during generation pass
c c	lenbac - number of nodes defining substructure during the
C	generation pass
С	nnod - number of unique nodes in the
С	substructure having DOFs, and
C	which define this substructure
C	during the use pass. Also, the number of nodes having master
C C	DOFs.
c	kunsym - unsymmetric matrix key
C	= 0 - symmetric
С	= 1 - unsymmetric
С	kstf - stiffness matrix present key
C	<pre>= 0 - matrix is not on file = 1 - matrix is on file</pre>
c c	kmass - mass matrix present key
C	= 0 - matrix is not on file
C	= 1 - matrix is on file
С	=-1 - Lumped mass vestor (Sparse only)
C	kdamp - damping matrix present key
C	= 0 - matrix is not on file = 1 - matrix is on file
C C	kss - stress stiffening matrx present
c	= 0 - matrix is not on file
C	= 1 - matrix is on file
С	nvect - number of load vectors
C	(at least 1 is required)
C	nWorkL,H - BCS workspace length (only for
C C	bacsub) nTermL,H - Number of terms in sparse matrix
c	lenUl - length of intermediate transformation
C	vector
С	sesort - DOF set sort key
C	= 0 - numbers are not sorted
C	= 1 - numbers are sorted in
c c	ascending order lenlst - maximum length of DOF set for
c	this substructure (maxn*numdof)
С	ptrLod - pointer to the start of the load
С	vectors (see also ptrLodh)
C	ntrans - transformed key
C	= 0 - substructure has not been transformed
C C	> 0 - substructure copied
c	from another substructure,
C	via either SESSYM or SETRAN
C	
· ·	ptrMtxL,H - pointer to the start of the
С	substructure matrices (iDiagL for
c c	substructure matrices (iDiagL for sparse matrices)
с с с	substructure matrices (iDiagL for sparse matrices) ptrXFM - pointer to the substructure
c c	substructure matrices (iDiagL for sparse matrices)
с с с	substructure matrices (iDiagL for sparse matrices) ptrXFM - pointer to the substructure transformations
c c c c c	substructure matrices (iDiagL for sparse matrices) ptrXFM - pointer to the substructure transformations ptrHED - pointer to the SUB file header namel - first four characters of the substructure file name, in
c c c c c c	substructure matrices (iDiagL for sparse matrices) ptrXFM - pointer to the substructure transformations ptrHED - pointer to the SUB file header namel - first four characters of the substructure file name, in integer form
c c c c c c	substructure matrices (iDiagL for sparse matrices) ptrXFM - pointer to the substructure transformations ptrHED - pointer to the SUB file header name1 - first four characters of the substructure file name, in integer form name2 - second four characters of the
c c c c c c	substructure matrices (iDiagL for sparse matrices) ptrXFM - pointer to the substructure transformations ptrHED - pointer to the SUB file header name1 - first four characters of the substructure file name, in integer form name2 - second four characters of the substructure file name, in
c c c c c c c	substructure matrices (iDiagL for sparse matrices) ptrXFM - pointer to the substructure transformations ptrHED - pointer to the SUB file header name1 - first four characters of the substructure file name, in integer form name2 - second four characters of the
	substructure matrices (iDiagL for sparse matrices) ptrXFM - pointer to the substructure transformations ptrHED - pointer to the SUB file header namel - first four characters of the substructure file name, in integer form name2 - second four characters of the substructure file name, in integer form
	substructure matrices (iDiagL for sparse matrices) ptrXFM - pointer to the substructure transformations ptrHED - pointer to the SUB file header namel - first four characters of the substructure file name, in integer form name2 - second four characters of the substructure file name, in integer form name3 - third four characters of the substructure file name, in integer form
	substructure matrices (iDiagL for sparse matrices) ptrXFM - pointer to the substructure transformations ptrHED - pointer to the SUB file header namel - first four characters of the substructure file name, in integer form name2 - second four characters of the substructure file name, in integer form name3 - third four characters of the substructure file name, in integer form name4 - fourth four characters of the
	substructure matrices (iDiagL for sparse matrices) ptrXFM - pointer to the substructure transformations ptrHED - pointer to the SUB file header namel - first four characters of the substructure file name, in integer form name2 - second four characters of the substructure file name, in integer form name3 - third four characters of the substructure file name, in integer form name4 - fourth four characters of the substructure file name, in
	substructure matrices (iDiagL for sparse matrices) ptrXFM - pointer to the substructure transformations ptrHED - pointer to the SUB file header name1 - first four characters of the substructure file name, in integer form name2 - second four characters of the substructure file name, in integer form name3 - third four characters of the substructure file name, in integer form name4 - fourth four characters of the substructure file name, in integer form
	substructure matrices (iDiagL for sparse matrices) ptrXFM - pointer to the substructure transformations ptrHED - pointer to the SUB file header namel - first four characters of the substructure file name, in integer form name2 - second four characters of the substructure file name, in integer form name3 - third four characters of the substructure file name, in integer form name4 - fourth four characters of the substructure file name, in
	substructure matrices (iDiagL for sparse matrices) ptrXFM - pointer to the substructure transformations ptrHED - pointer to the SUB file header name1 - first four characters of the substructure file name, in integer form name2 - second four characters of the substructure file name, in integer form name3 - third four characters of the substructure file name, in integer form name4 - fourth four characters of the substructure file name, in integer form ptrDOF - pointer to the DOF/node list
	substructure matrices (iDiagL for sparse matrices) ptrXFM - pointer to the substructure transformations ptrHED - pointer to the SUB file header name1 - first four characters of the substructure file name, in integer form name2 - second four characters of the substructure file name, in integer form name3 - third four characters of the substructure file name, in integer form name4 - fourth four characters of the substructure file name, in integer form ptrDOF - pointer to the DOF/node list ptrDST - pointer to the local DOF set

```
ptrTIT - pointer to the title
C
                                  ptrNOD - pointer to the unique nodes
С
С
                                           defining the substructure
                                  ptrXYZ - pointer to the coordinates of the
С
                                           unique nodes
С
                                  ptrEDG - pointer to the substructure edges
С
                                  ptrGDF - pointer to the global DOF set
C
C
                                  ptrCG - pointer to the element mass information
                                  thsubs - thermal key
C
С
                                           = 0 - structural
                                           = 1 - thermal
С
                                  ptrPOS - pointer to the sorted substructure
C
С
                                           DOF set to the original
С
                                  ptrORG - pointer to the DOF set of the model
                                           during the generation pass
C
                                  stfmax - maximum diagonal stiffness term
С
                                           (packed into an integer)
C
                                  ptrLodh- High 32 bits of 64 bit pointer
C
С
                                  nmodes - number of modes used to generate
                                           CMS s.e.
C
                                  keydim - dimensionality key
С
С
                                           = 1 - axisymmetric
                                           = 2 - 2-D
C
                                           = 3 - 3-D
С
С
                               cmsMethod - component mode synthesis method
C
                                  name5 - fifth four characters of the
С
                                           substructure file name, in integer
C
                                           form
                                  name6 - sixth four characters of the
С
                                           substructure file name, in integer
С
С
                                           form
С
                                  name7
                                         - seventh four characters of the
С
                                           substructure file name, in integer
                                           form
C
С
                                  name8 - eighth four characters of the
С
                                           substructure file name, in integer
                                           form
C
                                 nvnodes - number of virtual nodes that contain
С
                                           the modal coordinates
С
                                 ptrCTXM - coordinate transformation
C
                                 ptrColL,H - pointer to the iCol sparse matrix
С
C
                                             array
С
                                 ptrCofL,H - pointer to the of the
С
                                             sparse matrix Sk(1:nTerm),
                                             Sm(1:nTermL),Sc(1:nTermL),
C
                                             Ss(1:nTermL) Each matrix is a
С
С
                                             single large record
                                note: name1/2/3/4/5/6/7/8 are the
С
С
                                      inexc4 representation of the
C
                                      32 character filename.
                                      name1/2/5/6/7/8 will be "0"
C
                                      for pre rev 5.2 files - cwa
C
С
    Note: If ntrans > 0, records from position ptrDOF to ptrGDF will be
          identical to the data for the copied substructure.
C
                                  Substructure transformations (5*25 double
   XFM
                           125
С
            dр
                    1
С
                                  precisions). This record has meaning only
                                  if ntrans > 0. You can define up to five
С
C
                                  levels of transformations, with 25 variables
                                  in each level. Up to the first seven
C
                                  variables are used as follows:
C
С
                                  If the substructure was transferred (via the
С
                                  SETRAN command):
C
                                  1st variable - 1.0
С
С
                                  2nd variable - nodal increment
                                  3rd variable - reference number of
C
С
                                   coordinate system where substructure will
C
                                   be transferred
```

```
4th variable - reference number of
С
С
                                    coordinate system where substructure is
С
                                   presently defined
                                   5th variable - x coordinate increment
С
                                   6th variable - y coordinate increment
С
                                   7th variable - z coordinate increment
C
C
                                   If the substructure used symmetry (via the
                                    SESYMM command):
C
С
                                   1st variable - 2.0
                                   2nd variable - nodal increment
С
                                   3rd variable - number of coordinate
C
                                    component to be used in operation
С
С
                                    = 1 - x coordinate
C
                                    = 2 - y coordinate
                                    = 3 - z coordinate
С
                                   4th variable - reference number of
C
                                    coordinate system to be used for symmetry
С
C
                                    operation
                                  Substructure transformations
                            250
C
   CTXM
            dр
                     1
С
   DOF
            i
                     1
                           numdof
                                      Degrees of freedom per node (Global)
                                       (curdof(i),i=1,numdof)
C
С
                                      DOF reference numbers are:
C
         = 1, UY = 2, UZ = 3, ROTX= 4, ROTY= 5, ROTZ= 6, AX = 7, AY = 8
        = 9, VX =10, VY =11, VZ =12 ***** 13-18 are spares ********
************** PRES=19, TEMP=20, VOLT=21, MAG =22, ENKE=23, ENDS=24
C
С
                        ****** 27-32 are spares ***************
     EMF = 25, CURR = 26
C
   DST
            i
                           nmrow This record contains degrees of freedom for
С
С
                                   this substructure of the unique nodes, as
С
                                   used with this substructure, in ascending
С
                                   order. This index is calculated as
                                   (N-1)*numdof+DOF, where N is the node number
C
С
                                   and DOF is the DOF reference number given
С
                                   above
                                    (lsort(i),i=1,nmrow)
C
С
   POS
                           nmrow
                                  This record stores the positions of the
                                   local DOF set in relation to the generated
C
C
                                  DOF set. (lposit(i),i=1,nmrow)
С
   ORG
                           nmrow
                                  DOF set of the model as defined during the
С
                                   generation pass. This index is calculated as
                                   (N-1)*NUMDOF+DOF, where N is the position
C
С
                                  number of the node in the nodal equivalence
С
                                   table and DOF is the DOF reference number
                                   given above
C
                                    (lorig(i),i=1,nmrow)
С
   BAC
            i
                           lenbac This group describes nodes that defined the
C
                                   substructure during the generation pass of
C
                                   the analysis. Nodal data is stored in arrays
C
С
                                   equal to the number of used or referenced
С
                                   nodes. This table equates the number used
                                   for storage to the actual node number.
C
С
                                    (Back(i), i=1, lenbac)
   TIT
            i
                     1
                             20
                                   Substructure title (converted to integers -
C
С
                                   see inexc4)
С
   NOD
                            nnod
                                  This record describes unique nodes defining
                                   the substructure for the use pass of the
C
                                   analysis. These are also the nodes having
C
С
                                  master degrees of freedom.
                                    (node(i), i=1, nnod)
С
   XYZ
           dр
                  nnod
                                  This record describes the coordinates of a
C
C
                                   unique node, in the order X, Y, Z, THXY,
                                   THYZ, and THZX. Nodal order corresponds to
C
                                   that of the node list given above
С
C
                                    (xyzang(j,i),j=1,6)
```

```
This record contains beginning and ending
С
   EDG
           dр
                  nedge
С
                                  locations (X1,Y1,Z1,X2,Y2,Z2 coordinates) of
                                  a straight line comprising an edge of the
С
C
                                  substructure.
                           nmrow This record describes global degrees of
C
C
                                  freedom of the unique nodes in ascending
                                  order, as used during the analysis use pass.
C
С
                                  This index is calculated as (N-1)*32+DOF,
                                  where N is the node number and DOF is the
C
                                  DOF reference number given above
C
С
                                   (l(i), i=1, nmrow) (sorted)
                            10
                                  total mass, CGx, CGy, CGz, 6 moments of inertia
C
   CG
           αb
c The substructure matrices are written at this position in the file. One row
c of each matrix is written to the file at a time. i.e. the first row of each
   matrix is written, then the second row of each matrix, etc. this pattern
   continues until all nmrow rows of each matrix have been written to the file.
   МАТ
                                   Row of the stiffness matrix, if nmatrx > 0.
C
                           nmrow
C
                                    (ak(i,j),i=1,nmrow)
С
                    1
                           nmrow
                                   Row of the mass matrix, if nmatrx > 1.
                                    (am(i,j),i=1,nmrow)
C
С
                     1
                                   Row of the damping matrix, if nmatrx > 2.
C
                                    (ac(i,j),i=1,nmrow)
                                   Row of the stress stiffening matrix, if
С
            dp
                           nmrow
C
                                   nmatrx > 3.
С
                                    (gs(i,j),i=1,nmrow)
С
   LOD
            dp
                 nvect
                           nmrow
                                   This record contains the load vectors.
                                    (f(i),i=1,nmrow)
```

1.8. Description of the Component Mode Synthesis Matrices (CMS) File

This section explains the contents of the CMS matrices file (jobname.cms).

1.8.1. Standard ANSYS File Header

See The Standard Header for ANSYS Binary Files (p. 4) for a description of this set. File number (Item 1) is 8.

1.8.2. CMS File Format

```
*comdeck.fdcms
c --- description of cms(component modal synthesis) transformation file
     character*8 CMSNM
     parameter (CMSNM='cms
                    cmsfpL, cmsfp
     integer
                    cmsbk, cmsut
     common /fdcms/ cmsfpL, cmsbk, cmsut
     equivalence (cmsfp,cmsfpL)
c --- common variable desciption ------
              file position on file mode
co cmsfp
   cmsbk
              block number for file mode
CO
   cmsut
              file unit for file mode
CO
c --- See fddesc for documentation of how binary files are stored. -----
C
     --- file format
```

```
recid tells the identifier for this record. Not all records will have
С
      identifiers -- they are only indicated for those records whose
С
С
     record pointers are stored in the second file header.
С
      --- type tells what kind of information is stored in this record:
С
C
      i - integer
     dp - double precision
C
      cmp - complex
C
С
      --- nrec tells how many records of this description are found here
C
      --- lrec tells how long the records are (how many items are stored)
C
c ----
c recid
           type
                   nrec
                           lrec
                                     contents
С
            i
                           100
                                     standard ANSYS file header (see binhed for
                                     details of header contents)
C
            i
                    1
                            40
                                     .CMS FILE HEADER
C
C
С
                                      fun45,
                                              neqn ,
                                                      nirfm, nnorm, ncstm,
                                                 0,
                                                          Ο,
                                                                  0,
                                          0,
C
C
                                          0,
                                                  Ο,
                                                          Ο,
                                          0,
                                                  Ο,
                                                          Ο,
                                                                   Ο,
                                                                           0,
C
                                          0.
                                                  0.
                                                          0.
                                                                   0.
                                                                           0.
C
С
                                          0,
                                                  0,
                                                          0,
                                                                   0,
                                                                           0,
С
                                    ptrIRFS,ptrNORS,ptrCSTS,
                                                                   0,ptrIRFL,
                                    ptrNORL,ptrCSTL,
C
С
                                    each item in header is described below:
                                     fun45 - unit number
CC
С
                                            - number of equation in BCS
                                     nirfm - number of inertia relief modes
C
С
                                     nnorm - number of normal modes
                                     ncstm - number of constraint modes
                                            > 0 available in file
C
                                            < 0 NA in file
С
С
                          ptrIRFS,ptrIRFL - pointer to inertia relief modes
                          ptrNORS,ptrNORL - pointer to normal modes
C
                          ptrCSTS,ptrCSTL - pointer to constraint modes
C
C
С
                                            - position not used
             i
                                     BCS to ANS mapping (lBCStoANS(i), i= 1,neqn)
                     1
C
                          negn
C
С
   NOR
            dр
                 nnorm
                          neqn
                                     Normal Modes
C
                                     Inertia Relief Modes
С
            dр
                 nirfm
                          neqn
    CST
            αb
                 ncstm
                                     Constraint Modes
```

1.9. Description of the Full Stiffness-Mass File

nean

This section explains the contents of the full file (jobname.full).

1.9.1. Standard ANSYS File Header

See The Standard Header for ANSYS Binary Files (p. 4) for a description of this set. File number (Item 1) is 4.

1.9.2. FULL File Format

^{*}comdeck.fdfull

```
c *** copyright(c) 2008 SAS IP, Inc. All rights reserved.
c *** ansys, inc.
      ****** description of full stiffness-mass file *******
c *** mpg fdfull.inc < stff10 slvstr: full file description
      character*8 FULLNM
     parameter (FULLNM='full
                                 ')
      *** NOTE: if this variable is changed in the future it should be
               updated in spdefines.h also for symbolic assembly (jrb)
      integer
                     FULLHDLEN
                     (FULLHDLEN=80)
      parameter
                     fullfpL, fullfp
fullbk, fullut, wrLdstep, wrSbstep, wrEqiter
     LONGINT
      integer
     common /fdfull/ fullfpL, fullbk, fullut,
                     wrLdstep,wrSbstep,wrEqiter
     equivalence (fullfp,fullfpL)
    ****** common variable descriptions *******
co fullfpL
                file position on file full
                block number for file full
co fullbk
  fullut
                file unit for file full
С
      ****** file format (except for extopt=3,4) ********
    See fddesc for documentation of how binary files are stored.
С
      ****** file format *******
C
C
         recid tells the identifier for this record. Not all records will have
              identifiers -- they are only indicated for those records whose
C
С
              record pointers are stored in the second file header.
         type tells what kind of information is stored in this record:
C
C
              i - integer
             dp - double precision
C
             cmp - complex
C
        nrec tells how many records of this description are found here
C
         lrec tells how long the records are (how many items are stored)
c recid
                          lrec
                                   contents
          type
                  nrec
                          100
                                   standard ANSYS file header (see binhed for
           i
                   1
C
                                   details of header contents)
С
C
            i
                   1
                            60
                                    .FULL FILE HEADER
C
                                    fun04,
C
                                             neqn, nmrow, nmatrx,
С
                                    wfmax, lenbac, numdof,jcgtrmL,jcgtrmH,
С
                                    lumpm, jcgeqn, jcgtrm, keyuns, extopt,
                                    keyse, sclstf, nxrows,ptrIDXl,ptrIDXh,
C
                                   ncefull,ncetrm,ptrENDl,ptrENDh,
С
С
                                        Ο,
                                                Ο,
                                                        Ο,
                                                                0,
                                                                        0,
                                        0,
                                                0,
                                                        0,
С
                                                                0,
                                                                        0,
С
                                        0,
                                                Ο,
                                                        Ο,
                                                                0,
С
                                        0,
                                                Ο,
                                                        Ο,
                                                                0,
                                                                        0,
C
                                        Ο,
                                                0,
                                                        0,
                                                                Ο,
                                        0,
                                                0,
                                                        0,
                                                                0,
                                                                        0,
C
                                                        0,
                                        0,
                                                0,
                                                                0,
                                                                        0
C
c NOTE: If fun04 > 0, then the file was created with frontal assembly
       If fun04 < 0, then the file was created with symbolic assembly; see below
С
                      for its format
 ----- frontal assembled file -----
                                   each item in header is described below:
C
```

```
fun04 - unit number (full file is 4)
С
С
                                    neqn - number of equations on file
                                    nmrow - number of rows in matrices
С
                                    nmatrx - number of matrices on file
С
                                           - analysis type
С
                                    kan
                                    wfmax - maximum wavefront
C
                                    lenbac - number of nodes
                                    numdof - number of dofs per node
C
С
                                     jcgtrmL, jcgtrmH - number of coefficients
                                     lumpm - lumped mass key
С
                                              = 0 - default matrix type
C
                                              = 1 - lumped
С
С
                                     jcgeqn - number of jcg equations
C
                                     jcgtrm - pre-8.1 this is the number of
                                              coefficients in sparse jcg
С
                                              storage (otherwise this value
С
                                              must be 0 and jcgtrmL, jcgtermH
C
                                              must be used)
                                    keyuns - unsymmetric key
C
                                              = 0 - no unsymmetric matrices on
С
С
                                                     file
                                              = 1 - there is at least one
С
С
                                                     unsymmetric matrix on file
С
                                     extopt - mode extraction method
                                              = 0 - reduced
= 1 - lumped
C
С
                                              = 3 - unsymmetric Lanczos
C
                                              = 4 - damped Lanczos
С
                                              = 6 - block Lanczos
С
                                    keyse - superelement key; set if at least
С
С
                                              one superelement
                                     sclstf - scale factor for matrices
С
                                    nxrows - the maximum rank for this solution
C
                                    ptrIDXl- pointer to the matrix row indices.
С
                                    ptrIDXh- high part of row index pointer
C
                                    ncefull- Number of constraint equations on
С
                                              the full file
С
                                    ncetrm - Total number of terms in the
С
                                              constraint equations
C
                                    ptrENDl- Low part of 64 bit end of file ptr
С
C
                                    ptrENDh- High part of 64 bit end of file ptr
С
                                           - position not used
           i
                    1
                                    Degrees of freedom per node
                           numdof
C
                                    DOF reference numbers are:
С
          UX = 1, UY = 2, UZ = 3, ROTX = 4, ROTY = 5, ROTZ = 6, AX = 7, AY = 8
С
          AZ = 9, VX =10, VY =11, VZ =12 ***** 13-18 are spares ********
C
                            PRES=19, TEMP=20, VOLT=21, MAG =22, ENKE=23, ENDS=24
          ******
С
          EMF =25, CURR=26 ******* 27-32 are spares *******************
С
                                     (curdof(i),i=1,numdof)
C
            i
                    1
                                    Nodal equivalence table. This table equates
C
                           lenbac
С
                                     the number used for storage to the actual
С
                                    node number
                                      (Back(i), i=1, lenbac)
C
C
   NOTE: The next five records are repeated as a group neqn times.
   When the matrices get written, one row of each matrix is written to the file
C
С
    at a time. i.e. the first row of each matrix is written, then the second row
    of each matrix, etc. this pattern continues until all the rows of each
C
   matrix have been written to the file. If kan=3, the matrix rows will be
C
    complex valued, otherwise they will be double precision values.
   IDX
            i
                           varies
                                    Matrix row indices. The first
С
C
                                    item signifies what term in the row belongs
С
                                    to the pivot. The second term signifies what
С
                                    DOF is being eliminated, and the remaining
C
                                     items signify the new DOFs being introduced
С
                                     (if any). The length of this record will
                                    vary (actual length is returned from routine
C
```

C C				BINRD8) (111(i),i=1,m)
0 0 0 0	 i	1	varies	A second level of indexing for the matrix. Indicates positions and values of terms to be reduced. The length of this record will vary (actual length is returned from routine BINRD8) (index(i),i=1,n) for compressed rows
С	 dp/cmp	1	varies	Stiffness matrix.
с с с				If keyuns=0, this record will contain the non-diagonal terms of this column, the diagonal term itself, followed by the normalized F term.
c c c				If keyuns=1, this record will contain the non-diagonal terms of this column, the diagonal term itself, the non-diagonal terms of this row, followed by the normalized F term.
				<pre>If lumpm = 1, then the mass for this node is located after the F term. The length of this record will vary (actual length is returned from routine BINRD8) (krow(i),i=1,n),vload,(mass) (symmetric) (n-1 column) diag (n-1 row) load (dmass) (unsymmetric)</pre>
C	 dp/cmp	1	varies	Mass matrix. This record exists only if nmatrix > 1.
с с с				If keyuns=0, this record will contain the non-diagonal terms of this column, the diagonal term itself, followed by the normalized F term.
с с с				If keyuns=1, this record will contain the non-diagonal terms of this column, the diagonal term itself, followed by the non-diagonal terms of this row.
c c c				The length of this record will vary (actual length is returned from routine BINRD8) (mrow(i),i=1,n) (symmetric) (n-1 column) diag (n-1 row) (unsymmetric)
C				If lumpms=1, this record contains one double array with diag values
C C	 dp/cmp	1	varies	Damping matrix. This record exists only if nmatrx > 2.
с с с				If keyuns=0, this record will contain the non-diagonal terms of this column, the diagonal term itself, followed by the normalized F term.
с с с				If keyuns=1, this record will contain the non-diagonal terms of this column, the diagonal term itself, followed by the non-diagonal terms of this row.
с с с				The length of this record will vary (actual length is returned from routine BINRD8) (ceqn(i),i=1,n) (symmetric) (n-1 column) diag (n-1 row) (unsymmetric)
С	 		symbol	ic assembled file

```
i
                    1
                            80
                                    .FULL FILE HEADER
C
С
С
                                   fun04.
                                              neqn,
                                                       nmrow,
                                                               nmatrx,
                                                                              kan.
                                            lenbac,
                                                      numdof, ntermKl,
С
                                   wfmax,
                                                                         ntermKh,
                                                      ntermK,
                                                                keyuns,
С
                                   lumpm,
                                             nmrow,
                                                      nxrows, ptrSTFl, ptrSTFh,
С
                                   keyse,
                                            sclstf,
                                 ncefull, ntermMh, ptrENDl, ptrENDh, ptrIRHSl,
С
                                ptrIRHSh, ptrMASl, ptrMASh, ptrDMPl, ptrDMPh,
                                  ptrCEl, ptrCEh, nNodes, ntermMl, ntermDl,
C
                                 ptrDOF1, ptrDOFh, ptrRHS1, ptrRHSh, ntermDh,
С
                                 ngMaxNZ, ptrNGPHl, ptrNGPHh, minKdiag, maxKdiag,
С
                                minMdiag, maxMdiag, minDdiag, maxDdiag, ngTerml,
C
                                 ngTermh, ngTermCl, ngTermCh,ptrDiagKl,ptrDiagKh,
С
                               ptrDiagMl,ptrDiagMh,ptrDiagCl,ptrDiagCh, ptrSclKl,
С
C
                                ptrSclKh, Glbneqn, distKey, ngTermFl, ngTermFh,
                               GlbnNodes, GlbnVars , GlbfAcCE, lcAcLen,
С
                                                 0,
                                                           0,
                                                                      0,
                                                                                0,
C
                                       0,
                                       Ο,
                                                 Ο,
                                                            Ο,
                                                                      0,
                                                                                Ο,
C
                                   each item in header is described below:
C
                                    fun04 - negative of the unit number (-4)
С
                                          - number of equations on file
                                    negn
C
                                    nmrow - number of active DOF (neqn-BC)
С
С
                                    nmatrx - number of matrices on file
C
                                    kan
                                           - analysis type
                                    wfmax - maximum row size
С
                                    lenbac - number of nodes
C
                                    numdof - number of dofs per node
С
                                    ntermKl,ntermKh - number of terms in Stiffness
С
                                                      matrix
                                    lumpm - lumped mass key
С
С
                                             = 0 - default matrix type
                                             = 1 - lumped
C
С
                                    ntermK - pre-8.1 this is the number of terms
С
                                             in Stiffness matrix (otherwise this
С
                                             value must be 0 and ntermKl,ntermKh
                                             must be used)
С
                                    keyuns - unsymmetric key
С
                                              = 0 - no unsymmetric matrices on
C
                                                    file
С
C
                                             = 1 - there is at least one
С
                                                    unsymmetric matrix on file
С
                                    extopt - mode extraction method
                                             = 0 - reduced
C
С
                                             = 1 - lumped
С
                                             = 3 - unsymmetric Lanczos
                                             = 4 - damped Lanczos
С
                                             = 6 - block Lanczos
С
                                             = 7 - QRdamped
С
                                             = 8 - SuperNode
C
                                             = 9 - PCG Lanczos
C
                                    keyse - superelement key; set if at least
C
С
                                            one superelement
                                    sclstf - maximum absolute stiffness matrix term
С
                                    nxrows - the maximum rank for this solution
C
                                    ptrSTFl,h - pointer to Stiffness matrix
С
С
                                    ncefull
                                               - number of CE+CP equations
                                               - low part of 64 bit end of file ptr
С
                                    ptrENDl
С
                                    ptrENDh
                                               - high part of 64 bit end of file ptr
                                    ptrIRHSl,h - pointer to imaginary RHS (F)
С
С
                                    ptrMASl,h - pointer to Mass matrix
                                    ptrDMPl,h - pointer to Damping matrix
                                    ptrCEl,h - pointer to Gt and g matrices
C
С
                                    nNodes
                                               - number of internal Nodes
                                                 considered by symbolic assembly
С
                                    ntermMl,h - number of terms in Mass matrix
C
                                    ntermDl,h - number of terms in Damping matrix
С
С
                                    ptrDOFl,h - pointer to DOF info
C
                                    ptrRHSl,h - pointer to RHS (F)
                                                - maximum number of nodes per nodal
С
                                    ngMaxNZ
                                                 block in nodal graph structure
C
```

```
ptrNGPHl,h - pointer to vectors needed for
C
С
                                                 nodal graph structure
С
                                    minKdiag - minimum absolute stiffness matrix
                                               diagonal term
С
С
                                    maxKdiag - maximum absolute stiffness matrix
С
                                               diagonal term
                                    minMdiag - minimum absolute mass matrix
C
C
                                               diagonal term
                                    maxMdiag - maximum absolute mass matrix
C
С
                                               diagonal term
                                    minDdiag - minimum absolute damping matrix
С
C
                                               diagonal term
С
                                    maxDdiag - maximum absolute damping matrix
С
                                               diagonal term
C
                                    ngTerml,h - total number of nonzeroes in nodal graph
С
                                                   (expanded graph based value, no BC applied)
                                    ngTermCl,h - total number of nonzeroes in nodal graph
C
                                                  (compressed graph based value)
C
С
                                    ptrDiagKl,h - pointer to stiffness matrix DIAGONAL vector
                                                  (NOTE: this is a copy of the diagonal
C
                                                         values stored in the full matrix)
С
С
                                    ptrDiagMl,h - pointer to mass matrix DIAGONAL vector
                                                  (NOTE: this is a copy of the diagonal
C
                                                         values stored in the full matrix)
С
С
                                    ptrDiagDl,h - pointer to damping matrix DIAGONAL vector
C
                                                  (NOTE: this is a copy of the diagonal
С
                                                         values stored in the full matrix)
                                    ptrSclKl,h - pointer to stiffness matrix diagonal scaling
С
                                                  vector (may contain all 1.0's when not scaling)
С
                                    Glbneqn - global number of equations (this will match
С
                                               neqn at position 2 unless we are writing distributed
С
С
                                               "local" FULL files in Distributed ANSYS
C
                                    distKey - key denoting whether the FULL file is a single,
                                               global FULL file (0) or multiple, local FULL file (1)
C
С
                                    ngTermFl,h - total number of nonzeroes in nodal graph
                                                 as passed to the solver (after BC applied)
C
                                    GlbnNodes - non-zero if across CE in distributed full file
C
                                    GlbnVars - non-zero if across CE in distributed full file
С
                                    GlbfAcCE - total number of across CE
С
                                             - number of acrossCE where slaves are in my domain
С
                                    lcAcLen
                                    GlbfCE
                                              - total number of all the CE
С
C
                                             - position not used
С
            i
                    1
                           numdof
                                    Degrees of freedom per node
                                    DOF reference numbers are:
C
          UX = 1, UY = 2, UZ = 3, ROTX = 4, ROTY = 5, ROTZ = 6, AX = 7, AY = 8
С
          AZ = 9, VX =10, VY =11, VZ =12 ***** 13-18 are spares ********
C
                            PRES=19, TEMP=20, VOLT=21, MAG =22, ENKE=23, ENDS=24
          ******
C
          EMF = 25, CURR = 26
                            ****** 27-32 are spares ***************
C
            i
                    1
                           lenbac
                                    Nodal equivalence table. This table equates
                                    the number used for storage to the actual
C
                                    node number
C
c Stiffness Matrix. The next two records are repeated as a group neqn times.
                                    Matrix row indices. The last item
С
   STF
                           varies
С
                                    corresponds to the diagonal. The
                                    length of this record will vary (actual
С
С
                                    length is returned from routine BINRD8)
C
         dp/cmp
                    1
                           varies
                                    Matrix terms
                                    If keyuns=0, this record will contain the
C
                                     terms before the diagonal.
С
                                    If keyuns=1, this record will contain the
C
C
                                     the entire row.
C
c Load Vector
С
  RHS
         dp/cmp
                    1
                             neqn
                                    Load vector terms.
```

```
c Imaginary part of Load Vector
                                     Imaginary load vector terms.
  IRHS
           dp
                    1
                             negn
c Stiffness matrix diagonal vector
  DIAGK
           dp/cmp
                    1
                              neqn
                                     diagonal vector data for stiffness matrix
c Stiffness matrix diagonal scaling vector
           dp/cmp
                                     diagonal scaling vector for stiffness matrix
   SCLK
                             nean
c DOF information
                            nNodes
                                     Nodal extent vector. Number of DOFs at
С
   DOF
C
                                     each node
                              neqn
                                     DOF vector. If negative, this DOF
                                     constrained
C
                              neqn
                                     DOFs with imposed values
C
         dp/cmp
                            varies
                                     Imposed values
c Mass Matrix.
С
     if lumpm = 0:
       The next two records are repeated as a group neqn times.
C
                            varies
                                     Matrix row indices. The last item
С
                                     corresponds to the diagonal. The
С
                                     length of this record will vary (actual
С
С
                                     length is returned from routine BINRD8)
С
           dр
                            varies
                                     Matrix terms
     if lumpm = 1:
C
           dр
                                     Matrix diagonals
С
                              neqn
c Mass matrix diagonal vector
 DIAGM
           dρ
                    1
                              nean
                                     diagonal vector data for mass matrix
c Damping Matrix. The next two records are repeated as a group neqn times.
   DMP
                            varies
                                     Matrix row indices. The last item
С
С
                                     corresponds to the diagonal. The
                                     length of this record will vary (actual
C
                                     length is returned from routine BINRD8)
С
           αb
                    1
                            varies
                                     Matrix terms
C
c Damping matrix diagonal vector
  DIAGD
           dр
                    1
                              negn
                                     diagonal vector data for damping matrix
c G matrix if ncefull > 0.
                          ncefull
                                     List of slave DOFs
  CE
            i
C
С
           dр
                    1
                          ncefull
                                     g vector (constant terms)
                           ncefull
                                     imaginary g vector (constant terms). This
C
                                     vector only exists for harmonic analyses.
C
                                     (Antype == 3).
С
                                     Header; 1=nRows, 2=nRows, 3=1, 4=0
C
            i
                    1
            i
                    1
                            nRows
                                     Vector of 1's
                            nRows
                                     Number of non-zero terms in each row
```

```
Repeat for each row:
            i
                    1
                            varies
                                    Column indices
           dр
                    1
                            varies
                                    Column values
c Nodal graph vectors
            i
                                     number of nonzeroes for each node
  NGPH
                    1
                            nNodes
    Repeat for each node
            i
                            varies
                                     Index vector.
   Meaning of K11, K12, and G matrices:
С
      Given
C
          [K]\{x\} = \{F\}
С
С
      subject to the constraints
            {x1} = [G]{x2} + {g}
C
      where \{x1\} are the slave DOFs, \{x2\} the master DOFs
С
С
      This results in
          [K*]{x2} = {F*}
С
С
      where
          [K*] = [G]'[K11][G] + [G]'[K12] + [K21][G] + [K22]
С
          {F*} = [G]'{f1} + {f2} - [G]'[K11]{g} - [K21]{g}
c complex version of \{F^*\} decomposed into, we assume G' is always real
c and g could be complex denoted as g' == (g,gx):
          G' \ K11' \ g' = G' \ (K11,M11)*(g,gx)
С
С
                      = G' [K11*g - M11*gx, M11*g + K11*gx]
          K21' *g'
                     = (K21,M21)*(g,gx)
С
С
                      = (K21*g- M21*gx, K21*gx + M21*g)
```

Release 12.0 - © 2009 SAS IP, Inc. All rights reserved Contains proprietary and confidential information
nelease 12.0 - © 2009 3A3 ir, inc. Airrights reserved Contains proprietary and confidential information
of ANSYS Inc. and its subsidiaries and affiliates.

Chapter 2: Accessing Binary Data Files

This chapter explains the routines you need to read, write, or modify an ANSYS binary file. This collection of routines (called BINLIB) resides on your ANSYS distribution media.

The following topics are discussed in this chapter:

- 2.1. Accessing ANSYS Binary Files
- 2.2. Demonstration Routines
- 2.3. Results File Access Routines

2.1. Accessing ANSYS Binary Files

The BINLIB library is in the dynamic link library \Program Files\Ansys Inc\V120\ANSYS\custom\misc\<platform>\binlib.dll (on Windows systems (where <platform> is a directory that uniquely identifies the hardware platform version)) or the shared library /ansys_inc/v120/ansys/customize/misc/<platform>/libbin.so on UNIX systems (libbin.sl on HP systems).

2.1.1. Access Routines to Results, Substructure, and Matrix Files

Demonstration programs that use the BINLIB library for reading and writing ANSYS results, substructure, and matrix files are included on the installation media:

- ResRdDemo
- ResWrDemo
- rdsubs
- wrtsub
- rdfull
- bintst

On Windows Systems:

The FORTRAN source for these programs is located in \Program Files\Ansys Inc\V120\ANSYS\custom\misc\<platform> and the files are named ResRdDemo.F, ResWrDemo.F, rdsubs.F, wrtsub.F, and rdfull.F.

To link these demonstration programs, use the \Program Files\Ansys Inc\V120\ANSYS\custom\misc\<platform>\rdrwrt.bat procedure file and specify the program that you want to build on the command line. Valid command line options are ResRdDemo, ResWrDemo, rdsubs, wrtsub, rdfull, and userprog. For example, to build the program to read a results file, type:

\Program Files\Ansys Inc\V120\ANSYS\custom\misc\<platform>\rdrwrt ResRdDemo

Appropriate files will then be copied from \Program Files\Ansys Inc\V120\ANSYS\cus-tom\misc\<platform> to your working directory, compiled, and linked. The resulting executable will also be placed in your current working directory.

Use the userprog command line option when writing your own customized program, naming the routine userprog.F. The resulting executable will be named userprog.exe. When userprog is used, no files are copied from \Program Files\Ansys Inc\V120\ANSYS\custom\misc\<platform> to your working directory.

These files will be loaded onto your system only if you performed a custom installation and chose to install the customization tools.

On UNIX systems:

The FORTRAN source for these programs is located in /ansys_inc/v120/ansys/customize/misc and the files are named ResRdDemo.F, ResWrDemo.F, rdsubs.F, wrtsub.F, and rdfull.F.

To link these demonstration programs, use the /ansys_inc/v120/ansys/custom-ize/misc/rdrwrt.link procedure file and specify the program that you want to build on the command line. Valid command line options are ResRdDemo, ResWrDemo, rdsubs, wrtsub, rdfull, and userprog. For example, to build the program to read a results file, type:

/ansys_inc/v120/ansys/customize/misc/rdrwrt.link ResRdDemo

Appropriate files will then be copied from /ansys_inc/v120/ansys/customize/misc to your working directory, compiled, and linked. The resulting executable will also be placed in your current working directory. Procedure files are available in the /ansys_inc/v120/ansys/bin directory to run these programs, once linked. The procedure files are named ResRdDemo120, ResWrDemo120, rdsubs120, wrtsub120, and rdfull120.

Use the userprog command line option when writing your own customized program, naming the routine userprog.F. The resulting executable will be named userprog.e120. When userprog is used, no files are copied from /ansys_inc/v120/ansys/customize/misc to your working directory. The procedure file is named userprog120.

These files will be loaded onto your system only if you performed a custom installation and chose to install the customization tools.

2.1.2. Characteristics of ANSYS Binary Files

Before accessing ANSYS binary files, you need to know certain file characteristics:

- 1. An ANSYS binary file is a direct access, unformatted file. You read or write a record by specifying (as a number) what location to read or write.
- Before the ANSYS program actually writes data to a file on a disk, it uses buffers to store data in memory until those buffers become full. A block number designates these buffers. Most access routines use this block number.
- 3. By default, ANSYS files are external files. The standardized "external" format the files use enables you to transport them across different computer systems.
- 4. In addition to file names, ANSYS uses file numbers to identify the files. File handles and other information are associated with the file numbers.
- 5. Some binary files contain data values that point to the start of certain data (for example, the start of the data steps index table record). Both the ANSYS program and external binary files access routines use these pointers to locate data on the various binary files.
- 6. All data is written out as 32-bit integers. Double-precision data and pointers, therefore, take up two integer words. To create a 64-bit pointer from the two 32-bit integers, use the function largeIntGet.

2.1.3. Viewing Binary File Contents

To view the contents of certain ANSYS binary files, you issue the command /AUX2 or choose menu path Utility Menu>File>List>Binary Files or Utility Menu>List>File>Binary Files. (You can do so only at the Begin level.) The ANSYS program then enters its binary file dumping processor, AUX2, and dumps the binary file record by record.

In AUX2, you can use either the record number (**DUMP** command) or the record pointer (**PTR** command). If the file was written in parallel (-NP>2 on the command line), only the **PTR** command may be used.

2.1.4. Abbreviations

The input and output for the routines discussed in this chapter are described with the following abbreviations:

Type of variable is one of the following:

```
int - integer
dp - double-precision
log - logical (true or false)
char - character
```

Size of variable is one of the following:

```
sc - scalar variable ar(n) - array of size n
```

Intent of variable is one of the following:

```
in - input onlyout - output onlyinout - both an input and an output variable
```

2.1.5. binini (Initializing Buffered Binary I/O Systems)

2.1.6. Function sysiqr (Retrieving the Status of a File)

```
*deck,sysiqr
    function sysiqr (nunit,fname,lname_in,inqr_in)

c *** primary function: do a file system inquire (system dependent)

c *** Notice - This file contains ANSYS Confidential information ***

c input arguments:
    variable (typ,siz,intent) description
```

```
nunit.
              (int,sc,in)
                                 - fortran unit number (used only for inqr='0')
C
С
              (chr,sc,in)
                                 - name of file
С
     lname_in (int,sc,in)
                                 - length of file name (characters, max=50)
                                 - character key for information requested
С
     inqr_in (chr,sc,in)
                                   = 'E' - return whether file exists
С
                                       sysiqr = 1 - file exists
С
                                              = 0 - file does not exist
С
                                              < 0 - error occured
                                   = '0' - return whether file is open
C
С
                                       sysiqr = 1 - file is open
                                              = 0 - file is closed
С
                                              < 0 - error occured
C
С
                                   = 'N' - return unit number of file
С
                                       sysiqr > 0 - unit number for file
                                              = 0 - file not assigned to a unit
C
                                              < 0 - error occured
С
   output arguments:
C
С
     sysiqr (int,func,out)
                                 - the returned value of sysiqr is based on
C
                                         setting of ingr
```

2.1.7. Function binigr8 (Retrieving System-Dependent Parameters)

```
*deck,biniqr8
      function biniqr8 (nblk,key)
c *** primary function: get data about a block i/o buffer
c --- This routine is intended to be used in standalone programs.
c --- This routine should not be linked into the ANSYS program.
c *** Notice - This file contains ANSYS Confidential information ***
   input arguments:
C
      nblk
                                   - the block number for the inquiry
С
                (int,sc,in)
                                    or zero (see below)
C
C
                (int,sc,in)
                                   - key for information requested
                       nblk = 0 - return information about system/file
С
C
                                 key = 1 - return system block size
С
                                     = 2 - return number of integers per dp
                                     = 3 - return filename length
С
                                       5 = return integers per LONG
C
                       nblk > 0 - return information about this block
С
С
                                 key = 1 - return fortran unit number
                                     = 2 - return number of blocks in file
С
С
                                     = 3 - return length of page (32 bit words)
                                     = 4 - return open status
C
С
                                           0 - file close
                                           1 - file open
C
                                     = 5 - return file format
C
                                           0 - internal format
С
                                           1 - external format
С
                                     = 6 - return read/write status
C
                                           0 - both read & write
С
С
                                           1 - read
С
                                           2 - write
                                     = 7 - return current position on file
C
                                     = 8 - return maximum length of file
C
                                             (in words)
С
С
                                     = 9 - return starting word for this page
                                             in buffer
C
С
   output arguments:
     biniqr (int,func,out)
                                   - the returned value of binigr is based on
C
                                         setting of nblk and key
С
```

2.1.8. Function binset (Opening a Blocked Binary File or Initializing Paging Space)

```
*deck,binset
     function binset (nblk,nunit,ikeyrw,istart,paglen,npage,
                       pname,nchar,kext,Buffer4)
c *** primary function: initialize paging space for a blocked binary file.
                        binset should be used to open a blocked file
С
                        before binrd8 or binwrt8 are used. binclo should
                        be used to close the file.
c --- This routine is intended to be used in standalone programs.
c --- This routine should not be linked into the ANSYS program.
c *** Notice - This file contains ANSYS Confidential information ***
  input arguments:
C
     nblk
                (int,sc,in)
                                  - block number (1 to BIO_MAXFILES max)
С
                 (int,sc,in)
                                   - fortran unit number for the file
C
      nunit
                                     (if 0, bit bucket)
C
                (int,sc,in)
                                   - read/write flag
C
      ikeyrw
                                     = 0 - both read & write
С
                                     = 1 - read
C
C
                                     = 2 - write
                                     = 9 - read only
C
С
      istart
                (int,sc,in)
                                   - starting location in buffer array
                                     usually 1 for nblk=1, paglen*npage+1
С
                                     for nblk=2,etc.
C
      paglen
                                  - page length in integer*4 words for external
С
                (int,sc,in)
С
                                    files
С
                                    paglen should always be a multiple of
                                     512 words for efficiency
С
                                  - number of pages (1 to BIO_MAXBLOCKS max)
C
      npage
                (int,sc,in)
                (chr,ar(*),in)
                                  - name of the file
С
      pname
      nchar
                (int,sc,in)
                                  - number of characters in the file name(not
C
                                     used)
С
С
      kext.
                (int,sc,in)
                                  - no longer used, always external format
С
      Buffer4
                (i4, ar(*),inout) - work array for paging, should be
                                    dimensioned to paglen*npage*nblk (max)
C
С
   output arguments:
C
      binset.
              (int.func.out)
                                  - error status
                                     = 0 - no error
C
                                     <>0 - error occurred
C
      Buffer4
               (i4, ar(*),inout) - work array for paging
```

2.1.9. Subroutine bintfo (Defining Data for a Standard ANSYS File Header)

```
*deck,bintfo
     subroutine bintfo (title,jobnam,units,code)
c *** primary function:
                          set information necessary for binhed
\ensuremath{\text{c}} --- This routine is intended to be used in standalone programs.
c --- This routine should not be linked into the ANSYS program.
 *** Notice - This file contains ANSYS Confidential information ***
C
С
С
      typ=int,dp,log,chr,dcp siz=sc,ar(n) intent=in,out,inout
C
С
   input arguments:
      variable (typ,siz,intent)
C
                                    description
С
               (chr*80,ar(2),in) - main title and 1st subtitle
С
      jobnam (chr*8,sc,in)
                                  - jobname
      units
               (int.sc.in)
                                  - units
C
                                    = 0 - user defined units
C
                                    = 1 - SI (MKS)
С
                                    = 2 - CSG
С
```

```
= 3 - U.S. Customary, using feet
С
                                    = 4 - U.S. Customary, using inches
С
С
                                    = 6 - MPA
                                     = 7 - uMKS
С
                                   - code defining 3rd party vendor
С
      code
C
                                     (contact ANSYS, Inc. for code assignment)
C
   output arguments:
C
      none
```

2.1.10. Subroutine binhed (Writing the Standard ANSYS File Header)

```
*deck,binhed
      subroutine binhed (nblk,nunit,filpos,buffer)
c *** primary function:
                           put standard header on a binary file, all
                           permanent binary files should have this header
c *** secondary functions: return the first data position
c --- This routine is intended to be used in standalone programs.
c --- This routine should not be linked into the ANSYS program.
c *** Notice - This file contains ANSYS Confidential information ***
   input arguments:
      nblk
               (int,sc,in)
                                  - block number of open binary file
C
                                    (as defined with subroutine binset)
C
      nunit.
                                  - the unit number for this file
C
               (int.sc.in)
С
               (int,ar(*),inout) - work array for paging, should be the
                                    same array as used in binset
C
С
   output arguments:
      filpos
                                  - the position after the header
С
               (int,sc,out)
      buffer
               (int,ar(*),inout) - work array for paging
С
    ******* ANSYS standard header data description (100 words) ********
C
C
   loc
        no. words
                    contents
    1
            1
                     fortran unit number
C
            2
    2
                     file format
C
С
                     = 0 - internal format
                     = 1 - external format
C
            1
                     time in compact form (ie 130619 is 13:06:19)
    3
C
                     date in compact form (ie 19981023 is 10/23/1998)
            1
С
            1
                     units
                     = 0 - user defined units
С
С
                     = 1 - SI (MKS)
                     = 2 - CSG
С
                     = 3 - U.S. Customary, using feet
С
С
                     = 4 - U.S. Customary, using inches
                     = 6 - MPA
C
                     = 7 - uMKS
С
С
            1
                     User_Linked
   10
                     revision in text format ' 5.0' (inexc4)
C
            1
                     date of revision release for this version
С
  12
            3
                     machine identifier - 3 4-character strings
С
   15
                     jobname - 2 4-character strings
   17
                     product name - 2 4-character strings
С
                     special version label - 1 4-character string
   19
            1
C
   20
                     user name - 3 4-character strings
С
   23
            3
                     machine identifier - 3 4-character strings
   26
            1
                     system record size at file write
C
   27
С
                     maximum file length
С
   28
            1
                     maximum record number
  31
            8
                     jobname - 8 4-character strings
C
                     main title - 20 4-character strings
С
           20
           20
C
   61
                     first subtitle - 20 4-character strings
С
                     split point of file
   95
                      NOTE: Split files are not support by binlib!
                     LONGINT of file size at write
c 97-98
```

2.1.11. Subroutine binrd8 (Reading Data from a Buffered File)

```
*deck,binrd8
      subroutine binrd8 (nblk,LongLocL,leng,ivect,kbfint,Buffer4)
c ******* buffer read routine *******
c *** Notice - This file contains ANSYS Confidential information ***
   input arguments:
   nblk
         (int,sc,in)
                           - block number. see fd___(i.e. fdtri for tri
C
C
                             (as defined with subroutine bioset)
C
   LongLocL(LONG,sc,inout) - location in integer*4 words of the startin
С
                             position on the file.
С
          (int,sc,inout) - number of words to read into ivect. (must be
С
    leng
                             less or equal to dimension given to ivect in
C
C
                             the calling routine). if ivect is to be used
                             as integers, use as is. if ivect is to be
C
С
                             used for double precision numbers, it must be
                             increased by multiplying it by INTPDP.
C
                             if negative, skip record and do not return
C
С
                             data(results).
                             data(results).
C
    Buffer4 (i4, ar(*),inout) - work array for paging, should be the
C
С
                                    same array as used in binset
  output arguments:
C
С
    LongLocL(LONG,sc,inout) - location in integer*4 words of the current
                             position on the file. It is updated after
С
                             each read operation
С
           (int,sc,inout) - tells you how many items it actually read(in
С
                             integer words).
C
C
                             if zero, end of file(error case)
С
    ivect (int,ar(*),out) - results (can be either integer or double
C
                             precision in the calling routine)
С
    kbfint (int,sc,out)
                           - key for type(used only for AUX2 dump)
С
                            = 0 double precision data
                            > 0 integer data(usually the same as leng)
C
    Buffer4 (i4,ar(*),inout) - work array for paging
```

Versions of binrd8/binwrt8 exist without the "8" suffix (binrd/binwrt) that take a regular integer for the second argument. These subroutines, therefore, cannot address large files where the file position exceeds 2**31. Use the binrd8/binwrt8 versions for any new programs.

2.1.12. Subroutine binwrt8 (Writing Data to a Buffered File)

```
*deck,binwrt8
      subroutine binwrt8 (nblk,LongLocL,leng,ivect,kbfint,Buffer4)
c *** primary function: buffer write routine
\ensuremath{\text{c}} --- This routine is intended to be used in standalone programs.
c --- This routine should not be linked into the ANSYS program.
c *** Notice - This file contains ANSYS Confidential information ***
  input arguments:
С
С
   nblk
          (int,sc,in)
                           - block number. see fd___(i.e. fdtri for tri
С
    LongLocL(LONG,sc,inout) - location in integer words of the starting
С
                              position on the file.
С
С
    leng
           (int,sc,in)
                           - number of words to read from ivect. (must be
                              less or equal to dimension given to ivect in
```

```
the calling routine). if ivect is to be used
C
                             as integers, use as is. if ivect is to be
С
С
                             used for double precision numbers, it must be
                             increased by multiplying it by INTPDP.
С
    ivect (int,ar(*),in) - data to be written onto the file(can be either
С
                             integer or double precision in the calling
                             routine)
C
C
   kbfint (int,sc,in)
                          - key for type(used only for AUX2 dump)
                            = 0 double precision data
C
С
                            > 0 integer data(usually the same as leng)
    Buffer4 (int,ar(*),inout) - work array for paging, should be the
С
                                    same array as used in binset on this
С
                                    block
С
C
   output arguments:
    LongLocL(LONG,sc,inout) - location in integer words of the current
С
C
                             position on the file. It is updated after
                             each write operation
C
   ivect (int,ar(*),out)- vector containing record to be written
   Buffer4 (int,ar(*),inout) - work array for paging
```

Versions of binrd8/binwrt8 exist without the "8" suffix (binrd/binwrt) that take a regular integer for the second argument. These subroutines, therefore, cannot address large files where the file position exceeds 2**31. Use the binrd8/binwrt8 versions for any new programs.

2.1.13. Subroutine exinc4 (Decoding an Integer String into a Character String)

```
*deck.exinc4
      subroutine exinc4 (ichext,chin,n)
c primary function: decode externally formatted integer versions of 4-character
                    strings to plain 4-character strings (used to convert data
С
                    from externally formatted files to data for interally
                    formatted files)
C
C
c *** Notice - This file contains ANSYS Confidential information ***
C
   input arguments:
С
С
      ichext
               (int,ar(n),in)
                                  - externally formatted integer form of
                                    4-character strings
C
С
                (int,sc,in)
                                  - number of strings to convert
С
  output arguments:
C
С
              (char,ar(n),out) - strings in character form
С
c *** mpg exinc4 < sectionlist ansres getsecnm: int -> ch4 conversion
```

2.1.14. Subroutine inexc4 (Coding a Character String into an Integer String)

```
*deck.inexc4
      subroutine inexc4 (chin,ichext,n)
c primary function: encode plain 4-character strings into externally formatted
                    integer versions of 4-character strings (used to convert
C
                    data from internally formatted files to data for
C
С
                    externally formatted files)
С
c *** Notice - This file contains ANSYS Confidential information ***
С
   input arguments:
С
               (char,ar(n),in) - strings in character form
C
               (int.sc.in)
                                  - number of strings to convert
C
С
  output arguments:
```

2.1.15. Subroutine binclo (Closing or Deleting a Blocked Binary File)

```
*deck,binclo
      subroutine binclo (nblk,pstat,Buffer4)
c *** primary function: close a blocked file, every block/file opened with
                        binset should be closed with binclo
c *** secondary function: the file can be deleted by specifying 'D' in pstat
c --- This routine is intended to be used in standalone programs.
c --- This routine should not be linked into the ANSYS program.
c *** Notice - This file contains ANSYS Confidential information ***
  input arguments:
C
C
      nblk
               (int,sc,in)
                                 - the block number to close
                                    (as defined with subroutine binset)
С
С
     pstat (chr,sc,in)
                                  - keep or delete flag
                                   = 'K' - keep file
С
                                   = 'D' - delete file
C
С
      Buffer4 (int,ar(*),inout) - work array for paging, should be the
                                    same array as used in binset
С
c output arguments:
      Buffer4
               (int,ar(*),inout) - work array for paging
```

2.1.16. Subroutine largeIntGet (Converting Two Integers into a Pointer)

2.2. Demonstration Routines

The demonstration routines demonstrate several ways to use the binary file access routines provided with ANSYS. The programs described below (all available on your distribution media; see *Accessing ANSYS Binary Files* (p. 53) for their location) demonstrate other tasks that the binary access routines can do.

2.2.1. Program bintst (Demonstrates Dumping a Binary File and Copying It for Comparison Purposes)

The bintst program dumps a binary file with the name file.rst to the screen. It then takes that file, copies it to a new file, file2.rst, and dumps the new file to the screen for comparison purposes.

2.2.1.1. Common Variables:

Variable	Type, Size, Intent	Description
iout	int, sc, comm	The output unit number
intpdp	int, sc, comm	The number of integers per double precision word
lenfrm	int, sc, comm	The number of characters in the filename
reclng	int, sc, comm	The system record length

Note

The bintst program is not part of the binlib.a library. It is included here only to aid you.

2.2.2. Subroutine bintrd (Demonstrates Printing a Dump of File Contents)

```
*deck.bintrd
     subroutine bintrd (pname)
c *** primary function: bin file dump utility
c *** Notice - This file contains ANSYS Confidential information ***
c *** ansys(r) copyright(c) 2008
c *** ansys, inc.
      typ=int,dp,log,chr,dcp siz=sc,ar(n)
                                            intent=in,out,inout
С
С
С
 input arguments:
     variable (typ,siz,intent)
                                 description
C
                                 - name of binary file which is to
С
              (chr,sc,in)
С
                                   be dumped to the screen
С
 output arguments:
C
С
     none.
С
   common variables:
С
     iout (int,sc,comm)
С

    output unit number

     intpdp
             (int,sc,comm)
                               - number of integers per double precision word
С
С
     lenfnm (int,sc,comm)
                                - number of characters in the filename
     reclng (int,sc,comm)
                                 - system record length
С
С
                           NOTE: bintrd is not part of binlib.a. it is
С
                                 included only as an aid to users.
C
```

Note

The bintrd routine and the bintwr routine described below are not part of binlib.a. This chapter includes it only to aid you. You can find the source for this routine on the ANSYS distribution media.

Both subroutines require the following common:

COMMON/BINTCM/ IOUT, INTPDP, LENFNM, RECLNG

- *Iout* is the output unit number.
- Intpdp is the number of integers per double precision word.
- Lenfnm is the number of characters in the filename.
- Reclng is the system record length.

2.2.3. Subroutine bintwr (Demonstrates Copying Binary File Contents)

```
*deck,bintwr
     subroutine bintwr (pname, nname)
c *** primary function: bin file copy utility
c *** Notice - This file contains ANSYS Confidential information ***
c *** ansys(r) copyright(c) 2008
c *** ansys, inc.
С
     typ=int,dp,log,chr,dcp siz=sc,ar(n) intent=in,out,inout
C
С
С
  input arguments:
    variable (typ,siz,intent)
                                 description
C
С
     pname
               (chr,sc,in)
                           - name of binary file which is to be copied
С
c output arguments:
     variable (typ,siz,intent) description
С
     nname (chr,sc,out) - name of new binary file which is a copy
C
                                  of pname
С
c common variables:
     iout (int,sc,comm)
C
                               - output unit number
                               - number of integers per double precision word
С
     intpdp
              (int,sc,comm)
С
     lenfnm
             (int,sc,comm)
                               - number of characters in the filename
С
     reclng
              (int,sc,comm)
                              - system record length
С
                          NOTE: bintwr is not part of binlib.a. it is
С
                                included only as an aid to users.
С
```

2.2.4. Program wrtsub (Demonstrates Writing an ANSYS Substructure File)

2.2.5. Program rdsubs (Demonstrates Reading a Substructure File)

Subroutine rdsubs demonstrates how you read an ANSYS substructure file. This demonstration program can handle up to MAXNODE nodes and MAXDOF degrees of freedom.

```
*deck,rdsubs
     program rdsubs
c primary function:
                    demonstrates use of binary access routines
c secondary function: read an ANSYS substructure file
c *** Notice - This file contains ANSYS Confidential information ***
c *** copyright(c) 2008 SAS IP, Inc. All rights reserved.
c *** ansys, inc.
  ******************
C
  * Reads a substructure file. To be used as base for 3rd party
C
  * development of routines for reading ANSYS substructure files.
С
     This demonstration program can handle up to:
     MAXNODE Nodes and MAXDOF DOFs
```

2.2.6. Program rdfull (Demonstrates Reading and Reformatting the .FULL File)

Program rdfull demonstrates how to read and reformat the .FULL file. ANSYS writes the full file if the **PSOLVE**,ELFORM, **PSOLVE**,ELPREP, **PSOLVE**,TRIANG sequence is used. You can also use the **WRFULL** command.

If you want to use the free stiffness and mass matrices, make sure that there are no constraints on your model.

```
*deck,rdfull
     program rdfull
c *** primary function:
                           demonstrates use of binary access routines
c *** secondary function: Read and reformat full file
c *** copyright(c) 2008 SAS IP, Inc. All rights reserved.
c *** ansys, inc.
С
  NOTICE - A new assembly process, termed 'symbolic assembly', has
С
           replaced the old assembly process, termed 'frontal
C
С
            assembly', and is now the default assembly process for
            most analyses. This program demonstrates how to read
            and reformat the .FULL file that was created using
C
            frontal assembly or symbolic assembly. ANSYS writes the
С
С
            .FULL file if the PSOLVE, ELFORM
                              PSOLVE, ELPREP
C
С
                              PSOLVE, TRIANG
            sequence is used. ANSYS will also write the .FULL file
C
С
            when the sparse, ICCG, or JCG solver is used, as well as
            when most mode extraction methods are used.
С
С
    Be sure to set up for modal ANTYPE, 2
C
     and Block Lanczos
                                  MODOPT, LANB, nmode, 0, 0, OFF
        (nmode is not used - it can be any value
C
С
       If the free-free stiffness and mass matrices are desired,
         make sure there are no constraints on the model.
C
```

2.2.7. Program ResRdDemo (Demonstrates Reading a Results File)

Program ResRdDemo demonstrates how to read a results file using the results file access routines. The file must be named test.rst and the file contents are written to the screen.

This file resides in the subdirectory \Program Files\Ansys Inc\V120\ANSYS\custom\misc\<platform> (on Windows systems) or /ansys_inc/v120/ansys/customize/misc (on UNIX systems).

2.2.8. Program ResWrDemo (Demonstrates Writing a Results File)

Program ResWrDemo demonstrates how to write an ANSYS-readable results file. This file resides in the subdirectory \Program Files\Ansys Inc\V120\ANSYS\custom\misc\<platform> (on Windows systems) or /ansys_inc/v120/ansys/customize/misc (on UNIX systems).

2.3. Results File Access Routines

You can use the low-level routines in described in *Accessing ANSYS Binary Files* (p. 53) to retrieve data from the results file. Alternatively, you can use the routines described in this section that retrieve the data specific to the format of the results file.

These files reside in the subdirectory \Program Files\Ansys Inc\V120\ANSYS\cus-tom\misc\<platform> (on Windows systems) or /ansys_inc/v120/ansys/customize/misc (on UNIX systems). See Access Routines to Results, Substructure, and Matrix Files (p. 53) for information on compiling and linking these routines.

2.3.1. Overview of the Routines

For each data record in the results file, routines exist that:

- Read the record index and allocate space for the data. These are named ResRdrecordBegin, where record is a descriptive name of the record, e.g., ResRdNodeBegin
- Read the data itself. These are named ResRdrecord, e.g., ResRdNode
- Deallocate space for the data. These are named ResRdrecordEnd, e.g., ResRdNodeEnd

Below is a complete listing of all the routines with the indentation indicating the required nested calling sequence:

```
function ResRdBegin (Nunit, Lunit, Fname, ncFname, Title, JobName,
  subroutine ResRdGeomBegin (MaxType, MaxReal, MaxCsys)
    subroutine ResRdTypeBegin (NumType)
      function ResRdType (itype,ielc)
    subroutine ResRdTypeEnd
    subroutine ResRdRealBegin (NumReal,NumPerReal)
      function ResRdReal (iReal,Rcon)
    subroutine ResRdRealEnd
    subroutine ResRdCsysBegin (NumCsys)
      function ResRdCsys (iCsys,Csys)
    subroutine ResRdCsysEnd
    subroutine ResRdNodeBegin
      function ResRdNode (iNode, xyzang)
    subroutine ResRdNodeEnd
    subroutine ResRdElemBegin
      function ResRdElem (iElem, nodes, ElemData)
    subroutine ResRdElemEnd
  subroutine ResRdGeomEnd
  function ResRdSolBegin (key, lstep, substep, ncumit, kcmplx, time,
    subroutine ResRdDispBegin
      function ResRdDisp (node, Disp)
```

```
subroutine ResRdDispEnd
   subroutine ResRdRforBegin (nRForce)
      function ResRdRfor (node,idof,value)
   subroutine ResRdRforEnd
   subroutine ResRdBCBegin (BCHeader)
      subroutine ResRdFixBegin (BCHeader,nFixed)
        function ResRdFix (node,idof,value)
      subroutine ResRdFixEnd
      subroutine ResRdForcBegin (BCHeader,nForces)
        function ResRdForc (node,idof,value)
      subroutine ResRdForcEnd
   subroutine ResRdBCEnd
   subroutine ResRdEresBegin
      function ResRdEstrBegin (iElem)
        function ResRdEstr (iStr.Str)
      subroutine ResRdEstrEnd
   subroutine ResRdEresEnd
 subroutine ResRdSolEnd
subroutine ResRdEnd
```

These routines are contained in the file ResRd.F. See the demonstration routine ResRdDemo.F on the distribution medium for an example of the usage of these routines.

The memory allocation scheme is described in Memory Management Routines in the *Guide to ANSYS User Programmable Features*.

The following sections describe the data-reading routines. See the file ResRd.F and its corresponding include deck ResRd.inc for listings of the corresponding Begin/End routines.

2.3.2. ResRdBegin (Opening the File and Retrieving Global Information)

```
*deck,ResRdBegin
     function ResRdBegin (Nunit, Lunit, Fname, ncFname, Title, JobName,
                          Units, NumDOF, DOF, UserCode,
                          MaxNode, NumNode, MaxElem, NumElem,
                          MaxResultSet, NumResultSet)
    x
c primary function:
                      Open result file and return global information
c object/library: ResRd
  input arguments:
                               - Fortran Unit number for file (ANSYS uses 12)
     Nunit (int.sc.in)
C
С
     Lunit
                              - Current print output unit (usually 6 <STDOUT>)
C
     Fname
              (ch*(ncFname),sc,in) - The name (with extension) for the file
     ncFname (int,sc,in)
                               - Number of characters in Fname
C
 output arguments:
C
     Title (ch*80,ar(2),out) - Title and First subtitle
С
С
     JobName (ch*32,sc,out) - Jobname from file
     Units (int,sc,out)
                                - 0, unknown 1, SI 2, CSG
C
С
                                  3, U.S. Customary - foot
                                 4, U.S. Customary - inch
С
                                 6, MPA
C
     NumDOF
              (int,sc,out)
                               - Number of DOF per node
С
С
     DOF
              (int,ar(*),out) - The DOFs per node
     UserCode (int,sc,out)
                               - Code for this application
C
     MaxNode (int,sc,out)
                               - Maximum node number used
С
     NumNode (int,sc,out)
                               - Number of nodes attached to elements
C
     MaxElem (int,sc,out)
C
                               - Maximum element number used
С
     NumElem (int,sc,out)
                               - Number of elements used
     MaxResultSet (int,sc,out) - Maximum number of result sets (usually 1000)
C
     NumResultSet (int,sc,out) - Number of result sets on file
C
                               - 0, successful other, error in file open
     ResRdBegin (int,sc,out)
```

2.3.3. ResRdGeomBegin (Retrieving Global Geometry Information)

2.3.4. ResRdType (Retrieving Element Types)

```
*deck,ResRdType
    function ResRdType (itype,ielc)
c primary function: Read an element type record

c object/library: ResRd

c input arguments:
    itype (int,sc,on) - Element type number

c output arguments: none
    ielc (int,ar(IELCSZ),out) - Element characteristics
    ResRdType (int,sc,out - number of words read
```

2.3.5. ResRdReal (Retrieving Real Constants)

2.3.6. ResRdCsys (Retrieving Coordinate Systems)

2.3.7. ResRdNode (Retrieving Nodal Coordinates)

```
*deck,ResRdNode
     function ResRdNode (iNode, xyzang)
c primary function:
                      Get a node
c object/library: ResRd
  input arguments:
                               - node sequence number
     iNode
            (int,sc,in)
C
                                   (1 to nXYZnode)
С
  output arguments:
C
              (dp,ar(6),out) - x,y,z,thxy,thyz,thzx for node
С
     ResRdNode (int,sc,out)
                                - Node number
```

2.3.8. ResRdElem (Retrieving Elements)

```
*deck,ResRdElem
      function ResRdElem (iElem, nodes, ElemData)
c primary function: Read an element
c object/library: ResRd
   input arguments:
      iElem (int,sc,in)
                                - The element number
C
С
   output arguments:
                                 - Number of nodes
C
     ResRdElem(int,sc,out)
С
     nodes
            (int,ar(n),out)
                                - Element nodes
     ElemData (int,ar(10),out) - Element information
C
                                           - material reference number
С
                                     type - element type number
C
                                           - real constant reference number
C
                                     real
С
                                     secnum - section number
С
                                     esvs
                                          - element coordinate system
                                     death - death flag
C
                                             = 0 - alive
С
                                             = 1 - dead
С
С
                                     solidm - solid model reference
                                     shape - coded shape key
C
                                     elnum - element number
C
                                     pexcl - P-Method exclude key
```

2.3.9. ResRdSolBegin (Retrieving Result Set Location)

```
*deck,ResRdSolBegin
     function ResRdSolBegin (key, lstep, substep, ncumit, kcmplx, time,
                             Title, DofLab)
                      Read the solution header records
c primary function:
c object/library: ResRd
  input arguments:
C
                                - 1, find by lstep/substep
С
     key (int,sc,in)
                                  2, find by ncumit
С
С
                                  3, find by time
C
     lstep
              (int,sc,in/out)
                                - Load step number
                                - Substep of this load step
С
     substep
              (int,sc,in/out)
С
     ncumit
              (int,sc,in/out)
                                - Cumulative iteration number
                                - 0, Real solution 1, Imaginary solution
             (int,sc,in)
C
     kcmplx
С
              (int,sc,in/out)
                               - Current solution time
C
  output arguments:
     Title
              (ch*80,ar(5),out) - Title and 4 subtitles
C
              (ch*4,ar(nDOF),out)- Labels for DOFs
C
     DofLab
```

```
c ResRdSolBegin (int,sc,out) - 0, requested solution set found
c 1, not found
```

2.3.10. ResRdDisp (Retrieving Nodal Solution)

```
*deck,ResRdDisp
    function ResRdDisp (node,Disp)
c primary function: Retrieve a nodal displacement

c object/library: ResRd

c input arguments:
    node (int,sc,in) - Node number

c output arguments: none
    Disp (dp,ar(nDOF),out) - Displacements
c ResRdDisp(int,sc,out) - Number of displacements
```

2.3.11. ResRdRfor (Retrieving Reaction Solution)

```
*deck,ResRdRfor
     function ResRdRfor (node,idof,value)
c primary function: Retrieve a reaction force
c object/library: ResRd
 input arguments:
                         - External node number
    node (int,sc,in)
С
С
     idof
             (int,sc,in)
                              - Internal dof number
c output arguments:
                              - Value of reaction force
     value (dp,sc,in)
С
     ResRdRfor (int,sc,out)
                              - Number of returned values (0 or 1)
```

2.3.12. ResRdFix (Retrieving Applied Nodal Constraints)

```
*deck,ResRdFix
      function ResRdFix (node,idof,value)
c primary function: Retrieve a constraint value
c object/library: ResRd
c input arguments:
С
     node (int,sc,in)
                               - External node number
                               - Internal dof number
     idof
              (int,sc,in)
С
 output arguments:
     value (dp,ar(4),in)
                               - Real, Imag, RealOld, ImagOld
C
     ResRdFix (int,sc,out)
                               - Number of returned values (0 or 4)
```

2.3.13. ResRdForc (Retrieving Applied Nodal Loads Solution)

```
*deck,ResRdForc
     function ResRdForc (node,idof,value)
c primary function: Retrieve an applied force value
c object/library: ResRd
c input arguments:
     node (int,sc,in) - External node number
С
     idof
             (int,sc,in)
                               - Internal dof number
  output arguments:
C
     value
            (dp,ar(4),in)
                                - Real, Imag, RealOld, ImagOld
C
     ResRdForc (int,sc,out)
                                - Number of returned values (0 or 4)
С
```

2.3.14. ResRdEstr (Retrieving Element Solutions)

```
*deck,ResRdEstr
     function ResRdEstr (iStr,Str)
c primary function: Get an element's results
c object/library: ResRd
  input arguments:
С
     iStr
             (int,sc,in)
                               - element record number (1-25)
  output arguments:
С
С
     ResRdEstr (int,sc,out)
                                  - Number of element values
             (dp,ar(nStr),out) - element values
С
```

Chapter 3: The CDWRITE (CDB) File Format

This chapter discusses how to write a coded database file, <code>Jobname.CDB</code>, that can be used to export a model from ANSYS into another application. The <code>Jobname.cdb</code> file contains model data in terms of ANSYS input commands.

The following topics are discussed:

- 3.1. Using the CDWRITE Command
- 3.2. Coded Database File Commands

3.1. Using the CDWRITE Command

To export a model from the ANSYS program to another application, use menu path **Main Menu> Preprocessor> Archive Model> Write** or the **CDWRITE** command within the general preprocessor, PREP7. This produces a coded database file called Jobname.cdb. You specify the jobname using **Utility Menu> File> Change Jobname** or the **/FILNAME** command. If you supply no jobname, the ANSYS program uses the default name "file".

The Jobname.cdb file contains selected geometry (nodes and elements), load items, and other model data in terms of ANSYS input commands. (For a complete list of data in the file, see the **CDWRITE** description in the *Command Reference*.) You can convert this information to a format compatible with the program into which you are importing it. The next few pages describe special considerations and commands you may need to do this conversion.

Note

Files created by the **CDWRITE** command have the active coordinate system set to Cartesian (**CSYS**, 0).

ANSYS may create parameters in the **CDWRITE** file that start with an underscore (_), usually an "_z." Such parameters are for ANSYS internal use and pass information to the ANSYS GUI.

3.1.1. Customizing Degree of Freedom Labels: the /DFLAB Command

The ANSYS program uses a set of default labels for the degrees of freedom. You use these labels when entering boundary conditions, or ANSYS uses the labels when writing the Jobname. cdb file.

You can change the labels to reflect the degrees of freedom of the other program by issuing the command /DFLAB. If you are customizing the DOF labels, /DFLAB must be the first command you enter within the ANSYS program. You may want to include the command in your START. ANSfile. You can use /DFLAB only at the Begin processing level.

/DFLAB assigns or reassigns the "displacement" and "force" labels in the ANSYS DOF list. For example, degree of number 1 is predefined to have a displacement label of UX and a force label of FX, but you can assign new labels to this DOF using by issuing **/DFLAB**. Changing predefined labels generates a warning message.

The format for the /DFLAB command is:

/DFLAB, NDOF, LabD, LabF

NDOF

ANSYS degree of freedom number (1 to 32)

LabD

Displacement degree of freedom label to be assigned (up to four characters)

LabF

Force label to be assigned (up to four characters)

You can also use /**DFLAB** to assign labels to spare degree of freedom numbers. Spare displacement and force labels are from 13 to 18 and from 27 to 32. All other DOF numbers are predefined, as follows:

DOF Num- ber	Corresponding Displacement Label	Corresponding Force Label
1	UX	FX
2	UY	FY
3	UZ	FZ
4	ROTX	MX
5	ROTY	MY
6	ROTZ	MZ
7	AX	CSGX
8	AY	CSGY
9	AZ	CSGZ
10	VX	VFX
11	VY	VFY
12	VZ	VFZ
19	PRES	FLOW
20	TEMP	HEAT
21	VOLT	AMPS
22	MAG	FLUX
23	ENKE	NPKE
24	ENDS	NPDS
25	EMF	CURT
26	CURR	VLTG

3.2. Coded Database File Commands

In the coded database file Jobname. CDB, most ANSYS commands have the same format they have elsewhere. (See the *Command Reference* for command-specific information.) However, the format for some commands differs slightly in the Jobname. CDB file. The format for these commands is described below.

The **CDWRITE** command has an UNBLOCKED and a BLOCKED option. The UNBLOCKED option will write all data out in command format, the default BLOCKED option will write certain data items in a fixed format, especially those which could potentially contain large amounts of data, such as nodal data.

3.2.1. CE Command

The **CE** command defines the constant term in a constraint equation. The command format in Jobname. CDB is:

```
CE, R5.0, Type, LENGTH, NCE, CONST
```

Type

The type of data to be defined. DEFI is the valid label.

LENGTH

The total number of variable terms in the constraint equation.

NCE

The constraint equation reference number.

CONST

The constant term of the equation.

Another version of the **CE** command defines the variable terms in a constraint equation. You must issue this version of the command after the **CE** command described above. This command repeats until all terms are defined.

The alternate format for the **CE** command is:

```
CE, R5.0, Type, N1, Dlab1, C1, N2, Dlab2, C2
```

Type

The type of data to be defined. NODE is the valid label.

N1

The node number of the next term.

Dlab1

The DOF label of N1.

C1

The coefficient of N1.

N2

The node number of the next term.

Dlab2

The DOF label of N2.

C2

The coefficient of N2.

3.2.2. CP Command

The **CP** command defines a coupled node set. You repeat the command until all nodes are defined. The command format in Jobname . CDB is:

```
CP,R5.0,LENGTH,NCP,Dlab,N1,N2,N3,N4,N5,N6,N7
```

LENGTH

The total number of nodes in the coupled set

NCP

The coupled node reference number

Dlab

The degree of freedom label for the set

N1,N2,N3,N4,N5,N6,N7

The next seven node numbers in the coupled set

3.2.3. CMBLOCK Command

The **CMBLOCK** command defines the entities contained in a node or element component. The command format in Johname. CDB is:

```
CMBLOCK,Cname,Entity,NUMITEMS
Format
```

Cname

Eight character component name.

Entity

Label identifying the type of component (NODE or ELEMENT).

NUMITEMS

Number of items written.

Format

Data descriptors defining the format. For **CMBLOCK** this is always (8i10).

The items contained in this component are written at 10 items per line. Additional lines are repeated as needed until all NumItems are defined. If one of the items is less than zero, then the entities from the item previous to this one (inclusive) are part of the component.

3.2.4. EBLOCK Command

The **EBLOCK** command defines a block of elements. The command syntax is:

```
EBLOCK, NUM_NODES, Solkey
Format
```

NUM NODES

The number of nodes to be read in the first line of an element definition.

Solkey

The solid model key. The element is part of a solid model if the keyword SOLID appears here. When Solkey = SOLID, Field 8 (the element shape flag) may be left at zero, and Field 9 is the number of nodes defining this element.

Format

Data descriptors defining the format.

The format of the element "block" is as follows for the SOLID format:

- Field 1 The material number.
- Field 2 The element type number.
- Field 3 The real constant number.
- Field 4 The section ID attribute (beam section) number.
- Field 5 The element coordinate system number.
- Field 6 The birth/death flag.

- Field 7 The solid model reference number.
- Field 8 The element shape flag.
- Field 9 The number of nodes defining this element if Solkey = SOLID; otherwise, Field 9 = 0.
- Field 10 The exclude key (p-elements).
- Field 11 The element number.
- Fields 12-19 The node numbers. The next line will have the additional node numbers if there are more than eight.

The format without the SOLID keyword is:

- Field 1 The element number.
- Field 2 The type of section ID.
- Field 3 The real constant number.
- Field 4 The material number.
- Field 5 The element coordinate system number.
- Fields 6-15 The node numbers. The next line will have the additional node numbers if there are more than ten.

The final line of the block will be a -1 in field 1.

If you are in the GUI, the **EBLOCK** command must be contained in an externally prepared file and read into ANSYS (i.e., **CDREAD**, **/INPUT**, etc.).

3.2.5. EDCADAPT Command

The **EDCADAPT** command specifies adaptive meshing control for explicit dynamics analysis. The command format in Johname. CDB is:

 $\verb|EDCADAPT,R5.3|, FREQ,TOL,OPT,MAXLVL,BTIME,DTIME,LCID,ADPSIZE,ADPASS,IREFLG,ADPENE,ADPTH,MAXELADPASS,ADP$

FREO

The time interval between adaptive mesh refinement.

TOL

The adaptive angle tolerance (in degrees).

OPT

The adaptivity option.

MAXLVL

The maximum number of mesh refinement levels.

BTIME

The birth time to begin adaptive meshing.

DTIME

The death time to end adaptive meshing.

LCID

The curve ID defined by **EDCURVE**

ADPSIZE

The minimum element size to be adapted, based on the element edge length.

ADPASS

The one-pass or two-pass adaptivity option.

TREFTG

The uniform refinement level flag.

ADPENE

Adaptive mesh flag for starting adaptivity when approaching (positive ADPENE) or penetrating (negative ADPENE) the tooling surface

ADPTH

Absolute shell thickness level below which adaptivity should begin.

MAXEL

The maximum number of elements at which adaptivity will be terminated.

NOTE: This command is also listed in the *Command Reference*. The format listed here contians information specific to the CDREAD/CDWRITE file.

3.2.6. EDCGEN Command

The **EDCGEN** command is used to define a contact definition for explicit dynamics. The command format in Jobname . CDB is:

EDCGEN, R5.3, Option, Cont, Targ, Lkey, FS, FD, DC, VC, VDC, V1, V2, V3, V4, BTIME, DTIME, BOXID1, BOXID2

Option

The label identifying the contact behavior.

Cont

The contact surface, identified by component name, part ID, or part assembly ID.

Targ

The target surface, identified by component name, part ID, or part assembly ID.

Lkey

A key identifying the meaning of Cont and Targ (component, part or part assembly).

FS

The static friction coefficient.

FD

The dynamic friction coefficient.

DC

The exponential decay coefficient.

VC

The coefficient of viscous friction.

VDC

The viscous damping coefficient in percent of critical damping.

V1, V2, V3, V4

Additional input for some contact types. See **EDCGEN** in the *Command Reference* for more information.

RTTMF

The birth time for which conatct definition will become active.

DTIME

The death time for which conatct definition will become inactive.

BOXID1

Contact volume as defined using EDBOX

BOXTD2

Target volume as defined using EDBOX

NOTE: This command is also listed in the *Command Reference*. The format listed here contians information specific to the CDREAD/CDWRITE file.

3.2.7. EDCURVE Command

The **EDCURVE** command is used to define a curve for an explicit dynamics analysis. The command format in Jobname . CDB is:

EDCURVE, R5.3, Option, LCID, Length, 0.0, Par1, Par2

Option

The **EDCURVE** command option. The only valid option is "ADD."

LCID

The curve ID.

Length

The number of data values for the abcissa array (Par1) and the ordinate array (Par2).

Par1

The abcissa values, repeat Length number of times.

Par2

The ordinate values, repeat Length number of times.

NOTE: This command is also listed in the *Command Reference*. The format listed here contians information specific to the CDREAD/CDWRITE file.

3.2.8. EDDRELAX Command

The **EDDRELAX** command activates initialization to a prescribed geometry or dynamic relaxation for the explicit analysis. The command format in Jobname. CDB is:

EDDRELAX, R5.4, Option, NRCYCK, IRELAL, EDTTL, DRTOL, DFFCTR, DRTERM, TSSFDR

Option

EDDRELAX command option. Valid options are "ANSYS" (relaxation is based on the implicit analysis, see the **EDDRELAX** command in the *Command Reference*) or "DYNA," where the relaxation parameters are controlled within the LS-DYNA analysis. The following arguments are valid for *Option*= DYNA only.

NRCYCK

The number of iterations between the convergence checks.

IRELAL

Automatic control based on Papadrakakis not active (0) or active (1).

EDTTL

The convergence tolerance when automatic control is used.

DRTOL

The convergence tolerance.

DFFCTR

The dynamic relaxation factor.

DRTERM

The termination time for dynamic relaxation.

TSSFDR

The scale factor for each computed time step.

NOTE: This command is also listed in the *Command Reference*. The format listed here contians information specific to the CDREAD/CDWRITE file.

3.2.9. EDLCS Command

The **EDLCS** command is used to define a local coordinate system for explicit dynamics. The command format in Jobname . CDB is:

EDLCS, R5.3, Option, CID, X1, Y1, Z1, X2, Y2, Z2, X3, Y3, Z3

Option

The **EDLCS** command option. The only valid option is "ADD."

CID

The coordinate system ID.

X1,Y1,Z1,

The X,Y,Z coordinate of a point on the local X-axis.

X2,Y2,Z2,

The X,Y,Z coordinate of a point on the local X-Y plane.

X3,Y3,Z3,

The X,Y,Z coordinate of the local origin.

NOTE: This command is also listed in the *Command Reference*. The format listed here contians information specific to the CDREAD/CDWRITE file.

3.2.10. EDLOAD Command

The **EDLOAD** command is used to define loading conditions for explicit dynamics. The command format in Jobname . CDB is:

EDLOAD, R5.3, Option, Lab, KEY, Cname, Length, PHASE, Par1, Par2, LCID, SCALE, BTIME, DTIME

Option

The **EDLOAD** command option. The only valid option is "ADD."

Lab

The load labels.

Key

The coordinate system number defined by EDLCS or the element face number for the pressure loading.

Cname

The name of the existing component or part number to which this load will be applied.

Length

The number of data values for the time array (Par1) and the load array (Par2).

Phase

Phase of the analysis in which the load curve is to be used.

Par1

The time values, with the number of values in the string defined by the Length argument (above).

Par2

The load values, with the number of values in the string defined by the Length argument (above).

LCTD

The curve ID, created using the **EDCURVE** command. If LCID is nonzero, then Length=1, and Par1 and Par2 will be equal to 0.

Scale

The Scale Factor applied to the load curve.

Btime

The birth time.

Dtime

The death time.

NOTE: This command is also listed in the *Command Reference*. The format listed here contians information specific to the CDREAD/CDWRITE file.

3.2.11. EDPREAD Command

The **EDPREAD** command is used to internally write the part information to the Jobname.CDB file for explicit dynamics. Prior to Release 8.0, the command format in Jobname.CDB is:

```
EDPREAD, R5.4, Nmat, Npart
Type, Mat, Real, Used
```

Nmat

The number of materials.

Npart

Number of parts, and also, the number of times to repeat the second Type,Mat,Real,Used input line.

Type

The element type number.

Mat

The material number.

Real

The real constant set number.

Used

The flag indicating if the part is used (1), or not used (0).

For Release 8.0 and beyond, the command format is:

```
EDPREAD,R8.0,Nmat,Npart,Part ID Type, Mat, Real, Used
```

Nmat

The number of materials.

Npart

Number of parts, and also, the number of times to repeat the second Type,Mat,Real,Used input line.

PartID

The part number.

Type

The element type number.

Mat

The material number.

Real

The real constant set number.

Used

The flag indicating how many elements use PartID. If USED = 0, PartID is not used.

3.2.12. EDWELD Command

The **EDWELD** command is used to define a spotweld or a generalized weld for explicit dynamics.

There are two command formats (for spot and generalized welds). The command format for the spotweld appears in Jobname . CDB as follows:

```
EDWELD, R5.3, Option, NWELD, N1, N2, SN, SS, EXPN, EXPS
```

Option

The **EDWELD** command option. The only valid option is "ADD."

NWELD

The spotweld ID number.

N1

The node number of the first node connected by the spotweld.

N2

The node number of the second node connected by the spotweld.

SN

The normal force at the spotweld failure.

SS

The shear force at the spotweld failure.

EXPN

The exponent for spotweld normal force.

EXPS

The exponent for spotweld shear force.

The command format for the generalized weld appears in Jobname . CDB as follows:

```
EDWELD, R5.3, Option, NWELD, CNAME,, SN, SS, EXPN, EXPS, EPSF, TFAIL, NSW, CID
```

Option

The **EDWELD** command option. The only valid option is "ADD."

NWELD

The generalized weld ID number.

CNAME

The name of the node component.

SN

The normal force at the weld failure.

SS

The shear force at the weld failure.

EXPN

The exponent for weld normal force.

EXPS

The exponent for weld shear force.

EXPF

The effective plastic strain at ductile failure.

TFAIL

The time of failure of the weld.

NSW

The number of spotwelds for the generalized weld.

CID

The coordinate system ID as defined by the **EDLCS** command.

NOTE: This command is also listed in the *Command Reference*. The format listed here contians information specific to the CDREAD/CDWRITE file.

3.2.13. **EN Command**

The **EN** command is used to define an element . If an element contains more than eight nodes, the **EN** command is repeated until all nodes are defined. The command format in Jobname . CDB is:

```
EN, R5.5, Type, NUMN, I1, I2, I3, I4, I5, I6, I7, I8
```

The type of data to be defined. Valid labels are "ATTR" (read in element attributes), and "NODE" (read in nodes defining the element).

NUMN

The number of nodes.

11,12,13,1415,16,17,18

The integer values to be read:

- If Type is ATTR, the integer values are the element attributes. Attributes are in the order: NUMN,MAT,TYPE,REAL,SECNUM,ESYS,NUMELEM,SOLID,DEATH,EXCLUDE
- If Type is NODE, the integer values are the node numbers.

3.2.14. LOCAL Command

The **LOCAL** command defines a local coordinate system. The command format in Jobname. CDB is:

```
LOCAL, R5.0, Type, NCSY, CSYTYP, VAL1, VAL2, VAL3
```

Туре

The type of data to be defined. Valid labels are LOC (read in system origin), ANG (read in rotation angles), and PRM (read in system parameters).

NCSY

The coordinate system reference number.

CSYTYP

The coordinate system type (0, 1, 2, or 3).

VAL1,VAL2,VAL3

Values to be read:

- If Type is LOC, values are the system origin in global Cartesian coordinates.
- If Type is ANG, values are the rotation angles in degrees.

If Type is PRM, values are the first and second parameters of the system.

3.2.15. M Command

The **M** command defines a master degree of freedom. The command format in Johname . CDB is:

M,R5.0, NODE, Dlab

NODE

The node number

Dlab

The DOF label

3.2.16. MPDATA Command

The **MPDATA** command defines a material property data table. You repeat the command until all properties are defined. The command format in Jobname. CDB is:

MPDATA, R5.0, LENGTH, Lab, MAT, STLOC, VAL1, VAL2, VAL3

LENGTH

The total number of temperatures in the table.

Lab

The material property label. See the MP command description in Command Reference for valid labels.

MAT

The material reference number.

STLOC

The starting location in the table for the next three property values.

VAL1,VAL2,VAL3

Property values assigned to three locations in the table starting at STLOC.

3.2.17. MPTEMP Command

The **MPTEMP** command defines a temperature table. You repeat the command until all temperature values are defined. The command format in Jobname . CDB is:

MPTEMP, R5.0, LENGTH, STLOC, TEMP1, TEMP2, TEMP3

LENGTH

The total number of temperatures in the table

STLOC

The starting location in the table for the next three temperature values

TEMP1,TEMP2,TEMP3

Temperatures assigned to three locations in the table starting at STLOC

3.2.18. N Command

If the UNBLOCKED option is used with the **CDWRITE** command, then the **N** command defines a node. This is also the method used for defining nodes in . CDB files before ANSYS 5.4. The command format in Jobname . CDB is:

N,R5.3,Type,NODE,SOLID,PARM,VAL1,VAL2,VAL3

Туре

The type of data to be defined. Valid labels are LOC (read in coordinates) and ANG (read in rotation angles).

NODE

The node number.

SOLID

The solid model reference key. Not present for Type= ANG.

PARM

Line parameter value (0 if not on line). Not present for Type= ANG.

VAL1,VAL2,VAL3

Values to be read:

- If Type is LOC, values are the coordinates in the global Cartesian system.
- If Type is ANG, values are the rotation angles in degrees.

3.2.19. NBLOCK Command

The **NBLOCK** command defines a block of nodes. This is the recommended method for inputting nodes into the ANSYS data base. The command syntax is:

```
NBLOCK, NUMFIELD, Solkey, NDMAX, NDSEL Format
```

NUMFIELD

The number of fields in the blocked format.

Solkey

The solid model key. The node is part of a solid model if the keyword SOLID appears here.

NDMAX

The maximum node defined.

NDSEL

The number of nodes written.

Format

Data descriptors defining the format.

The format of the node "block" is as follows:

- Field 1 Node number.
- Field 2 The solid model entity (if any) in which the node exists (if SOLID key).
- Field 3 The line location (if the node exists on a line and if SOLID key).
- Field 4 6 The nodal coordinates.
- Field 7 9 The rotation angles (if NUMFIELD > 3).

Only the last nonzero coordinate/rotation is output; any trailing zero values are left blank.

The final line of the block is always an **N** command using a -1 for the node number.

The following example shows a typical **NBLOCK** formatted set of node information. Note that this example has no rotational data. It contains only the first six fields.

```
NBLOCK,6,SOLID, 159707, 113145
(3i8,6e20.13)
```

```
1
                      0-8.94000000000E+02 0.0000000000E+00-2.78000000000E-14
      3
                      0-3.269000000000E+01 5.16200000000E+02 1.12000000000E+02
      4
                      0-8.831000000000E+02 3.99800000000E+02 5.84900000000E+02
 157130
                      0-1.6831820040000E+03 2.0010350750000E+03-2.000000000000E+02
 157131
              0
                      0-1.6831785460000E+03 1.9922750340000E+03-4.900000000000E+02
                      0-1.6107399970000E+03 1.9408449890000E+03 5.2888200000000E+02
 157132
              0
N,R5.3,LOC,
               -1,
```

If you are in the GUI, the **NBLOCK** command must be contained in an externally prepared file and read into ANSYS (i.e., **CDREAD**, /**INPUT**, etc.).

3.2.20. R Command

The **R** command defines a real constant set. You repeat the command until all real constants for this set are defined. The command format in Jobname . CDB is:

```
R, R5.0, NSET, Type, STLOC, VAL1, VAL2, VAL3
```

NSET

The real constant set reference number.

Type

The type of data to be defined. LOC is the valid label.

STLOC

The starting location in the table for the next three constants.

VAL1, VAL2, VAL3

Real constant values assigned to three locations in the table starting at STLOC.

3.2.21. RLBLOCK Command

The **RLBLOCK** command defines a real constant set. The real constant sets follow each set, starting with Format1 and followed by one or more Format2's, as needed. The command format is:

```
RLBLOCK, NUMSETS, MAXSET, MAXITEMS, NPERLINE Format1
Format2
```

NUMSETS

The number of real constant sets defined

MAXSET

Maximum real constant set number

MAXITEMS

Maximum number of reals in any one set

NPERLINE

Number of reals defined on a line

Format1

Data descriptor defining the format of the first line. For the **RLBLOCK** command, this is always (2i8,6g16.9). The first i8 is the set number, the second i8 is the number of values in this set, followed by up to 6 real constant values.

Format2

Data descriptors defining the format of the subsequent lines (as needed); this is always (7g16.9).

The real constant sets follow, with each set starting with Format1, and followed by one or more Format2's as needed.

3.2.22. SECBLOCK Command

The **SECBLOCK** command retrieves all mesh data for a user-defined beam section as a block of data. You repeat the command for each beam section that you want to read. The command format is:

SECBLOCK Format1 Format2 Format3

Format1

The First Line section. The first value is the number of nodes, and the second is the number of cells.

Format2

The Cells Section. The first 9 values are the cell connectivity nodes. The 10th (last) value is the material ID (MAT).

Format3

The Nodes Section. This section contains as many lines as there are nodes. In this example, there are 27 nodes, so a total of 27 lines would appear in this section. Each node line contains the node's boundary flag, the Y coordinate of the node, and the Z coordinate of the node. Currently, all node boundary flags appear as 0s in a cell mesh file. Because all node boundary flags are 0, **SECBLOCK** ignores them when it reads a cell mesh file.

Sample User Section Cell Mesh File

Following is a sample excerpt from a custom section mesh file for a section with 27 nodes, 4 cells, and 9 nodes per cell:

```
First Line:
                  27
                        4
                              3
                                        9
                                              2
                                                                   5
                                                                         2
                         1
                                  11
                                                  6
Cells Section:
                         7
                             9
                                             8
                                                                   15
                                  23
                                        21
                                                  16
                                                        2.2
                                                             14
                                                                         1
                             11
                                  25
                                        23
                                             10
                                                  18
                                                        24
                                                                   17
                         9
                                                             16
                                                                         1
                                  27
                                        25
                                             12
                                                  20
                             13
                                                        26
                                                             18
                                                                   19
Nodes Sec-
                         0.0
                                        0.0
                  0
tion:
                  Ω
                         0.025
                                        0.0
                         0.05
                                        0.0
                  0
                         5.0175
                                        0.0
                                        10.00
                  0
                         19.98
                  0
                         20.00
                                        10.00
```

3.2.23. SFBEAM Command

The **SFBEAM** command defines a surface load on selected beam elements. Remaining values associated with this specification are on a new input line with a (4f16.9) format. The command format in Jobname. CDB is:

```
SFBEAM, ELEM, LKEY, Lab, R5.0, DIOFFST, DJOFFST
```

ELEM

The element number.

LKEY

The load key associated with these surface loads.

Lab

A label indicating the type of surface load. PRES (for pressure) is the only valid label.

DIOFFST

Offset distance from node I.

DJOFFST

Offset distance from node J.

3.2.24. SFE Command

The **SFE** command defines a surface load. Values associated with this specification are on a new input line with a (4f16.9) format. The command format in Jobname. CDB is:

```
SFE, ELEM, LKEY, Lab, KEY, R5.0
```

ELEM

The element number.

LKEY

The load key associated with this surface load.

Lab

A label indicating the type of surface load: Valid labels are:

- PRES (pressure)
- CONV (convection)
- HFLU (heat flux)
- IMPD (impedance)
- SEL (substructure load vector)
- SELV (S. E. load vectors)
- CHRG (charge density)

KEY

A value key. If it is 1, the values are real (film coefficient if convection). If it is 2, values are imaginary (bulk temperature if convection).

Chapter 4: ANSYS Graphics File Format

Graphics written to a file (/**SHOW**,<filename>) are written in an ASCII coded format. This chapter provides information on the graphics file content and format. You can use the DISPLAY program to read and plot this file.

The following topics are discussed in this chapter:

- 4.1. Pixmap Format for Graphic Display Files
- 4.2. Neutral Graphics File Format
- 4.3. Decoding a Graphics File: an Example

4.1. Pixmap Format for Graphic Display Files

The ANSYS graphics display is KPX pixels high by KPX * 1.33 pixels wide.

KPX is the resolution specified by the /GFILE,SIZE command (where SIZE is the pixel resolution) or by choosing menu path Utility Menu>PlotCtrls>Redirect Plots>To File. Default resolution is 800.

IX1,IY1 is the lower left corner of the z-buffer image.

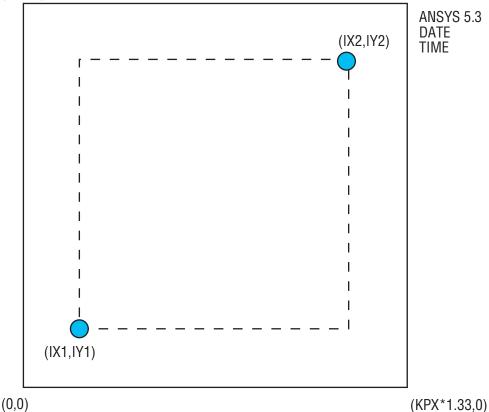
IX2,IY2 is the upper right corner of the z-buffer image.

The image should be mapped to the hardcopy device accordingly.

The following graphic illustrates the items described above:

Figure 4.1: Display Format for Z-buffered Graphics





4.2. Neutral Graphics File Format

The neutral graphics file is an 80-byte, ASCII coded file with fixed length records. It contains plot directives representing the image of a display, as formed in ANSYS, encoded onto a host-independent, printable character set.

Most ANSYS users will not need to know the format of the graphics file. However, in rare cases, you may want to edit your graphics file or, as a programmer, you may need to know the file format to write a program that reads it. Although the file is ASCII coded, it can be difficult to interpret. This section gives details about the file format.

4.2.1. Characters the Graphics File Uses

The host-independent printable character set consists of the ASCII characters listed below:

- Numerals 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9
- · Uppercase alphabetic characters A through Z
- The following characters: \$ () * + , . < = >
- The space character, " ".

4.2.2. Graphics File Directives

Graphics files contain a set of directives that define various aspects of how ANSYS displays a plot, such as window coordinates, colors for graphs and test, line dimensions, and so on. Each directive consists of a command character followed by one or more parameters.

Within a graphics file, one directive directly follows the preceding directive. For example, below is the first line of a graphics file:

(BBAAA2A0AAAAAPPPLPO>AP\$MEKLKBAJANSYS 5.3\$MEKLEFALNOV 15 1996\$MEKKOJAI10:01:40

The text of this example line breaks down as follows:

(BBAAA	The Start-Plot directive, beginning with command character. (B, B, A, A, and A are the values of parameters defining the plot environment. (Parameters for all plot directives, and their possible values, are explained later.)
2A	The Text-Size directive, which determines the type size of displayed text strings. The 2 is the command character, and A represents the size value.
0aaaaaappplpO	The Window directive, which sets the coordinates for the displayed image. 0 is the command character. AAAAAA represents the first set of coordinates (the lower left corner of the image), and PPPLPO represents the second coordinate set (the right upper corner of the image).
>AP	The Text-Color directive, which sets the color of displayed text. > is the command character. AP is a parameter value specifying the color.
\$MEKLKBAJANSYS 5.3	The first of several Text directives. \$ is the command character, MEKLKB are the coordinates for the text, AJ is the number of characters in the string, and ANSYS 5.3 is the text string itself.
\$MEKLEFALNOV 15 1996	A second Text directive, defining the position and length of the string NOV 15 1996.
\$MEKKOJAI10:01:40	A third Text directive, defining the position and length of the string 10:01:40

4.2.2.1. Parameter Types for Graphics File Directives

The descriptions of graphics file directives in the next section include discussions of the parameter or parameters for each directive. There are five types of parameters:

Parameter Type	Parameter Attributes	Valid Parameter Values
Int	1 byte, base 16 (letters A through P)	0 through 15

Parameter Type	Parameter Attributes	Valid Parameter Values
Long	2 bytes, base 16 (letters A through P)	0 through 255
Byt3	3 bytes, base 16 (letters A through P)	0 through 65535
Ху	6 bytes, base 16 (letters A through P)	0 through 4095, mapped to coordinate space of -1.0 to 1.67
String	An array of <i>Nchar</i> characters	Characters from the common character set.

4.2.2.2. Directive Descriptions

The next few pages describe each of the graphics file directives. Parameters are always specified in the order shown below.

Graphics Directive	Command Character	Parameters	Parameter Types
Start_Plot	(keras - Defines whether the display surface is cleared prior to the plot $(0 = do not clear)$ the surface, $1 = clear$ it)	Int, Int, Int, Int
		kras - Defines whether the display uses raster mode or vector mode (1 = raster mode, 0 = vector mode)	
		kcntr - Defines whether the display uses a contour color map or shading color map (1 = contour, 0 = shading)	
		kdocu - Defines whether the Docu column is compressed (1 = do not compress, 0 = compress)	
		ispare - A spare value	
Window	0	x1,y1,x2,y2 (x and y coordinates)	Xy, Xy
Area-Color	<	iclra - Sets the color for the displayed area. (See "Color Specification" below.)	Long
Graph-Color	=	iclrg - Sets the color for the displayed graph. (See "Color Specification" below.)	Long
Text-Color	>	icIrt - Sets the color for displayed text. (See "Color Specification" below.)	Long
Text-Size	2	tsize - Defines the size of displayed text (0 = normal, 1 = small)	Int
Line-Type	,	Itype - Defines the type of lines used in the display (0 = solid, 1 = dashed)	Int
Line-Width	1	lwidth - Defines the width of displayed lines (0 = normal, 1 to 5 = larger line size)	Int

Graphics Directive	Command Character	Parameters	Parameter Types
Marker Size	3	size - Defines the size of the node marker (0 = the smallest size, 15 = the largest size)	int
Anno Size	4	Annotation text size * 1000	Long
Pixmap Style	5	= 1 - Do not include background pixels	Int
Font Control	6	= 1 - Small font	Int
Text Justifica- tion	7	= 0 - Right justified = 1 - Left justified	Int
Point	*	x,y - Defines a point at coordinates x,y	Xy
Move		x,y - Moves to coordinates x,y	Xy
Draw	<u> -</u>	x,y - Draws a line to coordinates x,y	Xy
Text	\$	x,y - Sets coordinates for where text will display nchar - Defines the number of displayed characters	Xy, Long, String
		string - Defines the text string itself	
Normal	/	inorm - This value, divided by 255, is cos(), where is the viewing direction and the surface normal of subsequent polygons	Long
Polygon	+	n - Defines the number of polygon vertices kedge - Defines whether the polygon edge is displayed (0, = do not display edge, 1 = display it) xy1,xyn - Defines coordinates for the polygon	Int, Int, XyXy
No-Op	none	no parameters	none
End-Plot)	no parameters	none
Pixmap Z	Z	kpx - Defines the pixel resolution x1,y1 - Sets coordinates for the lower left corner of the z-buffer image x2,y2 - Sets coordinates for the upper right corner of the z-buffer image The following three parameters are runlength encoded data which repeats until all pixels are read in, as defined by the window	Byt3, Xy, Xy, Long, Long, Long,
		n - Defines the number of pixels in a row inrm - Sets the normal for pixels	

 Command Character		Parameter Types
	iclr - Sets the color for the pixmap	

4.2.2.3. Color Specification

Below is the list of color specifications used by the directives that set colors for areas, graphs, and text. If more than a single intensity of a color is available, use the value specified by the **Normal** directive to complete the selection. *Normal* of 0 represents the lowest intensity and *normal* of 255 represents the highest intensity.

Value	Color
0	Black
1	Cyan
2	Blue-Magenta
3	Red
4	Cyan-Blue
5	Magenta-Red
6	Green
7	Orange
8	Magenta
9	Yellow-Green
10	Blue
11	Green-Cyan
12	Yellow
13	Dark Gray
14	·
15	Light Gray White
	white
16	Decembed for firture was
127	Reserved for future use
127	Dl
128	Blue
:	:
	Cyan
	:
	Green
	Indices 128 through 255 rep resent the color spectrum used to display the Low (Blue) to High (Red) contour values.
	: Valley
	Yellow

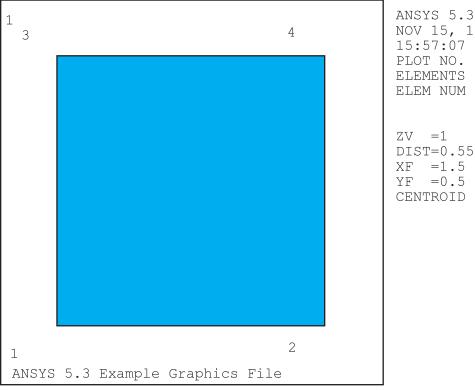
Value	Color
	:
	Orange
:	:
255	Red

4.3. Decoding a Graphics File: an Example

The following example shows you the following:

- The ANSYS command stream used to create a simple graphics plot, shown in Figure 4.2: Example Display of a Graphics File (p. 93) below
- The encoded graphics file that these commands produce
- The decoded graphics plot directives

Figure 4.2: Example Display of a Graphics File



```
NOV 15, 1996
15:57:07
PLOT NO.
            1
ELEMENTS
ELEM NUM
ZV = 1
DIST=0.55
XF = 1.5
YF = 0.5
CENTROID HIDDEN
```

4.3.1. The Example Command Stream

To create the graphics display shown in Figure 4.2: Example Display of a Graphics File (p. 93), you would issue the following ANSYS commands:

```
/PREP7
/TITLE, ANSYS 5.3 Example Graphics File
N,1,1
N,2,2
NGEN, 2, 2, 1, 2, 1, , 1
ET,1,42
E,1,2,4,3
/PNUM,ELEM,1
```

/PNUM,NODE,1 /SHR,.1 /SHOW,F33 EPLOT FINISH

4.3.2. Example Graphics File Contents

The commands listed above produce the display shown in *Figure 4.2: Example Display of a Graphics File* (p. 93) and the following graphics file:

(BBAAA2A0AAAAAPPPLPO<AA>AP\$MEKLKBAJANSYS 5.3\$MEKLEFALNOV 16 1996\$MEK KOJAI15:57:07\$MEKKIMAMPLOT NO. 1\$MEKKDAAIELEMENTS\$MEKJNEAIELEM NUM2 B0AAAAAALPOLPO<AB/PP+EBBBHBBHKOGBBHKOGKOGBBHKOG\$FPPFPPAB1\$AILAILAB1\$L HCAILAB2\$LHCLHCAB4\$AILLHCAB32A0AAAAAAPPPLPO.AAAAAA-LPOAAA-LPOLPO-AAAL PO-AAAAAA>AB\$ABLLKBAB1>AP\$MEKJBLAGZV =1\$MEKILPAJDIST=0.55\$MEKIGCAIXF =1.5\$MEKIAGAIYF =0.5\$MEKHKKAPCENTROID HIDDEN\$ABOABOCA ANSYS 5.3 Ex ample Graphics File)

The decoded plot directives are:

(BBAAA	Start-Plot:/ERASE, raster mode
2A	Text-Size: Default
0AAAAAAPPPLPO	Window: 0.0 4095,3070
<AA</td><td>Area-Color: Black</td></tr><tr><td>>AP</td><td>Text-Color: White</td></tr><tr><td>\$MEKLKBAJANSYS 5.3</td><td>Text: 3146 2977 "ANSYS 5.3"</td></tr><tr><td>\$MEKLEFALNOV 16 1996</td><td>Text: 3146 2885 "NOV 15 1996"</td></tr><tr><td>\$MEKKOJAI15:57:07</td><td>Text: 3146 2793 "15:57:07"</td></tr><tr><td>\$MEKKIMAMPLOT NO. 1</td><td>Text: 3146 2700 "PLOT NO. 1"</td></tr><tr><td>\$MEKKDAAIELEMENTS</td><td>Text: 3146 2608 "ELEMENTS"</td></tr><tr><td>\$MEKJNEAIELEM NUM</td><td>Text: 3146 2516 "ELEM NUM"</td></tr><tr><td>2B</td><td>Text-Size: Small</td></tr><tr><td>0AAAAAALPOLPO</td><td>Window: 0 0 3070 3070</td></tr><tr><td><AB</td><td>Area-Color: Cyan</td></tr><tr><td>/PP</td><td>Normal: 255</td></tr><tr><td>+ЕВВВНВВНКОСВВНКОСКОСВВНКОС</td><td>Polygon: 279, 279, 2790, 279 2790, 2790 279, 2790</td></tr><tr><td>\$FPPFPPAB1</td><td>Text: 1535 1535 "1"</td></tr><tr><td>\$AILAILAB1</td><td>Text: 139 139 "1"</td></tr><tr><td>\$LHCAILAB2</td><td>Text: 2930 139 "2"</td></tr><tr><td>\$LHCLHCAB4</td><td>Text: 2930 2930 "4"</td></tr></tbody></table>	

\$AILLHCAB3	Text: 139 2930 "3"
2A	Text-Size: Default
0AAAAAAPPPLPO	Window: 0,0 4095,3070
. AAAAAA	Move: 0,0
-LPOAAA	Draw: 3070,0
-LPOLPO	Draw: 3070,3070
-AAALPO	Draw: 0,3070
-AAAAA	Draw: 0,0
>AB	Text-Color: Cyan
\$ABLLKBAB1	Text: 27 2977 "1"
>AP	Text Color: White
\$MEKJBLAGZV =1	Text: 3146 2331 "ZV =1"
\$MEKILPAJDIST=0.55	Text: 3146 2239 "DIST=0.55"
\$MEKIGCAIXF=1.5	Text: 3146 2146 "XF =1.5"
\$MEKIAGAIYF =0.5	Text: 3146 2054 "YF =0.5"
\$MEKHKKAPCENTROID HIDDEN	Text: 3146 1962 "CENTROID HIDDEN"
\$ABOABOCA ANSYS 5.3 Example Graphics File	Text: 30 30 "ANSYS 5.3 Example Graphics File"
)	End-Plot
	No-Op

Release 12.0 - © 2009 SAS IP, Inc. All rights reserved Contains proprietary and confidential information
nelease 12.0 - © 2009 3A3 ir, inc. Airrights reserved Contains proprietary and confidential information
of ANSYS Inc. and its subsidiaries and affiliates.



Chapter 5: Using User Programmable Features (UPFs)

The ANSYS program has an open architecture, allowing you to write your own routines or subroutines in C or in Fortran and either link them to ANSYS or use them as external commands. Some standard ANSYS features originated as user programmable features (UPFs).

You can take advantage of UPFs if you are licensed for any of the following products:

- ANSYS Multiphysics
- ANSYS Mechanical
- ANSYS Structural
- ANSYS Emag Low Frequency
- ANSYS Emag High Frequency
- ANSYS PrepPost

Other versions of the ANSYS program do not support UPFs.

For more information about C and Fortran compilers, see the ANSYS, Inc. installation guide specific to your operating system:

ANSYS, Inc. UNIX/Linux Installation Guide ANSYS, Inc. Windows Installation Guide

5.1. What Are UPFs?

User programmable features are ANSYS capabilities you can use to write your own routines. Using UPFs, you can tailor the ANSYS program to your organization's needs. For instance, you may need to define a new material behavior, a special element, or a modified failure criterion for composites. You can even write your own design optimization algorithm that calls the entire ANSYS program as a subroutine.

UPFs provide the following capabilities:

- To read information into or fetch information from the ANSYS database, you can create subroutines and either link them into the ANSYS program or use them in the external command feature (see *Appendix A* (p. 303) for more information about external commands). If you link these subroutines into ANSYS, you are limited to 10 database access commands. Such commands, created through either method, operate at all levels of ANSYS operation, including the begin, preprocessor, general postprocessor, time-history postprocessor, and solution levels. For more information about accessing the ANSYS database, see *Chapter 7, Accessing the ANSYS Database* (p. 215).
- ANSYS provides a set of routines you can use to specify various types of loads, including BF or BFE loads, pressures, convections, heat fluxes, and charge densities. These routines are described under Subroutines for Customizing Loads (p. 178).
- Another group of UPFs enables you to modify and monitor existing elements. For details, see *Subroutines* for *Modifying and Monitoring Existing Elements* (p. 149).

- Another set of UPF routines enables you to define the following material properties: plasticity, creep, swelling law, viscoplasticity, hyperelasticity, and layered element failure criteria. To see inputs and outputs for these routines, see *Subroutines for Customizing Material Behavior* (p. 156).
- Several sets of UPFs enable you to define new elements and to adjust the nodal orientation matrix. See Creating a New Element (p. 113) for more information.
- You can customize UPF userop to create a custom design optimization routine. For more information, see *Creating Your Own Optimization Subroutine* (p. 204).
- You can call the ANSYS program as a subroutine in a program you have written. To learn how, see Running ANSYS as a Subroutine (p. 193).

5.2. What You Should Know Before Using UPFs

Before you do anything with linked UPFs, contact your on-site ANSYS system support person to get the permissions needed to access the appropriate ANSYS files.

The UPF subroutines are written in Fortran; some extensions are used. They contain comments intended to give you enough detail to develop your own versions of the subroutines.

User routines that can be modified have the term "USERDISTRIB" in the first line of the routine. These routines are provided with the ANSYS distribution media. You can modify these routines to tailor the ANSYS program to your specific needs. Certain other routines described in this document are not provided on the distribution media, but can be called using the provided header information.

To use UPFs successfully, you need strong working knowledge of the following:

- The ANSYS program.
- The UPF subroutines themselves. Study the UPF subroutines before customizing them, and make sure that you fully understand the subroutines, as well as any applicable functions. Unless you review them carefully, a few UPF subroutines may seem like a maze with many logic paths to consider. You may have to set special variables correctly in order to run your customized ANSYS program without errors. Even if you have in-depth knowledge of the ANSYS input and your desired outputs, you still need to ensure that everything that needs to be done in the UPF subroutines is done properly in your custom version.
- Fortran. Besides knowing how to write Fortran subroutines, you must be sure that the level of the Fortran compiler is as least as high as the level mentioned in your ANSYS installation manual. For more information on Fortran compilers, please refer to the ANSYS Installation Guide specific to your operating system. You also need to know what to do should the computer abort the program due to an arithmetic error, a file read error, a memory access error, and so on.
- The mathematics of the phenomenon you are planning to include.

Important

- UPFs are not available or will behave unpredictably in certain data center environments or on some hardware configurations. You should take special care when using UPFs on parallel systems. It is a good practice to verify your coding with single processing by using the -np,1 option before you run your analysis. For additional information, consult your ANSYS installation manual or your on-site ANSYS system support person
- Carefully consider whether you wish to use UPFs, especially if you are linking them into ANSYS (rather than into a shared library for use as external commands). When you add your own routines to ANSYS by either method, you are creating a customized, site-dependent version of the program. ANSYS, Inc.

considers the use of UPFs a nonstandard use of the program, one that the ANSYS Quality Assurance verification testing program does not cover. Therefore, you are responsible for verifying that the results produced are accurate and that your customizations do not adversely affect unchanged areas of the ANSYS program.

- Although the flexibility that UPFs offer can be highly attractive, UPF usage is a complicated process that can introduce errors. Consider what you want your customizations to accomplish. You may be able to customize ANSYS more easily and safely with macros than with UPFs.
- For shared-memory ANSYS, all user-programmable features are supported except for common block variables in Fortran and external variables in C or C++. You should avoid overwriting the values of these variables by multiple cores at the same time.
- For Distributed ANSYS, all user-programmable features are supported except for global (often in common block) variables in Fortran and global (often external) variables in C or C++. You should avoid overwriting the values of these variables; they should have the same values across all cores used in the distributed solution.

For other guidelines for nonstandard uses of the ANSYS program, see the *Advanced Analysis Techniques Guide*.

5.3. Planning Your UPFs

UPFs can range from a simple element output routine for customized output to a complex user optimization. Before you start programming, ask yourself these questions:

- Does the capability you want already exist in the ANSYS program? Remember, a capability may not be obvious at first, especially to a novice ANSYS user.
- Does your proposed subroutine fit into the ANSYS program architecture and specifications? For example, you can not program a user element that has more than 32 degrees of freedom per node.

Use your experience and judgment to answer these questions. If you need help to do so, consult your ANSYS Support Distributor. If you can respond "no" to both questions, then the user routine you are planning will be both useful and feasible.

5.4. Studying the ANSYS User Routines

Your ANSYS distribution medium contains the source codes for all user routines:

- If you are running the ANSYS program under UNIX/Linux, the source code for the UPF routines resides in directory /ansys_inc/v120/ansys/customize/user/.
- If you are running the ANSYS program under Windows, the source code for the UPF routines resides in directory Program Files\Ansys Inc\V120\ansys\customize\user\.

Most of the user routines have at least simple functionality, so print out the routines of interest before you start programming. All source routines are concatenated onto file user.for.Delete the routines you do not want and make appropriate changes to the others.

5.5. Programming in Languages Other than Fortran

If you access UPFs by compiling and linking a custom version of ANSYS, the preferred method is to design and program your custom routine in Fortran. Although you can use languages other than Fortran, in each case Fortran must provide the interface to the rest of the ANSYS program. If you do use a language other than Fortran, such as the C programming language, your code may require a Fortran shell.

You need to take care when calling Fortran subroutines from C subroutines. You must use the symbol associated with the Fortran subroutine when invoking the subroutine from a C function. This symbol typically differs slightly from the Fortran subroutine name, and is extremely system dependent.

On many UNIX systems, you build this symbol name by taking the Fortran subroutine name, converting it to lower case, and appending an underscore. For example, the symbol name for the Fortran subroutine **HeapInquire** would be **heapinquire**. You would have to use the symbol **heapinquire** in the invoking C function to avoid an unsatisfied external reference when the program is linked.

Keep in mind that the instance described above is just an example. Compilers from different vendors may construct the symbols differently. Please consult the manuals for your specific compiler for information on how to call Fortran subroutines from C functions.

For more information on Fortran compilers please refer to the ANSYS Installation Guide specific to your operating system.

5.6. Developing UPFs: a Suggested Strategy

When developing UPFs by compiling and linking a custom version of ANSYS, you can avoid problems and reduce debugging time by following a gradual, orderly process. Start with a trivial test. Then, add a few changes at a time so that if something goes wrong, the error that caused the problem should be isolated and relatively easy to locate.

The example procedure below illustrates this type of gradual process. The example assumes that you are creating a new element for the ANSYS program using the method described in *Creating a New Element by Directly Accessing the ANSYS Database* (p. 128). You develop and test it by performing these steps:

- 1. Get the applicable element subroutines for uel101 from the ANSYS distribution medium. Add a small change (such as a misspelling in an output heading), then compile and link the subroutines.
- 2. Using a production version of the ANSYS program, run several analysis problems using LINK8 (and maybe other elements) to form a base for comparison.
- 3. Replacing LINK8 with USER101, run the same problem on your custom version of ANSYS.
- 4. Compare the results from Steps 2 and 3. If they show discrepancies other than the misspelled output heading, resolve them before you go on to Step 5.
- 5. Choose the standard ANSYS element that most closely resembles your new custom element, and run some problems on a production version of ANSYS using that element.
- 6. Modify the element subroutines to match the element you chose in Step 5. Then, compile and link those subroutines into a custom version of ANSYS.
- 7. Again, compare the results from Steps 5 and 6. If they don't match, resolve the discrepancies before moving on to Step 8.
- 8. Modify your element subroutines to include the features you want. Then, compile and link the subroutines into a custom version of ANSYS.
- 9. Test the changes with a series of increasingly complex problems for which you already know the answers.

5.7. Include Decks

In addition to the subroutines and functions described in this chapter, most of the include decks (files with the extension .inc) used by ANSYS are on your ANSYS distribution medium. These include decks, also called *commons*, contain important but relatively small amounts of data. The ANSYS program also handles large amounts of data using various access routines (GET and PUT), as described elsewhere in this manual.

To insert include decks in a subroutine or function, use the INCLUDE (or an analogous) statement. *Do not modify an include deck under any circumstances*. The following table lists some of the more commonly used ANSYS include files and the definitions they contain:

Include File	Description
acel- cm.inc	Contains accelerations and angular velocities
ansys- def.inc	Defines general ANSYS parameters. You must include this common to retrieve the parameter values of MEM_INTEGER, MEM_DOUBLE, MEM_COMPLEX, or MEM_REAL.
cmopt.inc	Contains optimization variables
ech- prm.inc	Defines parameters for element characteristics
el- ccmt.inc	Defines element characteristics (comments only)
ele- com.inc	Contains element-specific information
el- parm.inc	Defines pointers for the element data array
eluc- om.inc	Defines the element degree of freedom pointers
ety- com.inc	Element type data
imp- com.inc	Used by all routines and functions in the ANSYS program
outp- cm.inc	Defines output control information
soptcm.inc	Contains solution options and keys
stack.inc	Defines stack storage. You must include this common in any routines that access stack space.
step- cm.inc	Contains load step information
us- vrcm.inc	Defines storage of user-defined variables

5.8. Linking User Routines

After you make your changes to the user routines supplied on your ANSYS distribution medium, you can either:

- Link your routines into shared libraries (as discussed starting in *Appendix A* (p. 303)).
- Compile and link your custom routines into the ANSYS program itself. This is discussed for UNIX systems
 in Compiling and Linking UPFs on UNIX/Linux Systems (p. 104) and for Windows systems in Compiling and
 Linking UPFs on Windows Systems (p. 105). You may need superuser or root privileges to run the procedure
 that does the linking.

5.9. Compiling and Linking UPFs on UNIX/Linux Systems

As mentioned previously, the source files for the user routines reside in subdirectory /ansys_inc/v120/ansys/customize/user/. If you modify any of these subroutines, select the **Relink ANSYS** option from **ANS_ADMIN120** utility to link these changes.

When you run a user-linked version of the ANSYS program, the ANSYS output will include the following:

```
NOTE: This ANSYS version was linked by Licensee
```

The **Relink ANSYS** option compiles all Fortran files (files ending with .F) and all C files (files ending with .c) in the current working directory. The procedure then loads all object files (files ending with .o) along with the default ANSYS objects and libraries in /ansys_inc/v120/ansys/customize/user/<platform>. For Distributed ANSYS the location is: /ansys_inc/v120/ansys/customize/user/<platform>/dis/native, where <platform> is replaced by the folder representative of your operating system. The new executable file created will be named ansyscust.e120 and will reside in the working directory.

Fortran files are assumed to be Fortran (some extensions are allowed), and C files are assumed to be ANSI C. For more information on Fortran compilers please refer to the ANSYS Installation Guide specific to your operationg system.

When relinking on UNIX or Linux systems, you can choose to link a distributed version of ANSYS as well. If you choose to link a distributed version, the executable (ansyscustdis.e120) must reside in the same directory path on all systems. However, you need to link it on only one system; you can then copy the executable to the other systems.

The Installation and Configuration Guide specific to your operating system lists the compilers you will need to use UPFs.

```
ANSYS, Inc. UNIX/Linux Installation Guide ANSYS, Inc. Windows Installation Guide
```

Creating a Shared Library You can also set up UPFs on some UNIX and Linux systems through a shared library as an alternative to creating a custom ANSYS executable. All Fortran (* . F), C (* . c), and C++ (* . cpp) files that you want to include in your shared library should reside in your working directory. To compile all * . F, * . c, and * . cpp routines, issue the following command:

```
/ansys_inc/v120/ansys/customize/user//ANSUSERSHARED
```

If the compile was successful, you will be asked if a shared file is to be created. Choose **Yes** to create a shared library named libansuser.so (or .sl).

To use this library, set the ANS_USER_PATH environment variable to point to the working directory where the libansuser shared library resides. Use one of the following commands, depending on the UNIX shell you are using:

```
setenv ANS_USER_PATH workingdirectory

Or
    export ANS_USER_PATH=workingdirectory
```

When you run a user-linked version of the ANSYS program, the ANSYS output will echo the value of **ANS_USER_PATH** and will include the following:

```
NOTE: This ANSYS version was linked by Licensee
```

To return to the original version of ANSYS, unset the **ANS_USER_PATH** environment variable.

ANSYS recommends using the ANSUSERSHARED script as a template to try compilers that are not supported by ANSYS, Inc., such as the GNU compilers. To do so, edit the ANSUSERSHARED script, making changes to the appropriate platform logic. Note that if you do use compilers other than those listed in the ANSYS Installation and Configuration Guide specific to your operating system, you will need to debug (i.e., find missing libraries, unsatisfied externals, etc.) them yourself. ANSYS, Inc. does not provide assistance for customers using unsupported compilers or if the resulting objects are not compatible with the ANSYS executable(s) as distributed.

Note

This shared library method is not available on Windows or IBM platforms.

5.10. Compiling and Linking UPFs on Windows Systems

The procedure for compiling and linking UPFs on Windows Systems allows you to link the ANSYS Shared Memory Parallel (SMP) executable as well as the Distributed versions of ANSYS executables.

As mentioned previously, the source files for the user routines reside in subdirectory Program Files\Ansys Inc\V120\ansys\customize\user\.

Note

If you intend to modify any of the user routines, make a duplicate copy of the Program Files\Ansys Inc\V120\ansys\customize\user\ directory to preserve the original files for later use, if necessary.

If you modify any of the user routines, move them to the folder(s) they are linking in. By default those folders are:

- ANSYS-SMP: Program Files\Ansys Inc\V120\ansys\custom\user\<platform>
- Distributed ANSYS-HPMPI version: Program Files\Ansys Inc\V120\ansys\custom\user\<platform>\DANSYSHPMPI
- Distributed ANSYS-MSMPI (Winx64 only): Program Files\Ansys Inc\V120\ansys\custom\user\<platform>\DANSYSMSMPI

Where cplatform> is a directory that uniquely identifies the hardware platform version: "Intel" for 32-bit Windows, "Winx64" for 64-bit Windows.

If you modify any of these subroutines, select the **Relink ANSYS** option from the **ANS_ADMIN120** utility to link these changes into the ANSYS program. This procedure will ask which versions you intend to relink and then compiles all Fortran files (files ending with .F) and all C files (files ending with .c) in the Program Files\Ansys Inc\V120\ansys\custom\user\<platform> directory. The procedure then loads all object files (files ending with .obj), along with the default ANSYS objects and libraries and creates custom ansys executables. The executable file(s) created will be named ansys. exe and will reside in the folders described above.

Caution

When creating custom ANSYS executables, the executables must be named ansys.exe. This requirement is due to shared library usage.

Note

On any Windows system, if you are attempting to create a relinked version of ANSYS by using ANSCUST instead of using the **ANS_ADMIN120** utility (as recommended above), error messages may occur. These messages may state that object files have been created, but the ANSYS executable has not been created; or the errors may state that some libraries necessary to complete the link cannot be found. These errors usually indicate that required compiler environment variables are not set. To avoid these errors, use the following workaround when relinking ANSYS with ANSCUST:

- Pick Start > Programs > Microsoft Visual Studio 2005 > Visual Studio Tools > Visual Studio 2005 Command Prompt, which should open a new command prompt window.
- In this command prompt window, issue the drive letter and all the necessary cd commands to move to the directory where the customized files exist (example: C:\Program Files\Ansys Inc\V120\ansys\custom\user\).
- Type ANSCUST in this command window. The process of creating the new user-customized ANSYS version will begin.

After relinking the ANSYS executable, the program can be executed by either of the following two methods:

- 1. To execute the relinked version of the ANSYS program:
 - Click Start>Programs>ANSYS 12.0>Mechanical APDL Product Launcher
 - In the launcher, select the Customization/Preferences tab, then browse to the path which contains
 the relinked ansys.exe. Select other desired options then pick Run to execute the customized
 ansys.exe.
- 2. To execute the relinked ansys.exe from a Command Prompt window, use one of the following commands.
 - · Interactive:

Batch:

```
ansys120 -b -p roduct variable> -j jobname -i <input file> -o <output file> -custom <path>
```

where "path" indicates the full path to the relinked ansys.exe.

Note

The commands above are for the SMP version of ANSYS. Consult the Distributed ANSYS manual for instructions on running Distributed ANSYS.

Note

Output from a user-linked version will contain the following statement:

This ANSYS version was linked by Licensee

Note

You will need all the compilers specified in the Installation and Configuration Guide specific to your operating system to use these user programmable features. The user programmable features are loaded onto the system only if you perform a custom installation and choose to install the customization tools.

5.11. Activating UPFs

The ANSYS program activates many UPFs through a specific user action. This can be through a command option or a user selection. To activate user elements created using the method described in *Creating a New Element via the User-Defined Element API* (p. 115), you need **USRELEM** and **USRDOT** commands, as well as **ET** and **TYPE** commands. To activate a user element created using the method described in *Creating a New Element by Directly Accessing the ANSYS Database* (p. 128), you must select it as one of the element types in a model using the **ET** command, and then set the element attribute pointer using the **TYPE** command), and define elements using the solid modeling or direct generation method. To define user material described in *Subroutine UserMat (Creating Your Own Material Model)* (p. 156), *Subroutine UserPL (Writing Your Own Plasticity Laws)* (p. 171), *Creep Subroutine UserCreep* (p. 165), and *Subroutine usersw (Writing Your Own Swelling Laws)* (p. 173), you need to activate them with the corresponding **TB** commands.

UPFs that are not activated by the means described above must be activated by either of the following methods:

- Issuing the USRCAL command
- Choosing menu path Main Menu>Preprocessor>Loads>-Load Step Opts->Other>User Routines or Main Menu>Solution>-Load Step Opts->Other>User Routines.

To activate or deactivate the routines, issue the command **USRCAL**, Rnam1, ...Rnam9, where Rnam1 and Rnam9 are the names of specific routines. You can specify up to nine routines with one **USRCAL** command, or you can issue multiple **USRCAL** commands.

Issue the command **USRCAL**,NONE to deactivate all valid user subroutines. To list the status of the routines, issue the command **USRCAL**,STAT.

For a list of the user routines that the **USRCAL** command (or its equivalent menu paths) affects, see the **USRCAL** command description in the *Command Reference*.

If you do not activate the UPFs in this manner, standard ANSYS logic will be used by default. For instance, when you apply a convection load, standard ANSYS logic is the default even if you have a user convection routine linked in. The user convection routine must be activated by the **USRCAL** command or its menu equivalent.

5.12. Running Your Custom Executable

You can run a custom executable from the **Customization/Preferences** tab of the launcher:

Enter the full pathname to the custom executable in the **ANSYS Custom Exe** field. Do not include the -custom argument.

When run from the command prompt, if no path is specified after the <code>-custom</code> argument, the <code>ansys120</code> script searches the current working directory for the custom ANSYS executable (<code>ansyscust.e120</code> by default on UNIX or <code>ansys.exe</code> on Windows). If the custom ANSYS executable resides in a separate directory (or has a name other than <code>ansyscust.e120</code> on UNIX), you can specify a different path and filename after the <code>-custom</code> argument.

Caution

If you are running on a Windows system and you create a custom ANSYS executable, the executable must be named ansys.exe. This requirement is due to shared library usage.

On UNIX, you can also run your custom executable via command line.

ansys120 -custom /pathname/ansyscust.e120

5.13. Verifying Your Routines

After compiling and linking your new user routine, test and verify it using whatever procedures you think are adequate. Remember, verifying that your customized version of the ANSYS program works properly is *your* responsibility.

Make certain that your custom version of the ANSYS program performs correctly for the combinations of elements, analysis types, materials, boundary conditions, and so on. that you plan to use. Confirm that the logic you introduced is correct and does not produce any unwanted side effects.

In testing your custom user routines, you also should verify that the changes you have made do not affect standard, non-customized ANSYS features. To do so, you can compare the results of a set of problems from the *Verification Manual* run on the standard version and on the customized version. Input for these problems is also available on your ANSYS distribution medium.

Always remember: your last step, a series of steps, or even your concept may be wrong. Proceed in clear steps, and verify your work as often as possible. Keep intermediate versions of your modified source code on backup media.

Note

If you contact your site's ANSYS system support person or any ANSYS, Inc. representative about the performance of a custom version of ANSYS, always tell him or her explicitly that you are using a user programmable feature. If you feel that an error exists in an unrelated feature of the ANSYS program, demonstrate the suspected error in an non-customized, production version of the program before you report the error to an ANSYS. Inc. representative.

5.14. Debugging Commands

To debug errors in your user routines, you can use commands and other features not documented in the *Command Reference*. Use these commands only for extremely small models with few solution iterations (otherwise, they will generate an excessive amount of output). /TRACK and /DEBUG are described in detail below. Two other useful commands are OUTEQ and /NERR. The command OUTEQ,ON can be used to output results from all equilibrium iterations. The command /NERR,,,,-1 causes errors to be reported as before, but

the run continues anyway, normally terminating with either a) system abort or b) incorrect answers. The /NERR,,,-1 command is intended for program debugging and may generate erroneous results. You should remove this statement before generating solutions for production use.

5.14.1. Tracking the Path of Program Logic

The /TRACK command issues a message when the program logic enters and leaves some of the higher level subroutines. Subroutines TrackBegin and TrackEnd (see *Chapter 8, Subroutines for Users' Convenience* (p. 285)) set up the /TRACK command. Then, issue the command using the format below

```
/TRACK, MonLevel, PrintLevel, SumLevel
```

MonLevel

The level for timing monitoring.

PrintLevel

The level for enter/exit printout.

SumLevel

The level at which the timing sum is output.

Each of these arguments can be any value between 0 and 9 (default is 0).

You can use the /TRACK command to identify which section of code is causing the program to abort. For example, to flag up to eight levels of subroutines to determine when the program logic enters and leaves them, you would issue the command /TRACK,,8.

5.14.2. Debugging Elements and Solutions

The /**DEBUG** command generates debugging at various points in the output. You can specify one of three formats for /**DEBUG**: solution debug format, element debug format, and general debug format.

5.14.2.1. Solution Debug Format

Issue the command using this format:

```
/DEBUG,-1,F1,F2,F3,F4,F5,F6,F7,F8,F9
F1
```

1 (provides basic solution control debugging)

F2

- 1 (provides transient debugging using Newmark constants)
- 2 (provides transient debugging using velocities and accelerations)

F3

- 1 (provides element matrix debugging and prints matrix + load vectors, before going into solve)
- 2 (provides element matrix debugging with load vectors only, before going into solve)
- 3 (provides element matrix debugging with matrix diagonals and load vectors, before going into solve)

F4

1 (provides auto time stepping debugging)

F5

1 (provides multifield debugging)

F6

1 (provides arc-length debugging)

F7

- 1 (provides basic Newton-Raphson debugging)
- 2 (provides Newton-Raphson debugging and prints out-of-balance forces or incremental displacement or each DOF)
- 3 (provides Newton-Raphson debugging and prints applied loads and n-r restoring force for each DOF)

F8

- 1,2 (provides displacement vector debugging with displacement pointers)
- 2 (provides displacement vector debugging with incremental displacement)
- 3 (provides displacement vector debugging with contact database)

F9

1 (provides temporary programmer debugging)

5.14.2.2. Element Debug Format

Issue the command using this format:

```
/DEBUG, -3, G1, G2, G3, G4, G5, G6, G7, G8, G9
```

G1

1 (provides basic element pass debugging)

G2

1 (provides element displacement and coordinate debugging)

G3

- 1 (provides element matrix debugging and prints matrix + load vectors, after the element routines)
- 2 (provides element matrix debugging with load vectors only, after the element routines)
- 3 (provides element matrix debugging with matrix diagonals and load vectors, after the element routines)

G4

1 (provides element load information debugging)

G5

1 (provides element real constant debugging)

G6

1 (provides element saved variable debugging)

G7

- 1 (provides element material property debugging with linear material properties)
- 2 (provides element material property debugging with nonlinear properties)

G8

- 1,2 (provides element nonlinear debugging with plasticity)
- 2 (provides element nonlinear debugging with large deformation)
- 3 (provides element nonlinear debugging with contact database)

G9

1 (provides temporary programmer debugging)

5.14.2.3. General Debug Format

Issue the command using this format:

```
/DEBUG, H1, H2, , H4, H5, , , , H9

H1

1 (provides file header record information)

2 (provides input line (character))

3 (provides input line (decoded))

H2
```

1 (provides element reordering and element checking debugging)

2 (provides meshing debugging)

H4

- 1 (provides nodal coordinate system transformation debugging)
- 2 (provides displacement updating debugging)

Н5

1 (provides pre-element debugging, element characteristics debugging, and element field load debugging)

Н9

- -1 (print the progress of the resume (or save) to isolate location of failure)
- -99 (resume only the command log information for subsequent **LGWRITE** command)

5.15. Other Useful Commands

Two other ANSYS commands, **NSVR** and **/UCMD**, can help you implement UPFs. (Neither command has an equivalent GUI path.) Use the **NSVR** command to define the number of extra variables that need to be saved for user programmable element options, such as user plasticity.

Issue the **/UCMD** command to make a user routine into a custom command. For more information, see *Defining Your Own Commands* (p. 194).

5.16. Generating Output

You can generate output controlled by the **/OUTPUT** command by using the Fortran write statement. The output unit for this statement is usually called IOTT. IOTT may be defined with the function wringr. See the discussion on the function wringr in *Chapter 8, Subroutines for Users' Convenience* (p. 285) for more details.

5.17. Reading Large Data Files More Rapidly

When files containing ANSYS-related data are large, loading them into the ANSYS program or writing them out to an external file can be a slow process. For example, consider an ANSYS problem file which contains nearly 462,000 lines, 150,000 of which contain nodal data and 97,383 of them containing data for elements. Because many of the lines in this file are in command format, the ANSYS program spends a lot of time reading it.

You can shorten the time ANSYS takes to read such files by including two commands in your programs, UPFs, or macros: **EBLOCK** and **NBLOCK**. The **NBLOCK** command converts nodal data into fixed format data blocks (which ANSYS can read more quickly than commands). The **EBLOCK** command places element data into a fixed format block, one line per element. These commands also compress displacement constraint data to one line per constraint node. See *Chapter 3, The CDWRITE (CDB) File Format* (p. 71) in the *Guide to Interfacing with ANSYS* for more information on the use of these commands.

Chapter 6: UPF Subroutines and Functions

This chapter describes the various subroutines, functions, and commands that allow you to customize the ANSYS program for your specific purpose. The first portion of each subroutine or function (consisting of comment lines) is shown in most cases.

User subroutines that can be modified have the term *USERDISTRIB* in the first line of the subroutine. For example, the first line of the userop subroutine looks like this:

*deck,userop USERDISTRIB

User subroutines that do not have *USERDISTRIB* in the first line cannot be modified and must be used as they are.

The following UPF topics are available:

- 6.1. Creating a New Element
- 6.2. Supporting Subroutines for Element Creation
- 6.3. Subroutines for Modifying and Monitoring Existing Elements
- 6.4. Subroutines for Customizing Material Behavior
- 6.5. Subroutines for Customizing Loads
- 6.6. Running ANSYS as a Subroutine
- 6.7. Defining Your Own Commands
- 6.8. Supporting Subroutines
- 6.9. Access at the Beginning and End of Various Operations
- 6.10. Creating Your Own Optimization Subroutine
- 6.11. Memory Management Subroutines
- 6.12. Parameter-Processing Subroutines
- 6.13. Miscellaneous Useful Functions

6.1. Creating a New Element

ANSYS offers two tools for creating a user-defined element:

- The user-defined element API
- Direct access to the ANSYS database and files

ANSYS recommends the user-defined element API in most cases. The direct-access method is generally for special-purpose use only, or if you are already using preexisting elements created with this method.

This table highlights the differences between the two methods:

Interface		Accessing ANSYS database and files directly
	serving much of the underlying user-	No special interface. If an element capability exists for an ANSYS element, it will exist here (with a few exceptions).

Interface	User-defined element API	Accessing ANSYS database and files directly
	of the database subroutines and the file structure is rarely necessary to use the interface.	The logic necessary for using this interface effectively is more complex.
Relative level of difficulty	Medium	High
Expected compatibility between versions	High	Medium
Element names	USER300	USER100 to USER105
Demonstration logic in- cluded on the ANSYS distribution media	4-node quad and 20-node brick elements	MASS21 and LINK8 elements
Typical linear material access subroutine	getMatProp	propev
New nonlinear material properties	Program in UserMat.	No special programming has been set up.
Existing nonlinear material properties	All ANSYS standard structural materials are accessible via ElemGetMat.	Limited capability. Accessible via plastx, creepx, and swellx.
Non-structural material properties	No special programming has been implemented.	No special programming has been implemented.
Number of different ele- ment types allowed	Effectively, no limit.	Effectively, no limit.
Element characteristic capability	Input the basic 10 element characteristics (via the USRELEM and USRDOF commands). All other characteristics default automatically.	Input all 140 element characteristics (using uec100). Approximately 30 are normally used, and the rest default unless required for special cases.
Applicable subroutines to be programmed	UserElem	uec100, uel100, and rarely uex100 and uep100. Subroutines uec101 to uec105 are analogous.
Access to database information	Generally through the interface.	Through subroutines (such as tmpget, prsget, svgidx, svrget, svpidx, svrput).
Access to file information	Through the interface.	Through pointers and subroutines (such as eldwrtL, eldwrnL).
Special features	Element convergence criteria Cutback control via element	None.
Capabilities <i>not</i> included	Control information unavailable for: Birth and death Superelement stress pass Initial stress Section input Input of fluences Swelling	Material TB labels PLASTIC, NLISO, AHYPER, HYPER, PRONY, SHIFT, ELASTIC, ANEL, SDAMP, SMA, CAST, EDP, and GURSON.

6.1.1. Input and Output Abbreviations

The descriptions of the subroutines or functions within this chapter describe both the input arguments and output arguments. Argument information includes the argument's type, size and intent.

• Argument *type* is one of the following:

int - integer

dp - double-precision

log - logical

chr - character

dcp - double-precision complex

• Argument size is one of the following:

sc - scalar variable

ar(n) - array variable of length n

func - functional return value

• Argument *intent* is one of the following:

in - input argument

out - output argument

inout - both an input and an output argument

6.1.2. Creating a New Element via the User-Defined Element API

Following is the general process for creating your own element via the user-defined element API.

Sep	Description	Comments	
1.	Specify the element type.	Issue the ET and TYPE commands. The name of the element must be USER300.	
2.	Define your new element according to the specified element type.	Issue the USRELEM command. Specify the element characteristics (such as the number of nodes, number of dimensions, number of real constants etc.).	
3.	Specify nodal DOFs.	Issue the USRDOF command. You can specify a maximum of 10 DOFs per USRDOF command; to define additional DOFs, issue the command again. Each node will have the same DOFs. Although you can specify any valid DOFs, the total number of DOFs for your element cannot exceed 480, and the number of DOFs for each node cannot exceed 32.	
4.	Define real constants.	If needed.	
5.	Create finite element models.	 Direct generation Create elements directly from nodes, using commands such as E, EGEN, EN, ENGEN, or EMORE. (You can also use the CDREAD command if the .cdb file is available.) This method is the only way to create an element with a topology different from that of any standard ANSYS element. Meshing commands This method is available only if your element has the same topology as that of a standard ANSYS element and you have 	

		specified any standard element shape (USRELEM KeyShape value) except ANYSHAPE.
6.	Apply boundary conditions and loads.	As needed.
7.	Specify solution options.	If your element has multi-field degrees of freedom (displacements and temperatures), disable default solution settings (SOLCONTROL ,OFF).
8.	Perform postpro- cessing.	Postprocessing occurs normally as with any other element. Only total strain (or equivalent quantities such as thermal gradient) and stress (or equivalent quantities such as thermal flux) are saved as regular result quantities. Other variables are saved as nonsummable miscellaneous variables in the results file.

Note

Steps 2 and 3 specify data for the ANSYS user-defined element API. All other steps represent standard ANSYS features.

Recommendations and Restrictions

The following recommendations and restrictions apply to user-defined element USER300:

- Verify that your input data for the USRELEM and USRDOF commands are consistent with the values used in the UserElem. F code. For example, if the number of dimensions (NDIM) specified via the USRELEM command is 2, do not change the number of dimensions specified in the UserElem. F subroutine from 2. A runtime error or incorrect results can occur if the values do not match.
- ANSYS may activate default solution settings automatically according to the USER300 element's DOFs, but the default solution control settings may not be optimal for your element. If any convergence difficulty occurs, try disabling the default solution settings (SOLCONTROL, OFF).
- The USER300 element does not support ANSYS section (SECxxx) commands. For composite beams and layered shells, you must input element data via real constants and code the UserElem. F subroutine accordingly.

6.1.2.1. Subroutine UserElem (Writing Your Own Elements)

The UserElem subroutine provides an interface to ANSYS code above the element level. UserElem supports shared memory and distributed parallel processing; however, you are responsible for ensuring that your code can use parallel processing.

The subroutine passes all data needed to create a user-defined element and returns all data and results from the element to update the ANSYS database and files. With this API, you can create virtually any element type *without* having to access ANSYS database and files directly. Two examples are included in this subroutine: a 4-node quadrilateral 2-D element, and 20-node brick structural element, both for geometric linear analysis. Key options (KEYOPT settings) switch the elements.

The following table shows the input and output arguments, and their definition and usage. Some argument names (such as those pertaining to element matrices and load vectors) are common to structural analyses; however, you can specify argument values appropriate to analyses in other engineering disciplines.

Argument	Input (I) or Output (O)	Definition	Purpose	How Defined
elId	I	Element number	Output information	At FE model creation
matId	I	Material number	Output information Call material sub- routines	At FE model creation
keyMtx	I	Formulation request	Specifying which matrices and load vectors to form	ANSYS code
lumpm	I	Mass matrix format: 0 = Consistent mass matrix 1 = Lumped mass matrix	Specifying how to form the mass matrix	LUMPM com- mand
nDim	I	Number of dimensions	Element coding	USRELEM command
nNodes	I	Number of element nodes	Element coding	USRELEM command
Nodes	I	Element node list Connectivity	Output	At FE model creation
nIntPnts	I	Maximum number of element integration points	Element coding	USRELEM command
nUsrDof	I	Number of element DOFs	Element coding The DOFs are ordered in the way in which they are listed via the USRDOF command for each node and repeated for all nodes All element matrices DOF values and load vectors must be arranged in the same way	USRELEM and USRDOF com- mands
kEStress	I	Element stress state	Element coding Calling material sub- routines if requested	USRELEM command
keyAnsMat	I	Element formulation key: 0 Write your own material formulation 1 Use standard ANSYS material subroutines and call Elem-GetMat subroutine	Specifying how to create material data	USRELEM command
keySym	I	Flag for symmetricity of ele- ment matrices	Element coding ANSYS assembly logic	USRELEM command
nKeyOpt	I	Maximum number of element key options allowed	Element coding	ANSYS code

		Example: If nkeyOpt = 2, only KEYOPT(1) and KEYOPT(2) are allowed.		
KeyOpt	I	Element key options KEYOPT(1) = 0~99	Branching the user- element codes to dif- ferent formulations. (This could be equival- ent to 100 x 100 differ- ent types of elements.)	ET command
temper	1	Nodal temperatures at current time	Temperature dependence and thermal loads	BF and BFE commands (if keyShape is specified in the UserE-lem subroutine)
temperB	I	Nodal temperatures at the end of the last substep	Temperature dependence and thermal loads	ANSYS code
tRef	I	Reference temperature	Temperature dependence and thermal loads	TREF com- mand
kTherm	1	Key indicating whether a thermal load exists: 1 = Calculate the thermal load 0 = No thermal load calculation	Element coding	
nPress	I	Number of pressure values	Element coding The size of the press vector	USRELEM command At FE model creation
Press	I	Pressures at nodes of element facets (available only when keyShape is specified via the USRELEM command) The pressure vector is ordered in the element with the same topology as that in the standard element library. Refer to that element for details.	Pressure load and pressure load stiffness	SF and SFE commands
kPress	I	Key indicating whether a pressure load exists: 1 = Calculate the pressure load 0 = No pressure load calculation	Element coding	ANSYS code
nReal	I	Number of real constants	Element coding	USRELEM command
RealConst	I	The list of real constants	Element coding	R command

nSaveVars	I	The number of variables saved in the .esav file for the element	Can pass material properties, section/layer information, element material control, and any element data Element coding The size of	USRELEM command
saveVars	I/O	The data saved in the .esav file ANSYS saves the data after exiting the UserElem subroutine and retrieves it immediately before entering UserElem again. It should include kinematic related variables only when the ANSYS material subroutine is called; otherwise, it should include both kinematic and material data. History dependent variables can only be saved/updated when the substep is converged (keyHisUpd = 1).	Element coding	UserElem subroutine
xRef	I	Initial coordinates of the element nodes Values in global Cartesian coordinates	Element coding	At FE model creation
xCur	I	Current (deformed) coordinates of element nodes Values in global Cartesian coordinate system, equal to xRef when nlgeom = off	Element coding	ANSYS code
TotValDofs	I	Total values of DOFs (displace- ments for structural analysis) Values in global Cartesian co- ordinate system	Element coding	ANSYS code
IncValDofs	I	Increment values of DOFs oc- curring at the current substeps Values in global Cartesian co- ordinate system	Element coding	ANSYS code
ItrValDofs	I	Iteration values of DOFs occur- ring at the last iteration Values in global Cartesian co- ordinate system	Element coding	ANSYS code
VelValDofs	I	First time derivatives of DOFs	Velocities	ANSYS code
AccValDofs	I	Second time derivatives of DOFs	Accelerations	ANSYS code

kfstps	I	Key indicating first time entering the element subroutine: 1 = First time 0 = Other than first time	Initializing data	ANSYS code
nlgeom	I	Flag indicating whether large displacement/deformation is in effect	Element coding	NLGEOM command
nrkey	I	Newton-Raphson algorithm key: 1 Any nonlinear analysis 0 Pure linear analysis	Output Element coding	
outkey	I	Key indicating output result type: 1 This is an output call, the substep is converged, and you can print/save element results 0 All other cases	Element coding	ANSYS code
elPrint	I	Key indicating whether any element output should appear in the print file: 0 = No 1 = Yes	Element coding	ANSYS code OUTPR com- mand
iott	I	Output file number	The Fortran output file number. All information written in the specified file appears in the ANSYS output file.	ANSYS code
keyHisUpd	I	Key to update history-dependent variables: 1 = The substep converged; ready to update history-dependent variables (such as equivalent plastic strain) 0 = Solution not yet converged; cannot update history-dependent variables	Element coding	ANSYS code
The following vignore them.	ariables are for o	debug, timing, and convergence-	control purposes only. \	ou can usually
ldstep	I	Current load step number	Output Debug	ANSYS code
isubst	I	Current substep number	Output	ANSYS code
ieqitr	I	Current iteration number	Output	ANSYS code
timval	I	Current time	Output	ANSYS code
keyEleErr	I/O	Formulation error key: 0 = No error (preset value) 1 = Error occurred in element formulation, possibly due to	Element coding	ANSYS code

		excessive deformation. (ANSYS will lessen deformation if possible by cutback.)		
keyEleCnv	I/O	Element convergence key: 1 = Converged (preset value before calling) 0 = Not converged	Provides manual control of convergence when you introduce any constraint at the element level (such as volumetric constraint for mixed u-P)	ANSYS code
End of special-	ourpose variabl	e group		1
eStiff	0	Small-deformation stiffness matrix In global Cartesian coordinate system	Solution	Requested when keyMtx(1) = 1
eMass	0	Mass matrix In global Cartesian coordinate system	Solution	Requested when keyMtx(2) = 1
eDamp	0	Damping matrix In global Cartesian coordinate system	Solution	Requested when keyMtx(3) = 1
eSStiff	0	Stress stiffness matrix In global Cartesian coordinate system	Solution	Requested when keyMtx(4) = 1
fExt	0	External load vector In global Cartesian coordinate system	Solution	Requested when keyMtx(5) = 1
fInt	0	Internal nodal force vector In global Cartesian coordinate system	Solution	Requested when keyMtx(6) = 1
elVol	0	Element volume	Output	UserElem subroutine
elMass	0	Element mass	Output	UserElem subroutine
elCG	0	Element centroid coordinates In global Cartesian coordinate system	Postprocessing	UserElem subroutine
nRsltBsc	I	Number of basic result data saved in result file	Specifying the size of RsltBsc	ANSYS code
RsltBsc	0	Basic result data saved in AN- SYS result file These variables are accessible via the PRESOL and PRNSOL	Postprocessing	UserElem subroutine

		commands in the standard way and can also be plotted if you specify a KeyShape value via the USRELEM command.		
nRsltVar		The number of result data to be saved in the result file as non-summable miscellaneous variables	Specifying the size of RsltVar	USRELEM command
RsltVar	0	The result data saved in the result file as non-summable miscellaneous variables The data is accessible via the PLESOL command only, but only one value for an element each time	Postprocessing	UserElem subroutine
nElEng	I	Number of energies Fixed at 3	Solution	UserElem subroutine
elEnergy	0	Element energy vector: elEnergy(1) Strain energy elEnergy(2) Plastic energy elEnergy(3) Creep energy	Output	UserElem subroutine

```
*deck,UserElem
                                   USERSDISTRIB
      subroutine UserElem (elId, matId, keyMtx, lumpm, nDim, nNodes,
                          Nodes, nIntPnts, nUsrDof, kEStress,
    &
                          keyAnsMat, keySym, nKeyOpt, KeyOpt,
     &
                          temper, temperB, tRef, kTherm,
     &
                          nPress, Press, kPress, nReal, RealConst,
                          nSaveVars, saveVars, xRef, xCur,
     &
     &
                          TotValDofs, IncValDofs, ItrValDofs,
                          VelValDofs, AccValDofs,
     δ
                          kfstps, nlgeom, nrkey, outkey, elPrint, iott,
                          keyHisUpd, ldstep, isubst, ieqitr, timval,
     &
                          keyEleErr, keyEleCnv,
     &
                          eStiff, eMass, eDamp, eSStiff,
                          fExt, fInt, elVol, elMass, elCG,
                          nRsltBsc, RsltBsc, nRsltVar, RsltVar,
     &
     &
                          nElEng, elEnergy)
                 ****************
С
 *** Primary function: General User Element Subroutine
c *** Note:
       This routine is completed with an example, see more details later.
c *** Notice - This file contains ANSYS Confidential information ***
C
С
      PROGRAMMER SHOULD NOT CHANGE ANY PURE INPUT ARGUMENTS (marked by ....,in)!
С
      elId
С
               (int,sc,in)
                                  element number
С
      matId
                (int,sc,in)
                                  material number of this element
      keyMtx
               (int,ar(10),in)
                                  matrix and load vector form requests
C
                                     0 = not requested, 1 = requested
С
С
                                     see below for more details
      lumpm
               (int,sc,in)
                                  mass matrix format
C
```

```
= 0 no lumped mass matrix
C
С
                                      = 1 lumped mass matrix
С
      nDim
                 (int,sc,in)
                                    number of dimensions of the problem
                                          (defined on USRELEM command as NDIM)
С
                                      = 2 2D
С
                                      = 3 3D
C
      nNodes
                 (int,sc,in)
                                    number of nodes of the element
С
C
                                          (defined on USRELEM command as NNODES)
                 (int,ar(nNodes),in)node list of this element
      Nodes
С
С
      nIntPnts
                (int,sc,in)
                                    maximum number of integration points
                                          (defined on USRELEM command as NINTPNTS)
С
      nUsrDof
                 (int,sc,in)
                                    number of DOFs of this element (matrix and
C
С
                                       load vector size)
С
      kEStress
                (int,sc,in)
                                    kEStress
                                          (defined on USRELEM command as KESTRESS)
C
                                    key to indicate if ANSYS material
С
      keyAnsMat (int,sc,in)
C
                                       routine is going to be called
                                       (defined on USRELEM command as KEYANSMAT)
С
С
                                       = 0, No
                                       = 1, Yes
С
                                    key to indicate if element matrices
С
      keySym
                 (int,sc,in)
С
                                       is symmetric
                                         (defined on USRELEM command as KEYSYM)
C
С
                                       = 0, symmetric
С
                                       = 1, unsymmetric
C
      nKeyOpt
                 (int,sc,in)
                                    number of element key options able to be
C
                                       used in this routine
                (int,ar(nKeyOpt),in) values of element key option defined
С
      KeyOpt
                                       by et or keyopt command for the
С
                                       user elements, only the first
С
С
                                       nKeyOpt values are passed in and can
С
                                       be used to branch the routine for
C
                                       different formulations
                 (dp,ar(nNodes),in) nodal temperatures at current time
      t.emper
C
С
      temperB
                 (dp,ar(nNodes),in) nodal temperatures at the beginning of this
                                       incremental step (substep)
С
      tRef
                 (dp,sc,in)
                                    reference temperature
C
      kTherm
                 (int,sc,inout)
                                    input: flag for thermal loading
C
                                        = 1, Temperatures at nodes are different
C
                                        from the reference temperature,
С
                                        thermal loading might be needed.
С
C
                                        = 0, Temperatures at nodes are the same
С
                                        as the reference temperature,
                                        thermal loading is not needed.
C
                                    output: flag for thermal strains
C
      nPress
                 (int,sc,in)
                                    number of pressure values for this element
С
С
      Press
                 (dp,ar(nPress),in) applied elemental face load (pressure)
      kPress
                 (int,sc,in)
                                    flag for pressure loading
C
                                         = 1, pressure load is applied and
С
                                        equivalent nodal forces should be
С
C
                                        calculated
                                         = 0, no pressure loading
C
                (int,sc,in)
                                    number of real constants
C
      nReal
С
                                          (defined on USRELEM command as NREAL)
С
      RealConst (dp,ar(nReal),in)
                                    user defined real constants
                                    number of saved variables
C
      nSaveVars (int,sc,in)
                                        (defined on USRELEM command as NSAVEVARS)
С
С
      saveVars
                (dp,ar(nSaveVars),inout) user saved variables
                 (dp,ar(nDim,nNodes),in)
С
      xRef
С
                                    nodal coordinates in initial configuration
С
      xCur
                 (dp,ar(nDim,nNodes),in)
С
                                    nodal coordinates in current configuration
      TotValDofs (dp,ar(nUsrDof),in) total values of DOFs (displacements)
C
                                       from time = 0
С
      IncValDofs (dp,ar(nUsrDof),in) incremental values of DOFs (displacements)
С
                                       for the current step
С
C
      ItrValDofs (dp,ar(nUsrDof),in) iterative values of DOFs (displacements)
С
                                       for the current iteration
С
                                        (normally needed for debug only)
      VelValDofs (dp,ar(nUsrDof),in) first time derivatives of DOFs
C
С
                                                (velocities) (normally not needed)
      AccValDofs (dp,ar(nUsrDof),in) second time derivatives of DOFs
```

```
(accelerations) (normally not needed)
C
                                    key for the first iteration of first
С
      kfstps
                 (int,sc,in)
С
                                       substep of the first load step
С
                                        = 1 \text{ yes}
                                        = 0 no
С
                                    large deformation key [from nlgeom command]
С
      nlgeom
                 (int,sc,in)
                                       = 0 NLGEOM, OFF
С
C
                                       = 1 NLGEOM, ON
                 (int,sc,in)
                                    key to indicate a newton-raphson
C
      nrkey
С
                                        (incremental) procedure
                                        = 0 No
С
                                       = 1 Yes
C
                                    key to indicate if any element output is
С
      outkey
                 (int,sc,in)
С
                                        to be placed on the print file or the
                                       result file
C
                                        = 0 No
С
                                       = 1 Yes
C
      elPrint
                                    key to indicate if any element output is
С
                (int.sc.in)
С
                                        to be placed on the print file
                                        = 0 No
C
                                        = 1 Yes
С
С
                 (int,sc,in)
                                    print output file unit number
      keyHisUpd (int,sc,in)
                                    key to indicate if history-dependent
C
                                      variables need to be updated, like
С
С
                                      equivalent plastic strain, back stress
С
                                      etc. since the iteration is already
С
                                      converged
                                       = 0 not converged, don't need to update
C
                                           history dependent variables
С
                                        = 1 yes, converged, need to update
С
                                           history dependent variables
С
С
C
      --- The following 7 variable group can usually be ignored.
      --- The variables are used for debug, timing, and convergence control.
C
С
      ldstep
                (int,sc,in)
                                    current load step number
      isubst
                (int,sc,in)
                                    current substep number
C
                                    current equilibium iteration number
      iegitr
                (int,sc,in)
C
      timval
                (int,sc,in)
                                    current time value
С
      keyEleErr (int,sc,inout)
С
                                    key to indicate if there is any element
                                       formulation error, like negative Jacobian.
C
                                       The error could be caused by too
С
C
                                       large incremental step, illegal model.
С
                                       = 0 no error (preset value before calling)
С
                                        = 1 some error happens. ANSYS will
                                       decide to stop the analysis or cutback
C
С
                                       the substep (bi-section) based on other
С
                                       user input and information at higher
                                       level.
C
      keyEleCnv (int,sc,inout)
                                    key to flag if this element satisfies
С
С
                                       the user defined element convergence
C
                                       criterion.
                                        = 1, yes, the criterion is satisfied
C
                                         or don't have any criterion at all
C
С
                                         it is preset value before calling
С
                                        = 0, no, the element doesn't satisfy
                                         element convergence criterion. If
C
                                         this is the case, the iteration will
С
С
                                         not converge even when both force
                                         and displacement converge
С
С
        ---- end of 7 variable group ----
С
С
                                                                      requested if
      eStiff(dp,ar(nUsrDof,nUsrDof),inout) stiffness matrix
                                                                       keyMtx(1)=1
C
                                                                       keyMtx(2)=1
      eMass (dp,ar(nUsrDof,nUsrDof),inout) mass matrix
C
      eDamp (dp,ar(nUsrDof,nUsrDof),inout) damping matrix
                                                                       keyMtx(3)=1
С
      eSStiff(dp,ar(nUsrDof,nUsrDof),inout)stress stiffness matrix
                                                                      keyMtx(4)=1
С
                                            applied f vector
C
      fExt
                 (dp,ar(nUsrDof),out)
                                                                       kevMtx(5)=1
С
      fInt
                 (dp,ar(nUsrDof),out)
                                             internal force vector
                                                                       keyMtx(6)=1
      elVol
                 (dp,sc,out)
                                    element volume
C
С
      elMass
                 (dp,sc,out)
                                    element mass
                                    element centroid coordinates in current
C
      elCG
                 (dp,ar(3),out)
```

```
configuration
C
      nRsltBsc
                                    number of basic elemental results saved in
С
                (dp,sc,in)
С
                                     result files
С
      RsltBsc
                (dp,ar(nRsltBsc),out) basic elemental results
С
                                         (see EXPLANATION below)
С
      nRsltVar (int,sc,in)
                                    number of elemental results saved in
                                       result file as non-summable miscellaneous
С
C
                                         (defined on USRELEM command as NRSLTVAR)
С
С
      RsltVar
                (dp,ar(nRsltVar),out) variables to saved in result files as
                                        non-summable miscellaneous variables
C
                                        requested when outkey = 1
C
С
                (int,sc,in)
                                    number of energies (fixed at 3)
С
      nElEng
C
С
                (dp,ar(nElEng),out) elemental energy
C
                                       elEnergy(1), element strain energy
                                       elEnergy(2), element plastic strain energy
С
C
                                       elEnergy(3), element creep strain energy
С
      EXPLANATION OF RsltBsc
С
С
        Basic element results are saved and total number of result
C
С
      quantities is nRsltBsc, where:
С
             nRsltBsc = (7+7)* number of corner nodes at one element.
C
        To process the quantites by post processing properly, the results
С
      must be in the following order:
C
        1.) Stresses: Sx Sy Sz Sxy Syz Sxz Seqv at all corner points,
      followed by:
C
        2.) Strains : Ex Ey Ez Exy Eyz Exz Eeqv at all corner points
C
С
      where Seqv and Eeqv = equivalent stress and strain respectively
С
```

6.1.2.2. Subroutine ElemGetMat (Calling the ANSYS Standard Structural Material Library)

The ElemGetMat subroutine is the API to the ANSYS materials. When you issue the **USRELEM** command after setting the command's *KEYANSMAT* argument, the subroutine accesses the ANSYS standard structural material library. It allows you to focus on the kinematic portion of element formulation while ANSYS handles the material part of the formulation. When calling the subroutine, input the associated material data via the standard method. There is no need to access this subroutine, only to call it.

The following table shows the input and output arguments, and their definition and usage.

Argument	Input (I) or Output (O)	Definition	Purpose	How Defined
elId	I	Element number	Output	At FE model creation
matId	I	Material number	Output information Getting material data	At FE model creation
nDim	I	Number of dimensions of element geometry 2 = 2-D element geometry 3 = 3-D element geometry Specifying the size of the nodal coordinates	Material calculation	At FE model creation
nTens	I	Number of stress/strain tensor components:	Specifying the data size	UserElem subroutine

		4 = 2-D and ordered as x, y, z, xy 6 = 3-D and ordered as x, y, z, xy, yz, xz		
nDirect	I	Number of direct component of stress/strain tensors nDirect< or = nTens	Specifying the data size	UserElem subroutine
intPnt	I	Current integration point number	Output Data handling	UserElem subroutine
xCurIP	I	Coordinates of current integra- tion point Values in global Cartesian co- ordinate system	Material calculation	UserElem subroutine
TemperIP	I	Integration point temperatures at the current time	Evaluating temperat- ure-dependent materi- al data	UserElem subroutine
TemperIPB	l	Integration point temperatures at the end of the last incremental step	Evaluating temperat- ure-dependent materi- al data	UserElem subroutine
IncStrain	I	Strain components [1 (p. 127)] Incremental strain of the current substep when nlgeom = on Total strain when nlgeom = off	Material calculation	UserElem subroutine
defG0	I	Deformation gradient tensor at the end of previous substep [1]	Material updating	UserElem subroutine
defG	I/O	Total deformation gradient tensor at the current time [1]	The component in thickness direction is updated by material subroutines for plane stress and shell elements	UserElem subroutine
kTherm	I/O	Thermal loading key: 0 = No thermal loading 1 = Has thermal loading	Thermal load calculation	UserElem subroutine
cMat	0	Material Jacobian [1]	Forming stiffness	Material sub- routine
MatProp	0	Material data for element formulation	Forming mass matrix Handling transverse shear Output	Material sub- routine
Stress	0	Cauchy stress [1]	Forming geometric stiffness Calculating internal forces	Material sub- routine

Strain	0	Total strain components [1]	Output	Material sub- routine
StressTh	0	Total thermal stress components [1]	Output Calculating thermal loading	Material sub- routine
StrainTh	0	Total thermal strain components [1]	Output	Material sub- routine
StrainPl	0	Total plastic strain components [1]	Output	
StrainCr	0	Total creep strain components [1]	Output	
StressBk	0	Back stress components [1]	Output	
StrainSw	0	Swelling strain	Not yet supported	
EnergyD	0	Energy density: 1 = Elastic energy density 2 = Plastic energy density 3 = Creep energy density		
MatRotGlb	0	Rotation matrix	Rotation matrix from global Cartesian to ro- tated element coordin- ate system	Used only for solid ele-ments when nlgeom = on

1. All tensor component values in the subroutine are in the global Cartesian coordinate system for solid elements, and in the co-rotated element Cartesian coordinate system for link, beam and shell elements.

```
*deck,ElemGetMat
     subroutine ElemGetMat (elId, matId, nDim, nTens, nDirect,
                               intPnt, xCurIP, TemperIP,
                               TemperIPB, kTherm, IncStrain,
    &
    &
                               defGO, defG, CMat, MatProp,
                               Stress, Strain, StressTh, StrainTh,
    &
                               StrainPl, StrainCr, StressBk, StrainSw,
    &
                               EnergyD, MatRotGlb)
С
 *** Primary function: call ANSYS material routines to obtain material
                       data for USER300 elements
c *** Notice - This file contains ANSYS Confidential information ***
С
C
     input arguments
     ==========
С
С
     elId
                 (int,sc)
                                   element number
     matId
                 (int,sc)
                                  material number of this element
С
С
     nDim
                 (int,sc)
                                  number of dimensions of the problem
                                   = 2 2D
С
                                   = 3 3D
С
     nTens
                 (int,sc)
                                   number of stress/strain components
С
     nDirect
                 (int,sc)
                                  number of stress/strain direct
С
С
                                    components
С
     intPnt
                 (int,sc)
                                   current integration point number
С
     xCurIP
                 (dp,ar(3))
                                  coordinates of integration point
С
     TemperIP
                 (dp,sc)
                                   integration point temperatures at
C
                                     current time
     TemperIPB
                 (dp,sc)
                                   integration point temperatures at
C
                                     the end of last incremetal step
```

```
IncStrain
                 (dp,ar(nTens))
                                   strain for the current substep step when
С
С
                                     nlgeom = on
С
                                   total strain when nlgeom = off
                                   deformation gradient tensor at the end
     defG0
С
                 (dp,ar(3x3))
С
                                      of last incremental step
C
     input output arguments
                                   input desc
                                                 / output desc
C
C
     ========
                                                   ========
                (dp, ar(3x3))
                                   deformation gradient tensor at current
C
С
                                     time, updated for thichness change in
                                     plane stress when nlgeom=on
С
     kTherm
                                   flag for thermal loading
C
                (int.sc)
С
                                     input as:
С
                                     = 0 if temp = tref
                                     = 1 if temp .ne. tref
C
С
                                     gets reset to 0
                                                 if ALPX, ALPY, and ALPZ = 0
С
C
С
     output arguments
С
     ===========
                (nTens*nTens)
                                   material Jacobian matrix
С
     cMat
     MatProp
                 (dp,ar(5))
                                   commonly used materail properties
С
                                   MatProp(1),Gxz: shear modulus
С
                                   MatProp(2), Gyz: shear modulus
С
С
                                   MatProp(3),Gxy: shear modulus
                                   MatProp(4), density
C
С
                                   MatProp(5), nuxy
С
     Stress
                (dp,ar(nTens))
                                  total stress
                (dp,ar(nTens))
                                  total strain
     Strain
C
     StressTh (dp,ar(nTens))
                                thermal stress
С
     StrainTh
                (dp,ar(nTens))
С
                                  thermal strain
С
     StrainPl
                (dp,ar(nTens))
                                  plastic strain
С
     StrainCr
                 (dp,ar(nTens))
                                   creep strain
                                   back stress for kinematic hardening
     StressBk
                 (dp,ar(nTens))
C
С
     StrainSw
                 (dp,sc)
                                   isotropic swelling strain
                                       (swelling capability not available yet)
C
     EnergyD
                 (dp,ar(3))
C
                                   energy density
С
                                   EnergyD(1) elastic energy density
С
                                   EnergyD(2) plastic energy density
C
                                   EnergyD(3) creep energy density
     MatRotGlb
С
               (dp,ar(3,3))
                                   The rotation matrix from global
C
                                   to material coordinate system
*******************
С
```

6.1.3. Creating a New Element by Directly Accessing the ANSYS Database

The next few pages describe the user subroutines and supporting subroutines you use to create new elements. Using these subroutines, you can create new element types, add them to the ANSYS element library, and use them as "regular" elements. You can create up to six independent element types (names USER100 - USER105). For demonstration purposes, example copies of the subroutines for MASS21, the structural mass element, and LINK8, the 3-D spar element, are included on the ANSYS distribution medium as uel100 and uel101 respectively.

6.1.3.1. User Subroutines

Subroutines uec100 through uec105 describe the element characteristics. Subroutine elccmt (on the distribution medium) describes the input for these subroutines in detail. You can use subroutines uex100 through uex105 to override default logic. Subroutines uec100 through uec105 define parameters such as:

- 2-D or 3-D geometry
- Degree of freedom set

- Symmetric or unsymmetric matrix
- Number of nodes
- Number of body loads (for example, temperatures)
- Number of surface loads (for example, pressures)
- Number of real constants
- Number of variables to be saved
- Number of rows in element matrices
- Linear or nonlinear element.

Subroutines uel100 through uel105 calculate the element matrices (stiffness, specific heat, and so on), the element load vector (force, heat flow, and so on), and any element output quantities. The element printout also is generated, and the variables to be saved are calculated and stored in the results file.

Other user subroutines available for manipulating element information include the following:

- Subroutines uep100 through uep105 provide printed output of line elements. The general ANSYS postprocessor, POST1, calls the subroutines, or you can call them using uel100 through uel105.
- Subroutine usertr allows access to the nodal transformations.
- Subroutine userac describes some of the data handling.

6.1.3.2. Subroutine uec100 (Defining Characteristics of the usr100 Subroutine)

```
*deck.uec100
                                 USERSDISTRIB
      subroutine uec100 (elcdn,ielc,kerr)
      **** this subroutine defines the characteristics of user100.
С
C
          *** copyright(c) 2008 SAS IP, Inc. All rights reserved.
С
          *** ansys, inc.
С
  *** Notice - This file contains ANSYS Confidential information ***
C
С
      typ=int,dp,log,chr siz=sc,ar(n) intent=in,out,inout
С
С
  input arguments:
С
      variable (typ,siz,intent)
                                  description
С
С
      ielc (int,ar(IELCSZ),inout) - element characteristics
С
С
      kerr
           (int,sc,inout)
                                 - error flag up to this point.
                                    (do not initialize to zero)
С
С
  output arguments:
C
      variable (typ,siz,intent) description
С
      elcdn (chr,sc,out)
                                - name of element
С
С
      ielc (int,ar(IELCSZ),inout) - element characteristics
С
                                 - error flag (set to 1 if error)
     kerr (int,sc,inout)
C
       note to programmers: the validity of keyopt values may be checked here
С
```

6.1.3.2.1. Subroutines uec101 through uec105

The input and output arguments for subroutines uec101, uec102, uec103, uec104, and uec105 is identical to the uec100 subroutine listed above.

6.1.3.3. Subroutine uex100 (Overriding Element Characteristic Defaults)

```
USERSDISTRIB
*deck,uex100
      subroutine uex100 (ielc,kerr)
      *** subroutine to override element characteristic defaults ***
      *** hence, this routine is needed only in rare cases.
C
C
С
          *** copyright(c) 2008 SAS IP, Inc. All rights reserved.
          *** ansys, inc.
C
c *** Notice - This file contains ANSYS Confidential information ***
C
      *** input and output are the same as for uec100, except that this
C
      \ensuremath{^{***}} logic is called after the defaulting logic is finished.
      *** this defaulting is done in ansys subroutine echdft(not a upf).
C
C
      *** as indicated above, this routine is rarely needed, but if it is
      *** desired to see the effect of echdft, you may print out the ielc array
      *** leaving uec100 and print it out again entering this routine.
C
C
      typ=int,dp,log,chr siz=sc,ar(n) intent=in,out,inout
C
С
   input arguments:
С
      variable (typ,siz,intent)
                                     description
      ielc (int,ar(IELCSZ),inout) - element characteristics
С
                                   - error flag up to this point.
C
      kerr
            (int,sc,inout)
С
                                    (do not initialize to zero)
C
  output arguments:
C
С
     variable (typ,siz,intent)
                                     description
С
      ielc (int,ar(IELCSZ),inout) - element characteristics
С
С
      kerr
             (int,sc,inout)
                                   - error flag (set to 1 if error)
      \ensuremath{^{***}} standard defaults are taken. the final results are given with
C
      *** the debug accessed with /debug,,,,,1
C
```

6.1.3.3.1. Subroutines uex101 through uex105

The source code for subroutines uex101, uex102, uex103, uex104, and uex105 is identical to the code for subroutine uex100 listed above.

6.1.3.4. Subroutine uel100 (Computing Element Matrices, Load Vectors, and Results)

```
*deck,uel100
                                 USERSDISTRIB
     subroutine uel100 (elem,ielc,elmdat,eomask,nodes,locsvrL,kelreq,
    x kelfil,nr,xyz,u,kelout,zs,zass,damp,gstif,zsc,zscnr,elvol,elmass,
    x center,elener,edindxL,lcerstL)
c --- general lumped mass is demonstrated -----
c *** primary function:
       1. compute element matrices, load vectors, and results
c *** secondary functions:
        2. maintain element solution data
c *** user programmable functions may not be used in parallel processing ***
c *** Notice - This file contains ANSYS Confidential information ***
          *** copyright(c) 2008 SAS IP, Inc. All rights reserved.
С
         *** ansys, inc.
C
  input arguments:
       elem (int,sc,in)
                                 - element label (number)
C
              (int,ar(IELCSZ),in) - array of element type characteristics
С
                                    (IELCSZ = array size, defined in echprm)
C
С
       elmdat (int,ar(EL_DIM),in) - array of element data
       eomask (int,sc,in)
                                  - bit pattern for element output
```

```
(see outpcm)
C
        nodes (int,ar(nnod),in)
                                    - array of element node numbers
С
С
                                    (nnod = number of nodes; 1 in this case)
        locsvrL (int,sc,in)
                                    - location of the saved variables
С
                                                on file .esav for this element
C
С
        kelreq (int,ar(10),in)
                                    - matrix and load vector form requests
                                    (indices for kelreq are given with output
С
C
                                                              arguments below)
        kelfil (int,ar(10),in)
                                    - keys indicating incoming matrices and
С
С
                                      load vectors (indices for kelfil are the
                                      same as given for kelreq with output
C
                                      arguments below)
C
С
        nr
               (int.sc.in)
                                    - matrix and load vector size
С
               (dp,ar(6,nnod),in) - nodal coordinates (orig) and rotation angle
        xyz
C
               (dp,ar(nr,5),in)
                                    - element nodal solution values
        11
C
  output arguments:
        kelout (int,ar(10),out)
                                    - keys indicating created matrices and
C
C
                                           load vectors (indices for kelout
                                              are the same as for kelreq below,
C
                                              as well as kelin and kelout later)
С
С
        zs
               (dp,ar(nr,nr),inout) - stiffness/conductivity matrix (kelreq(1))
               (dp,ar(nr,nr),inout) - mass matrix
                                                                     (kelreq(2))
        zass
C
               (dp,ar(nr,nr),inout) - damping/specific heat matrix (kelreq(3))
С
        damp
С
        gstif
               (dp,ar(nr,nr),inout) - stress stiffness matrix
                                                                     (kelreq(4))
                                   - applied f vector
C
        ZSC
               (dp,ar(nr),out)
                                                                     (kelreg(5))
C
        zscnr
               (dp,ar(nr),out)
                                   - n-r restoring f vector
                                                                     (kelreg(6))
                                     or imaginary f vector
C
                                                                     (kelreg(7))
        elvol (dp,sc,out)
                                    - element volume
С
        elmass (dp,sc,out)
                                   - element mass
C
        center (dp,ar(3),out)
C
                                   - centroid location
С
        elener (dp,ar(5),out)
                                    - element energies
С
        edindxL(LONG,ar(25),out)
                                   - element result data file indexes
                                   - position on result file
        lcerstL(LONG,sc,inout)
C
C
```

6.1.3.4.1. Subroutines uel101 through uel105

The input and output arguments for subroutines uel101, uel102, uel103, uel104, and uel105 is identical to subroutine uel100 listed above.

6.1.3.5. Subroutine uep100 (Printing Output for User Elements in POST1 via PRESOL, ELEM)

```
*deck,uep100
                                  USERSDISTRIB
      subroutine uep100 (iott,elem,nodes,mat, kept,tem,
     x kemn,fluen, kems,force, kens,sig, keel,epel,
     x keth,eptho,epswel,epino, kenl,sigepl,sigrat,hpres,epeq,
     x kepl,eppl, kecr,epcrp)
С
c *** primary function:
                           produce printed output for user100
C
          *** copyright(c) 2008 SAS IP, Inc. All rights reserved.
С
          *** ansys, inc.
С
  *** Notice - This file contains ANSYS Confidential information ***
С
   ****** this subroutine is provided for user information *******
C
С
 *** user programmable features may not be used in parallel processing ***
С
  input arguments:
               (int,sc,in)
                                   - output unit number
С
      iott
C
      elem
               (int,sc,in)
                                   - element number
               (int,ar(2),in)
                                   - node numbers
      nodes
C
С
      mat
               (int,sc,in)
                                   - material number
С
      kept
               (int,sc,in)
                                   - key to print temperatures
               (dp,ar(2),in)
                                   - nodal temperatures
C
      t.em
```

```
(inr,sc,in)
                                  - key to print fluences
C
      kemn
                                  - neutron fluences
С
               (dp,ar(2),in)
С
     kems
               (int,sc,in)
                                  - key to print moment forces
С
     force
               (int,sc,in)
                                  - member force fx
С
     kens
               (int,sc,in)
                                   - key to print strains
С
      sig
               (dp,sc,in)
                                  - axial stress
                                  - key to print elastic strain
     keel
              (int,sc,in)
C
C
      epel
              (dp,sc,in)
                                  - axial elastic strain
     keth
              (int,sc,in)
                                  - key to print thermal, initial, swelling strai
С
С
     eptho
               (dp,sc,in)
                                   - axial thermal strain
      epswel
               (dp,sc,in)
                                  - swelling strain
С
                                  - initial axial strain
C
     epino
               (dp,sc,in)
     kenl
               (int,sc,in)
                                  - key set if any nonlinear materials present
С
С
     sigepl
               (dp,sc,in)
                                  - stress in stress-strain curve
               (dp,sc,in)
C
     sigrat
                                  - stress ratio
С
     hpres
               (dp,sc,in)
                                  - hydrostatic pressure
С
     epeq
               (dp,sc,in)
                                  - plastic equivalent strain
                                  - key to print plastic strain
     kepl
               (int,sc,in)
C
      eppl
               (dp,sc,in)
                                  - plastic strain
               (int,sc,in)
С
     kecr
                                  - key to print creep strain
               (dp,sc,in)
                                   - creep strain
     epcrp
C
  output arguments:
C
                       none
```

6.1.3.5.1. Subroutines uep101 through uep105

The source code for subroutines uep101, uep102, uep103, uep104, and uep105 is identical to subroutine uep100 listed above.

6.1.3.6. Subroutine usertr (Adjusting the Nodal Orientation Matrix)

```
USERSDISTRIB
*deck,usertr
     subroutine usertr (node,tr)
c *** primary function: adjust nodal orientation matrix
     secondary function: study nodal orientation matrix
       accessed with ielc(notran) = -100
C
C
          *** copyright(c) 2008 SAS IP, Inc. All rights reserved.
С
          *** ansys, inc.
C
c *** Notice - This file contains ANSYS Confidential information ***
C
С
      typ=int,dp,log,chr,dcp siz=sc,ar(n) intent=in,out,inout
С
  input arguments:
C
     variable (typ,siz,intent)
                                description
С
С
               (int,sc,in)
                                - node number being acted upon
С
               (dp,ar(32,32),inout) - nodal to global orientation matrix
  output arguments:
C
С
     variable (typ,siz,intent)
                                   description
С
               (dp,ar(32,32),inout) - nodal to global orientation matrix
C
С
           tr is a matrix that is already defined based on the degrees
          of freedom selected.
C
          it does not normally need to be changed.
С
          it may be printed out here to study. its functional size is
С
          nr by nr, where nr is the number of degrees of freedom in the
          element
С
```

6.1.3.7. Subroutine userac (Accessing Element Information)

This subroutine is provided for demonstration purposes.

```
*deck.userac
                                  USERSDISTRIB
     subroutine userac (elem)
c *** primary function: To demonstrate user access of element information.
c --- Given the element number, all information about the element is avaiable.
c --- Starting with elmdat, one can get the element type, real constant number,
c --- the material number, the element coordinate system number, as well as
c --- the node numbers. Then, one can get more information about any or all
c --- of these things. The below demonstrates getting the element type and
c --- real constants.
          *** copyright(c) 2006 SAS IP, Inc. All rights reserved.
С
          *** ansys, inc.
C
c *** Notice - This file contains ANSYS Confidential information ***
C
С
   input arguments:
C
      variable (typ,siz,intent)
                                   description
              (int,sc,in)
                                 - element number
C
c output arguments:
C
     none
```

6.2. Supporting Subroutines for Element Creation

The subroutines described on the next few pages support the user subroutines used to create new elements (using the database-access method described in *Creating a New Element by Directly Accessing the ANSYS Database* (p. 128)).

6.2.1. Subroutine nminfo (Returning Element Reference Names)

```
*deck.nminfo
     subroutine nminfo (ielc,rname)
c *** primary function:
                          set element reference names
c *** secondary functions: none
c *** Notice - This file contains ANSYS Confidential information ***
C
С
  input arguments:
С
     variable (typ,siz,intent)
                                   description
С
      ielc
              (int,ar(*),inout)
                                   - element characteristic vector
С
     rname
               (chr,sc,in)
                                   - 8 character reference name
C
  output arguments:
С
     variable (typ,siz,intent)
                                   description
            (int,ar(*),inout)
                                   - element characteristic vector with
С
                                     element name encoded
С
C
```

6.2.2. Subroutine sygidx (Fetching the Index for Saved Variables)

```
*deck,svgidx
      subroutine svgidx (locsvr,svindx)
c *** primary function:
                          get the index for saved variables
c *** Notice - This file contains ANSYS Confidential information ***
  input arguments:
C
               (LONGINT,sc,in)
                                - pointer to location of index
      locsvr
  output arguments:
C
С
      svindx (int,ar(20),out) - the 20 word index of svr variables
                                   1,2-starting loc of this eles svr sets
C
С
                                   3- length of eles svr sets
```

```
4-11-relative starting loc for each set
С
С
                                       4-structural svrs
С
                                       5-thermal/electric/fluid svrs
                                       6-magnetic svrs
С
С
                                       7-nonlinear svrs
С
                                       8-plasticity svrs
                                       9-creep svrs
C
                                      10-coupled svrs
                                      11-user svrs
C
С
                                      12-initial strain svrs
                                      13-section data after FiberSIM conversion
С
                                                                  (shell181 only)
C
С
                                      14-20 spares
```

6.2.3. Subroutine syrget (Fetching Saved Variable Data for an Element)

```
*deck,svrget
      subroutine syrget (syindx,nset,nsyr,syr)
c *** primary function:
                          get svr data set for an element
c *** Notice - This file contains ANSYS Confidential information ***
   input arguments:
      svindx (int,ar(20),in) - index for svr for this element (see svgidx)
C
С
                               - the set number in this index
                                   = 1 - structural syrs
C
С
                                      2 - thermal/electric/fluid svrs
С
                                      3 - magnetic svrs
                                      4 - nonlinear svrs
C
С
                                   = 5 - plasticity svrs
C
                                      6 - creep svrs
                                      7 - coupled svrs
С
                                      8 - user svrs
С
                                      9 - initial stress syrs
C
C
                                                  (2,42,82,45,92,95 only)
С
                                   = 10 - section data after FiberSIM conversion
C
                                                                  (shell181 only)
С
                                   = 11-17 - spares (note that the first three
                                              items in svindx are not available)
С
                                 - number of dp words expected in this set
               (int,sc,inout)
C
      nsvr
С
   output arguments:
                                 - number of dp words in this set
               (int,sc,inout)
C
     nsvr
               (dp,ar(nsvr),in) - data in this set
С
      svr
```

6.2.4. Subroutine syrput (Writing an Element's Saved Variable Set)

```
*deck,svrput
      subroutine svrput (svindx,nset,leng,svr)
c *** primary function:
                          write out a svr data set for an element
c *** Notice - This file contains ANSYS Confidential information ***
   input arguments:
С
C
      svindx
               (int,ar(20),inout) - the index for svr for this element
                                                                     (see svgidx)
C
               (int,sc,in)
                                 - the set number in this index (same as syrget)
C
С
                                   = 1 - structural svrs
                                      2 - thermal/electric/fluid svrs
C
                                      3 - magnetic svrs
С
                                      4 - nonlinear svrs
C
                                      5 - plasticity svrs
C
                                      6 - creep svrs
С
                                      7 - coupled svrs
                                      8 - user svrs
С
```

```
= 9 - initial stress svrs
C
                                                   (2,42,82,45,92,95 only)
С
C
                                    = 10 - section data after FiberSIM conversion
С
                                                                   (shell181 only)
С
                                    = 11-17 - spares (note that the first three
C
                                              items in svindx are not available)
                                  - number of dp words in this set
С
               (dp,ar(leng),in) - data in this set
С
  output arguments:
               (int,ar(10,2),inout) - updated index
```

6.2.5. Subroutine sypidx (Writing the Saved Variable Element Index to a File)

```
*deck,svpidx
     subroutine svpidx (locsvr,svindx)
c *** primary function: write the svr element index onto file
c *** secondary functions: update the locsvr pointer to next element
c *** Notice - This file contains ANSYS Confidential information ***
  input arguments:
     locsvr (LONGINT,sc,inout)
                                    - pointer to start of svr for element
C
      svindx
              (int,ar(10,2),in)
                                     - index to svr for this element
C
                                         low and high parts of 64 bit address
C
С
  output arguments:
      locsvr
             (LONGINT,sc,inout)
                                     - pointer to start of svr for next element
```

6.2.6. Subroutine mreuse (Determining Which Element Matrices Can Be Reused)

```
*deck, mreuse
      subroutine mreuse (kelrqq,kelfil,elem,ielc,kmasrt,knlmg,kconve,
     x kpheno, kprop, nprop, propo, krvro, rvr, rvro, amodo, asymo, kelin)
c *** primary function:
          determine which Matrices can be REUSEd and which must be recomputed
C
C
           from iteration to iteration.
С
           Note: a few special elements have some supplementary logic
           to adjust these results further. No attempt as been made to
           include all such logic in these routines.
C
C
           Second note: this logic is essentially the same as the old
           sfrm logic. Hopefully, further simplifications and enhancements
С
           will be made in the future. (Especially in gap elements and in
C
C
           multilaver elements)
C
           the whole idea of kpheno, a holdover from the sfrm routines,
           needs to be looked at and possibly eliminated.
С
C
  *** Notice - This file contains ANSYS Confidential information ***
С
C
   input arguments:
C
               (int,ar(10),in)
                                   - request keys (needed for this analysis)
С
      kelrqq
С
      kelfil
               (int,ar(10),in)
                                   - keys indicating matrices on the file
      elem
                                    - element number
С
               (int,sc,in)
С
      ielc
               (int,ar(IELCSZ),in) - array of element type characteristics
      kmasrt (int,sc,in)
                                    - does the mass matrix have rotational DOF?
C
                                       0 - no
                                                  1 - yes(with nlgeom, sfrmln)
C
С
      knlmg
               (int,sc,in)
                                    - nonlinear magnetic curve exists in this
C
                                       element
С
                                        0 - no
                                                   1 - yes
C
      kconve
               (int,sc,in)
                                    - key indicating existence of convections
C
                                       in this element
С
                                        0.1 - no
                                                    2 or more - yes
С
                                       must be input as 'i' if not used, as is
С
                                       changed in this routine(for analyzer).
```

```
i = 0 must be used in calling routine
С
С
                                       if kpheno = 1.
С
      kpheno
               (int,sc,in)
                                    - key for type of phenomenon/level of check
                                       0 - structural like old sfrmln,1s,3n,3s,fl
С
                                       1 - thermal like old sfrmlc,1t,2t,3t
С
С
                                       2 - electrical/magnetic like some of old
С
                                                                      sfrmpo
C
                                       3 - general
                                                      like old sfrmoo
                                   - key indicating which material properties
C
      kprop
               (int,sc,in)
С
                                       in the prop vector that need to be
                                      checked (see below)
С
     nprop
               (int,sc,in)
                                   - number of properties
C
С
     prop
               (dp,ar(nprop),in)
                                   - current mat props
С
      propo
               (dp,ar(nprop),inout) - previous material properties
C
      krvro
               (int,sc,in)
С
             = 0 - real constants are used by this element, and the old
С
                    values(rvro) have been saved; or the element does not
                    use real constants. Any change of real constants
C
                    causes all matrices to be reformed.
             = 1 - real constants are used by this element and the old
C
                    values(rvro) have been saved. However, any change
С
                    of real constants will cause the run to terminate,
C
                    because the results would be too unpredictable.
C
С
                    (e.g. gap elements)
             = 2 - element is nonlinear, so do not bother to check
C
             = 3 - real constants are used by this element, and the old
C
С
                    values(rvro) have been saved. However, no checking is
                    done in this routine because of needed customized logic.
C
             = 4 - real constants are used by this element, but the old
C
                    values(rvro) have not been saved because it was
С
                    decided not to use this much storage. therefore, no check
С
                    can be made to determine if matrices should be reformed.
C
                    (e.g. 100 layer elements)
             = 5 - real constants are used by this element, but the old
C
С
                    values(rvro) have not been saved because the real
                    constants have no effect on matrix formulation.
C
                    (e.g. acoustic elements)
C
               (dp,ar(*),in)
С
      rvr
                                   - current real constants
               (dp,ar(*),inout)
                                   - previous real constants
С
      rvro
                                   - previous value of mode
C
      amodo
               (dp,sc,inout)
                                   - previous value of isym
С
      asymo
               (dp,sc,inout)
C
С
   output arguments:
               (dp,ar(nprop),inout) - current material properties
C
               (dp,ar(*),inout) - current real constants
C
     rvro
     amodo
               (dp,sc,inout)
                                   - current value of mode
С
С
      asvmo
               (dp,sc,inout)
                                   - current value of isym
     kelin
               (int,ar(10),out)
                                   - keys indicating matrices to form
C
```

6.2.7. Subroutine subrd (Reading Element Load Data for a Substructure Generation Run)

```
*deck, subrd
      subroutine subrd (iel,key,nd,vect,ka)
c *** primary function:
                           read element load data from file for substructure
                           generation run
c *** secondary functions: none
c *** Notice - This file contains ANSYS Confidential information ***
   input arguments:
               (int,sc,in)
                                  - element number
      iel
C
С
               (int,sc,in)
                                  - type of load data
С
                                   = 1 temperature
                                   = 2 fluences
C
С
                                   = 3 heat generation rates
С
                                   = 4 current densities
С
                                   = 9 end pressures (needed for beams/pipes)
```

```
=10 pressures
C
                                   =11 film coefficients
С
С
                                   =12 bulk temperatures
                                   =13 extra displacement shapes
С
                                   =14 thermal strains(eptho in el42)
С
                                   =15 thermal flux (as in el55)
С
                                   =16 initial strains(epino in el01)
С
                                   =17 magnetic virtual displacements
                                   =18 calculated source field(hsn in el96)
С
С
                                   =20 element load vector
                                   =30 copy - do not scale(tempev in el42)
C
                                       first load step only
C
                                  - number of data items
С
               (int,sc,in)
C
  output arguments:
               (dp,ar(nd),out)
                                  - array of load data
С
                                  - load activation key
C
      ka
               (int,sc,out)
                                    = 0 no load for this data
C
                                    = 1 load is active
```

6.2.8. Subroutine subwrt (Writing an Element Load Vector to a File for a Substructure Generation Run)

```
*deck.subwrt
     subroutine subwrt (iel,nvect,kkey,nd,vect,ref)
                          write element load vect to file for substructure
c *** primary function:
                           generation run
c *** secondary functions: none
c *** Notice - This file contains ANSYS Confidential information ***
  input arguments:
C
С
     iel
               (int,sc,in)
                                  - element number
С
      nvect
               (int,sc,in)
                                  - number of load vectors
С
                                    (current load step number)
                                  - type of load vect
C
               (int,sc,in)
                                  = 1 temperature
C
                                   = 2 fluences
С
                                   = 3 heat generation rates
C
C
                                   = 4 current densities
С
                                   = 9 end pressures
С
                                   =10 pressures
                                   =11 film coefficients
C
С
                                   =12 bulk temperatures
C
                                  =13 extra displacement shapes
                                   =14 thermal strains(eptho in el42)
C
                                   =15 thermal flux (as in el55)
С
                                   =16 initial strains(epino in el01)
С
                                   =17 magnetic virtual displacements
C
                                   =18 calculated source field(hsn in el96)
C
С
                                   =20 element load vector
                                  =30 copy - do not scale(tempev in el42)
С
                                 - number of vect items
С
      nd
               (int,sc,in)
               (dp,ar(nd),in)
                                 - array of load data
C
      vect.
                                  - reference value for zero load
               (dp,sc,in)
c output arguments: none
```

6.2.9. Subroutine rvrget (Fetching Real Constants for an Element)

```
typ=int,dp,log,chr,dcp siz=sc,ar(n),func
                                                    intent=in,out,inout
C
С
      variable (typ,siz,intent)
                                     description
         iel
                  (int,sc,in)
                                     - element number
C
         ireal
С
                  (int,sc,in)
                                     - real constant set number
C
         ielc
                  (int,ar(*),in)
                                     - elment type characteristics
C
      output arguments:
                                     - number of real variables
                  (int,sc,out)
C
        nrvr
С
         rvr
                  (dp,ar(*),out)
                                     - element real constants
c *** mpg magnetic element usage - iel ?
```

6.2.10. Subroutine propev (Evaluating a Group of Material Properties)

```
*deck,propev
      subroutine propev (iel,mtr,lp,tem,prop,n)
c *** primary function: to evaluate a group of material properties
      propev is used to pass two or more material property numbers
С
С
      thru the lp array to determine which temperature dependent
      material properties are to be evaluated.
С
      thus, the 3 propel calls:
C
С
         call propel (elem,mat, 1,tem,e(1))
         call prope1 (elem,mat,10,tem,alpha)
C
         call propel (elem,mat,13,tem,dens)
С
      should be combined as:
С
         integer lp(3)
C
         data lp /1,10,13/
C
         call propev (elem,mat,lp(1),tem,prop(1),3)
c *** Notice - This file contains ANSYS Confidential information ***
   input arguments:
C
      iel (int,sc,in)
                             - element number
С
                             - material number(input quantity mat, mat comma
С
      mtr (int,sc,in)
                             - keys for which specific value is requested
          (int,ar(n),in)
C
                                       each group must be in ascending
С
С
                                       order (ex,ey,ez, etc)
С
                                    if negative, a required property
C
                                    if zero, leave prop term unchanged
         ---- MP command labels -----
C
С
         EX = 1, EY = 2, EZ = 3, NUXY = 4, NUYZ = 5, NUXZ = 6, GXY = 7, GYZ = 8,
         GXZ = 9, ALPX=10, ALPY=11, ALPZ=12, DENS=13, MU =14, DAMP=15, KXX =16,
         KYY =17, KZZ =18, RSVX=19, RSVY=20, RSVZ=21, C =22, HF =23, VISC=24,
C
         EMIS=25, ENTH=26, LSST=27, PRXY=28, PRYZ=29, PRXZ=30, MURX=31, MURY=32,
С
         \texttt{MURZ=33, PERX=34, PERY=35, PERZ=36, MGXX=37, MGYY=38, MGZZ=39, EGXX=40,}
С
         EGYY=41, EGZZ=42, SBKX=43, SBKY=44, SBKZ=45, SONC=46, SLIM=47, ELIM=48,
C
         USR1=49, USR2=50, USR3=51, USR4=51, FLUI=53, ORTH=54, CABL=55, RIGI=56,
С
         HGLS=57, BM =58, QRAT=59, REFT=60, CTEX=61, CTEY=62, CTEZ=63, THSX=64,
С
С
         THSY=65, THSZ=66, DMPR=67, LSSM=68,
                                                  =69,
                                                           =79,
                                                                    =71,
                      =74,
                               =75,
                                         =76,
                                                  =77,
                                                           =78,
                                                                     =79,
                                                                              =80
С
                                                (see mpinit for uncommented code)
C
       (see chapter 2 of the elements volume of the user's manual
С
С
        for a detailed description))
                                  - temperature at which to evaluate material
С
               (dp,sc,in)
                                  - number of properties to be evaluated.
С
               (int,sc,in)
С
                                     (20 maximum)
                                     If n = 1, use propel instead.
С
   output arguments:
C
                                 - values of material property
               (dp,ar(n),out)
c *** mpg propev < ell17,el53,elofrm,gtprop,el96: evaluate material properties
```

6.2.11. Subroutine prope1 (Evaluating One Material Property)

```
*deck,propel
     subroutine propel (iel,mtr,icon,tem,prop1)
c *** primary function: to evaluate one material property
С
                           (if multiple material properties are to
С
                            be evaluated, use propev)
c *** secondary functions: to ensure that certain required props are present
c *** Notice - This file contains ANSYS Confidential information ***
  input arguments:
C
C
      iel
              (int,sc,in)
                                - element number
      mtr
              (int,sc,in)
                                - material number
C
                                - key for which specific value is requested
С
              (int,sc,in)
С
                                  (negative if property is required)
         ---- MP command labels -----
C
         EX = 1, EY = 2, EZ = 3, NUXY = 4, NUYZ = 5, NUXZ = 6, GXY = 7, GYZ = 8,
         GXZ = 9, ALPX=10, ALPY=11, ALPZ=12, DENS=13, MU = 14, DAMP=15, KXX = 16,
C
С
         KYY =17, KZZ =18, RSVX=19, RSVY=20, RSVZ=21, C =22, HF =23, VISC=24,
         EMIS=25, ENTH=26, LSST=27, PRXY=28, PRYZ=29, PRXZ=30, MURX=31, MURY=32,
C
        MURZ=33, PERX=34, PERY=35, PERZ=36, MGXX=37, MGYY=38, MGZZ=39, EGXX=40,
C
С
         EGYY=41, EGZZ=42, SBKX=43, SBKY=44, SBKZ=45, SONC=46, SLIM=47, ELIM=48,
С
         USR1=49, USR2=50, USR3=51, USR4=51, FLUI=53, ORTH=54, CABL=55, RIGI=56,
         HGLS=57, BM =58, QRAT=59, REFT=60, CTEX=61, CTEY=62, CTEZ=63, THSX=64,
C
         THSY=65, THSZ=66, DMPR=67, LSSM=68,
С
                                                =69,
                                                       =79,
                                                                  =71,
             =73.
                     =74,
                             =75, =76,
                                                =77,
                                                         =78,
                                                                  =79,
                                                                           =80
C
                                               (see mpinit for uncommented code)
C
               (dp,sc,in)
C
     tem
                                 - temperature at which to evaluate material
  output arguments:
                                - value of material property
      prop1
              (dp,sc,out)
```

6.2.12. Subroutine pstev1 (Evaluating Material Properties for 1-D Elements)

```
*deck,pstev1
      subroutine pstev1 (elem, matin, tem, prop)
c *** primary function:
                         to evaluate material properites for 1-d elements
c *** Notice - This file contains ANSYS Confidential information ***
C
С
   input arguments:
                                   - element number (for anserr)
C
     elem (int.sc.in)
C
              (int,sc,in)
                                   - material reference number
                                     if negative, no required properties
C
                                   - temperature for evaluation
C
              (dp,sc,in)
     t.em
С
С
  output arguments:
                                   - material properties: ex,nuxy,gxy,alpx,dens
              (dp,ar(5),out)
C
     prop
```

6.2.13. Subroutine tbuser (Retrieving User Table Data)

```
c mat (int,sc,in) - material property number
c numitm (int,sc,in) - the number of data items requested
c output arguments:
c tbprop (dp,ar(numitm),out) - array of tb data
```

6.2.14. Subroutine plast1 (Updating an Element's Plastic History)

```
*deck,plast1
     subroutine plast1 (option, elem, intpt, mat, kstartL, tem, dtem, e,
                         ktform, dens, flu, dflu, epel, eppl, statev, usvr,
                         epeq,plwork,sigepl,sigrat,et)
 *** primary function:
                          to update the plastic history (for 1 component)
С
                           used by: LINK1, LINK8, BEAM23, BEAM24, and
                                     SOLID65(reinforcing)
C
c *** secondary functions: to compute the material tangent matrix if requested
c *** Notice - This file contains ANSYS Confidential information ***
  input arguments:
С
С
     option (int,sc,in)
                                  - plasticity option
                                  - element number (label)
С
     elem
               (int,sc,in)
                                  - element integration point number
C
     intpt
              (int,sc,in)
              (int,sc,in)
                                  - material reference number
C
     kstartL (intL,sc,in)
                                   - virtual starting address of the data table
C
                                   - temperature at the end of this substep
C
     t.em
              (dp,sc,in)
С
     dtem
              (dp,sc,in)
                                   - temperature increment over this substep
              (dp,sc,in)
                                  - elastic modulus
С
     ktform
                                  - request key for tangent matrix formation
              (int.sc.in)
C
С
              (dp,sc,in)
                                  - material density
     flu
                                  - fluence at the end of this substep
C
              (dp,sc,in)
     dflu
              (dp,sc,in)
                                  - fluence increment over this substep
С
              (dp,sc,inout)
                                  - modified total strain (trial strain)
С
     epel
              (dp,sc,inout)
                                 - plastic strain at previous substep
C
     eppl
С
     statev
              (dp,ar(6),inout) - state variables at previous substep
С
     usvr
               (dp,ar(*),inout)
                                  - user-defined state variables (for userpl)
     epeq
              (dp,sc,inout)
                                   - effective plastic strain at prev substep
C
С
     plwork
              (dp,sc,inout)
                                   - accumulated plastic work at prev substep
  output arguments:
C
              (dp,sc,inout)
                                  - elastic strain
С
     epel
С
     eppl
               (dp,sc,inout)
                                   - updated plastic strain
                                  - updated state variables
              (dp,ar(6),inout)
     statev
C
С
     usvr
               (dp,ar(*),inout)
                                  - updated user-defined state variables
                                  - updated effective plastic strain
              (dp,sc,inout)
C
     epeq
C
     plwork
              (dp,sc,inout)
                                  - updated accumulated plastic work
     sigepl
              (dp,sc,out)
                                  - stress value on stress-strain curve
С
                                   - ratio of trial stress to yield stress
     sigrat
              (dp,sc,out)
C
              (dp,sc,out)
С
                                   - tangent modulus
  internal variables:
C
     deppl
              (dp,sc)
                                   - equivalent plastic strain increment
```

6.2.15. Subroutine plast3 (Updating an Element's Plastic History, 4 or 6 components)

```
c *** secondary functions: to compute the material tangent matrix if requested
c *** Notice - This file contains ANSYS Confidential information ***
C
   input arguments:
      option
              (int,sc,in)
                                   - plasticity option
C
C
      elem
               (int,sc,in)
                                   - element number (label)
      intpt
               (int,sc,in)
                                   - element integration point number
C
С
      mat
               (int,sc,in)
                                   - material reference number
              (intL,sc,in)
                                   - virtual starting address of the data table
C
      kstartL
                                   - number of stress/strain components (4 or 6)
C
      ncomp
               (int,sc,in)
С
      tem
               (dp,sc,in)
                                   - temperature at the end of this substep
С
      dtem
               (dp,sc,in)
                                   - temperature increment over this substep
     prop
               (dp,ar(9),in)
                                   - material property array (ex,ey,ez,
C
С
                                     gxy,gyz,gxz, uxy,uyz,uxz)
               (dp,ar(ncomp,ncomp),in) - elastic stress-strain matrix
C
                                  - request key for tangent matrix formation
      ktform
               (int,sc,in)
C
C
      dens
               (dp,sc,in)
                                   - material density
      flu
               (dp,sc,in)
                                   - fluence at the end of this substep
C
      dflu
               (dp,sc,in)
                                   - fluence increment over this substep
С
С
      epel
               (dp,ar(ncomp),inout) - modified total strain (trial strain)
               (dp,ar(ncomp),inout) - plastic strain at previous substep
      eppl
C
               (dp,ar(ncomp,6),inout) - state variables at previous substep
С
      statev
С
      usvr
               (dp,ar(*),inout)

    user-defined state variables (for pluser)

C
               (dp,sc,inout)
                                   - effective plastic strain at prev substep
      epeq
C
      plwork
               (dp,sc,inout)
                                   - accumulated plastic work at prev substep
                                   - plane stress key (form dtt if kplst=1)
C
      kplst
               (int,sc,in)
  output arguments:
С
               (dp,ar(ncomp),inout)- elastic strain
С
      epel
С
      eppl
               (dp,ar(ncomp),inout) - updated plastic strain
C
      statev
               (dp,ar(ncomp,6),inout)- updated state variables
               (dp,ar(*),inout) - updated user-defined state variables
C
      usvr
С
      epeq
               (dp,sc,inout)
                                   - updated effective plastic strain
      plwork
               (dp,sc,inout)
                                   - updated accumulated plastic work
C
      sigepl
               (dp,sc,out)
                                   - stress value on stress-strain curve
C
                                   - ratio of trial stress to yield stress
C
      sigrat
               (dp,sc,out)
С
      dt.
               (dp,ar(ncomp,ncomp),out) - material modulus modified by dscpar
               (dp,ar(ncomp,ncomp),out)- consistent tangent modulus
C
      dtt
                                           (formed only if kplst=1)
С
  internal variables:
      deppl
              (dp,sc)
                                   - equivalent plastic strain increment
```

6.2.16. Subroutine creep1 (Updating an Element's Creep History)

```
*deck,creep1
      subroutine creep1 (option,elem,intpt,mat,kstartL,epel,e,epcrp,
     x statev.usvr.tem.dtem.fluen.dflu.sig)
c *** primary function:
                          to update the creep history for 1-d elements
                           used by: LINK1, LINK8, BEAM23, BEAM24, and
                                     SOLID65(reinforcing)
c *** Notice - This file contains ANSYS Confidential information ***
  input arguments:
С
С
      option
               (int,sc,in)
                                   - creep option
               (int,sc,in)
                                   - element number (label)
      elem
C
С
      intpt
               (int.sc.in)
                                   - element integration point number
                                   - material reference number
С
      mat.
               (int,sc,in)
С
      kstartL (intL,sc,in)
                                   - virtual starting address of the data table
                                   - elastic strain
С
      epel
               (dp,sc,inout)
                                   - elastic modulus
C
               (dp,sc,in)
С
               (dp,sc,inout)
                                   - creep strain at previous substep
      epcrp
               (dp,ar(7),inout)
                                   - state variables at previous substep
С
      statev
               (dp,ar(*),inout)
                                   - user-defined state variables (for usercr)
C
      usvr
               (dp,sc,in)
                                   - temperature at the end of this substep
С
      tem
```

```
dt.em
               (dp,sc,in)
                                   - temperature increment over this substep
C
С
               (dp,sc,in)
                                   - fluence at the end of this substep
С
      dflu
               (dp,sc,in)
                                   - fluence increment over this substep
                                   - elastic strain adjusted for creep increment
С
      epel
               (dp,sc,inout)
                                   - stress (not really used)
С
               (dp,sc,inout)
  output arguments:
C
C
      epcrp
               (dp,sc,inout)
                                   - updated creep strain
                                   - updated state variables
C
      statev
               (dp,ar(7),inout)
               (dp,ar(*),inout)
С
     usvr
                                   - updated user-defined state variables
               (dp,sc,inout)
                                   - stress (recomputed if requested)
      siq
```

6.2.17. Subroutine creep3 (Updating an Element's Creep History, 3-D Elements)

```
*deck,creep3
     subroutine creep3 (option,elem,intpt,mat,kstartL,ncomp,epel,e,
     x posn,d,epcrp,statev,usvr,tem,dtem,fluen,dflu,kplst,sig,hsig)
c *** primary function: to update the creep history for 3-d elements
        used by: PLANE02, PLANE13, PIPE20, PLANE42, SHELL43, SOLID45,
                  SHELL51, PIPE60, SOLID62, SOLID65, PLANE82, SHELL91,
С
С
                  SOLID92, SHELL93, SOLID95, SHELL143, SOLID191
c *** Notice - This file contains ANSYS Confidential information ***
   input arguments:
С
C
     option
               (int,sc,in)
                                    - creep option
С
      elem
               (int,sc,in)
                                    - element number (label)
     intpt
               (int,sc,in)
                                   - element integration point number
C
               (int,sc,in)
                                   - material reference number
     mat
C
С
     kstartL (intL,sc,in)
                                   - virtual starting address of the data table
                                   - number of stress/strain components (4 or 6)
C
     ncomp
               (int,sc,in)
               (dp,ar(ncomp),inout)- elastic strain
С
     epel
                                   - elastic young'S MODULUS
С
               (dp,sc,in)
                                   - poisson'S RATIO
C
     posn
               (dp.sc.in)
С
               (dp,ar(ncomp,ncomp),in) - elastic stress-strain matrix
С
               (dp,ar(ncomp),inout) - creep strain at previous substep
     epcrp
C
      statev
               (dp,ar(ncomp*5+2),inout) - state variables at previous substep
С
     usvr
               (dp,ar(*),inout) - user-defined state variables (for usercr)
С
      tem
               (dp,sc,in)
                                   - temperature at the end of this substep
                                   - temperature increment over this substep
     dtem
               (dp,sc,in)
C
                                   - fluence at the end of this substep
С
     fluen
               (dp,sc,in)
                                   - fluence increment over this substep
С
     dflu
               (dp,sc,in)
                                   - plane stress/plane strain key
     kplst
               (int,sc,in)
С
С
      sig
               (dp,ar(ncomp),inout) - stresses (not used in input)
                                 - hydrostatic stress (not used in input)
C
     hsig
               (dp,ar(1),inout)
  output arguments:
С
               (\texttt{dp}, \texttt{ar}(\texttt{ncomp}), \texttt{inout}) - \texttt{elastic strain adjusted for creep increment}
C
      epel
               (dp,ar(ncomp),inout) - updated creep strain
С
      epcrp
С
      statev
               (dp,ar(ncomp*5+2),inout) - updated state variables
               (dp,ar(*),inout) - updated user-defined state variables
C
     usvr
               (dp,ar(ncomp),inout) - stresses (redefined if c13 > 0)
С
С
     hsiq
               (dp,sc,inout)
                                   - hydrostatic stress (redefined if c13 > 0)
```

6.2.18. Subroutine swell1 (Updating an Element's Swelling History)

```
option
              (int,sc,in)
                                  - swelling option
C
С
     elem
               (int,sc,in)
                                  - element number (label)
С
     intpt
               (int,sc,in)
                                  - element integration point number
     mat
               (int,sc,in)
                                  - material reference number
С
     kstartL (intL,sc,in)
                                  - virtual starting address of the data table
С
                                  - swell strain at previous substep
С
     epswel
               (dp,sc,inout)
               (dp,sc,inout)
                                  - elastic strain
C
     epel
C
               (dp,sc,in)
                                  - elastic young'S MODULUS
     fluen
               (dp,sc,in)
                                  - fluence at the end of this substep
C
С
     dfluen
              (dp,sc,in)
                                  - fluence increment over this substep
               (dp,sc,in)
                                  - temperature at the end of this substep
C
      t.em
               (dp,sc,in)
     dtem
                                 - temperature increment over this substep
C
               (dp,ar(*),inout)
                                  - user-defined state variables (for usersw)
С
C
  output arguments:
                                  - elastic strain adjusted for swelling inc
С
     epel
               (dp,sc,inout)
C
     epswel
              (dp,sc,inout)
                                  - updated swelling strain
              (dp,ar(*),inout)
                                  - updated user-defined state variables
C
```

6.2.19. Subroutine swell3 (Updating an Element's Swelling History, 3-D Elements)

```
*deck,swell3
     subroutine swell3 (option,elem,intpt,mat,kstartL,ncomp,epswel,
                        epel,e,nuxy,fluen,dfluen,tem,dtem,usvr)
c *** primary function:
                        to update the swelling history for 3-d elements
       used by: PLANE02, PLANE13, PIPE20, PLANE42, SHELL43, SOLID45,
С
                 SHELL51, PIPE60, SOLID62, PLANE82, SHELL91, SOLID92,
С
                 SHELL93, SOLID95, SHELL143, SOLID191
C
c *** Notice - This file contains ANSYS Confidential information ***
С
  input arguments:
     option (int,sc,in)
                                  - swelling option
C
С
     elem
               (int,sc,in)
                                  - element number (label)
С
     intpt
               (int,sc,in)
                                  - element integration point number
                                  - material reference number
              (int,sc,in)
С
     mat.
     kstartL (intL,sc,in)
                                  - virtual starting address of the data table
С
               (int,sc,in)
                                  - number of stress/strain components (4 or 6)
C
              (dp,sc,inout)
                                 - swell strain at previous substep
C
     epswel
              (dp,ar(ncomp),inout) - elastic strain
С
     epel
С
               (dp,sc,in)
                             - elastic young'S MODULUS
     е
                                  - poisson'S RATIO
     nuxy
              (dp,sc,in)
C
С
     fluen
              (dp,sc,in)
                                 - fluence at the end of this substep
     dfluen
              (dp,sc,in)
С
                                  - fluence increment over this substep
              (dp,sc,in)
                                  - temperature at the end of this substep
С
     tem
     dt.em
               (dp,sc,in)
                                  - temperature increment over this substep
С
              (dp,ar(*),inout) - user-defined state variables (for usersw)
C
     usvr
С
 output arguments:
              (dp,ar(ncomp),inout) - elastic strain adjusted for swelling inc
C
     epel
С
      epswel
               (dp,sc,inout)
                                  - updated swelling strain
      usvr
               (dp,ar(*),inout)
                                  - updated user-defined state variables
```

6.2.20. Function elLenPsvrBuf (Determining additional ESAV Record for Plasticity)

```
determine additional esave record for plasticity
C
С
     input arguments
     ==========
C
С
              (int,sc,in)
                                 - material ID
С
     pl0pt
              (int,sc,in)
                                 - plasticity option
             (int,sc,in)
                                 - number of strain components (1,4, or 6)
C
     ncomp
C
     output arguments
C
     elLenPsvrBuf (int,sc,out) - number of extra data items saved
С
     local variables
     _____
C
```

6.2.21. Function nlget (Retrieving Material Nonlinear Property Information)

```
*deck,nlget
     function nlget (mat,iprop,prop)
c *** primary function:
                          get a material non-linear property (TB) table.
c *** Notice - This file contains ANSYS Confidential information ***
      input arguments:
C
        variable (typ,siz,intent)
С
                                    description
                 (int,sc,in)
                                    - material number
C
                                    - property number (tbpnum in tblecm)
                 (int,sc,in)
C
         iprop
С
                                        use 13 for tb,user
                                        use 14 for tb,nl
C
С
     output arguments:
                                   description
        variable (typ,siz,intent)
C
С
        nlget (int,sc,out)
                                   - number of property values
               (dp,ar(nlget),out) - vector of the property values
С
                                      (the first 15(tbhdsz) items are a header,
C
С
                                       given below. The terms are defined in
С
                                       tblecm.inc)
         --- terms of the descriptor record:
C
        header(1) = tbtyp
С
С
        header(2) = tbtems
        header(3) = temloc
C
С
        header(4) = dprtem
        header(5) = tbrow
C
С
        header(6) = tbcol
        header(7) = rowkey
С
        header(8) = nxtloc
С
С
        header(9) = nxttem
С
        header(10) = temptr
        header(11) = tbpt
C
        header(12) = tbsiz
С
С
        header(13) = tbopt
        header(14) = hypopt
C
        header(15) = tbnpts
```

6.2.22. Subroutine usereo (Storing Data in the nmisc Record)

```
c *** Notice - This file contains ANSYS Confidential information ***
C
С
  input arguments:
С
   variable (typ,siz,intent)
                                   description
С
    elem
             (int,sc,in)
                                   - element number
             (int,sc,in)
                                   - output unit number
C
    iout
C
    nbsvr
           (int,sc,in)
                                   - number of basic element variables
    bsvr
             (dp,ar(nbsvr),in)
                                   - basic element variables
C
С
    nnrsvr (int,sc,in)
                                   - number of nonlinear element variables
             (dp,ar(nnrsvr),in)
                                   - nonlinear element variables
С
                                  - number of plasticity element variables
C
    npsvr
             (int,sc,in)
С
             (dp,ar(npsvr),in)
                                   - plasticity element variables
    psvr
С
    ncsvr
            (int,sc,in)
                                   - number of creep element variables
             (dp,ar(ncsvr),in)
                                   - creep element variables
C
    csvr
С
    nusvr
             (int,sc,in)
                                   - number of user-supplied element variables
             (dp,ar(nusvr),in)
С
    usvr
                                  - user-supplied element variables
    nnode (int,sc,in)
                                   - number of nodes
С
С
    nodes (int,ar(nnode),in)
                                   - node numbers
             (dp,ar(6,nnode),in) - nodal coordinates and rotations (virgin)
С
    XYZ
             (dp,sc,in)
                                   - element volume (or area if 2-d)
С
    vol
С
    leng
             (dp,sc,in)
                                   - element length (beams, spars, etc)
                                   - current time
    time
             (dp.sc.in)
C
                                   - current sub step time increment
С
    timinc (dp,sc,in)
С
    nutot (int,sc,in)
                                   - length of dof solution vector utot
C
    utot
             (dp,ar(nutot),in)
                                   - solution vector
    maxdat (int,sc,in)
С
                                   - size of user output array (3 x nnode)
C
                                      actually, = ielc(nmnmup)
C
c output arguments:
С
  variable (typ,siz,intent)
                                   description
    numdat (int,sc,out) - number of user output items in array udbdudbdat (dp,ar(maxdat),out) - user output items to be placed at the end
С
                                   - number of user output items in array udbdat
С
                                      of the nmisc record
C
```

6.2.23. Subroutine eldwrtL (Writing Element Data to a File)

```
*deck,eldwrtL
     subroutine eldwrtL (ielem,edtype,lcerstL,edindxL,nval,value)
c *** primary function:
                          output element data to result file.
c *** Notice - This file contains ANSYS Confidential information ***
C
      input arguments:
                                     - element number
        ielem (int.sc.in)
C
C
        edtype
                  (int,sc,in)
                                     - element data type (see elparm)
        lcerstL (LONG,sc,inout) - pointer to results file position
C
        edindxL (LONG,ar(NUMELEDATASETS),inout) - index to results file data
C
                                    - the total number of values
C
        nval
                   (int.sc.in)
С
                                         if edtype = EDEMS,
                                         this should -always- be ielc(nmsmis),
C
                                         unless there is a variable number, as
С
                                         in the layered shell elements.
         value
                   (dp,ar(nval),in)
                                      - output values (real)
```

6.2.24. Subroutine eldwrnL (Writing Element Nonsummable Miscellaneous Data to the Results File)

```
С
      input arguments:
                                - element number
С
        elem (int,sc,in)
        ielc
               (int,ar(IELCSZ),in) - element characteristic vector
С
С
                                        defined in elccmt
        lcerstL (LONG,sc,inout) - pointer to results file position
С
        edindxL (LONG, ar(NUMELEDATASETS), inout) - index to results file data
C
                                  - size of what the user wants to add
        nudb
                  (in,sc,inout)
        udbdat (dp,ar(*),in)
C
                                   - what the user wants to add
С
        nval
                  (int,sc,in)
                                    - the total number of values to
                                        be output(does not include nudb)
С
                                        this should -always- be ielc(NMNMIS),
C
С
                                        unless there is a variable number, as
С
                                        in the layered shell elements.
        value
                  (dp,ar(ndval),in) - output values
C
                               - dimension of value - must be no less than
С
        ndval
                  (int,sc,in)
С
                                         ielc(NMNMIS) + ielc(NMNMUP)
c *** mpg eldwrnL < el117,el126,el109,el53,el96,el97: write nmisc db
```

6.2.25. Subroutine trrot (Computing the Rotation Vector)

```
*deck,trrot
     subroutine trrot (tr,rot)
c *** primary function:
                         get the rotation vector from a transformation matrix
c *** Notice - This file contains ANSYS Confidential information ***
С
   input arguments:
С
             (dp,ar(3,3),in)
                                  - transformation matrix
C
С
  output arguments:
             (dp,ar(3),out)
С
                                  - rotation vector
     rot
```

6.2.26. Subroutine rottr (Computing the Transformation Matrix)

```
*deck,rottr
      subroutine rottr (rot,tr)
c primary function: compute transformation matrix from rotation vector *****
c *** Notice - This file contains ANSYS Confidential information ***
      ref(old): eqn. 20(transposed), rankin and brogan, jpvt, 108(1986)165-174.
С
     ref(new): eqn. (b.4), simo and vu-quoc, cmame, 58 (1986), 79-116
                  (removes singularities at pi and 2*pi)
С
С
C
  input arguments:
С
    variable (typ, siz, intent) description
             (dp,ar(4),in) - rotation parameter in radians
С
C
C
  output arguments:
С
    variable (typ,siz,intent)
                                description
              (dp,ar(3,3),out) - transformation matrix corresponding to rot
C
```

6.2.27. Subroutine xyzup3 (Updating an Element's 3-D Nodal Coordinates)

```
(int,sc,in)
                                  - number of nodes
C
     nnod
              (dp,ar(nr),in) - displacement vector
C
С
     nr
              (int,sc,in)
                                  - size of the u vector
              (dp,ar(nx,nnod),in) - coordinates to be updated
С
     XYZ
С
              (int,sc,in)
                                   - row size of xy
C
c output arguments:
               (dp,ar(3,nnod),out) - updated coordinates
```

6.2.28. Subroutine updrot (Updating the Rotation Pseudovector)

```
*deck,updrot
      subroutine updrot (v2,w1)
c primary function: update the rotation pseudovector for 3-d large rotations *****
c *** Notice - This file contains ANSYS Confidential information ***
C
      The updating of the pseudovector uses the mathematics of quarternions
C
      (ref: eqn. a5 of J. H. Argyris, CMAME, 32(1982)85-155). The
      pseudovector uses the nomalization proposed by Rankin and Brogan (ref:
С
С
      eqn. 15, JPVT, 108(1986)165-174).
      CMAME = Computer Methods in Applied Mechanics and Engineering
C
      JPVT = Journal of Presssure Vessel Technology (ASME)
C
C
   variable descriptions:
С
    input:
C
С
     v2
             - rotation increment
С
     w1
             - previous rotation pseudovector
C
    output:
С
             - updated pseudovector
C
        v1 = cos(v1/2) + 1/2*w1,
                                  w1 = 2*sin(v1/2)*e1
С
         v2 = cos(v2/2) + 1/2*w2,
                                   w2 = 2*sin(v2/2)*e2
         v21 = v2*v1 = cos(v21/2) + 1/2*w21 (quarternion multiplication)
C
         w1 =: v21 (w1 is updated)
```

6.2.29. Subroutine tmpget (Defining Current Temperature Loads)

```
*deck,tmpget
     subroutine tmpget (iel,ielc,nnod,nodes,ref,ndat0,begdat,dat,
     x enddat,tlvf)
     primary function: define the current temperature loads
c *** Notice - This file contains ANSYS Confidential information ***
 typ=int,dp,log,chr,dcp siz=sc,ar(n),func intent=in,out,inout
C
C
  input arguments:
С
  variable (typ,siz,intent) description
C
С
     iel
            (int,sc,in)
                               - element number
            (int,ar(IELCSZ),in) - array of element type characteristics
С
     ielc
     nnod (int,sc,in) - number of nodes in the nodes array
C
     nodes (int,ar(nnod),in) - list of nodes
C
            (dp,sc,in)
                               - reference temperature
С
     ref
     ndat
            (int,sc,in)
                               - number of data items to get
С
     begdat (dp,ar(ndat),in) - data at the beginning of this load step
С
C
  output arguments:
C
           (dp,ar(ndat),out) - data at this time point
C
      enddat (dp,ar(ndat),out)
                               - data at end of this load step
C
     tlvf (int,sc,out)
                                - thermal load vector flag
С
C
                                  Should the thermal load vector be computed
C
                                  = 0 - no, temperatures match tref
                                  = 1 - yes, temperatures do not match tref
C
С
                   Note, that even if tlvf = 0, temperatures may be used to
                   compute temperature-dependent material properties.
С
```

С

6.2.30. Subroutine prsget (Defining Current Pressure Loads)

```
*deck,prsget
     subroutine prsget (iel,ielc,nfac,ndat,begdat,dat,enddat,iexist)
     primary function: define the current pressure loads
     See also: PrsRTGet
C
c *** Notice - This file contains ANSYS Confidential information ***
     input arguments:
С
С
        iel
                 (int,sc,in)
                                   - element number
        ielc (int,ar(IELCSZ),in) - array of element type characteristics
C
                                   - number of pressure faces
С
               (int,sc,in)
С
        ndat
                 (int,sc,in)
                                  - number of pressure values
        begdat (dp,ar(ndat),in) - pressure at the beginning of load step
C
С
     output arguments:
С
        dat
                 (dp,ar(ndat),out) - pressures at this iteration
        enddat
                  (dp,ar(ndat),out) - pressure at end of this load step
С
                                    - flag if pressure exist
        iexist
C
                 (int,sc,out)
                                      = 0 - no pressure
C
                                      = 1 - yes pressure
C
```

6.2.31. Subroutine cnyget (Defining Current Convection Loads)

```
*deck.cnvget.
     subroutine cnvget (iel,ielc,nr,u,nfac,ndat,beghc,begtb,
    x hc,tb,endhc,endtb,iexist)
     primary function: define the current convection loads
c *** Notice - This file contains ANSYS Confidential information ***
С
      input arguments:
C
        iel
                (int,sc,in)

    element number

С
        ielc (int,ar(IELCSZ),in) - array of element type characteristics
        nr
                (int,sc,in) - dimension of u (temperature) vector
С
                 (dp,ar(nr),in) - most current temperatures
C
        11
        nfac
                 (int,sc,in)
                                 - number of convection faces
        ndat
С
                 (int,sc,in)
                                  - number of convection values
        beghc
                 (dp,ar(ndat),in) - hcoef at the beginning of load step
С
        begtb
                 (dp,ar(ndat),in) - tbulk at the beginning of load step
С
С
     output arguments:
                 (dp,ar(ndat),out) - hcoef at this substep
С
        hc
                  (dp,ar(ndat),out) - tbulk at this substep
C
        t.b
С
        endhc
                 (dp,ar(ndat),in)
                                   - hcoef at the end of this load step
        endtb
                 (dp,ar(ndat),in) - tbulk at the end of this load step
С
        iexist (int,sc,out)
                                    - flag if convection exist
C
С
                                      = 0 - no convection
С
                                      = 1 - constant convection (with time)
С
                                            does not require new element matrix
С
                                      = 2 - changing convection (with time)
                                             or deleted convection
C
                                            requires new element matrix
```

6.2.32. Subroutine hgnget (Defining Current Heat Generation Loads)

*deck,hgnget

```
subroutine hgnget (iel,ielc,nnod,nodes,ndat,begdat,dat,enddat,
С
      primary function: define the current heat generation loads
С
c *** Notice - This file contains ANSYS Confidential information ***
      typ=int,dp,log,chr,dcp siz=sc,ar(n),func intent=in,out,inout
C
C
     input arguments:
C
     variable (typ,siz,intent) description
С
        iel
                                    - element number
С
                (int,sc,in)
        ielc (int,ar(IELCSZ),in) - array of element type characteristics
C
       nnod (int,sc,in) - number of nodes in the nodes array
С
        nodes (int,ar(nnod),in) - list of nodes
С
        ndat (int,sc,in) - number of data items to get
begdat (dp,ar(ndat),in) - data at the beginning of this load step
C
С
С
     output arguments:
C
              (dp,ar(ndat),out) - data at this time point
        enddat (dp,ar(ndat),out) - data at end of this load step
С
         iexist (int,sc,out)
                                    - flag if heat generation exist
С
С
                                      = 0 - no heat generation
                                      = 1 - yes heat generation
C
```

6.2.33. Subroutine prinst (Computing principal stress and stress intensity)

```
*deck,prinst
     subroutine prinst (s)
    primary function: computes principal stresses and stress intensity
    secondary functions: none
c *** Notice - This file contains ANSYS Confidential information ***
  input arguments:
    variable (typ,siz,intent)
                                description
C
              (dp,ar(11),inout) - stress vector
С
С
                  s(1)=sx
C
                  s(2)=sy
С
                  s(3)=sz
С
                  s(4)=sigxy
                  s(5)=siqvz
C
                   s(6)=sigzx
C
С
   output arguments:
    variable (typ,siz,intent)
                                 description
C
С
      s (dp,ar(11),inout) - stress vector
С
                  s(7)=sig1
                  s(8) = sig2
C
                   s(9) = sig3
С
С
                  s(10)=s.i.
                  s(11)=sige
C
c ******* note: all changes to this routine must be made in
С
                 post1 (paprst)
```

6.3. Subroutines for Modifying and Monitoring Existing Elements

The next few pages describe user subroutines for modifying or monitoring existing ANSYS elements. These subroutines enable you to perform tasks including:

- Computing load vectors for frequency domain logic
- · Storing element output that users supply

- Modifying the orientation of material properties and stresses
- Modifying the orientation of material properties and stresses of layers within an element
- Performing a user-defined operation on a parameter for the COMBIN7 and COMBIN37 elements
- Providing a user-defined initial thickness for SHELL181, SHELL208, SHELL209, and SHELL281.
- Providing a user-defined initial stress for PLANE42, SOLID45, PLANE82, SOLID92, SOLID95, LINK180, SHELL181, PLANE182, PLANE183, SOLID185, SOLID186, SOLID187, SOLSH190, BEAM188, BEAM189, SHELL208, SHELL209, REINF264, REINF265, SHELL281, and SOLID285.
- Modifying SURF151 and SURF152 film coefficients and bulk temperatures based on information from FLUID116

6.3.1. Subroutine userou (Storing User-Supplied Element Output)

```
*deck,userou
                                 USERSDISTRIB
     subroutine userou (elem,iout,nbsvr,bsvr,nnrsvr,nrsvr,npsvr,psvr,
     x ncsvr,csvr,nusvr,usvr,nnode,nodes,xyz,vol,leng,time,
    x timinc,nutot,utot,maxdat,numdat,udbdat)
С
c *** primary function:
                          store user supplied element output
                           in nmisc record
C
C
С
         in order to activate this user programmable feature,
         the user must enter the usrcal command.
C
С
С
          *** copyright(c) 2008 SAS IP, Inc. All rights reserved.
         *** ansys, inc.
c *** Notice - This file contains ANSYS Confidential information ***
С
             this routine is called by almost every element
С
             the data is stored on the nmisc record.
C
             warning: other data may be stored between the
C
С
                       documented data and this data.
             in order to see the actual information on the nmisc
C
              record, insert the command:
                  dblist,elp,elnum1,elnum2,elinc,11
C
С
                       where elnum1 = the first element
С
                             elnum2 = the last element
                             elinc = the element increment number
C
С
              after a set command in post1.
С
  input arguments:
C
    variable (typ,siz,intent)
С
                                  description
                                  - element number
С
    elem (int,sc,in)
            (int,sc,in)
                                 - output unit number
С
    iout
    nbsvr (int,sc,in)
                                 - number of basic element variables
С
            (dp,ar(nbsvr),in)
С
    bsvr
                                  - basic element variables
С
    nnrsvr (int,sc,in)
                                  - number of nonlinear element variables
            (dp,ar(nnrsvr),in)
                                  - nonlinear element variables
С
    nrsvr
                                 - number of plasticity element variables
    npsvr
            (int,sc,in)
С
            (dp,ar(npsvr),in) - plasticity element variables
С
    psvr
С
    ncsvr (int,sc,in)
                                  - number of creep element variables
            (dp,ar(ncsvr),in)
                                  - creep element variables
    csvr
С
С
    nusvr
            (int,sc,in)
                                  - number of user-supplied element variables
С
                                         (= nstv on the nsvr command)
C
    usvr
            (dp,ar(nusvr),in) - user-supplied element variables
    nnode (int,sc,in)
                                - number of nodes
С
    nodes (int,ar(nnode),in)
                                  - node numbers
С
С
            (dp,ar(6,nnode),in)
                                  - nodal coordinates and rotations (virgin)
    XYZ
    vol
            (dp,sc,in)
                                  - element volume (or area if 2-d)
С
                                 - element length (beams, spars, etc)
    leng
            (dp,sc,in)
C
С
    time
            (dp,sc,in)
                                 - current time
С
    timinc (dp,sc,in)
                                  - current sub step time increment
    nutot (int,sc,in)
                                  - length of dof solution vector utot
C
С
    utot
            (dp,ar(nutot),in)
                                  - solution vector
```

```
maxdat (int,sc,in)
C

    size of user output array (3 x nnode)

C
С
  output arguments:
С
    variable (typ,siz,intent)
                                    description
С
    numdat (int,sc,out)
                                    - number of user output items in array udbdat
С
                                        (maximum size of numdat is ielc(NMNMUP)
                                        which is usually three times the number
C
C
                                        of nodes.
    udbdat (dp,ar(maxdat),out)
                                    - user output items to be placed at the end
C
С
                                      of the nmisc record
```

6.3.2. Subroutine useran (Modifying Orientation of Material Properties)

```
*deck.useran
                                 USERSDISTRIB
     subroutine useran (vn.vref.elem.thick.xyzctr.bsangl)
     user written routine to modify orientation of material properties
     applicable to: shell43,63,91,93,99, solid46,64,191
C
        accessed by keyopt
C
С
         *** copyright(c) 2006 SAS IP, Inc. All rights reserved.
         *** ansys, inc.
c *** Notice - This file contains ANSYS Confidential information ***
C
 **** warning *** do not change any arguments other than bsangl.
С
                  if you do, your results are probably wrong.
C
С
  input(do not change)---
C
            = vector normal to element
C
С
              = unit vector orienting element, essentially edge i-j
C
              = element number
              = total thickness of element at this point (see note below)
С
     xyzctr = location of element centroid or integration point
C
С
С
     bsangl = output from this subroutine. it represents the angle(s)
C
              between vref and the desired orientation. it may have
С
              the default orientation coming in to useran.
                This will be combined with the angles derived from
С
                the ESYS command.
С
           use 1 angle for 2-d elements and shells
С
           use 3 angles for 3-d solids
```

6.3.3. Subroutine userrc (Performing User Operations on COMBIN7 and COMBIN37 Parameters)

```
*deck,userrc
                                  USERSDISTRIB
     subroutine userrc (elem,ireal,type,nusvr,usvr,parm,parmld,
    x c1,c2,c3,c4,fcon)
     primary function: user operation on parameter for combin7 and combin37
C
       accessed with keyopt(9) = 1
С
          *** copyright(c) 2008 SAS IP, Inc. All rights reserved.
С
          *** ansys, inc.
C
c *** Notice - This file contains ANSYS Confidential information ***
C
  input arguments:
C
С
    variable (typ,siz,intent)
                                  description
С
    elem
            (int,sc,in)
                                  - element number
    ireal
             (int,sc,in)
                                  - element real constant number
C
            (int,sc,in)
                                  - element type number
С
    type
    nusvr (int,sc,in)
                                  - number of user-supplied element variables
C
С
                                     (input with the NSVR command)
            (dp,ar(nusvr),inout) - user-supplied element variables
    usvr
```

```
(dp,sc,in)
                                   - current value of the paramater
C
     parm
С
    parmld
             (dp,sc,in)
                                   - value of the parameter at previous time ste
С
     c1
             (dp,sc,in)
                                   - real constant cl
    c2
С
             (dp,sc,in)
                                   - real constant c2
     с3
С
             (dp,sc,in)
                                   - real constant c3
C
     c4
             (dp,sc,in)
                                   - real constant c4
C
  output arguments:
   variable (typ,siz,intent)
                                    description
С
С
            (dp,ar(nusvr),inout) - user-supplied element variables
                                       may be sent .rst file with usereo
С
    fcon
                                    - result of calculation
C
             (dp,sc,out)
С
С
      either c1 or c3 must be nonzero for this logic to be accessed,
```

6.3.4. Subroutine UEIMatx (Accessing Element Matrices and Load Vectors)

```
*deck.UElMatx
                                   USERSDISTRIB
     subroutine UElMatx (elem,nr,ls,zs,zsc,uelm,ielc,nodes,
                         ElDofEachNode,elmdat,xyzang,lenu)
c primary function:
                      User routine to access element matrices and load vectors.
                      Needs to have USRCAL, UELMATX to be accessed.
                      Called after the call to the element routine and
C
С
                       before the solver.
                       May be used to monitor and/or modify the element matrices
C
С
                                                         and load vectors.
          *** copyright(c) 2008 SAS IP, Inc. All rights reserved.
С
С
          *** ansys, inc.
     typ=int,dp,log,chr,dcp siz=sc,ar(n)
                                             intent=in,out,inout
C
  input arguments:
C
C
     variable (typ,siz,intent)
                                      description
     elem
              (int,sc,in)
                                   - User element number
C
              (int,sc,in)
                                   - number of rows in element matrix
C
     nr
С
     ls
              (int,ar(nr),in)
                                   - Dof Index vector for this element matrix
              (dp,ar(nr,nr,4),inout)- K,M,C,SS matrices for this element
С
     zs
              (dp,ar(nr,2),inout) - Element load vector and N-R correction vec
C
     ZSC
                                   - Nodal displacements for this element
С
     uelm
              (dp,ar(nr,5),in)
С
     ielc
              (int,ar(*),in)
                                   - Element type characteristics
     nodes
              (int,ar(*),in)
                                   - Nodes for this element
C
С
     ElDofEachNode (int,ar(nr),in) - list of dofs for each node in Global
                                - Element data for this element
C
     elmdat (int,ar(10),in)
C
     xyzang
             (dp,ar(6,*),in)
                                   - X,Y,Z,THXY,THYZ,THZX for each element node
     lenu
              (int,sc,in)
                                   - Length of global displacement vector
С
C
  output arguments:
              (dp,ar(nr,nr,4),inout) - K,M,C,SS matrices for this element
С
              (dp,ar(nr,2),inout) - Element load vector and N-R correction vec
C
      WARNING: any CHANGES to these (or any other) arguments will have a direc
С
С
      impact on the solution, possibly giving meaningless results. The normal
      usage of this routine is simply monitor what is happening.
```

6.3.5. Subroutine UTHICK (Getting User-defined Initial Thickness)

```
*** ansys, inc.
C
С
С
     input arguments
     =========
С
                  (type,sz,i/o) description
С
                  (int,sc,i) element number
(int sc,i) element TYPE (181 etc.)
С
      elemId
                  (int,sc,i)
C
     elemType
C
     matId
                  (int,sc,i)
                              material number
     realId
                  (int,sc,i)
                              real constant set number
C
     numDomIntPts
                  (int,sc,i)
С
                                number of integration points
      curCoords
                  (dp,ar(3,numDomIntPts),i)
С
                                current coordinates
С
С
С
     output arguments
     ==========
C
С
                  (dp,ar(3,numDomIntPts),o)
C
                                thickness at the integration points
C
c --- parameters
```

6.3.6. Subroutine UsrFictive (Providing User-defined Fictive Temperature Relationship)

```
*deck, UsrFictive
                                      USERSDISTRIB
     subroutine UsrFictive (tref,toffst,tem,ftl, veinpt, ftc)
c *** primary function: allow users to write their own
                          fictive temperature relationship
C
                       this logic is accessed with c5 = 11 on the tb, evisc table
c *** secondary function:
                             demonstrate the use of a user-written
                             fictive temperature relationship
С
                       this routine could also be used to modify the viscoelastic
С
                       data during solution, i.e., to make the viscoelastic
С
                       coefficients themselves time-dependent.
 *** notice- this routine contains ansys,inc. confidential information ***
C
C
          *** copyright(c) 2008 SAS IP, Inc. All rights reserved.
C
          *** ansys, inc.
C
C
   input arguments:
C
     variable (type,sze,intent)
                                   description
C
С
              (dp,sc,in)
                                   - reference temperature
     toffst
С
              (dp,sc,in)
                                  - temperature offset from absolute zero
               (dp,sc,in)
                                   - temperature at the end of this substep
С
     tem
                                   - previous fictive temperature
С
     ftl
               (dp,sc,in)
     veinpt (dp,ar(95),inout) - table from tb,evisc
С
C
  output arguments:
С
     variable (type,sze,intent)
                                   description
С
С
      veinpt (dp,ar(95),inout)
                                   - table from tb, evisc
      ftc
               (dp,sc,in)
                                   - fictive temperature
C
```

6.3.7. Subroutine Us_Surf_Str (Captures surface stresses)

```
input arguments:
C
С
     variable (typ,siz,intent)
                                    description
                               element number
С
     elem
             (int,sc,in)
С
     face
             (int,sc,in)
                                face number
C
     area
             (dp, sc,in)
                                face area (or length)
             (dp, sc,in)
                                face temperature
     temp
C
                                face pressure
    pressure(dp, sc,in)
             (dp,ar(4),in)
C
     ep
                               face strains
С
     stress (dp,ar(11),in)
                               face stresses
c output arguments: none
```

6.3.8. Subroutine uflex (Computes flexibility factors for PIPE288 and PIPE289)

```
*deck.uflex
                                 USERSDISTRIB
      subroutine uflex (elemId,pressInt,pressExt,ex,pois, sflex,twten)
c *** primary function:
                          to (re)compute the flexibility factors
                            for pipe288 and pipe289
С
C
                          this is accessed by inputting the axial flexibility factor
C
С
 *** secondary functions: none
c *** Notice - This file contains ANSYS Confidential information ***
C
С
          *** copyright(c) 2008 SAS IP, Inc. All rights reserved.
          *** ansys, inc.
C
С
      typ=int,dp,log,chr,dcp siz=sc,ar(n)
                                               intent=in,out,inout
C
C
С
      input arguments:
               (int,sc,in) - element number
C
   elemId
            (dp,ar(2),in) - internal pressures at end nodes
   pressInt
C
   pressExt (dp,ar(2),in) - external pressures at end nodes
                                       Pressures include hydrostatatic but
C
C
                                        not hydrodynamic effects.
                (dp,sc,in) - Young's Modulus
С
   ex
                (dp,sc,in) - Poisson's ratio
C
   pois
С
          (dp,ar(6),inout) - input flexibility factors
C
                               (axial, bending about element z,
                                bending about element y, twist, y shear, z shear)
C
             (dp,sc,inout) - twist-tension factor
С
С
С
      output arguments:
C
   sflex (dp,ar(6),inout) - output flexibility factors
                                (axial, bending about element z,
C
С
                                bending about element y, twist, y shear, z shear)
   twten
             (dp,sc,inout) - twist-tension factor
С
C
                                 USERSDISTRIB
*deck,uflex
      subroutine uflex (elemId,pressInt,pressExt,ex,pois, sflex,twten)
c *** primary function:
                          to (re)compute the flexibility factors
С
                            for pipe288 and pipe289
                          this is accessed by inputting the axial flexibility factor
C
                            as -10.
C
 *** secondary functions: none
С
c *** Notice - This file contains ANSYS Confidential information ***
С
          *** copyright(c) 2008 SAS IP, Inc. All rights reserved.
С
          *** ansys, inc.
C
С
      typ=int,dp,log,chr,dcp
                                               intent=in.out.inout
C
                               siz=sc,ar(n)
С
      input arguments:
               (int,sc,in) - element number
  elemId
  pressInt (dp,ar(2),in) - internal pressures at end nodes
```

```
pressExt (dp,ar(2),in) - external pressures at end nodes
                                       Pressures include hydrostatatic but
С
                                        not hydrodynamic effects.
                (dp,sc,in) - Young's Modulus
С
  ex
                (dp,sc,in) - Poisson's ratio
С
  pois
   sflex (dp,ar(6),inout) - input flexibility factors
С
                               (axial, bending about element z,
C
                                bending about element y, twist, y shear, z shear)
             (dp,sc,inout) - twist-tension factor
C
  t.wt.en
С
      output arguments:
С
  sflex (dp,ar(6),inout) - output flexibility factors
C
                               (axial, bending about element z,
C
С
                                bending about element y, twist, y shear, z shear)
             (dp,sc,inout) - twist-tension factor
C
  t.wt.en
```

6.3.9. Subroutine usflex (Computes the flexibility factor for PIPE16, PIPE17, PIPE18, and PIPE60)

```
*deck.usflex
                                  USERSDISTRIB
     subroutine usflex (etype,elem,rvrm,kff,prs,ex, flexi,flexo)
c *** primary function: to (re)compute the flexibility factor
                           for pipe16, pipe17, pipe18, and pipe60
C
C
                          this is accessed by inputting the flexibility factor
                           as any negative number.
C
c *** secondary functions: none
C
c *** Notice - This file contains ANSYS Confidential information ***
C
          *** copyright(c) 2008 SAS IP, Inc. All rights reserved.
С
          *** ansys, inc.
С
C
С
     typ=int,dp,log,chr,dcp siz=sc,ar(n) intent=in,out,inout
C
  input arguments:
C
     variable (typ,siz,intent)
                                  description
С
                                - pipe element type (16, 17, 18 or 60)
С
     etype
              (int,sc,in)
                                - element number
C
     elem
              (int,sc,in)
               (dp,ar(*),in)
                                - real constants
C
                                - keyopt for flexibility factor
C
     kff
              (int,sc,in)
С
                                    (not used for pipe16 or pipe17)
С
     prs
               (dp,ar(5),in)
                                - pressures
                                - young's Modulus
               (dp,sc,in)
C
     ex
С
     flexi
               (dp,sc,inout)
                                - effective in-plane flexibility factor
С
     flexo
              (dp,sc,inout)
                                - effective out-of-plane flexibility factor
                                    (not used for pipe16 or pipe17)
С
С
С
  output arguments:
     variable (typ, siz, intent) description
C
      flexi
               (dp,sc,inout)
                                - effective in-plane flexibility factor
C
     flexo
              (dp,sc,inout)
                                - effective out-of-plane flexibility factor
C
                                     (not used for pipe16 or pipe17)
С
```

6.3.10. Subroutine UsrShift (Calculates pseudotime time increment)

```
c input arguments:
С
    timinc (dp,sc,in)
                                - time increment
С
     temp (dp,sc,in)
                                - current temperature, t_n+1
     dtemp (dp,sc,in)
toffst (dp,sc,in)
                                - temperature increment, t_n+1 - t_n
С
                                - temperature offset to absolute zero
С
                                  (specified by TOFFST command)
С
     propsh (dp,ar,in)
                                - Constants for shift function
C
                                  (User's input using TB, SHIFT, , , , USER)
                                - number of user defined constants
     nTerms (int,ar,in)
С
С
                                  (specified in TB, SHIFT, , , nTerms, USER)
c output arguments:
           (dp,sc,out)
                                - pseudotime increment
C
   dxi
    dxihalf (dp,sc,out)
                                 - pseudotime increment over the upper half span
C
```

6.4. Subroutines for Customizing Material Behavior

This section describes the following subroutines that you can use to modify or monitor material behavior:

- 6.4.1. Subroutine UserMat (Creating Your Own Material Model)
- 6.4.2. Subroutine UserHyper (Writing Your Own Hyperelasticity Laws)
- 6.4.3. Subroutines UserCreep and UserCr (Defining Viscoplastic/Creep Material Behavior)
- 6.4.4. Subroutine user_tbelastic (Defining Material Linear Elastic Properties)
- 6.4.5. Subroutine UserPL (Writing Your Own Plasticity Laws)
- 6.4.6. Subroutine usersw (Writing Your Own Swelling Laws)
- 6.4.7. Subroutine userck (Checking User-Defined Material Data)
- 6.4.8. Subroutine UserVisLaw (Defining Viscosity Laws)
- 6.4.9. Subroutine userfric (Writing Your Own Friction Laws)
- 6.4.10. Supporting Function egen

Using the "_MATL" String

If you write a material-behavior subroutine using the MPDATA, MPDELE, TB, or TBDELE command, be aware that when the string "_MATL" appears in the MAT field of the command, the command interprets the string to mean the currently active material (as defined via the MAT,MAT command).

The "_MATL" string is used in conjunction with the library (LIB) option of the **MPREAD** and **MPWRITE** commands. When you issue **MPWRITE** with the LIB option, the command inserts "_MATL" in lieu of the specified material number as these commands are written to the material library file. When ANSYS reads a material library file written in this format, it interprets "_MATL" to mean the currently active material. Do not use the "_MATL" string outside the scope of the **MPREAD** command.

6.4.1. Subroutine UserMat (Creating Your Own Material Model)

The UserMat subroutine allows you to write your own material constitutive equations within a general material framework using current-technology elements.

UserMat is a tool for advanced users. Expertise in material constitutive modeling and software programming is necessary. Developing a custom material constitutive model requires validation and testing. ANSYS strongly recommends testing both single elements and multiple elements with various loading conditions to ensure correct results. UserMat supports shared memory and distributed parallel processing; however, you are responsible for ensuring that your code can use parallel processing.

The following UserMat topics are available:

- 6.4.1.1. UserMat Overview
- 6.4.1.2. Stress, Strain, and Material Jacobian Matrix
- 6.4.1.3. The UserMat API

6.4.1.4. UserMat Variables

6.4.1.5. Table (TB) Commands for UserMat

6.4.1.6. Material Constitutive Integration with UserMat

6.4.1.7. UserMat Restrictions

6.4.1.8. Accessing Material and Element Data for UserMat

6.4.1.9. Utility Functions for UserMat

For a UserMat subroutine example, see Appendix C (p. 313).

6.4.1.1. UserMat Overview

The UserMat subroutine defines the material stress-strain relationship of a material and applies to any ANSYS analysis procedure involving mechanical behavior. The subroutine supports current-technology elements only and does not apply to legacy elements.

The subroutine is called at every material integration point of the elements during the solution phase. ANSYS passes in stresses, strains, and state variable values at the beginning of the time increment and strain increment at the current increment, then updates the stresses and state variables to the appropriate values at the end of the time increment.

Input values and the number of state variables (if used) for UserMat are specified via the **TB** command. For more information, see *Table (TB) Commands for UserMat* (p. 161).

Further Reading

ANSYS recommends the following resource to help you understand the ANSYS implementation of user-defined materials:

Hughes, Thomas J.R. and James Winget. "Finite Rotation Effects in Numerical Integration of Rate Constitutive Equations Arising in Large-Deformation Analysis." [International Journal for Numerical Methods in Engineering]. 15.9 (1980): 1413-1418.

6.4.1.2. Stress, Strain, and Material Jacobian Matrix

The stress measure (σ) used by the subroutine is the Cauchy stress (true stress), and the strain measure (ε) is the logarithmic strain (true strain). The strains and incremental strains passed into UserMat are the total mechanical strains from which the thermal strains (if they exist) are subtracted.

UserMat must also provide the material Jacobian matrix defined as $\frac{\partial \Delta \sigma_{ij}}{\partial \Delta \epsilon_{ij}}$. $\frac{\Delta \sigma_{ij}}{\partial \Delta \epsilon_{ij}}$ is the stress increment, and $\frac{\Delta \epsilon_{ij}}{\partial \Delta \epsilon_{ij}}$ is the strain increment.

UserMat is based on the current configuration for nonlinear geometry analysis (**NLGEOM**,ON). ANSYS uses a co-rotational approach to account for rigid body rotation. Because ANSYS already accounts for the strains passed into UserMat for the rigid body rotation, there is no need to apply additional rotation within UserMat.

Stress, strain, and the material Jacobian tensors are stored in a vector or matrix format.

The order of components for all tensors is as follows:

3-D stress state 11, 22, 33, 12, 23, 13 2-D plane stress/strain and axisymmetric stress states

11, 22, 33, 12

Beam element stress states

11, 13, 12

Link element stress state

11

The order of components for the material Jacobian matrix is as follows:

3-D stress state

1111	1122	1133	1112	1123	1113
2211	2222	2233	2212	2223	2213
3311	3322	3333	3312	3323	3313
1211	1222	1233	1212	1223	1213
2311	2322	2333	2312	2323	2313
1311	1322	1333	1312	1323	1313

2-D plane strain and axisymmetric stress states

1111	1122	1133	1112
2211	2222	2233	2212
3311	3322	3333	3312
1211	1222	1233	1212

2-D plane stress states

1111	1122	1112
2211	2222	2212
1211	1222	1212

Beam element stress states

1111	1113	1112
1311	1313	1312
1211	1213	1212

Link element stress state

1111

6.4.1.3. The UserMat API

Following is the interface for the UserMat subroutine:

```
nDirect, nShear, ncomp, nStatev, nProp,
    &
                         Time, dTime, Temp, dTemp,
                        stress, statev, dsdePl, sedEl, sedPl, epseq,
    &
                        Strain, dStrain, epsPl, prop, coords,
    &
                         tsstif, epsZZ,
                         var1, var2, var3, var4, var5,
    &
    &
                        var6, var7, var8)
INTEGER
                      matId, elemId,
   &
    &
                      kDomIntPt, kLayer, kSectPt,
                      ldstep, isubst, keycut,
   &
                      nDirect, nShear, ncomp, nStatev, nProp
    DOUBLE PRECISION
                               dTime,
                      Time,
                                         Temp,
                                                   dTemp,
                      sedEl,
                               sedPl,
                                                  epsZZ
                                        epseq,
    DOUBLE PRECISION
                      stress (ncomp ), statev
                                                  (nStatev),
                      dsdePl (ncomp,ncomp),
    &
                      Strain (ncomp ), dStrain (ncomp ),
                      epsPl (ncomp ), prop coords (3),
    &
                                                  (nProp ),
    &
                      defGrad_t(3,3),
                                        defGrad(3,3),
    &
                       tsstif (2)
```

6.4.1.4. UserMat Variables

The UserMat subroutine uses the following Input, Input/Output, and Output variables. Do not change them in the subroutine code.

	UserMat Input Arguments
matld	Integer variable containing the material ID number.
elemId	Integer variable containing the element number.
kDomIntPt	Integer variable containing the material integration point number.
kLayer	Integer variable containing the layer number.
kSectPt	Integer variable containing section point number.
ldstep	Integer variable containing load step number.
isubst	Integer variable containing substep number.
nDirect	Number of direct components of the stress or strain vector at material point.
nShear	Number of shear components of the stress or strain vector at material point (engineering components).
ncomp	Total number of the stress or strain components at material point (nDirect + nShear).
nstatev	Number of state variables, specified by the $NPTS$ value in the TB ,STATE command.
nProp	Number of material constants, specified by the $NPTS$ value in the TB , USER command.
Temp	Double-precision variable contains the current temperature.
dTemp	Double-precision variable contains the current temperature increment.
Time	Double-precision variable contains the total time at the beginning of the time increment.
dTime	Double-precision variable contains the current time increment.
Strain	Double-precision array contains the total strains at the beginning of the time increment. Array size is ncomp.

	Thermal strains (defined via MP ,ALPHA and temperature load), if any, are subtracted from the total strains; therefore, the strains passed to UserMat are the mechanical strains only.
	For large-deformation problems, (NLGEOM ,ON), the strain components are updated to account for rigid body rotation before they are passed to UserMat and are approximate the logarithmic strains.
dStrain	Double-precision array contains current strain increments. Array size is ncomp. As with the Strain array, this value contains the mechanical strain increments only. Thermal strain increments (if any) are subtracted from the total strains increments.
prop	Double-precision array contains the material constants defined via TB ,USER and TBDATA commands. Array size is nProp. Array prop contains the material constants at current temperature point.
coords	Double-precision array contains the current coordinates of the material integration points. Array size is 3.
defGrad_t	Double-precision matrix contains deformation gradient at the beginning of the time increment. The matrix size is 3×3 . The matrix components DefGrad_ $t_{(i,j)}$ are equivalent to deformation gradient F_{ij} at the beginning of the time increment.
defGrad	Double-precision matrix contains current deformation gradient. The matrix size is 3 x 3. The matrix components $DefGrad_{(i,j)}$ are equivalent to deformation gradient F_{ij} at the current time.

	UserMat Input/Output Arguments
stress	Double-precision array containing the stresses. Its size is defined by the ncomp input value. The stress measure is the "true" stress. It is passed as the values of stresses at the beginning of the time increment and must be updated to the values of stress at the end of the time increment.
	For finite-deformation problems, the stresses are rotated to account for rigid body motion before they are passed in, and thus only the co-rotational portion of stress integration is required in UserMat.
statev	Double-precision array containing the state variables. Its size is defined via the TB ,STATE command. It is passed as the values of state variables at the beginning of the time increment and must be updated to the values of the state variables at the end of the time increment.
	For finite-deformation problems, any vector or tensor type of state variables must be rotated to account for rigid body motion before they are used in any constitutive calculations; the rotation matrix (rotatM) is passed in for that purpose.
epspl	Double-precision array containing the plastic strains. The strain measure is the "true" strain. Its size is defined by the ncomp input value. It is passed as the values of the plastic strains at the beginning of the time increment and must be updated to the values of the plastic strains at the end of the time increment.
	For finite-deformation problems, the plastic strains have been rotated to account for rigid body motion before they are passed in.
sedEl	Elastic work. It is used for output purposes only and does not affect the solution.
sedPl	Plastic work. It is used for output purposes only and does not affect the solution.

UserMat Output Arguments			
	These values must be updated in the subroutine code.		
keycut	Integer variable as key for loading bisection/cut control:		
	0 - No bisect/cut (default) 1 - Bisect/cut		
	Set keycut = 1 when UserMat experiences convergence difficulty when solving constitutive equation integration. The bisect/cut factor is determined by ANSYS solution control.		
epsZZ	Strain component at an out-of-plane direction for the plane stress state. This value is required when the thickness change is taken into account in plane stress or shell elements.		
tsstif(2)	Transver shear stiffness:		
	Tsstif(1) - GXZ Tsstif(2) - GYZ		
dsdePl(ncomp,ncomp)	Double-precision array containing the material Jacobian matrix $\partial \Delta \sigma_{ij} / \partial \Delta \epsilon_{ij}$. Here, the values represent the stress/strain increments, respectively. The dsdePl(i,j) value denotes the change in the i-th stress component at the end of the time increment caused by an change of the j-th strain component.		
	By default, ANSYS assumes that the element stiffness matrix is symmetric; therefore, you must provide a <i>symmetric</i> material Jacobian matrix <i>even if it is unsymmetric</i> . If your material requires an unsymmetric material Jacobian matrix, an element key option (KEYOPT(5) = 1) is available for current-technology plane and solid elements to define the unsymmetric stiffness matrix.		

6.4.1.5. Table (TB) Commands for UserMat

When creating your own material model, first define the material by specifying input values for the UserMat subroutine (**TB**,USER). It is also necessary to specify the number of state variables used, if applicable (**TB**,STATE).

Following is more information about defining your material and specifying the number of state variables used. For detailed information about the **TB** command and arguments, see the *Command Reference*.

TB,USER Command

Issue the **TB** command using the following syntax:

TB, USER, MAT, NTEMPS, NPTS

where

MAT = User material ID number

NTEMPS = Number of temperature points.

NPTS = Number of material constants at a given temperature point.

The material properties at an intermediate temperature point are interpolated and passed to the UserMat subroutine.

Define temperatures and material constants via TBTEMP and TBDATA commands, respectively.

Example 6.1 Defining the Material for UserMat

```
tb,user,1,2,4
! Define material 1 as a user
! material with two temperatures
! and four data points at each
! temperature point.

tbtemp,1.0
! first temp.

tbdata,1,19e5, 0.3, 1e3,100,
! Four mat. constants for one temp.

tbtemp,2.0
! Second temp.

tbdata,1,21e5, 0.3, 2e3,100,
! Four mat. constants for two temps.
```

TB,STATE Command

If you intend to use state variables with the UserMat subroutine, it is necessary to first specify the number of state variables. Issue the **TB** command using the following syntax:

```
TB,STATE,MAT,,NPTS

where

MAT = User material ID number

NPTS = Number of state variables that you intend to use.
```

The command defines only the *number* of state variables and must always be associated with a user material ID. No temperatures or data are associated with the command.

By default, ANSYS initializes state variables to zero at the beginning of an analysis. Use the **TBDATA** command to initialize your own values for state variables.

Example 6.2 Defining the Number of State Variables for UserMat

```
tb,state,1,,8 ! Define material 1 with eight state variables tbdata,1,c1,c2,c3,c4,c5,c6,c7,c8 ! Initialize the eight state variables.
```

6.4.1.6. Material Constitutive Integration with UserMat

The UserMat subroutine supports current-technology elements with all key options. However, a different material constitutive integration is necessary for the various stress states, such as general 3-D, plane stress, and beam (with or without shear-stress components).

To ensure overall numerical stability, verify that the integration scheme implemented in the subroutine is stable. ANSYS always uses the full Newton-Raphson scheme for the global solution to achieve a better convergence rate. The material Jacobian matrix (dsdePl(i,j)) must be consistent with the material constitutive integration scheme for a better convergence rate of the overall Newton-Raphson scheme.

6.4.1.7. UserMat Restrictions

The following restrictions apply to the UserMat subroutine:

• The subroutine supports current-technology elements only and is not applicable to legacy elements.

For more information, see Legacy vs. Current Element Technologies in the *Element Reference*.

• The state variables (defined via the **TB**,STATE command) are supported only by full graphics in the POST1 postprocessor.

Because POST1 does not switch to full graphics automatically, you must issue a /GRA,FULL command to do so.

The subroutine is not intended for modeling incompressible elastic materials, such as hyperelastic materials

A special treatment such as a penalty approach may be needed to ensure incompressibility. In any case, if the material exhibits nearly incompressible behavior, use a finite tangent bulk modulus.

6.4.1.8. Accessing Material and Element Data for UserMat

Following is the interface for accessing the ANSYS material and element data:

```
get_ElmData (inquire, elemId, kIntg, nvect, vect)
c --- argument list
      CHARACTER*4
                            inquire
      INTEGER
                            elemId, kIntg, nvect
      DOUBLE PRECISION
                            vect(nvect)
         definition
C
           inquire
                           - query argument (string, see variables)
С
            elemId
                           - element number
           kIntg
                           - gauss intg. number
С
С
           nvect
                           - number of vector to be inquired
                           - vector to be inquired
            vect
```

get_ElmData Input Arguments	
inquire	String variable containing a query argument.
elemId	Integer variable containing the element number.
kDomIntPt	Integer variable containing the material integration point number.
nvect	Integer variable containing number of variable to be retrieved.
Vect(*)	Variable array containing the retrieved variables.

Valid inquire Argument Variables	
ESYS	Element coordinate system (ESYS).
ISIG	Initial stress.
TREF	Reference temperature.

6.4.1.9. Utility Functions for UserMat

The following functions are available for use with UserMat. ANSYS provides them for your convenience.

Utility Functions for UserMat	
vzero(a ,n)	Initializes array a to zero.
	The value n is the array dimension.
vmult((a,b,n,c))Multiplies vector a by constant c and outputs ($b = a * c$) as vector b .	

	The value n is the dimension for both arrays.
vmult1(a ,n,c)	Multiplies vector \mathbf{a} by constant \mathbf{c} and outputs the result as itself (that is, $\mathbf{a} = \mathbf{a} * \mathbf{c}$).
	The value n represents the array dimension.
	Multiplies two double-precision matrices and outputs the result as c (that is, $c = a * b$).
c , na, nb, nc, n1, n2, n3)	The value na is number of rows in matrix a , nb the number of rows in matrix b , and nc the number of rows in matrix c .
	The $n1$ value is the number of rows in matrix c to fill, and $n2$ the number of columns in matrix c to fill.
	The value $n3$ is the number of <i>columns</i> in matrix a and the number of <i>rows</i> in matrix b to work with. (The number of columns in a and rows in b is the same in order to generate the inner product correctly.)

6.4.2. Subroutine UserHyper (Writing Your Own Hyperelasticity Laws)

Use the subroutine UserHyper when you issue the **TB** command with the HYPER option, and with the command option TBOPT = 100.

```
*deck,UserHyper
                                     USERSDISTRIB
      subroutine UserHyper(
                           prophy, incomp, nprophy, invar,
     &
                           potential, pInvDer)
С
С
      *** Example of user hyperelastic routine
C
            This example uses Arruda hyperelasticity model
С
С
            which is the same ANSYS TB, BOYCE
C
      input arguments
С
      _____
С
                   (dp,ar(*),i)
                                    material property array
C
      prophy
С
      nprophy
                   (int,sc,i)
                                     # of material constants
                  dp,ar(3)
                                     invariants
С
      invar
С
С
      output arguments
      ==========
С
С
      incomp
                  (log,sc,i)
                                      fully incompressible or compressible
С
      potential
                   dp,sc
                                      value of potential
                                      der of potential wrt i1,i2,j
С
      pInvDer
                    dp,ar(10)
                                       1 - der of potential wrt il
С
С
                                       2 - der of potential wrt i2
                                       3 - der of potential wrt ilil
C
С
                                       4 - der of potential wrt ili2
                                       5 - der of potential wrt i2i2
С
                                       6 - der of potential wrt ilj
С
                                       7 - der of potential wrt i2j
                                       8 - der of potential wrt j
С
                                       9 - der of potential wrt jj
С
C*
С
c --- parameters
C
```

6.4.3. Subroutines UserCreep and UserCr (Defining Viscoplastic/Creep Material Behavior)

ANSYS provides two subroutines to allow you to specify your own creep equations to define the time-dependent viscoplastic/creep behavior of materials, UserCreep and UserCr.

Use the subroutine UserCreep when you issue the **TB** command with the CREEP option, and with the command option TBOPT = 100.

Use the subroutine UserCr when you issue the **TB** command with the CREEP option, with TBOPT = 0, and data constant C6 = 100.

Subroutine UserCreep is incorporated with an implicit time integration algorithm, while subroutine UserCr is incorporated with an explicit time integration algorithm. In general, the implicit time integration algorithm is more effective for long time periods. The explicit creep algorithm is more effective for short time periods such as transient analyses with very small time increments, or when the creep behavior of materials is not that significant. In the finite deformation analysis, you should interpret the strain variables in the subroutines as logarithmic strains, and you should interpret the stresses as true stresses.

6.4.3.1. Creep Subroutine UserCreep

Use the subroutine UserCreep to define viscoplastic/creep behavior of materials. The subroutine is applicable when you issue the **TB** command with the CREEP option, and with TBOPT = 100.

UserCreep supports shared memory and distributed parallel processing; however, you are responsible for ensuring that your code can use parallel processing.

The subroutine is called at all integration points of elements for which the material is defined by this command. ANSYS always uses implicit time integration for this creep option. You can use plasticity options (BISO, BKIN, MISO, NLISO, PLASTIC) to define the plastic behavior of materials. Creep and plastic strain will be calculated simultaneously when both creep and plasticity are defined for a material. Through this subroutine, you can specify a "uniaxial" creep law that will be generalized to the multi-axial state by the general time-dependent viscoplastic material formulation implemented in ANSYS. You can use and update internal state variables in the subroutine. The number of state variables has to be defined by **TB**,STATE.

Please see the **TB** command description for more information.

```
*deck,usercreep
                                 USERSDISTRIB
     SUBROUTINE usercreep (impflg, ldstep, isubst, matId , elemId,
                        kDInPt, kLayer, kSecPt, nstatv, nprop,
                         prop , time , dtime , temp , dtemp ,
    &
                         toffst, Ustatev, creqv , pres , seqv
                         delcr , dcrda)
    æ
*** primary function ***
С
          Define creep laws when creep table options are
C
          TB, CREEP with TBOPT=100.
С
С
          Demonstrate how to implement usercreep subroutine
С
С
           Creep equation is
              dotcreq := k0 * seqv ^n * creqv ^m * exp (-b/T)
С
С
              seqv is equivalent effective stress (Von-Mises stress)
С
              creqv is equivalent effective creep strain
C
С
                   is the temperature
              k0, m, n, b are materials constants,
С
C
          This model corresponds to primary creep function TBOPT = 1
```

```
С
                                                          gal 10.01.1998
С
С
C'
С
С
     input arguments
С
     =========
      impflg (in ,sc
C
                                          Explicit/implicit integration
                                          flag (currently not used)
C
               (in ,sc
                                          Current load step
С
      ldstep
                          ,i)
      isubst
               (in ,sc
                          ,i)
                                          Current sub step
С
      matId
               (in ,sc
                                         number of material index
C
                         ,i)
С
      elemId
               (in ,sc
                         ,i)
                                         Element number
С
      kDInPt
               (in ,sc
                         ,i)
                                         Material integration point
                         ,i)
      kLayer
               (in ,sc
                                         Layer number
C
С
      kSecPt
               (in ,sc
                         ,i)
                                         Section point
С
      nstatv
               (in ,sc
                          ,i)
                                         Number of state variables
      nprop
                         ,i)
                                         size of mat properties array
               (in ,sc
C
С
               (dp ,ar(*),i)
                                         mat properties array
C
      prop
                                         This array is passed all the creep
С
С
                                          constants defined by command
                                         TBDATA associated with TB, CREEP
C
С
                                          (do not use prop(13), as it is used
С
                                          elsewhere)
С
                                          at temperature temp.
С
      time
                                          Current time
                                         Current time increment
С
      dtime
                                          Current temperature
С
      temp
                                          Current temperature increment
C
      dtemp
С
      toffst
                (dp, sc,
                          i)
                                          temperature offset from absolute zero
С
      seqv
                (dp ,sc
                        , i)
                                         equivalent effective stress
С
      creqv
                (dp ,sc
                        , i)
                                          equivalent effective creep strain
                        , i)
                                         hydrostatic pressure stress, -(Sxx+Syy+Szz)/3
                (dp ,sc
C
      pres
С
С
     input output arguments
                                          input desc
                                                        / output desc
C
     ========
                                                          ========
                                          user defined iinternal state variables at
С
      Ustatev (dp,ar(*), i/o)
С
                                          time 't' / 't+dt'.
                                         This array will be passed in containing the
C
                                          values of these variables at start of the
C
                                          time increment. They must be updated in this
C
                                          subroutine to their values at the end of
С
С
                                          time increment, if any of these internal
                                          state variables are associated with the
C
С
                                         creep behavior.
С
C
     output arguments
С
      ============
С
      delcr
               (dp ,sc , o)
                                          incremental creep strain
C
      dcrda
               (dp,ar(*), o)
                                          output array
                                          dcrda(1) - derivitive of incremental creep
C
                                                    strain to effective stress
С
С
                                          dcrda(2) - derivitive of incremental creep
С
                                                    strain to creep strain
C
     local variables
С
С
     ==========
С
      c1,c2,c3,c4 (dp, sc, 1)
                                         temporary variables as creep constants
С
                  (dp ,sc, 1)
                                          temporary variable
С
                  (dp ,sc, 1)
                                          temporary variable
C
C*************************
C
С
 --- parameters
```

6.4.3.2. Creep Subroutine UserCr

In contrast to the UserCreep subroutine, for the UserCr subroutine, you need to specify the creep strain tensor. A detailed explanation of this subroutine follows.

```
*deck,usercr
                                   USERSDISTRIB
      subroutine usercr (elem,intpt,mat,ncomp,kfirst,kfsteq,e,posn,d,
     x proptb, timval, timinc, tem, dtem, toffst, fluen, dfluen, epel, epcrp,
     x statev, usvr, delcr)
c *** primary function:
                           allow users to write their own creep laws.
                             this logic is accessed with c6 = 100
С
 *** secondary function:
                              demonstrate the use of user-written creep laws
 *** Notice - This file contains ANSYS Confidential information ***
C
С
C
           *** copyright(c) 2006 SAS IP, Inc. All rights reserved.
C
C
          *** ansys, inc.
С
   input arguments:
C
      variable (type,sze,intent)
                                     description
C
C
С
               (int,sc,in)
                                    - element number (label)
      elem
C
      intpt
                (int,sc,in)
                                    - element integration point number
                                    - material reference number
               (int,sc,in)
C
      mat.
С
      ncomp
                (int,sc,in)
                                    - no. of stress/strain components (1,4 or 6)
С
                                      1 - x
С
                                      4 - x,y,z,xy
С
                                       6 - x, y, z, xy, yz, xz
С
      kfirst
               (int,sc,in)
                                     - 1 if first time through, 0 otherwise
                                       (useful for initializing state variables
С
                                      to a non-zero value)
C
                                     - 1 if first equilibrium iteration of a
С
      kfsteg
               (int,sc,in)
С
                                      substep, 0 otherwise
C
C
С
                (dp,sc,in)
                                    - elastic young'S MODULUS
      е
С
      posn
                (dp,sc,in)
                                    - poisson'S RATIO
      d
                (dp,ar(ncomp,ncomp),in)- elastic stress-strain matrix
C
С
      proptb
               (dp,ar(72),in)
                                    - material properties input on tb commands
C
                                 (do not use proptb(13), as it is used elsewhere)
               (dp,sc,in)
                                    - current time value
C
      timval
      timinc
                (dp,sc,in)
                                    - time increment over this substep
C
               (dp,sc,in)
                                    - temperature at the end of this substep
С
      tem
      dtem
               (dp,sc,in)
                                    - temperature increment over this substep
С
С
      toffst
                (dp,sc,in)
                                    - temperature offset from absolute zero
                                    - fluence at the end of this substep
      fluen
               (dp,sc,in)
C
                                    - fluence increment over this substep
С
      dfluen
                (dp,sc,in)
С
                (dp,ar(ncomp),inout)- elastic strain
C
      epel
C
      epcrp
                (dp,ar(ncomp),inout) - creep strain from previous substep
               (dp,ar(ncomp*5+2),inout) - state variables from previous
C
      statev
С
                                       (converged) substep. This variable is for
                                       explicit creep only and refers to a
С
С
                                      different internal variable than that
С
                                      defined by TB, stat which is used by
                                       implicit creep (usercreep) and usermat.
C
               (dp,ar(nuval,nintp),inout)- additional state variables from
C
      usvr
С
                                      previous equilibrium iteration (saved
С
                                       if the nsvr command is used)
С
C
С
   output arguments:
      variable (type,sze,intent)
                                     description
C
C
                (dp,ar(ncomp),inout) - elastic strain adjusted for creep increment
C
      epel
С
      epcrp
                (dp,ar(ncomp),inout) - updated creep strain
      statev
                (dp,ar(ncomp*5+2),inout) - updated state variables
C
               (dp,ar(nuval,nintp),inout) - updated additional state variables
      usvr
```

```
- equivalent creep strain increment (used
С
      delcr
               (dp,sc,out)
С
                                     for creep ratio calculation)
С
  fortran parameters (to be defined by the user):
С
С
     variable (type)
                                  description
С
     nuval
              (int)
                                   - number of additional state variables per
                                    integration point
C
C
     nintp
             (int)
                                   - maximum number of integration points of
                                     an element to be used with this routine
С
С
                                     (14 is the maximum)
         note: nuval x nintp = nstv(on nsvr command); cannot exceed 840!
С
C
  internal variables:
С
С
     variable (type,sze)
                           description
С
     con
               (dp,sc)

    temporary variable

С
     del
               (dp,ar(6))
                              - creep strain increments
                              - equivalent elastic strain (before creep)
С
     epet
              (dp,sc)
     ept
              (dp,ar(6))
                              - total strain
C
      eptot
              (dp,sc)
                              - equivalent total strain, elastic + creep
      sigen
              (dp,sc)
С

    equivalent stress (before creep)

      temabs
              (dp,sc)
                              - temperature on the absolute scale
С
```

6.4.4. Subroutine user_tbelastic (Defining Material Linear Elastic Properties)

Subroutine user_tbelastic provides an interface for defining your own material linear elastic properties (**TB**,ELASTIC). The following topics are available:

```
6.4.4.1. Overview of the user_tbelastic Subroutine
```

6.4.4.2. Data Types Supported by user thelastic

6.4.4.3. Table (TB) Command for user thelastic

6.4.4.4. User Interface for user thelastic

6.4.4.5. The user the lastic API

6.4.4.6. Usage Example for user_tbelastic

6.4.4.1. Overview of the user tbelastic Subroutine

The user_tbelastic subroutine can define material linear elastic properties as a function of temperature or coordinates. The subroutine is called at the material integration points of elements for which the definition of material elastic properties is a user option. The material properties defined are based on the material coordinate system of the elements.

Element Support

You can use the subroutine with most current-technology elements, including the following: LINK180, BEAM188, BEAM189, PLANE182, PLANE183, SOLID185, SOLID186, SOLID187, SOLSH190, SHELL181, SHELL208, SHELL209, REINF264, REINF265, SHELL281, SOLID272, SOLID273, SOLID285, PIPE288, PIPE289, and ELBOW290.

Material Model Support

You can use the subroutine with most ANSYS nonlinear material models, including the following: BISO, BKIN, MISO, KINH, NLSIO, PLASTIC, CHABOCHE, CAST IRON, CREEP, EDP (Drucker-Prager model), SMA (shape memory alloy), PRONY, and GURSON.

For more information about these material models, see the documentation for the **TB** command in the *Command Reference*.

6.4.4.2. Data Types Supported by user_tbelastic

The user_tbelastic subroutine can define the following types of material property data:

· Isotropic elasticity with two constants

Define the Young's modulus (EX) and Poisson's ratio (NUXY) material constants

· General orthotropic elasticity with nine constants

Define the normal modulus, shear modulus, and minor Poisson's ratios. The order is as follows: EX, EY, EZ, GXY, GXZ, GYZ, NUXY, NUXZ, NUYZ. All nine constants must be defined; no default values are assigned.

· Anisotropic elasticity with 21 constants

Define the material elastic stiffness matrix. The matrix consists of 21 constants, and all must be defined.

6.4.4.3. Table (TB) Command for user tbelastic

Issue a TB command using the following syntax to access the user_tbelastic subroutine interface:

The ELASTIC argument accesses the elastic material property data table. (For more information, see the documentation for the **TB** command's ELASTIC option in the *Command Reference*.)

The mat value represents the material number, and the npts value is the number of material constants.

The USER argument accesses the interface to the user_tbelastic subroutine.

6.4.4.4. User Interface for user_tbelastic

The user_tbelastic interface consists of six arguments, as follows:

- · Four input arguments for the element number, material number, coordinate array, and temperature
- One input/output argument for the number of material constants
- One output argument consisting of the material constants array

The syntax is as follows: SUBROUTINE user_tbelastic(elemId, matId, coords, temp, nprop, prop)

Argument	Input (I) or Out- put (O)	Definition	
elemId	I	Element number	
matId	I	Material number	
coords	I	Coordinates of material integration point at initial configuration (geometry)	
temp	I	Current temperature at material integration point	
nprop	1/0	Number of constants to be returned (input) or actually returned (output), as follows: 2 - isotropic elasticity 9 - orthotropic elasticity 21 - anisotropic elasticity	

		The value for this argument is obtained via the TB ,ELASTIC command, and is passed into the subroutine. However, you can redefine this value in the subroutine, which then returns it.
prop	0	The material elastic constants to be defined

6.4.4.5. The user_tbelastic API

Following is the interface to the user_tbelastic subroutine:

```
USERSDISTRIB
*deck,user tbelastic
    SUBROUTINE user_tbelastic(elemId, matId, coords, temp,
                        nprop, prop)
*** primary function ***
С
С
         user interface for elastic material constants
C
С
С
    input arguments
С
    _____
     elemId (in, sc
                   , i)
                         Element number
C
     matId (in, sc , i)
                         Number of material index
С
С
     temp
           (dp, sc , i)
                         Current temperature
C
     coords
           (dp, ar(5), i)
                          Coordinates at initial configuration
С
С
                           For continuum elements:
                            1-3: coordinates of integration point
C
                            4-5: not used
C
                           For line elements:
C
С
                            1-3: coordinates of integration point
С
                                   along line member axis
                            4-5: offsets in element y and z directions
C
С
С
     output arguments
С
    Number of constants
С
            (in, sc , o)
                            2 - isotropic
С
                                         elasticity
                             9 - orthotropic elasticity
С
                             21 - anisotropic elasticity
C
            (dp, ar(*), o)
                          Material elastic constants (stiffness)
C
     prop
С
    local variables
C
    ==========
C************************
C
c --- parameters
```

6.4.4.6. Usage Example for user_tbelastic

In this example, three elements in parallel are subjected to uniaxial tension.

Element 1 is a SOLID185 element defined via the ANSYS MP command with linear isotropic elasticity.

Element 2 is a SOLID185 element defined via the user-defined elastic material properties interface.

Element 3 is a SHELL181 element defined via the user-defined elastic material properties interface.

Solid elements are a unit cubic with a 1 mm edge. The shell element is a unit square with a 1 mm edge. The Young's modulus is 210E6 MPa, and the Poisson's ratio is 0.3.

Example ANSYS Input

```
/batch
/com
/com example for user elastic material property interface
/com
/com element 1 solid185 defined via ANSYS standard MP command
/com element 2 solid185 defined using ansys elastic material interface
/com element 3 shell181 defined using ansys elastic material interface
/prep7
esize,,1
et,1,185
et,2,181
mp, ex, 1, 210e6
mp, nuxy, 1, 0.3
tb,elastic,2,1,2,user
                         ! user-defined elastic material interface
! SOLID185 element
mat,2
block,,1,,1,,1
vmesh,1
mat,1
block,,1,,1,,1
vmesh,2
! SHELL181 element
sectype, 1, shell
secdata, 0.100000,1
secdata, 0.100000,2
rect,,1,,1
secn,1
mat,2
type,2
amesh,1
elist,all,all
nsel,s,loc,x
d,all,ux
nsel,s,loc,y
d,all,uy
nsel,s,loc,z
d,all,uz
/solu
nsel,s,loc,x,1
d,all,ux,0.05
alls
solve
fini
/post1
set,1
pres,s
pres, epel
fini
```

6.4.5. Subroutine UserPL (Writing Your Own Plasticity Laws)

ANSYS recommends using current-technology elements and the UserMat subroutine for defining your own material model. However, if you are using a legacy element type and wish to define a plasticity or viscoplas-

ticity material model, the UserPL subroutine is applicable to the following legacy elements: LINK1, LINK8, PIPE20, BEAM23, BEAM24, PLANE42, SOLID45, PIPE60, SOLID62, SOLID65, PLANE82, SOLID92, and SOLID95.

```
*deck,userpl
                                   USERSDISTRIB
      subroutine userpl (elem,intpt,mat,ncomp,kfirst,kfsteq,e,nu,dens,
     x prop,d,ktform,timval,timinc,tem,dtem,toffst,flu,dflu,epel,eppl,
     x statev, usvr, epeq, plwork, sigepl, sigrat, depeq, dt)
С
c *** primary function:
                           allow users to write their own plasticity laws.
С
                             this logic is accessed with tb, user.
С
                             the below demonstration logic is the same as using
                             tb, bkin, without adaptive descent (nropt,,,off).
C
                             Other plasticity rules may require internal
C
С
                             iterations and/or the more general definition of
С
                             plasticity theory, discussed in the Theory
                             Manual.
C
  *** secondary function:
                              demonstrate the use of user-written plasticity laws
С
                   in this routine:
С
С
                            update the nonlinear strain history
                      b.
                            compute the material tangent matrix if requested
C
С
  *** Notice - This file contains ANSYS Confidential information ***
С
C
С
           *** ansys(r) copyright(c) 2008
C
С
          *** ansys, inc.
C
С
   input arguments:
С
      variable (type,sze,intent)
                                     description
С
C
      elem
               (int.sc.in)
                                    - element number (label)
С
      intpt
                (int,sc,in)
                                    - element integration point number
С
      mat
               (int,sc,in)
                                    - material reference number
                                    - no. of stress/strain components (1,4 \text{ or } 6)
               (int,sc,in)
C
      ncomp
C
                                      1 - x
C
                                      4 - x, y, z, xy
С
                                      6 - x, y, z, xy, yz, xz
      kfirst
                (int,sc,in)
                                    - 1 if first time through, 0 otherwise
С
                                       (useful for initializing state variables
C
C
                                      to a non-zero value)
С
      kfsteq
                (int,sc,in)
                                    - 1 if first equilibrium iteration of a
C
                                      substep, 0 otherwise
С
С
                (dp,sc,in)
                                    - average elastic modulus
                                    - average poisson ratio
                (dp,sc,in)
C
      nu
                                    - current material density (mass/volume)
С
                (dp,sc,in)
С
      prop
                                    - linear material property array
                (dp,ar(9),in)
С
                                      (ex,ey,ez, gxy,gyz,gxz, nuxy,nuyz,nuxz)
C
                (dp,ar(1),in)
                                      if ncomp=1 (ex)
                (dp,ar(ncomp,ncomp),in)- elastic stress-strain matrix
C
      ktform
               (int,sc,in)
                                    - request key for tangent matrix formation
C
С
                                       (=1, form tangent .ne.1, do not form)
С
С
      timval
                (dp,sc,in)
                                    - current time value
                                    - time increment over this substep
С
      timinc
               (dp,sc,in)
C
                (dp,sc,in)
                                    - temperature at the end of this substep
С
С
      dt.em
                (dp,sc,in)
                                    - temperature increment over this substep
      toffst
                (dp,sc,in)
                                    - temperature offset from absolute zero
C
      flu
                (dp,sc,in)
                                    - fluence at the end of this substep
C
      dflu
                                    - fluence increment over this substep
               (dp,sc,in)
C
С
С
      epel
                (dp,ar(ncomp),inout)- modified total strain (trial strain)
                                       epel = eptot - eppl - eptherm - ...
C
С
                                       if a large strain analysis, epel is
С
                                      rotation neutralized and is the hencky
                                       (i.e. log) strain
C
                (dp,ar(ncomp),inout) - plastic strain from previous substep
C
      eppl
С
С
                (dp,ar(ncomp,6),inout) - state variables from previous substep
      statev
                (dp,ar(nuval,nintp),inout) - additional state variables from
C
      usvr
```

```
previous equilibrium iteration (saved
C
                                      if the nsvr command is used)
С
С
                                    - effective plastic strain from prev substep
С
      epeq
                (dp,sc,inout)
                                    - accumulated plastic work from prev substep
С
      plwork
               (dp,sc,inout)
С
  output arguments:
C
C
      variable (type,sze,intent)
                                     description
C
С
      epel
                (dp,ar(ncomp),inout)- elastic strain
                (dp,ar(ncomp),inout) - updated plastic strain
C
      eppl
C
                (dp,ar(ncomp,6),inout)- updated state variables
С
      statev
                (dp,ar(nuval,nintp),inout) - updated additional state variables
С
      usvr
C
                                    - updated effective plastic strain
С
      epeq
                (dp,sc,inout)
C
      plwork
               (dp,sc,inout)
                                    - updated accumulated plastic work
C
C
      sigepl
                (dp,sc,out)
                                    - stress value on stress-strain curve at epeq
      sigrat
               (dp,sc,out)
                                    - ratio of trial stress to yield stress
С
      depeq
               (dp,sc,out)
                                     - increment in plastic strain (equivalent)
С
С
                                       (used for auto time stepping - time step
                                      is reduced if it exceeds .05)
С
С
                (dp,ar(ncomp,ncomp),out)- material tangent modulus
С
      dt.
C
С
   fortran parameters (to be defined by the user):
      variable (type)
C
                                    description
      numinp
               (int)
                                     - number of data items in the user-defined
С
                                      data table (tbdat commands)
C
                                    - number of additional state variables per
C
      nuval
               (int.)
С
                                      integration point
C
      nintp
                (int)
                                     - maximum number of integration points of
                                      an element to be used with this routine
C
С
                                      (14 is the maximum)
С
          note: nuval x nintp = nstv(on nsvr command); cannot exceed 840!
C
   internal variables:
С
С
      variable (type, sze)
                                description
      b
               (dp,ar(6,6))
                                - 2nd derivative of the yield function
C
                (dp,ar(6,12))
                                - part of deff
C
C
      con
               (dp,sc)
                                - temporary variable
С
      deppl
               (dp,ar(6))
                                - plastic strain increment
С
      dfds
                (dp,ar(6))
                                - derivative of the yield function (normal)
                                - plastic multiplier
      dlamb
               (dp,sc)
C
               (dp,ar(6))
                                - shifted strain
С
      ep
      epshfo
С
                (dp,ar(6))
                                - initial shift strain (center of the yield surf)
      epshft
                                - shift strain (center of the yield surface)
               (dp,ar(6))
С
                                - tangent modulus (stress/total strain)
C
      et
               (dp,sc)
                                - plastic tangent modulus (stress/plastic strain)
С
      h
               (dp,sc)
C
      n2
               (int,sc)
                                - ncomp squared, ncomp*ncomp
      seqtr
               (dp,sc)
                                - equivalent (von mises) trial stress
C
      sigtr
               (dp,ar(6))
                                - trial stress
C
С
      sigy
               (dp,sc)
                                - yield stress
C
      vect.
               (dp,ar(6))
                                - temporary vector
```

6.4.6. Subroutine usersw (Writing Your Own Swelling Laws)

```
*** ansys, inc.
c *** Notice - This file contains ANSYS Confidential information ***
С
  input arguments:
C
                                 swelling option
С
     option
               (int,sc,in)
С
      elem
               (int,sc,in)
                                 element number (label)
               (int,sc,in)
                                 element integration point number
     intpt
C
C
               (int,sc,in)
                                 material reference number
                                 nonlinear material table (tb commands)
               (dp,ar(*),in)
C
     proptb
С
     ncomp
               (int,sc,in)
                                 number of strain components (=1, 4, or 6)
                                     1 - truss or beam elements
С
                                     4 - 2-d solids and pipe elements
C
С
                                     6 - 3-d solids and most shells
С
      epswel
               (dp,sc,inout)
                                  total accumulated swelling strain
                                       before this substep
C
               (dp,ar(ncomp),inout) elastic strain
С
      epel
С
               (dp,sc,in)
                                 elastic modulus
               (dp,sc,in)
                                 poisson'S RATIO
     nuxv
C
      fluen
               (dp,sc,in)
                                 total fluence (bf or bfe commands)
      dfluen
                                 increment in fluence for this substep
C
               (dp,sc,in)
               (dp,sc,in)
                                 temperature (bf or bfe commands)
С
      tem
С
      dtem
               (dp,sc,in)
                                 increment in temperature for this substep
     toffst
                                 offset of temperature scale from absolute zero
               (dp,sc,in)
C
С
                                                 (toffst command)
      timvll
               (dp,sc,in)
                                 time at the end of this substep
C
      timync
               (dp,sc,in)
                                 the increment of time this substep
C
С
      usvr
               (dp,ar(*),inout) user-defined state variables(optional)
C
    output arguments:
C
                                  total accumulated swelling strain
С
      epswel
               (dp,sc,inout)
С
                                       after this substep
С
               (dp,ar(ncomp),inout) elastic strain adjusted
      epel
C
                                                   for swelling increment
               (dp,ar(*),inout) updated user-defined state variables
C
      usvr
```

6.4.7. Subroutine userck (Checking User-Defined Material Data)

```
*deck,userck
                                  USERSDISTRIB
      subroutine userck (curmat,ntb,tb)
c *** primary function:
                           check the user-defined material data,
                            input with the TB, user command.
c *** secondary functions: none
C
С
          *** copyright(c) 2006 SAS IP, Inc. All rights reserved.
          *** ansys, inc.
С
  *** Notice - This file contains ANSYS Confidential information ***
C
C
   input arguments:
                                 - current material number
С
     curmat (int,sc,in)
С
     ntb
               (int,sc,in)
                                 - dimension of tb
               (dp,ar(ntb),in)
                                - input table
C
С
С
   output arguments:
С
     none
```

6.4.8. Subroutine UserVisLaw (Defining Viscosity Laws)

```
Primary function: to provide a user defined viscosity
С
                          relationship in terms of the following:
C
                          pressure, temperature, position, time,
C
                          velocity, & velocity-gradient
C
                          This routine is for use with the FLOTRAN
С
                          elements, Fluid141 and Fluid142 only.
C
C
       In order to activate this subroutine the user must issue
C
       FLDA, PROT, VISC, USRV command.
C
       In addition the initial value of viscosity must be specified via
C
С
       FLDA, PROP, IVIS, value. This value is not available in this routine.
C
       Optionally the user may specify 4 additional coefficients
C
C
       which are available in this routine by the commands:
C
       FLDA, NOMI, VISC, value1
С
       FLDA, COF1, VISC, value1
C
       FLDA, COF2, VISC, value2
C
       FLDA, COF3, VISC, value3
C
          *** copyright(c) 2008 SAS IP, Inc. All rights reserved.
С
          *** ansys, inc.
C
C
C
       input arguments:
С
       variable
                   (typ, siz, intent)
                                        description
C
       dudx
                   (dp,sc,in)
                                        velocity gradient component
C
                                        velocity gradient component
       dudy
                   (dp,sc,in)
C
       dudz
                                        velocity gradient component
                   (dp,sc,in)
C
       dvdx
                   (dp,sc,in)
                                        velocity gradient component
C
       dvdy
                   (dp,sc,in)
                                        velocity gradient component
С
       dvdz
                   (dp,sc,in)
                                        velocity gradient component
С
       dwdx
                   (dp,sc,in)
                                        velocity gradient component
С
       dwdv
                                        velocity gradient component
                   (dp,sc,in)
C
       dwdz
                   (dp,sc,in)
                                        velocity gradient component
C
                   (dp,sc,in)
                                        velocity component
С
                                        velocity component
       v
                   (dp,sc,in)
С
       W
                   (dp,sc,in)
                                        velocity component
C
       х
                   (dp,sc,in)
                                        position component
С
       У
                   (dp,sc,in)
                                       position component
C
                   (dp,sc,in)
                                       position component
С
       kGeom
                   (int,sc,in)
                                        analysis type
C
       Vis
                   (dp,sc,in)
                                        old viscosity
C
       Temp
                   (dp,sc,in)
                                        absolute temperature
С
       Tref
                                        reference temperature (Absolute also)
                   (dp,sc,in)
С
       Pres
                   (dp,sc,in)
                                       pressure
С
       Pref
                   (dp,sc,in)
                                        reference pressure
С
                   (dp,ar(4),in)
       Cf
                                        input coefficients
C
       Mfrac
                   (dp,ar(6),in)
                                        species mass fractions
C
       DfNSpec
                   (int,sc,in)
                                        defined number of species
C
       Time
                   (dp,sc,in)
С
       toffst
                   (dp,sc,in)
                                        Temperature offset for absolute scale.
С
       output arguments:
С
       variable
                   (typ,siz,intent)
                                        description
C
       VisNew
                   (dp,sc,out)
                                        new viscosity
```

6.4.9. Subroutine userfric (Writing Your Own Friction Laws)

This subroutine is applicable to the following contact elements: CONTA171, CONTA172, CONTA173, CONTA174, CONTA175, CONTA176, CONTA177, and CONTA178.

```
The below demonstration logic is the same as using
С
                           tb, fric for isotropic Coulomb friction.
С
С
                           Other friction laws may require more general
                           definition of friction forces.
С
  *** secondary function: demonstrate the use of user-written friction laws
С
C
                           in this routine:
                           a. update history variables
С
C
                           b. compute consistent tangent matrix
C
С
  *** Notice - This file contains ANSYS Confidential information ***
С
C
          *** ansys(r) copyright(c) 2009
С
С
          *** ansys, inc.
C
С
   input arguments:
С
      variable (type,sze,intent)
                                     description
C
С
      elem
                (int,sc,in)
                                    - element number (label)
                (int,sc,in)
                                    - material reference number
C
      mat
      intpt
                (int,sc,in)
                                    - element integration point number
С
С
      nkeyopt
               (int,sc,in)
                                     - number of key options
                (int,ar(nkeyopt),in) - array containing key options
      keyopt
C
С
                                      keyopt(1): Select degree of freedom
С
                                      keyopt(2) : Contact algorithm
С
                                       ... so on (see ANSYS documentation)
С
      nrl
                (int,sc,in)
                                    - number of real constants
                                     - array containing real constants
С
      rlconst (dp,ar(nrl),in)
                                       Elements CONTA171 to CONTA177
С
                                       rlconst(1) : FKN
С
                                      rlconst(2) : FTOLN
С
С
                                       ... so on (see ANSYS documentation)
С
                                       Element CONTA178
                                      rlconst(1) : FKN
C
С
                                      rlconst(2) : GAP
                                       ... so on (see ANSYS documentation)
C
                (int,sc,in)
                                    - no. of friction stress components (1 or 2)
C
      ncomp
                                     - no. of user-defined friction properties
С
      npropu
                (int,sc,in)
                                    - user-defined material properties
С
      uprop
                (dp,ar(npropu),in)
      kfirst
                                     - 1 if first time through, 0 otherwise
C
                (int,sc,in)
                                       (useful for initializing state variables
С
C
                                       to a non-zero value)
С
      kfsteq
                (int,sc,in)
                                     - 1 if first equilibrium iteration of a
С
                                       substep, 0 otherwise
                                    - normal penalty stiffness
                (dp,sc,in)
C
      kn
С
                (dp,sc,in)
                                     - tangential penalty stiffness
      kt.
С
                                       (an initial guess is provided but
                                       the user must pick a suitable
С
                                        value that allows minimal tangential
С
С
                                       slip during sticking without
C
                                       adversely affecting the convergence;
                                        a possible choice could be kt=mu*kn)
C
      elen
                (dp,sc,in)
                                     - length of contact element
C
С
      kstat.
                (int,sc,inout)
                                      contact status
С
                                       1 : stick
                                       2 : sliding
C
С
                                       3 : open contact (near)
С
                                      4 : open contact (far)
                                    - current time value
С
      timval
                (dp,sc,in)
С
      timinc
                (dp,sc,in)
                                     - time increment over this substep
С
      tcont
                (dp,sc,in)
                                    - contact surface temperature
С
                                       (from temperature DOF or temperature load)
                (dp,sc,in)
                                    - target surface temperature
C
      ttarg
                                       (only from temperature DOF)
C
С
      toffst
                (dp,sc,in)
                                    - temperature offset from absolute zero
      dslip
                (dp,ar(ncomp),in)
                                    - slip increment (current substep)
С
                (dp,ar(ncomp),inout)- accumulated slip (previous substep)
C
      slip
С
      pres
                (dp,sc,in)
                                     - normal pressure/force (current substep)
С
                                      > 0 : compression
                                       < 0 : tension
C
С
      tau
                (dp,ar(ncomp),inout) - frictional stress (previous substep)
                (dp,ar(nuval,nintp),inout) - additional state variables from
C
      usvr
```

```
previous equilibrium iteration (saved
C
                                      if the nsvr command is used)
С
С
      kupdhis (int,sc,in)
                                    - key to indicate if history-dependent
                                      variables (user defined) need to be
С
                                      updated after the substep has converged
С
С
                                      1 : update (converged)
                                      0 : do not update (not converged)
С
C
   output arguments:
C
С
      variable (type,sze,intent)
                                     description
С
                                    - updated contact status
C
      kstat
                (int.sc.inout)
               (dp,ar(ncomp),inout)- updated accumulated slip
С
      slip
С
      tau
                (dp,ar(ncomp),inout)- updated frictional stress
      dt
                (dp,ar(5,5),out)
                                    - material tangent modulus
C
                                      rows and columns of dt matrix are
С
C
                                      associated to:
                                      row 1 : frictional stress in direction 1
С
С
                                      row 2 : frictional stress in direction 2
                                      row 3 : normal pressure
С
                                      row 4 : blank
С
                                      row 5 : blank
С
                                      col 1 : sliding in direction 1
C
                                      col 2 : sliding in direction 2
С
С
                                      col 3 : normal gap
                                      col 4 : blank
C
C
                                      col 5 : blank
                                      relevant components to be filled in are:
C
                                      dt(1,1): d(taul)/d(slip1)
С
                                      dt(1,2): d(tau1)/d(slip2)
С
                                      dt(1,3): d(tau1)/d(normal gap)
C
С
                                      dt(2,1): d(tau2)/d(slip1)
C
                                      dt(2,2): d(tau2)/d(slip2)
                                      dt(2,3): d(tau2)/d(normal gap)
C
С
                                      dt(3,3): d(pres)/d(normal gap)
                                      dt(3,3) set to kn internally
С
      usvr
                (dp,ar(nuval,nintp),inout)- updated additional state variables
C
                                      For example, mu value and absolute
C
                                      accumated slip could be output as follows:
C
                                      usvr(1,intpt) : mu
C
                                      usvr(2,intpt) : abs. acc. slip in dir1
С
                                      usvr(3,intpt) : abs. acc. slip in dir2
C
С
                                      Use NSVR command to size usvr array and
                                      set nuval to same value as number of
C
                                      variables on NSVR commands
C
                                      Use userou.F to save these values
С
С
                                      on NMISC record for output purposes
      fdiss
С
               (dp.sc.out)
                                     - incremental frictional dissipation
                                      per unit area
C
                                     - incremental elastic stored energy
С
      elener
               (dp,sc,out)
C
                                      per unit area
C
   fortran parameters (to be defined by the user):
C
С
      variable (type)
                                    description
C
      nuval
               (int)
                                     - number of additional state variables per
C
                                      integration point
                                     - maximum number of integration points of
С
      nintp
С
                                      an element to be used with this routine
                                      (14 is the maximum)
С
С
          note: nuval x nintp = nstv(on nsvr command); cannot exceed 840!
C
   internal variables:
C
      variable (type,sze)
                                description
C
               (dp,sc
                                - friction coefficient
C
      mu
                           )
      dtfac
                (dp,sc)
                                - temporary variable
С
      taulim
                (dp,sc)
                                - limit frictional stress
C
C
      taueq
               (dp,sc)
                                - equivalent frictional stress
С
      dir1
                (dp,sc)
                                - slip increment direction 1
С
      dir2
                (dp,sc)
                                - slip increment direction 2
      dslipeq
                                - equivalent slip increment
C
               (dp,sc)
      oldt1
                                - frictional stress 1 from prev substep
С
                (dp,sc)
                                - frictional stress 2 from prev substep
      oldt2
C
                (dp,sc)
```

```
c eldslip (dp,ar(2)) - elastic slip increment
c err (dp,ar(2)) - data array for diagnostic message
c
```

6.4.10. Supporting Function egen

The function egen (kcomp,ep,nuxy) (function) combines kcomp strain components (ep) per:

```
*deck,egen
     function egen (kcomp,ep,posn)
c primary function: combines strain components to give an "overall" strain
                     used in creep and plasticity calculations
c secondary functions: none
С
    formulation of overall value is by:
С
С
      / -*((ep - ep ) + (ep - ep ) + (ep - ep ) + -*(ep
                                                                  + ep ))
C
     \/ 2 1 2 2 3 3 1 2 4 5
C
                              (1 + posn)
c *** Notice - This file contains ANSYS Confidential information ***
     typ=int,dp,log,chr,dcp siz=sc,ar(n) intent=in,out,inout
C
C
  input arguments:
     variable (typ,siz,intent) description
C
     kcomp (int,sc,in) - number of components of strain
C
            (dp,ar(6),in) - the strain components
(dp,sc,in) - poisson's ratio
С
    posn (dp,sc,in)
C
 output arguments:
С
    egen (dp,func,out)
                              - the combined strain value
C
C
```

6.5. Subroutines for Customizing Loads

This section describes user subroutines that you can use to modify or monitor existing ANSYS element loading. Activate these subroutines by issuing the **USRCAL** command or by selecting an equivalent menu path.

```
6.5.1. Subroutine usrefl (Changing Scalar Fields to User-Defined Values)
```

- 6.5.2. Subroutine userpr (Changing Element Pressure Information)
- 6.5.3. Subroutine usercy (Changing Element Face Convection Surface Information)
- 6.5.4. Subroutine userfx (Changing Element Face Heat Flux Surface Information)
- 6.5.5. Subroutine userch (Changing Element Face Charge Density Surface Information)
- 6.5.6. Subroutine userfd (Computing the Complex Load Vector for Frequency Domain Logic)
- 6.5.7. Function userpe (Calculating Rotation Caused by Internal Pressure)
- 6.5.8. Subroutine UsrViscEl (Performs Viscoelastic Computation)
- 6.5.9. Subroutine usrsurf116 (Modifying SURF151 and SURF152 Film Coefficients and Bulk Temperatures)
- 6.5.10. Subroutine User116Cond (Computing the conductance coefficient for FLUID116)
- 6.5.11. Subroutine User116Hf (Computing the film coefficient for FLUID116)
- 6.5.12. Subroutine userPartVelAcc (Computing Particle Velocities and Accelerations)

6.5.1. Subroutine usrefl (Changing Scalar Fields to User-Defined Values)

```
c *** primary function: change the scalar fields (temperatures, fluences,
                   heat generation, etc.) to what user desires.
С
  *** secondary functions: none
C
          in order to activate this user programmable feature,
С
С
          the user must enter the usrcal command.
С
          this routine is called at each substep of each load step
          for which element or nodal temperatures(etc) are used.
С
С
          it is called for each equilibrium iteration.
          the call to get the standard ansys input element or nodal values
C
          is made just before entering this routine.
C
C
          *** copyright(c) 2008 SAS IP, Inc. All rights reserved.
С
          *** ansys, inc.
C
С
  *** Notice - This file contains ANSYS Confidential information ***
С
C
С
      typ=int,dp,log,chr,dcp siz=sc,ar(n)
                                              intent=in,out,inout
C
  input arguments:
C
С
      variable (typ,siz,intent)
                                    description
                                  - type of data desired
               (int,sc,in)
C
С
                                    = 1 temperatures
С
                                    = 2 fluences
C
                                    = 3 heat generation rates
                                    = 4 moisture contents
C
                                    = 5 magnetic virtual displacements
C
      iel
                                 - element number
               (int,sc,in)
C
             (int,ar(IELCSZ),in) - array of element type characteristics
C
      ielc
                                 - number of nodes
C
      nnod
               (int,sc,in)
               (int,ar(nnod),in) - list of nodes
С
      nodes
C
      time
               (dp,sc,in)
                                  - time of current substep
                                 - default value (e.g. tunif)
      defalt.
               (dp,sc,in)
C
                                 - size of dat array
С
               (int,sc,in)
С
               (dp,ar(nd),inout) - array of data as normally computed by element
                                   as selected by key
С
С
С
   output arguments:
C
      variable (typ,siz,intent)
                                   description
               (dp,ar(nd),inout) - array of data passed back to element
С
C
                                      this data represents values at the end
С
                                      of the load step
C
         the input argument dat may be used in one of three ways:
C
             1. it may be simply passed thru
С
C
             2. it may be used as a flag(e.g. if dat(1) = -3.0, use
                                    a certain set of logic)
C
             3. it may be completely ignored and instead defined with new logic
```

6.5.2. Subroutine userpr (Changing Element Pressure Information)

```
*deck,userpr
                                  USERSDISTRIB
     subroutine userpr (ielc,elem,time,ndat,dat)
c *** primary function:
                           change element pressure information.
          *** copyright(c) 2008 SAS IP, Inc. All rights reserved.
C
          *** ansys, inc.
C
          in order to activate this user programmable feature,
          the user must enter the 'usrcal, userpr' command.
          this routine is called at each substep of each load step for which
          pressures are used. it is called for each equilibrium iteration.
C
          it is called once per element.
С
          the call to get the standard ansys input pressures is made just before
          entering this routine.
```

```
С
      input arguments:
С
         variable (typ,siz,intent)
                                      description
С
               (int,ar(IELCSZ),in) - array of element type characteristics
С
         elem
                 (int,sc,in)
                                    - element number for operation.
С
         time
                  (dp,sc,in)
                                    - time of current substep
C
         ndat
                  (int,sc,in)
                                    - number of pressure items for this element
                  (dp,ar(ndat,2),inout) - the element pressure vector
С
         dat
C
                                       (has input values for each corner
                                       of each face)
C
C
      output arguments:
         variable (typ,siz,intent)
                                      description
C
С
                  (dp,ar(ndat,2),inout) - the element pressure vector
                                        (defines input values for each corner
C
С
                                        of each face)
C
                                       dat(1:ndat,1) - real pressures
                                       dat(1:ndat,2) - complex pressures
C
                                                        (surface elements only)
         the input array dat may be used in one of three ways:
C
           1. it may be simply passed thru
C
C
           2. it may be used as a flag(e.g. if dat(1) = -3.0, use
                  a certain set of logic)
C
             it may be completely ignored and instead defined with new logic
```

6.5.3. Subroutine usercy (Changing Element Face Convection Surface Information)

```
*deck,usercv
                                  USERSDISTRIB
      subroutine usercv (elem,ielc,time,nr,u, ndat,hc,tb)
c *** primary function: change element face convection surface info
C
С
          *** copyright(c) 2006 SAS IP, Inc. All rights reserved.
          *** ansys, inc.
С
C
          in order to activate this user programmable feature,
C
          the user must enter the 'usrcal, usercv' command.
С
C
С
         the input arguments hc and tb may be used in one of three ways:
             1. they may be simply passed thru.
C
             2. they may be used as a flag(e.g. if hc(2) = -3.0, use
C
                                    a certain set of logic).
С
С
             3. they may be completely ignored.
                                     and instead redefined with new logic
C
С
          this routine is called during each substep of each load step.
          it is called for each equilibrium iteration.
C
          it is called once per element. it is called only during the heat
C
          flow load vector formulation stage, and not during the heat flow
C
С
          evaluation stage.
          the call to get the standard ansys input convection surfaces
C
          is made just before entering this routine, so this information is
C
          available to be modified, if desired.
С
С
          velocity-dependent film coefficients can be computed by inputting the
C
С
          velocity as the input film coefficient or bulk temperature or
С
          by inputting the velocity as a function of location in space.
          routine could then compute the effective film coefficient.
C
С
С
      input arguments:
         variable (typ,siz,intent)
                                      description
C
                  (int,sc,in)
                                    - element number for operation.
C
С
         ielc
               (int,ar(IELCSZ),in) - array of element type characteristics
                                    - time of current substep
         time
С
                  (dp,sc,in)
```

```
(int,sc,in)
                                     - number of nodal temperatures
C
         nr
С
                                            of the element
С
                   (dp,ar(nr),in)
                                     - vector of most recent values of the
С
                                         temperatures
C
                   (int,sc,in)
                                      - number of data points per element
С
                                         for example, for solid70, ndat = 24 = 6*4
                                         where 6 = faces per element
С
                                               4 = corners per face
C
                 (dp,ar(ndat),inout) - film coefficients
С
         hc
С
                                         (has input values for each corner
                                         of each face)
C
         tb
                (dp,ar(ndat),inout) - bulk temperature
C
                                         (has input values for each corner
C
С
                                         of each face)
C
С
      output arguments:
C
         variable (typ,siz,intent)
                                        description
                (dp,ar(ndat),inout) - film coefficients
C
С
                                         (defines input values for each corner
С
                                         of each face)
                (dp,ar(ndat),inout) - bulk temperature
С
         tb
С
                                         (defines input values for each corner
                                         of each face)
C
```

6.5.4. Subroutine userfx (Changing Element Face Heat Flux Surface Information)

```
*deck,userfx
                                   USERSDISTRIB
      subroutine userfx (ielc,elem,time,nr,u, ndat,dat)
     primary function: change element face heat flux surface info
          *** copyright(c) 2008 SAS IP, Inc. All rights reserved.
C
          *** ansys, inc.
C
C
          in order to activate this user programmable feature,
C
C
          the user must enter the 'usrcal, userfx' command.
C
          this routine is called during each substep of each load step.
C
          it is called for each equilibrium iteration.
C
          it is called once per element. it is called only during the heat
C
С
          flow load vector formulation stage, and not during the heat flow
С
          evaluation stage.
          the call to get the standard ansys input heat flux surfaces
C
С
          is made just before entering this routine, so this information is
C
          available to be modified, if desired.
C
      input arguments:
С
С
         variable (typ,siz,intent)
                                       description
                (int,ar(IELCSZ),in) - array of element type characteristics
C
         ielc
         elem
                  (int,sc,in)
                                    - element number for operation.
C
         time
                  (dp,sc,in)
                                     - time of current substep
C
С
         nr
                  (int,sc,in)
                                     - number of nodal temperatures
                                           of the element
C
                                     - vector of most recent values of the
C
         u
                  (dp,ar(nr),in)
С
                                        temperatures
C
         ndat
                  (int,sc,in)
                                     - number of data points per element
                                        for example, for solid70, ndat = 24 = 6*4
С
C
                                        where 6 = faces per element
С
                                              4 = corners per face
         dat
                (dp,ar(ndat),inout) - fluxes
C
                                        (has input values for each corner
C
                                        of each face)
C
C
      output arguments:
C
         variable (typ,siz,intent)
                                       description
C
С
                (dp,ar(ndat),inout) - fluxes
                                        (defines input values for each corner
С
```

```
c of each face)
```

6.5.5. Subroutine userch (Changing Element Face Charge Density Surface Information)

```
*deck,userch
                                  USERSDISTRIB
      subroutine userch (ielc,ielem,time,nr,u, ndat,dat)
c *** primary function: change element face charge density surface info
          in order to activate this user programmable feature,
C
С
          the user must enter the usrcal command.
С
          this routine is called during each substep of each load step.
С
С
          it is called once per element. it is called only during the heat
C
         flow load vector formulation stage, and not during the heat flow
          evaluation stage.
C
          the call to get the standard ansys input charge densities of surfaces
         is made just before entering this routine, so this information is
C
С
          available to be modified, if desired.
С
          *** copyright(c) 2006 SAS IP, Inc. All rights reserved.
C
          *** ansys, inc.
С
С
C
     input arguments:
        variable (typ,siz,intent)
                                      description
С
         ielc (int,ar(IELCSZ),in) - array of element type characteristics
С
               (int,sc,in) - element number for operation.
С
        time
                  (dp,sc,in)
                                   - time of current substep
                 (int,sc,in)
С
        nr
                                    - number of nodal temperatures
С
                                          of the element
С
                  (dp,ar(nr),in)
                                    - vector of most recent values of the
                                       temperatures
C
С
                                    - number of data points per element
         dat
                (dp,ar(ndat),inout) - fluxes
С
C
     output arguments:
С
        variable (typ,siz,intent)
С
                                      description
               (dp,ar(ndat),inout) - fluxes
C
С
C
         the input argument dat may be used in one of three ways:
С
             1. they may be simply passed thru.
             2. they may be used as a flag(e.g. if dat(2) = -3.0, use
С
                                    a certain set of logic).
C
             3. they may be completely ignored.
С
                                     and instead redefined with new logic
C
```

6.5.6. Subroutine userfd (Computing the Complex Load Vector for Frequency Domain Logic)

```
*deck,userfd
     subroutine userfd (nr,kcbrm,kpfor,ktrsur,isur,
     x cb,do,doext,aread,alenv,denswat,faclen,conac,fluidt,visc,
     x watbas, watcur, watwav, xyzup, tr, accel, puvel, u, zass,
     x forl,zsc,zsc2,pdyn,holdwv)
c *** primary function: compute complex load vector for frequency domain logic
                         for pipe59
c *** secondary functions: none
      -- accessed with keyopt(12) = 2
C
          *** copyright(c) 2006 SAS IP, Inc. All rights reserved.
C
          *** ansys, inc.
C
c *** Notice - This file contains ANSYS Confidential information ***
C
```

```
input arguments:
C
                                  - matrix size
C
               (int,sc,in)
С
      kchrm
               (int,sc,in)
                                  - key for reduced matrices/cable option
      kpfor
               (int,sc,in)
                                  - keyopt for hydrodynamic printout
С
                                  - keyopt for surface treatment(unfinished)
С
      ktrsur
               (int,sc,in)
                                 - surface flag
С
      isur
               (int,sc,in)
               (dp,sc,in)
                                 - buoyancy coefficient (real constant)
C
C
      do
               (dp,sc,in)
                                  - outside diameter of pipe
               (dp,sc,in)
                                  - outside diameter of insulation
      doext
C
С
      aread
               (dp,sc,in)
                                  - area of displaced water
                                  - length of element
      alenv
               (dp,sc,in)
C
                                 - water density
C
      denswat
              (dp.sc.in)
                                 - wetted fraction of pipe
С
      faclen
               (dp,sc,in)
С
      conac
               (dp,sc,in)
                                  - added mass per unit length
                                  - fluid temperature
      fluidt.
               (dp,sc,in)
C
С
      visc
               (dp,sc,in)
                                  - viscosity
                                 - water basic table
C
      watbas
               (dp,ar(*),in
               (dp,ar(*),in
      watcur
                                 - water current table
C
C
      watwav
               (dp,ar(*),in
                                  - water wave table
               (dp,ar(3,2),in)
                                 - updated coordinates
С
      xyzup
               (dp,ar(3,3),in)
                                 - local to global transformation matrix
С
      tr
С
      accel
               (dp,ar(3),in)
                                  - acceleration vector
               (int,sc,in)
                                  - index for velocities in u matrix
      puvel
C
                                 - displacements and velocities
С
               (dp,ar(nr,5),in
С
      zass
               (dp,ar(nr,nr),in) - mass matrix
C
      forl
               (\mathrm{dp},\mathrm{ar}(12),\mathrm{inout}) - force vector in element coordinates
C
      zsc
               (dp,ar(nr),inout) - real load vector for frequency domain
               (dp,ar(nr),inout) - complex load vector for frequency domain
C
      zsc2
C
C
   output arguments:
               (dp,ar(12),inout) - force vector in element coordinates
С
      forl
С
               (dp,ar(nr),inout) - real load vector for frequency domain
C
      zsc2
               (dp,ar(nr),inout) - complex load vector for frequency domain
                                  - dynamic pressure
      pdvn
               (dp,ar(2),out)
C
                                - wave information held for printout
С
      holdwy
               (dp,ar(60),out)
```

6.5.7. Function userpe (Calculating Rotation Caused by Internal Pressure)

```
*deck,userpe
                                   USERSDISTRIB
      function userpe (prs,rvrp,angle,ex,nuxy)
                       calculate the rotation caused by internal pressure
c primary function:
                       on an elbow element
C
C
                       This function is only called by el18(pipe18)
                                                if keyopt(5) = 1
C
c *** Notice - This file contains ANSYS Confidential information ***
          *** copyright(c) 2008 SAS IP, Inc. All rights reserved.
С
          *** ansys, inc.
C
      typ=int,dp,log,chr,dcp
                               siz=sc,ar(n)
   input arguments:
С
      variable (typ,siz,intent)
                                   description
C
                                 - pressure vector
               (dp,ar(5),in)
C
      prs
С
               (dp,ar(11),in)
                                 - real constants(see elements manual)
      rvrp
C
      angle
               (dp,sc,in)
                                 - subtended angle
                                 - Young's modulus
               (dp,sc,in)
C
      ex
                                 - Poisson's ratio
C
      nuxy
               (dp,sc,in)
С
  output arguments:
С
      variable (typ,siz,intent)
                                   description
                                  - rotation caused by internal pressure on the
C
      userpe (dp,sc,out)
                                       elbow element
С
```

6.5.8. Subroutine UsrViscEl (Performs Viscoelastic Computation)

```
*deck, UsrViscEl
                                     USERSDISTRIB
     subroutine UsrViscEl (veinpt,ncomp,tem,dtem,ex,gxy,eex,egxy,phil,
        zil,gll,g2l,hsm,hbm,smcm,bmcm,epsl,epst,ftl,dftl,tref,tvc,dvc,
        dsig,rsig,cm,kerr,cml)
     ***** perform visco-elastic computation - stiffness pass
C
C
c ***** notice- this routine contains ansys, inc. confidential information ****
C
          *** copyright(c) 2008 SAS IP, Inc. All rights reserved.
С
          *** ansys, inc.
C
C
  input arguments:
       veinpt (dp,ar(95),in)
                                       - viscoelastic input data
С
С
       ncomp (int,sc,in)
                                       - number of components (4 or 6)
               (dp,sc,in)
                                      - temperature
С
                                      - incremental temperature
С
       dtem
              (dp,sc,in)
                                      - bulk modulus at infinite time
              (dp,sc,in)
С
       ex
               (dp,sc,in)
                                      - shear modulus at infinite time
C
       qxy
               (dp,ar(ncomp,ncomp),in) - material matrix for shear modulus
C
       smcm
С
              (dp,ar(ncomp,ncomp),in) - material matrix for bulk modulus
                                   - strain for previous iteration
С
       epsl
              (dp,ar(ncomp),in)
              (dp,ar(ncomp),in)
                                      - total strain for current iteration
C
       epst
С
       tref
              (dp,sc,in)
                                       - reference temperature
C
C
  output arguments:
                                       - effective bulk modulus
С
               (dp,sc,out)
                                       - effective shear modulus
С
       egxy
              (dp,sc,out)
              (dp,sc,inout)
                                      - previous shift factor
С
       phil
С
       zil
              (dp,sc,inout)
                                      - previous pseudo time
                                     - recursive shear relaxation
              (dp,ar(ncomp,10),out)
       gll
C
С
       g21
              (dp,ar(ncomp,10),out)
                                      - recursive bulk relaxation
                                      - recursive shear relaxation
С
       hsm
              (dp,ar(10),out)
       hbm
              (dp,ar(10),out)
                                     - recursive bulk relaxation
C
       ftl
              (dp,sc,inout)
                                      - previous fictive temperature
       dftl
              (dp,ar(10),out)
С
                                      - incremental fictive temperature
С
       tvc
               (dp,sc,inout)
                                       - total volume change
С
       dvc
               (dp,sc,out)
                                      - incremental volume change
                                      - stress change
              (dp,ar(ncomp),out)
С
       dsiq
                                      - stress relaxation
С
       rsig
              (dp,ar(ncomp),out)
С
       cm
               (dp,ar(ncomp,ncomp),out) - total material matrix
C
       kerr
               (int.sc.out)
                                      - error kev
С
  argument of convenience:
              (dp,ar(ncomp,ncomp),none) - no value (used only to avoid simplify
       cml
C
                                          logic due to variable array sizes)
C
С
```

6.5.9. Subroutine usrsurf116 (Modifying SURF151 and SURF152 Film Coefficients and Bulk Temperatures)

```
USERSDISTRIB
*deck,usrsurf116
     subroutine usrsurf116 (elem,ielc,center,jdim,kaxis,time,nr,u,
                                        temvel, hc, tb, key)
                     omeg,ndat,
c *** primary function: change element convection surface info
C
      for surf151 and/or surf152 based on information from fluid116.
      It is called by el151 and el152.
С
C
          in order to activate this user programmable feature,
C
          the user must have used fluid116 with keyopt(2) = 1.
          Further, surf151 and/or surf152 must have keyopt(5) = 1
          (include extra node). Finally, for this routine to do anything,
C
          key(1) and/or key(2) must be reset in this routine to a
          nonzero number. There is no usrcal control over this routine.
С
С
```

```
*** copyright(c) 2008 SAS IP, Inc. All rights reserved.
C
          *** ansys, inc.
C
С
  *** Notice - This file contains ANSYS Confidential information ***
C
С
      input arguments:
С
         variable (typ,siz,intent)
                                       description
                (int,sc,in)
                                     - element number for operation.
С
C
         ielc
                (int,ar(IELCSZ),in) - array of element type characteristics
                                     - coordinates of center of surface element
         center (dp,ar(3),in)
С
С
         idim
                (int,sc,in)
                                     - dimensionality key
                                          1 = 2d
C
                                          2 = axisymmetric
C
С
                                          3 = 3d
С
         kaxis
                (int,sc,in)
                                     - axis of rotation (keyopt(3) for el152)
                                         (see getv116 for definition)
C
                                     - time of current substep
С
         time
                 (dp,sc,in)
C
         nr
                (int,sc,in)
                                     - number of nodal temperatures
                                           of the element
С
С
                 (dp,ar(nr),in)
                                     - vector of most recent values of the
                                        temperatures
С
                (dp,sc,in)
                                     - spin real constant (may be from table)
C
         omeg
С
         ndat
                 (int,sc,in)
                                     - number of data points per element
                (dp,ar(ndat),inout) - film coefficients
         hc
C
С
                                        (has input values for each corner
С
                                        of element)
C
         t.b
                (dp,ar(ndat),inout) - bulk temperature
C
                                        (has input values for each corner
C
                                        of element)
С
      output arguments:
С
С
         variable (typ,siz,intent)
                                       description
С
         temvel (dp,sc,out)
                                     - user defined bulk temperature in excess of
C
                                        fluid node temperature
                 (dp,ar(ndat),inout) - film coefficients
         hc
C
С
                                        (defines input values for each corner
                                        of element)
С
С
         tb
                (dp,ar(ndat),inout) -bulk temperature(includes any modification)
                                        (defines input values for each corner
C
C
                                        of element)
                                     - key if to use this logic
C
         key
                (int,ar(2),out)
                                        key(1) = 0 = no new film coefficient
С
                                        key(1) = 1 = define new film coefficient
C
С
                                        key(2) = 0 = no new bulk temperature
                                        key(2) = 1 = define new bulk temperature
C
                                         (if key(2) = 1, the adiabatic wall
C
                                                   temperature logic is not used).
С
С
          this routine is called during each substep of each load step.
C
          it is called for each equilibrium iteration.
C
          it is called once per element. it is called only during the heat
С
          flow load vector formulation stage, and not during the heat flow
C
          evaluation stage.
C
          the call to get the standard ansys input convection surfaces
C
С
          is made just before entering this routine, so this information is
          available to be modified, if desired.
C
C
          This routine may be thought of as a specialized version of usercv.
С
С
          Indeed, el151 and el152 also call usercv. Either (or both, rarely)
          could be used.
С
С
С
          velocity-dependent film coefficients and bulk temperatures can
С
          be computed by using the velocities and other information from
          fluid116.
C
          Details of this procedure are:
C
             -- SURF151 or SURF152 are 'pasted' onto the actual solid model.
С
             -- flow rate is input to or is computed by FLUID116,
C
C
                  with KEYOPT(2) = 1
             -- flow rate may be a function of time
C
С
             -- the user defines nodes on the FLUID116 network to be the same
                  nodes as the 'extra' nodes of SURF151 or SURF152. If more
C
                  than one FLUID116 element is attached to one of these nodes,
С
C
                  the velocities are averaged.
```

```
c -- SURF151 or SURF152 calls this routine, indirectly, to compute
c the film coefficient and bulk temperature. This routine,
c in turn, gets the average velocity at the 'extra' node
c using 'getv116', as shown below. Other quantities brought
c in by getv116 are also averaged.
```

6.5.10. Subroutine User116Cond (Computing the conductance coefficient for FLUID116)

```
*deck, User116Cond
                                       USERSDISTRIB
     subroutine User116Cond(elem,prop,rvr,aleng,re,fric,uptot,uttot,
    x bco)
c primary function: compute bc for conductance coefficient for fluid116
c *** Notice - This file contains ANSYS Confidential information ***
С
          *** copyright(c) 2008 SAS IP, Inc. All rights reserved.
C
          *** ansys, inc.
C
   input arguments:
                                 - element number
C
     elem
             (int.sc.in)
С
     prop
               (dp,ar(4),in)
                                 - material property vector
С
                                    order is: dens, visc, kxx, c
               (dp,ar(24),in)
                                 - real constant vector
C
     rvr
               (dp,sc,in)
                                 - element length
С
     aleng
С
     re
               (dp,sc,in)
                                 - reynold's number
     fric
               (dp,sc,in)
                                 - friction factor
С
               (dp,ar(2),in
                                - nodal pressure values from previous iteration
     uptot
               (dp,ar(4),in
С
     uttot
                                - nodal temperature values from prev iteration
                                 - the conductance coefficient from TB, fcon
C
     bco
               (dp,sc,inout)
c output arguments:
              (dp,sc,inout)
                                 - the desired conductance coefficient
```

6.5.11. Subroutine User116Hf (Computing the film coefficient for FLUID116)

```
*deck, User116Hf
                                     USERSDISTRIB
      subroutine User116Hf (elem,prop,rvr,aleng,re,uptot,uttot,hf)
c primary function: compute hf for film coefficient for fluid116
c *** Notice - This file contains ANSYS Confidential information ***
С
          *** copyright(c) 2008 SAS IP, Inc. All rights reserved.
C
          *** ansys, inc.
C
   input arguments:
C
С
     elem
               (int,sc,in)
                                 - element number
               (dp,ar(4),inout) - material property vector
С
     prop
                                    order is: dens, visc, kxx, c
C
               (dp,ar(18),in)
                                 - real constant vector
С
C
     aleng
               (dp,sc,in)
                                 - element length
С
               (dp,sc,in)
                                 - reynold's number
     uptot
               (dp,ar(2),in
                                 - nodal pressure values from previous iteration
С
                                 - nodal temperature values from prevs iteration
               (dp,ar(4),in
     ut.t.ot.
C
                                 - the film coefficient from TB, hflm
C
               (dp,sc,inout)
c output arguments:
               (dp,sc,inout)
                                 - the desired film coefficient
```

6.5.12. Subroutine userPartVelAcc (Computing Particle Velocities and Accelerations)

The userPartVelAcc subroutine is the primary component of the API for inputting your own wave and current information. The API supports the hydrodynamic capability available with current-technology pipe elements (PIPE288 and PIPE289). The userPartVelAcc subroutine works with the following subroutines:

- userPartVelAccSetup, which initializes the data for use by userPartVelAcc, and
- userWavHt, which calculates the wave height for a user-defined wave.

For your convenience, three I/O service subroutines are called by the userPartVelAcc subroutine: wvhybl, wvarqu, and wavHtSec, all described below.

```
*deck,userPartVelAcc
                     USERDISTRIB
                                       parallel
     subroutine userPartVelAcc (elemId, secId, xyzg, doIns, depth, denswat,
    x pCur, watcur, nw, watwav, timval,
        argu, eta, vxyz, axyz, ar, pdynam, phead)
    ***** primary function: compute particle velocities and accelerations
С
                            due to waves and current
С
    **** secondary function: compute dynamic pressures
С
C
С
          *** copyright(c) 2008 SAS IP, Inc. All rights reserved.
          *** ansys, inc.
C
 *** Notice - This file contains ANSYS Confidential information ***
C
С
С
  input arguments:
     elemId (int,sc,in)
                                - element id
C
              (int,sc,in)
                                - section id of element
С
     secId
                                - coordinates of point of interest
              (dp,ar(3),in)
C
     xyzg
С
     doIns
              (dp,sc,in)
                                - outside diameter with insulation
С
     depth
              (dp,sc,in)
                                - water depth
С
     denswat (dp,sc,in)
                                - water density
С
     pCur
              (int,sc,in)
                                - pointer in current table
С
                                      i.e. first item is at watcur(pCur+1)
              (dp,ar(*),in)
                                - water current table
С
     watcur
               (dp,sc,in)
C
                                - number of wave components
              (dp,ar(*),in)
С
     watwav
                                - water wave table
              (dp,sc,in)
C
     timval
                                - current time value
С
  output arguments:
С
С
       While the below 7 arguments are output, they can also
С
       be used as input, based on other ANSYS input.
              (dp,sc,out) - position in wave (radians)
С
     arqu
С
               (dp,sc,out)
                                - total wave height
                              - particle velocities
С
     vxyz
              (dp,ar(3),out)
              (dp,ar(3),out) - particle accelerations
С
     axyz
              (dp,sc,out)
                                - radial particle acceleration
С
С
     pdynam
             (dp,sc,out)
                                - dynamic pressure head
              (dp,sc,out)
                                - pressure head
С
     phead
#include "impcom.inc"
#include "ansysdef.inc"
#include "acelcm.inc"
#include "sectionsize.inc"
     --- arguments
     integer
     x elemId, secId, pCur, nw
     double precision
     x xyzg(3), doIns, depth, denswat, watcur(*), watwav(*), timval,
     x argu,eta,vxyz(3),axyz(3),ar,pdynam,phead
      --- local
     integer
     x iw, in, kmf, kpeak
     double precision
```

```
x con, tpt, ao, rkd, asint, eti, etidot, ss, cs, ca, de,
    x vr,etadot,r,cswav(2),tothead,facdep,rk,dperr(2)
      external
                       erhandler, wavHtSec
                       wvargu, wvhybl
      double precision wvargu, wvhybl
      also, remove the following, used only for the ANSYS code analyzer.
C
      in = elemId*secId
      con = watcur(pCur+1) + timval
      The below demonstrates a simple Airy wave sequence.
C
      The current is not demonstrated.
C
C ==========
      watwav(n) = nth location in digested water wav table
C
С
      de = distance from the ocean floor
      de = depth + xyzg(3)
      --- de is sometimes negative for a large deflection problem
C
      --- this is because routine WATERL has to leave a tolerance
      --- for gaps at the mudline
C
      if (de .lt. TINY) de = TINY
      kmf
           = nint (watwav(15))
      kpeak = nint (watwav(13))
      con = watwav(12)*DTORAD
      cswav(1) = cos (con)
      cswav(2) = sin (con)
      r = cswav(1)*xyzg(1) + cswav(2)*xyzg(2)
С
      calculate eta (water height over integration point)
      eta = 0.0d0
      call wavHtSec (0,xyzg(1),doIns,depth,watwav(1),nw, eta,etadot)
      --- eta = total wave height over point being studied
C
С
             = 0.0d0
      vr
             = 0.0d0
      ar
      axyz(3) = 0.0d0
      pdynam = 0.0d0
      ****** small amplitude (airy) wave theory ******
C
      **** compute effect of all components combined, before continuing on.
С
C
      ---facdep is always positive because eta must be less than depth
      facdep = depth/(depth + eta)
      **** now continue on.
C
      --- loop on waves
C
      do iw = 1, nw
         in = ptr\_Layer + 6*(iw-1) + 1
         --- compute water height due to only this wave (eti)
C
         rk = DPPI2/watwav(in+3)
         tpt = DPPI2/watwav(in+1)
         argu = wvargu (kpeak,kmf,watwav(in),r,doIns)
         ao = watwav(in)*0.5d0
         eti = ao*cos (argu)
         asint = ao*sin (argu)
         etidot = tpt*asint
         rkd = rk*depth
         ca = rk*de
         *** cs = cosh(ca)/sinh(rkd)
C
         cs = wvhybl (2, ca, rkd)
         *** ss = sinh(ca)/sinh(rkd)
C
         ss = wvhybl (3,ca,rkd)
С
         --- compute dynamic pressure
         ---- accel(3) is from the acelcm include deck and is the verical acceleration.
C
         pdynam = pdynam + denswat*accel(3)*
                          eti*wvhybl (0,rk*de*facdep,rkd)
                                  *** (cosh(rk*de*facdep)/cosh(rkd)
С
         **** compute particle velocities
C
         **** vr
С
                     = radial velocity
С
         **** vxyz(3) = vertical velocity
                           + cs*tpt*eti
         vr
                 = vr
         vxyz(3) = vxyz(3) + ss*etidot
         *** compute particle accelerations
С
```

```
--- first compute derivatives of cs and ss
                          + tpt*cs*( etidot)
                = ar
         axyz(3) = axyz(3) + tpt*ss*(-eti*tpt)
      enddo
      phead = pdynam/(accel(3)*denswat)
      tothead = phead - xyzg(3)
      if (tothead .lt. 0.0d0) then
         dperr(1) = tothead
         dperr(2) = xyzg(3)
         call erhandler ('userPartVelAcc',5000,2,
          'The ocean wave theory gives a negative water pressure head
         (%G) at vertical coordinate %G under the wave crest. This
         may be because the input wave length is too small.',
          dperr(1),' ')
      endif
      compute x and y components of radial velocity and acceleration
      vxyz(1) = vr*cswav(1)
      vxyz(2) = vr*cswav(2)
      axyz(1) = ar*cswav(1)
      axyz(2) = ar*cswav(2)
C
      adjust for currents here
      return
      end
```

6.5.12.1. Subroutine userPartVelAccSetup (Initializing Data for Use by the user-PartVelAcc Subroutine)

This subroutine initializes the data for the userPartVelAcc subroutine.

```
USERDISTRIB
                                            parallel
*deck,userPartVelAccSetup
      subroutine userPartVelAccSetup (secId, secIdCurr, secIdWave,
     x
                                      kch.
                                      nsize, nsizec, nsizew,
    х
                                      dWork,dWorkC,dWorkW,
                                      rkd, wvmax)
c *** primary function: set up and checking of user wave (and current) theory)
c *** secondary functions: none
c *** Notice - This file contains ANSYS Confidential information ***
C
С
  input arguments:
С
                                   - ocean basic data id
С
     secId
              (int,sc,in)
С
     secIdCurr(int,sc,in)
                                   - ocean current data id
C
     secIdWave(int,sc,in)
                                   - ocean wave data id
                                   - key for checking or defaulting (not used by PIPE288)
С
              (int,sc,in)
     nsize
              (int,sc,in)
                                   - size of ocean basic data
C
С
     nsizec
              (int,sc,in)
                                   - size of ocean current data
С
     nsizew
              (int,sc,in)
                                   - size of ocean wave data
              (dp,ar(*),inout)
                                   - ocean basic data
С
     dWork
              (dp,ar(*),inout)
     dWorkC
                                   - ocean current data
С
     dWorkW (dp,ar(*),inout)
С
                                   - ocean wave data
С
С
  output arguments:
              (dp,ar(*),inout)
С
      dWork
                                   - ocean basic data
               (dp,ar(*),inout)
      dWorkC
                                  - ocean current data
C
      dWorkW
              (dp,ar(*),inout)
                                   - ocean wave data
C
               (dp,sc,out)
     rkd
                                   - value of k*d
C
      wvmax
              (dp,sc,out)
                                    - total amplitude
#include "impcom.inc"
#include "ansysdef.inc"
#include "acelcm.inc"
#include "sectionsize.inc"
#include "soptcm.inc"
```

```
external erhandler
      integer secId,secIdCurr,secIdWave,kch,
     x nsize, nsizec, nsizew,
                                 iw,in,i,nw,kwav,kcrc,kmf
      double precision dWork(*),dWorkC(*),dWorkW(*),rkd,wvmax,
     x rko,rkdnew,ch,th,dwwl,tpd, depth,dperr(4)
      dperr(1) = secIdWave
С
      depth = dWork(11)
                                   ! depth
           = nint (dWorkW(6))
                                   ! number of wave components
      kwav = nint (dWorkW(11))
                                   ! type of wave
      kcrc = nint (dWorkW(14))
                                   ! how to combine wave and current
      kmf = nint (dWorkW(15))
                                   ! MacCamy-Fuchs flag
      --- dummy logic
      th = dWorkC(1)
      iw = kcrc+kmf+kch+kwav+nsize+nsizec+nsizew+secId+secIdCurr
      tpd = DPPI2*depth
      wvmax = 0.0d0
      do iw = 1.nw
         dperr(2) = iw
         in = ptr\_Layer + 6*(iw-1)
C
С
         *** compute wave length from other parameters
         dwwl = acend(3)*dWork(in+2)**2/DPPI2
         dwwl = deep water wave length
С
С
         if (dWork(in+4) .eq. 0.0d0) then
            ***** compute wave length if needed *****
C
            rko = tpd/dwwl
            rkd = rko
            *** test to see if deep water estimate is correct
С
            if (rko .lt. DPPI) then
               do i = 1,100
                  th = tanh (rkd)
                  ch = cosh (rkd)
                  rkdnew = rkd + (rko - rkd*th)/(rkd/ch**2 + th)
                  if (abs (rkd-rkdnew) .le. 1.d-7) go to 500
                  rkd = rkdnew
               enddo
               call erhandler ('userPartVelAccSetup',5060,3,
                'The following wave length has not converged after 100
     x
                iterations.',0.0d0,' ')
    х
С
500
               rkd = rkdnew
            endif
            dWork(in+4) = tpd/rkd
            if (dWork(16) .eq. 1) then
               dperr(3) = dwwl
               dperr(4) = dWork(in+4)
               call erhandler ('userPartVelAccSetup',5070,1,
                'Ocean ID %I, wave number %I (OCTABLE command) has:
    х
                응 /
                       deep water wave length = %G
                %/ small amplitude wave length = %G
                %/ Called from userPartVelAccSetup.F',
                                                        dperr(1),' ')
            endif
         endif
         if (antype .ne. 3) then
            *** previous th not usable if wavelength input directly
C
            rkd = tpd/dWork(in+4)
         **** compute total wave height for later checking ****
C
         wvmax = wvmax + abs (dWork(in+1))
      enddo
C
      return
```

end

6.5.12.2. Subroutine userWavHt

The userWavHt subroutine calculates the wave height of a user-defined wave for the userPartVelAcc subroutine.

```
*deck,userWavHt
                               parallel
      subroutine userWavHt (ktyp,xyzg,doext,depth,watwav,nw, eta,etadot)
      *** primary function: calculate wave height for user wave (kwav > 9)
C
С
                                 over point at xyzg of the element
      *** secondary functions: none
С
C
  *** Notice - This file contains ANSYS Confidential information ***
С
C
   input arguments:
C
С
      ktyp
               (int.sc.in)
                                    - key what argument is to be used
С
                                         0 - compute value in wvargu
                                         1 - pass directly in doext position
С
С
                                           ktyp = 1 used for stream function only
                                    - updated coordinates of point of interest in
C
      xvzq
               (dp,ar(3),in)
С
      doext
               (dp,sc,in)
                                    - outside diameter with insulation
С
                                       if ktyp = 1, doext = argu
      dept.h
               (dp,sc,in)
                                    - water depth
C
С
               (dp,ar(*),in)
                                    - water wav table
      watwav
               (int,sc,in)
                                     - number of waves
С
      ทพ
C
   output arguments:
С
С
      eta
               (dp,sc,out)
                                    - wave height (initialized before entering
С
                                                this routine)
С
      etadot
               (dp,sc,out)
                                    - time derivative of wave height
#include "impcom.inc"
#include "ansysdef.inc"
#include "sectionsize.inc"
      external wvargu
C
      integer ktyp,nw,
                         iw,in,kwav,kmf,kpeak
      double precision wvargu,
     x xyzg(3), doext, depth, watwav(*), eta, etadot,
     x r,con,ao,argu,tpt
      iw = ktyp
      depth = depth
      **** user programmable wave height (like airy)
C
      con = DTORAD*watwav(12)
      r = cos (con)*xyzg(1) + sin (con)*xyzg(2)
      kwav = nint (watwav(11)) ! must be 10 or greater
      kwav = kwav
      kmf = nint (watwav(15))
      kpeak = nint (watwav(13))
      etadot = 0.0d0
      do iw = 1,nw
         in = ptr\_Layer + 4*(iw-1) + 1
         argu = wvargu (kpeak,kmf,watwav(in),r,doext)
         ao = watwav(in)*0.5d0
         eta = eta + ao*cos (argu)
         tpt = DPPI2/watwav(in+1)
         etadot = etadot + ao*tpt*sin (argu)
      enddo
      return
      end
```

6.5.12.3. Subroutine wvhybl

The wvhybl subroutine computes the ratio of two hyperbolic functions and is intended for use with wave loading. It is a utility subroutine called by the userPartVelAcc subroutine.

```
*deck,wvhybl
                parallel
      function wvhybl (kclass,x,y)
c *** primary function: to compute the ratio of two hyperbolic functions,
С
                          specialized to the needs of wave loading.
                          The options are as given with kclass below.
C
С
                          Further, only positive values of x and y are used
C
c *** secondary functions: none
С
c *** Notice - This file contains ANSYS Confidential information ***
С
   input arguments:
     variable (typ,siz,intent)
                                  description
C
                                 - 0 - \cosh(x)/\cosh(y)
С
     kclass (int,sc,in)
                                 -1 - sinh(x)/cosh(y)
С
                                 -2 - cosh(x)/sinh(y)
C
                                 -3 - \sinh(x)/\sinh(y)
С
               (dp,sc,in)
С
     x
                                 - argument of numerator
               (dp,sc,in)
                                 - argument of denominator
C
С
С
  output arguments:
     variable (typ, siz, intent) description
C
      wvhybl (dp,sc,out)
                                 - resulting fraction
```

6.5.12.4. Subroutine wvargu

The wvargu subroutine computes the appropriate position with regard to the wave. It is a utility subroutine called by the userPartVelAcc subroutine.

```
*deck, wvargu
                             parallel
      function wvargu (kpeak,kmf,wavdat,r,doext)
      *** primary function: to find appropriate position wrt wave
C
      *** secondary functions: none
С
c *** Notice - This file contains ANSYS Confidential information ***
С
С
   input arguments:
C
     kpeak
             (int.sc.in)
                                 - keyopt for when peak effect occurs
     kmf
                                 - key for maccamy-fuchs adjustment
С
               (int,sc,in)
С
     wavdat.
              (dp,ar(6),in)
                                 - wave data (from water wave table)
                                        wavdat(1) = wave amplitude(not used)
C
                                        wavdat(2) = period
С
                                        wavdat(3) = phase shift
C
С
                                        wavdat(4) = wave length
С
               (dp,sc,in)
                                 - radial location of point of interest
                                 - effective outside diameter of pipe
               (dp,sc,in)
C
      doext
С
С
   output arguments:
                                 - wave position(as determined by the argument)
C
     wvargu (dp,sc,out)
                                      output in radians
```

6.5.12.5. Subroutine wavHtSec

The wavHtSec subroutine calculates wave height over point at xyzg of the element. It is a utility subroutine called by the userPartVelAcc subroutine.

```
*deck,wavHtSec
                            parallel
                                                                        pck
      subroutine wavHtSec (ktyp,xyzg,doext,depth,watwav,nw, eta,etadot)
      --- primary function: Calculate wave height
С
                                over point at xyzg of the element.
С
С
                            If KWAVE = 10, this routine calls userWavHt.
      --- secondary functions: none
С
C
c *** Notice - This file contains ANSYS Confidential information ***
С
  input arguments:
С
     ktyp (int,sc,in)
                                   - key what argument is to be used
C
С
                                        0 - compute value in wvargu
С
                                        1 - pass directly in doext position
C
                                          ktyp = 1 used for stream function only
               (dp,ar(3),in)
                                   - updated coordinates of point of interest in
С
      xyzg
С
      doext
               (dp,sc,in)
                                   - outside diameter with insulation
                                     if ktyp = 1, doext = argu
С
С
      depth
               (dp,sc,in)
                                   - water depth
               (dp,ar(*),in)
С
      watwav
                                   - water wav table
С
               (int,sc,in)
                                   - number of waves
С
c output arguments:
                                   - wave height (initialized before entering he
С
     eta
             (dp,sc,out)
С
      etadot (dp,sc,out)
                                  - time derivative of wave height
C
                                               (used for kwav=1(wheeler) only)
```

6.6. Running ANSYS as a Subroutine

To call the ANSYS program, use the following:

```
program ansys
```

If you are running ANSYS on a UNIX system (but not under Windows), you also can call the ANSYS program as a subroutine in a program you've written. To do so, use the following:

```
subroutine ansys
```

For multiple calls to subroutine ANSYS, you must open and close standard input in the calling subroutine. (Usually, input and output are Fortran units 5 and 6, respectively.) The calling subroutine can't use the database access subroutines. However, other user programmable features can use the database access subroutines freely.

There may be times when ANSYS exits abnormally. Check the file.err file to see if ANSYS wrote an exit code to the file before ending. These error codes may help you to understand the problem ANSYS had:

Table 6.1 ANSYS Exit Codes

Code	Explanation	Code	Explanation
0	Normal Exit	14	XOX Error
1	Stack Error	15	Fatal Error
2	Stack Error	16	Possible Full Disk
3	Stack Error	17	Possible Corrupted or Missing File
4	Stack Error	18	Possible Corrupted DB File
5	Command Line Argument Error	21	Authorized Code Section Entered

Code	Explanation	Code	Explanation
6	Accounting File Error	25	Unable to Open X11 Server
7	Auth File Verification Error	30	Quit Signal
8	Error in ANSYS or End-of-run	31	Failure to Get Signal
11	User Routine Error	>32	System-dependent Error
12	Macro STOP Command		

6.7. Defining Your Own Commands

ANSYS, Inc. supplies a set of user subroutines, named user01 through user10, which you can use to define custom ANSYS commands. To do so, follow these steps:

- 1. Insert the code for the functions you want to perform into subroutine user01 (or user02, etc.).
- 2. Link the subroutine into the ANSYS program.
- 3. Issue the ANSYS command /**UCMD** to define a name for a custom command that calls and executes your subroutine. Use the command format shown below:

```
/UCMD, Cmd, SRNUM
```

Cmd

The name for your new command. It can contain any number of characters, but only the first four are significant. The name you specify can not conflict with the name of any ANSYS command or the names of any other commands or macros.

SRNUM

The number of the subroutine your command should call; that is, a value between 01 and 10. For example, suppose that you create and link in a user subroutine for a parabolic distribution of pressure, and you name that subroutine user01. Issuing the command shown below creates a new command, PARB, that when issued calls your parabolic pressure distribution subroutine:

```
/UCMD,PARB,1
```

To make these "custom command" subroutines available in all your ANSYS sessions, include the /**UCMD** commands in your start-up file (START.ANS).

You also can use /UCMD to remove a custom command. To do so, simply use a blank value for Cmd, as shown below:

```
/UCMD,,1
```

This command removes the PARB command. To list all user-defined command names, issue the command /UCMD,STAT.

6.7.1. Function user01

```
| special coding. accesss to this routine is by the command usr1.
    usrl may be changed by the user using the command /ucmd. the
С
   user may then use this routine to call his/her special routines.
   | ansys routines to access information in the ansys database may be
С
    found in the "ansys programmer's manual", available from ansys, inc
C
    see user02 for a simple example usage.
   routines user03 to user10 are also available.
   С
  input arguments: none
  output arguments:
C
     user01 (int,sc,out)
                                - result code (should be zero)
С
С
                                   (which is ignored for now)
С
    Functions for accessing data on the command line
C
     integer function intinfun(iField) - gets an integer from field iField
C
     double precision function dpinfun(iField) - gets double precision
     character*4 ch4infun(iField) - gets (upper case) 4 characters
C
     character*8 ch8infun(iField) - gets (mixed case) 8 characters
C
     character*32 ch32infun(iField) - gets (mixed case) 32 characters
С
C
#include "impcom.inc"
#include "ansysdef.inc"
      external wringr
      integer wringr
      integer user01, iott
      iott = wrinqr(2)
          ***** USER'S CODE IS INSERTED HERE *****
      write (iott,2000)
 2000 format (//' **** CALL TO ANSYS, INC DUMMY USER01 *****'//)
          \ensuremath{^{\star\star\star\star\star}} do not return this result code in a real user routine
      user01 = -654321
         ***** instead return a zero *****
     user01 = 0
C
      return
      end
```

6.7.2. Function user02 (Demonstrates Offsetting Selected Nodes)

```
*deck,user02
                   user
    function user02()
c *** primary function:
                     user routine number 02
     --- This demonstration offsets selected nodes with the command:
С
            usr2,dx,dy,dz
        *** copyright(c) 2006 SAS IP, Inc. All rights reserved.
С
        *** ansys, inc.
c *** Notice - This file contains ANSYS Confidential information ***
C /*******************
  | see user01 for additional information on user routines
C
  input arguments: none
C
  output arguments:
    user02 (int,sc,out)
                          - result code (should be zero)
С
                             (which is ignored for now)
```

```
Functions for accessing data on the command line
C
    integer function intinfun(iField) - gets an integer from field iField
С
С
    double precision function dpinfun(iField) - gets double precision
    character*4 ch4infun(iField) - gets (upper case) 4 characters
С
    character*8 ch8infun(iField) - gets (mixed case) 8 characters
С
    character*32 ch32infun(iField) - gets (mixed case) 32 characters
     ************
#include "impcom.inc"
#include "ansysdef.inc"
     external TrackBegin, TrackEnd
     external wringr,ndingr,ndgxyz,ndpxyz,erhandler, dpinfun
     integer wringr,ndingr,ndgxyz
     double precision dpinfun
     integer user02, iott, maxnp, i ,ksel
     double precision xyz(3), offset(3)
     ***** start timing check *****
C
     call TrackBegin ('user02')
     maxnp = ndingr(0,DB_MAXDEFINED)
          ***** get the desired offsets from the command line *****
C
     offset(1) = dpinfun(2)
     offset(2) = dpinfun(3)
     offset(3) = dpinfun(4)
     do i = 1, maxnp
         ksel = ndgxyz (i,xyz(1))
         if (ksel .eq. 1) then
            xyz(1) = xyz(1) + offset(1)
            xyz(2) = xyz(2) + offset(2)
            xyz(3) = xyz(3) + offset(3)
            call ndpxyz (i,xyz(1))
         endif
     enddo
          ***** write to output file *****
     iott = wrinqr(WR_OUTPUT)
     write (iott, 2000)
2000 format (/' NODE OFFSET COMPLETE '/)
         ***** write to GUI window *****
     call erhandler ('user02',3000,
                  2, 'NODE OFFSET COMPLETE', 0.0d0,'')
          **** required return value ****
     user02 = 0
          **** end timing check ****
C
     call TrackEnd ('user02')
     return
     end
```

6.7.3. Function user03 (Demonstrates Using ANSYS Memory)

```
************
   | see user01 for additional information on user routines
  input arguments: none
  output arguments:
C
С
     user03 (int,sc,out)
                                - result code (should be zero)
                                   (which is ignored for now)
C
     ****************
С
     Functions for accessing data on the command line
С
     integer function intinfun(iField) - gets an integer from field iField
С
С
     double precision function dpinfun(iField) - gets double precision
    character*4 ch4infun(iField) - gets (upper case) 4 characters
character*8 ch8infun(iField) - gets (mixed case) 8 characters
C
С
     character*32 ch32infun(iField) - gets (mixed case) 32 characters
C
     ************
#include "impcom.inc"
#include "ansysdef.inc"
      external TrackBegin, TrackEnd
     external wringr, ndingr, ndgxyz, ndnext, fAnsMemAlloc,
               fAnsMemFree, erhandler, parreturn, parstatus
               wringr, ndingr, ndgxyz, ndnext
      integer
      PTRFTN
               fAnsMemAlloc
               user03, iott, i, ksel, numnp, node, istat
     double precision xyz(3), xmean, ymean, zmean, stdxyz(3),
               sodx, sody, sodz
c pointers:
     pointer (pdXnodeL,Xnode)
     pointer (pdYnodeL,Ynode)
     pointer (pdZnodeL,Znode)
     double precision Xnode(*), Ynode(*), Znode(*)
        ***** call to start timing *****
C
      call TrackBegin ('user03')
     Get nodal xyz locations and calculate standard deviation of
     x coordinates, y coordinates, & z coordinates
С
      get number of currently selected nodes
     numnp = ndingr(0,DB_NUMSELECTED)
      istat = 1
      if (numnp .le. 0) go to 999
      allocate memory for x, y, & z coordinates of nodes
C
      pdXnodeL = fAnsMemAlloc(numnp,MEM_DOUBLE,'XCoords ')
      pdYnodeL = fAnsMemAlloc(numnp,MEM_DOUBLE,'YCoords ')
     pdZnodeL = fAnsMemAlloc(numnp,MEM_DOUBLE,'ZCoords ')
     loop through all selected nodes
      i = 1
     node = 0
     xmean = 0.0d0
     ymean = 0.0d0
      zmean = 0.0d0
     node = ndnext(node)
      if (node .gt. 0) then
          get xyz coordinates
С
         ksel = ndgxyz(node,xyz(1))
         store this node's xyz coordinates
```

```
Xnode(i) = xyz(1)
         Ynode(i) = xyz(2)
         Znode(i) = xyz(3)
         while we're looping, accumulate sums to calculate means
С
         xmean = xmean + xyz(1)
         ymean = ymean + xyz(2)
          zmean = zmean + xyz(3)
         increment index
C
          i = i + 1
         loop back up for next selected node
C
         goto 10
      endif
     node = 0, at the end of node list
С
      calculate mean of xyz coordinates
C
      xmean = xmean / numnp
      ymean = ymean / numnp
      zmean = zmean / numnp
      calculate standard deviation for xyz coordinates
      sodx = 0
      sody = 0
      sodz = 0
      do i = 1, numnp
         sodx = sodx + (Xnode(i) - xmean)**2
         sody = sody + (Ynode(i) - ymean)**2
         sodz = sodz + (Znode(i) - zmean)**2
      enddo
      stdxyz(1) = sqrt(sodx / (numnp-1))
      stdxyz(2) = sqrt(sody / (numnp-1))
      stdxyz(3) = sqrt(sodz / (numnp-1))
      ***** write to output file *****
      iott = wringr(WR_OUTPUT)
      write (iott,2000) xmean,ymean,zmean,
    x
                       stdxyz(1),stdxyz(2),stdxyz(3)
2000 format (/' MEAN FOR X COORDINATES:',G12.5/
              ' MEAN FOR Y COORDINATES: ',G12.5/
               ' MEAN FOR Z COORDINATES: ',G12.5/
               ' STD FOR X COORDINATES: ',G12.5/
               ' STD FOR Y COORDINATES: ',G12.5/
               ' STD FOR Z COORDINATES: ',G12.5)
     ***** write to GUI window *****
     call erhandler ('user03',5000,2,
     x 'STD FOR X COORDINATES: %G %/
     x STD FOR Y COORDINATES: %G %/
     x STD FOR Z COORDINATES: %G', stdxyz(1),'')
      ***** set _STATUS to 0 for success *****
      istat = 0
     release dynamically allocated memory
C
      call fAnsMemFree (pdZnodeL)
      call fAnsMemFree (pdYnodeL)
      call fAnsMemFree (pdXnodeL)
      **** required return value *****
999
     user03 = 0
      ***** set _RETURN to number of nodes processed *****
      call parreturn (dble(numnp))
      ***** set _STATUS for success (0) or no nodes (1) *****
C
      call parstatus (istat)
```

```
c ***** call to end timing *****
  call TrackEnd ('user03')
  return
  cond
```

6.7.4. Function user04

```
*deck,user04
                       user
     function user04()
c *** primary function: user routine number 04; demonstrates gettting a
С
                         list of nodes attached to a keypoint, line, or area
          *** copyright(c) 2006 SAS IP, Inc. All rights reserved.
C
         *** ansys, inc.
c *** Notice - This file contains ANSYS Confidential information ***
  /**********************
   | see user01 for additional information on user routines
c input arguments: none
  output arguments:
                               - result code (should be zero)
C
     user04 (int,sc,out)
                                   (which is ignored for now)
С
C
     Functions for accessing data on the command line
С
     integer function intinfun(iField) - gets an integer from field iField
     double precision function dpinfun(iField) - gets double precision
С
     character*4 ch4infun(iField) - gets (upper case) 4 characters
    character*8 ch8infun(iField) - gets (mixed case) 8 characters
C
     character*32 ch32infun(iField) - gets (mixed case) 32 characters
С
#include "impcom.inc"
#include "ansysdef.inc"
      external wringr, ndkpnt, ndline, ndarea, ch4infun, intinfun
      integer wringr, ndkpnt, ndline, ndarea, intinfun
      character*4 ch4infun
     integer user04, iott, listk(20),listl(20),lista(20),
               i, num,ktype,
                                  nkpnts, nlines, nareas
     character*4 type, lab2
     iott = wringr (WR_OUTPUT)
С
      --- setup with: /UCMD,GNSME,4
C
        !gnsme,group,num,type
         ! group = kp, ln, or ar
С
        ! num = entity number of kp, ln, or ar
С
        ! type = interior, or all
     lab2 = ch4infun(2)
      write (iott,2010) lab2
 2010 format(/' group name (type of entity) = ',a4)
     num = intinfun(3)
     write (iott, 2020) num
 2020 format (' entity number =',i4)
      if (lab2 .ne. 'KP ' ) then
         type = ch4infun(4)
         if (type .eq. 'INTE') then
           write (iott,2030)
```

```
format (' interior nodes only ')
2030
           ktype = 0
        elseif (type .eq. 'ALL ') then
           write (iott, 2040)
2040
           format (' all (interior and edge/end) nodes ')
           write (iott, 2050)
2050
           format ('Only INTE or ALL are acceptable in last field',
            ' on user-written gnsme command')
        endif
     endif
     if (lab2 .eq. 'KP ' ) then
        nkpnts = ndkpnt (num, listk(1))
        write (iott,2110) nkpnts
2110
        format (' number of nodes on keypoint = ',i4)
        write (iott,2115) (listk(i),i=1,nkpnts)
2115
        format (' node on keypoint = ',i4)
     elseif (lab2 .eq. 'LN ' ) then
        nlines = ndline (num,ktype,listl(1))
        write (iott,2120) nlines
2120
        format (' number of nodes on line = ',i4)
        write (iott,2125) (listl(i),i=1,nlines)
        format (' list of nodes on line'/(3x,i4))
2125
     elseif (lab2 .eq. 'AR ' ) then
        nareas = ndarea (num,ktype,lista(1))
        write (iott,2130) nareas
        format (' number of nodes on area = ',i4)
2130
        write (iott,2135) (lista(i),i=1,nareas)
2135
        format (' list of nodes on area'/(3x,i4))
     else
        write (iott,2150)
2150
        format (' Only KP, LN, or AR are acceptable on user-written ',
        'gnsme command')
     endif
     user04 = 0
     return
     end
```

6.7.5. Functions user05 through user10

The source code for user subroutines user05, user06, user07, user08, user09, and user10 is identical to function user01 shown above.

6.8. Supporting Subroutines

The following routines may be used for general applications.

6.8.1. Function GetRForce (Getting Nodal Reaction Force values)

```
*deck, GetRForce
      function GetRForce (Node, Label, Value)
c primary function:
                       Get the K * u - F at a node from the rfsum vector.
С
                        warning: This routine is called after the elements
                        are formed, but before solution. Therefore,
C
                        F is from the current iteration, but
                        u is from the previous iteration. At convergence,
C
С
                        this difference will have little effect.
С
                        The computations are done immediately after the
                        call to UElMatx.
C
```

```
Use the RFSUM command to ask for the summation.
C
                 Use *GET,Parm,NODE,num,RF,DOFLAB to access the reaction
C
С
                          sum from the command line.
c secondary functions: Return pointer for fast access
c object/library: usr
c *** Notice - This file contains ANSYS Confidential information ***
      Prolog is not CONFIDENTIAL INFORMATION
С
   input arguments:
C
      variable (typ,siz,intent)
                                   description
С
               (int,sc,in)
                                 - Node Number (User)
С
      Label
               (ch*4,sc,in)
                                 - DOF Label (Upper Case)
                                    'UX ','UY ','TEMP','VOLT','ROTY', etc
C
С
   output arguments:
      GetRForce (int,func,out)
                                 - status/pointer
C
C
                                    = 0 - data not valid
                                    > 0 - Rfsum pointer to data for fast access
C
                                          see comments below
С
С
      Value
               (dp,sc,out)
                                 - Solution value for Node, Label
                                    All results are in the nodal coordinate
C
С
                                    system
   example usage:
C
C
        external GetRForce
        integer GetRForce, ptr, Node2
C
        double precision Value
c #include "handlecom.inc" (if Value = Rfsum(ptr) form is to be used)
C
        ptr = GetRForce (Node2,'UY ',Value)
C
      later...
        Value = Rfsum(ptr)
```

6.8.2. Function GetStackDisp (Getting Current Displacement Values)

```
*deck,GetStackDisp
      function GetStackDisp (Node,Label,Value)
c primary function:
                       Get the displacement at a node from the disp vector
c secondary functions: Return pointer for fast access
c object/library: usr
c *** Notice - This file contains ANSYS Confidential information ***
      Prolog is not CONFIDENTIAL INFORMATION
      typ=int,dp,log,chr,dcp siz=sc,ar(n) intent=in,out,inout
   input arguments:
C
      variable (typ,siz,intent)
                                   description
C
                                 - Node Number (User)
      Node
              (int,sc,in)
C
С
      Label
               (ch*4,sc,in)
                                 - DOF Label (Upper Case)
C
                                   'UX ','UY ','TEMP','VOLT','ROTY', etc
С
   output arguments:
С
      variable (typ,siz,intent)
                                   description
С
      GetStackDisp (int,sc,out) - status/pointer
С
                                   = 0 - data not valid
                                   > 0 - UDisp pointer to data for fast access
C
                                         see comments below
С
      Value
               (dp,sc,out)
                                 - Solution value for Node, Label
c example usage:
```

```
c external GetStackDisp
c#include "handlecom.inc" (only if UDisp(ptr) form is used
c integer GetStackDisp, ptr, Node2
c double precision Value

c ptr = GetStackDisp (Node2,'UY ',Value)

c later...
c Value = UDisp(ptr)
```

6.8.3. Subroutine ElResultStrt (Getting Load Data from Analysis Results)

```
*deck,ElResultStrt
     subroutine ElResultStrt (Label,Comp,LabAvg,TypeData,nVal,iLoc)
c *** primary function: (post1) Load data for later ElResultGet
c *** Notice - This file contains ANSYS Confidential information ***
              (prolog is not confidential)
С
  input arguments:
                                Result TypeResult Component (8 char for ESTR)
    Label (ch*4,sc,in)
C
     Comp
              (ch*4,sc,in)
С
     LabAvg (ch*4,sc,in)
                                   - 'AVG ' or 'NOAV' ('AVG ' default)
С
С
  output arguments:
     TypeData (int,sc,out)
                                   - Code for data type
C
                                   - Number of values per point
     nVal (int,sc,out)
C
С
                                     If 0, no data
                                     - Location of Comp in values
C
     iLoc
              (int,sc,out)
```

6.8.4. Subroutine ElResultGet (Getting Results Values at Selected Points)

```
*deck,ElResultGet
     subroutine ElResultGet (nPoints, ebest, elcord, TypeData, iLoc,
    x
                           nVal,result)
                         (post1) Get results at selected points
c *** primary function:
c *** Notice - This file contains ANSYS Confidential information ***
С
               (prolog is not confidential)
  input arguments:
С
                                    - Number of evaluation points
С
     nPoints (int,sc,in)
                     *** from ElInterp ***
C
С
     ebest (int,ar(nPoints),in) - Element(s) containing points
     elcord (dp,ar(3,nPoints),in) - Element coordinates
С
                     *** from ElResultStrt ***
C
                             - Data type code
- Start of selected data
С
     TypeData (int,sc,in)
С
     iLoc
               (int,sc,in)
                                    - Number of results per point
     nVal
              (int.sc.in)
C
c output arguments:
     Result (dp,ar(nvar,nPoints),out) - Array of results
```

6.8.5. Subroutine ElInterp (Finding Element Coordinates)

```
c primary function:
                      Find element numbers containing xyz points
c secondary functions: Find element coordinates of these points
c object/library: upf
c *** Notice - This file contains ANSYS Confidential information ***
     (Prolog is not CONFIDENTIAL INFORMATION)
c input arguments:
   piFEML (ptr,sc,in)
                                 - If non 0, pointer of a FEM Object
   nPoints
                                - Number of points to find (do in one group)
            (int,sc,in)
   xyzPoints(dp,ar(3,nPoints),in) - XYZ coordinates of each point
C
  tolInsidein(dp,sc,in)
                                - Tolerance for point inside element
C
С
                                     (0.0d0 defaults to 1.0d-4)
   tolOutsidein(dp,sc,in)
                                 - Maximum distance outside to be associated
C
С
                                     with an element (0.0d0 defaults to 0.25)
C
   MoveTol
            (dp,sc,in)
                                 - Node move tolerance (0.0d0, no move)
c output arguments:
            (int,ar(nPoints),out) - Best element number for each point
c ebest
            (dp,ar(3,nPoints),out) - Element coordinates of the point
   elcord
```

6.9. Access at the Beginning and End of Various Operations

You can access the logic just before an ANSYS run begins or just after a run ends, and at many other intermediate points, by using the ANSYS subroutines listed below. These subroutines can perform actions such as evaluating results or performing calculations. (None of the subroutines have input or output arguments.)

Issue the **USRCAL** command (or use an equivalent menu path) to activate or deactivate these subroutines.

User Subroutine	Is Called
UAnBeg	At ANSYS start-up
USolBeg	Before solution
ULdBeg	Before a load step
USsBeg	Before a substep
UItBeg	Before an iteration
UItFin	After an iteration
USsFin	After a substep
ULdFin	After a load step
USolFin	After solution
UAnFin	At the end of an ANSYS run

Subroutines USSBeg, UItBeg, UItFin and USSFin default to reading a command macro file from the current working directory whose name is subroutine.mac (that is, ussfin.mac is read by USSFin.F). No user action to relink the ANSYS program is required for the command macro to be read except that the calling subroutine must be activated by the USRCAL command. The design of the command reading ability of these subroutines is limited to APDL parameter setting commands (*GET, *SET, a = value, etc) and testing for general ANSYS commands is limited. Commands which are known to work include *DIM, *STATUS.

Commands which require another line (*MSG, *VWRITE) are not allowed. Other commands which are known to not work are the solution loading commands (D, F, SFE, and so on). If these capabilties are required, the user will need to create a Fortran subroutine and link this subroutine into ANSYS, as described in *Chapter 5*, *Using User Programmable Features (UPFs)* (p. 99).

While parameter substitution into commands is not permitted, USSBeg, and so on were designed to be used in conjunction with dynamic tables and parameter substitution from the user subroutine. As an example, consider an ANSYS table defined as d5 = f(par1), If d5 contains values of displacement as a function of PAR1, then d5 may be used as a constraint, as

```
*dim,d5,table,10,1,1,PAR1
d5(1)=0,.1,.25,
/solu
d,5,ux,%d5%
```

Modify the value of PAR1 in USSBeg. MAC and the constraint on node 5, ux can then be modified in the middle of a load step.

The following is an example of a valid input that may be read by USSBeg, UItBeg, UItFin and USSFin.

```
/COM, SAMPLE ussfin.mac
a=5
b=nx(1)
                                    ! *get function is ok
*get,c,active,solu,Time,cpu
                                ! *get is ok
*dim,array,,6
                                  ! array parameters are ok
arrav(1) = 1
array(2) = 2
array(3) = 3
arrav(4) = 4
array(5) = 5
array(6) = 6
*vleng,3
                                     ! vector operations are ok
*vfun,array(4),copy,array(1)
*stat
*stat,array(1)
array(1) =
nnode = ndinqr(0,14)
*dim,array,,nnode
*vget,array(1),NODE,1,NSEL
*stat,array(1)
array(1) =
/eof
/COM, COMMANDS BELOW THIS LINE ARE KNOWN TO NOT WORK
p,1,6,2000
                                   ! commands DO NOT work
d,1,uy,.1
*msq.note
THIS IS A TEST MESSAGE
*vwrite,array(1)
(/b = ,f10.4)
```

6.10. Creating Your Own Optimization Subroutine

One way to customize the ANSYS program for your site's specific needs is to substitute your own external optimization for the standard ANSYS optimization algorithm. You can do so by linking a user subroutine within the ANSYS program.

You can find additional information on design optimization techniques and procedures in the *Advanced Analysis Techniques Guide*.

6.10.1. Linking a Custom Optimization Program to ANSYS

If you are performing iterative analyses for design optimization, and you wish to use the standard ANSYS optimizer, you simply choose GUI path **Main Menu>Design Opt** or issue the ANSYS command **/OPT**.

However, if you plan to use an optimization subroutine of your own design, you must do the following:

- Define parameters for your external optimization subroutine, using either the OPUSER command or GUI path Main Menu>Design Opt>Method/Tool. (For more information about OPUSER, see the Command Reference.)
- Either issue the OPTYPE,USER command or choose menu path Main Menu>Design Opt>Method/Tool and specify User Optimizer.
- 3. Issue the **OPEXE** command or choose GUI path **Main Menu>Design Opt>Run**.

Issuing the **OPTYPE**,USER command (or its GUI equivalent) instructs the ANSYS program to bypass the standard ANSYS design optimizer logic and instead execute your custom optimization subroutine.

The userop subroutine (below) includes a flow chart showing how a user-defined optimization subroutine interacts with the ANSYS program. It also contains a commented example that shows you sample ANSYS command input for design optimization, done with a user-customized version.

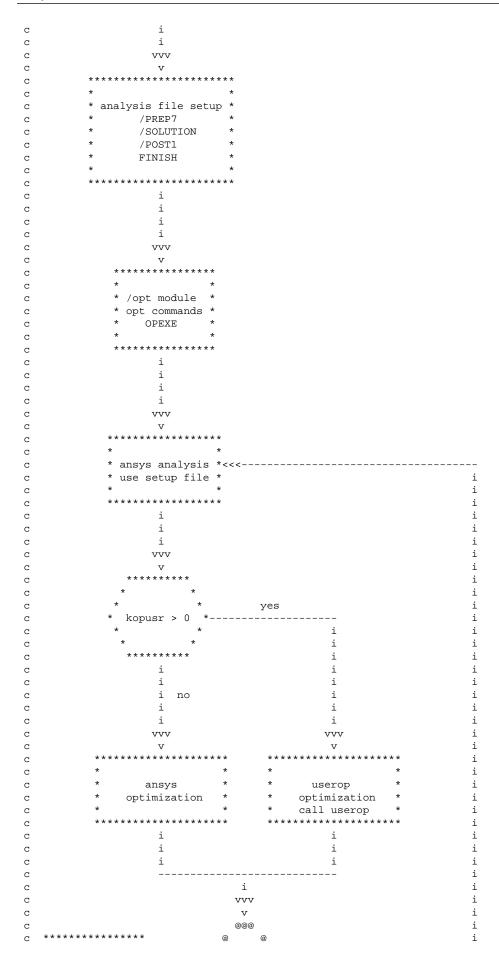
For information about the kopusr variable and the userop subroutine, see the next section.

6.10.2. Subroutine userop (Defining a Custom Optimization Subroutine)

Instead of creating your custom design optimization subroutine from scratch, you can simply tailor the userop subroutine available on your ANSYS distribution medium. Defined within this subroutine are a number of variables to which you assign the values you want.

Below is a listing of the header information foruserop which includes the inputs and outputs and an example user optimization:

```
*deck,userop
                                  USERSDISTRIB
      subroutine userop (iott,nterm,maxparm,optvar)
С
        primary purpose: user-supplied optimization routine
          accessed with optype, user
С
            other input comes from the opuser command
C
          *** copyright(c) 2008 SAS IP, Inc. All rights reserved.
C
 *** Notice - This file contains ANSYS Confidential information ***
C
C
c*
C
c incoming arguments:
C
C
С
      iott = ansys output unit
С
      nterm = looping control parameter
C
               passed back to routine opterm. this variable should be set
С
               as follows:
C
С
                         nterm = 0 if optimization looping should stop
                         nterm = 1 if optimization looping should continue
С
C
      maxparm = maximum number of parameters (first dimension on optvar)
С
С
С
      optvar = optimization variables (see documentation in cmopt.inc)
C
c*
C
  *** simplified flowchart of how user optimization interfaces with ansys
С
C
С
C
C
                   start
С
С
                     i
C
```



```
i
C
                     yes
                  *<<---@ terminate
С
                  *(nterm=0) @ looping?@
С
                                                                       i
                                                                       i
С
                               @
                                     @
                                 @@@
С
С
                                  i
                                  i no
C
C
                                  i (nterm=1)
C
                                 \nabla\nabla\nabla
С
                                  v
                                                                       i
C
C
                    * move optimization results *---- to new ---->>i
С
                        into parameter vector *
С
                                                    analysis
C
                    ********
С
C
c *** sample input for user optimization ***
C
c!!!----
          minimize y = x1**2 + x2**2
           subject to : 1 < x1 < 10
c!!! ---
c!!! ---
                        1 < x2 < 10
c!!! ---
                         g = x1 - x2 > 1
С
C
      /batch.list
                         ! use batch mode
С
      x1=5
                          ! initial value for dv x1
      x2=5
                         ! initial value for dv x2
С
     /prep7
                         ! enter prep
С
      y = x1*x1 + x2*x2 ! define function y
С
      g = x1 - x2! define function g
С
С
      finish
                         ! finish prep
С
      /opt
                          ! enter opt
      opvar,x1,dv,1,10 ! define 1st dv (range 1 to 10)
C
С
      opvar,x2,dv,1,10 ! define 2nd dv (range 1 to 10)
С
      opvar,y,obj,,,1 ! define objective function (tol = 1)
     opvar,g,sv,1 ! define state variable (lower limit 1)
optype,user ! opt method is user supplied
С
С
      opuser,10,1.1,2.0 ! defime user opt modifiers (max iterations=10)
С
                         ! execute opt
C
      opexe
      oplist,all
                         ! list results
С
      finish
C
```

When you finish customizing the userop subroutine, you relink it using either of the methods described in the Installation and Configuration Guide. You must include the **cmopt** common block in your subroutine, to allow it to pass common data (such as design variables, state variables, and objective function values) to and from the ANSYS program.

6.10.3. Structuring Your Input

When your userop optimization subroutine finishes executing, program control returns to the ANSYS design optimizer, so that the ANSYS commands on Jobname.LOOP can execute for the next optimization loop. To use your own customization subroutine, you should issue the following ANSYS commands (in the order shown):

```
/OPT
OPTYPE,USER ! ANSYS optimization logic bypassed
OPUSER, ! user-defined optimization to be done
OPVAR ! design and state variables & objective function values defined
.
.
.
OPEXE ! looping begins
```

6.11. Memory Management Subroutines

ANSYS provides UPF subroutines you can use for memory management.

6.11.1. Using the Memory-Management Subroutines

ANSYS uses a dynamic memory manager that overlays the system *malloc* and *free* functions and provides a mechanism for accessing the memory from Fortran as well as c and c++. Since the UPF subroutines are provided in Fortran, we will be discussing the Fortran access subroutines.

You may certainly use the system *malloc* and *free* functions or, for Fortran, the *allocate* system function. However, you may end up competing with ANSYS for memory, and for large problems there may be insufficient system memory to perform the function.

Dynamic memory is done through *Cray-style* pointers, where a dynamically allocated array is defined via the construct

```
pointer (piArray,Array)
integer Array(*)
```

and memory space for the array is allocated by assigning the pointer, in this case piArray, to the allocated memory space:

```
piArray = fAnsMemAlloc (size,...)
```

To use the ANSYS memory manager in a UPF, follow these steps:

1. 1. Define the dynamically allocated arrays:

```
pointer (piArray,Array), (pdData,Data)
integer Array(*)
double precision Data(*)
```

2. Initialize the pointers as follows:

```
piArray = PTRFTNNULL
pdData = PTRFTNNULL
```

3. Allocate space for an array or arrays, as follows:

For integer numbers:

```
piArray = fAnsMemAlloc(ileng,MEM_INTEGER,C16Label)
```

For double-precision numbers:

```
pdArray = fAnsMemAlloc(dleng,MEM_DOUBLE,C16Label)
```

For complex numbers:

```
pcArray = fAnsMemAlloc(cleng,MEM_COMPLEX,C16Label)
```

For real numbers:

```
prArray = fAnsMemAlloc(rleng,MEM_REAL,C16Label)
```

Where the arguments are:

- xleng is the desired size of the array
- MEM_xxx is the keyword indicating the type of data

C16Label is a character*16 name of the memory block

You must include the ansysdef.inc include file to get the parameter values of MEM_INTEGER, MEM_DOUBLE, MEM_COMPLEX, and MEM_REAL.

Note

If there is insufficient memory, fAnsMemAlloc returns "PTRFTNNULL".

- 4. Use the arrays.
- 5. Deallocate the space using the fAnsMemFree subroutine, as follows:

```
call fAnsMemFree (piArray)
```

The next two sections provide input and output listings for the memory management subroutines.

For an example, see *Function user03* (*Demonstrates Using ANSYS Memory*) (p. 196), which appears earlier in this chapter.

6.11.2. Function fAnsMemAlloc (Allocating Space and Returning a Pointer)

```
*deck.fAnsMemAlloc
      function fAnsMemAlloc (iLen, key, c16Label)
c primary function:
                      Get A Block of Space from mem manager and Return Pointer
c keywords: integer function for mem allocate
c object/library: mem
c *** Notice - This file contains ANSYS Confidential information ***
  input arguments:
                                - length of the block (in data elements)
     iLen (int,sc,in)
С
     c16Label (chr*16,sc,in) - 16 character name for the Block
     key (int,sc,in)
                                - type of data for this block (see ansysdef)
C
С
  output arguments:
      fAnsMemAlloc (PTRFTN,sc,out) - Pointer to this data block -- needs to be
C
C
                                      tied to a local variable in the calling
                                      routine
```

6.11.3. Subroutine fAnsMemFree (Deallocating Space)

6.12. Parameter-Processing Subroutines

The ANSYS distribution medium contains three subroutines you can use for parameter processing: pardim, parevl, and pardef.

6.12.1. Subroutine pardim (Creating a Dimensioned Parameter)

```
*deck,pardim
     subroutine pardim (cName,labl4,nDim,nxyz,cLabels)
c *** primary function:
                         create a dimensioned parameter
        *dim,parm32,type,d1,d2,d3,cName1,cName2,cName3
C
        *dim,parm32,type,d1,cName1
        *dim,parm32,type,d1,d2,d3,d4,d5,cName1,cName2,cName3,cName4,cName5
C
c *** Notice - This file contains ANSYS Confidential information ***
C
  input arguments:
C
     cName (chr*32,sc,in) - the name of the parameter to create
              (chr*4,sc,in) - 'TABL' or 'ARRA' or 'CHAR' or 'STRI'
     labl4
C
                               - Dimension of array
C
     nDim
              (int,ar(nDim),in) - the dimensions of the array
     nxyz
C
    cLabels (chr*32,ar(nDim),in) - Names for the directions in table
C
c output arguments: none
```

6.12.2. Function parevl (Finding and Evaluating a Parameter)

```
*deck,parevl
     subroutine parevl (ParName, nDim, subc, lvl, dpValue, chValue, kerr)
c *** primary function: find and evaluate a parameter
c *** Notice - This file contains ANSYS Confidential information ***
C
С
  input arguments:
     ParName (chr*(PARMSIZE),sc,in) - the name of the parameter
C
С
                                        (must be upper case, left justified)
               (int,sc,in)
                                      - the number of subscripts (0,scaler)
С
              (dp,ar(*),in)
                                     - values for the subscripts (if any)
С
     subc
                                     - 0,1 no error output 2, report error
     lvl
              (int,sc,in)
С
С
                                        -1, set kerr flag with no anserr call
C
c output arguments:
                                   - the value of the parameter (may be a
С
     dpValue (dp,sc,out)
                                    packed character*8
C
    chValue (ch*128,sc,out)
                                    - character output
     kerr (int,sc,out)
C
                                    - error flag (0,ok -1,output is packed
С
                                      0=ok, 1=error, 2=error but TINY is used
                                       -2, output is string in chValue
C
 *** mpg parevl<dpValueFunct<dpinfun<vopwrt<voper<rdmac<rdcmd<utcmd: *get par
С
С
          getfun<funevl<evall gtparm<r5get vopwrt</pre>
```

6.12.3. Subroutine pardef (Adding a Parameter)

```
C
   input arguments:
C
      cNameIn (chr*(PARMSIZE),sc,in) - name of parameter
С
                               cNameIn is a character variable that
                               contains the name of the parameter that
С
                               is to be defined. (Length = PARMSIZE characters)
С
С
                            - 0, dp 1, character 2, string
С
      ctype (int.sc.in)
                               ctype is an integer key which describes
C
                               the type of data that the parameter data
С
С
                               holds. This would also indicate the
                               contents of "value" (arg 5).
С
                               0=double precision data
C
С
                               1=character data packed in value
С
                               2=character data in string
C
                            - number of subscripts
С
      nval
             (int,sc,in)
                               nval is the number of subscripts that the
C
                               "cNameIn" (arg 1) contains.
C
С
                               1=single dimensioned variable (ex. x(10))
                               2=double dimensioned variable (ex. y(10,3))
С
                               3=triple dimensioned variable (ex. z(10,3,2))
С
С
                               -1=delete this parameter from the internal
                               tables.
C
С
      subc (dp,ar(*),in)
                            - values of subscripts
C
                               subc is a double precision vector that
C
C
                               contains the subscripts of "cNameIn" (arg 1).
                               There should be enough values defined to
C
                               match "nval" (arg 3). For example if "x"
C
                               was dimensioned as "x(10,3,2)" and you wanted
С
                               to set "x(5,1,1)=123.0", then "nval" (arg 3)
C
                               should be set to 3, and "subc" should be set
C
C
                               to 5.0, 1.0, 1.0, and "value" (arg 5) should
                               be 123.0. Another example is if "y" was
C
С
                               dimensioned to as "y(20,20)" and you were
                               setting "y(5,8)=987", then "nval" (arg 3) should
C
                               be set to 2 and "subc" should be set to 5.0,
C
                               8.0, 0.0, and "value" (arg 5) should be 987.0.
C
C
                               Remember subroutine "pardef" is only storing
C
                               a data value of "cNameIn" or "cNameIn(x,y,z)". The
С
                               proper dimensions were set by a "*dim" command.
C
С
                               Please note that although the values of "subc"
C
                               should be double precision, subroutine "pardef"
C
                               uses the intrinsic "nint" (nearest integer)
С
C
                               function to get and use the integer equivalent.
C
                               You should also note the "nval" (arg 3) and
C
С
                             "subc" (arg 4) must fall within the range that was
C
                               set with a "*dim" or "*set" command or an error
                               will occur.
C
C
С
      valuein(dp,sc,in)
                            - the value for this parameter
C
                               (should be a packed character*8 if
                               ctype=1. To pack a char into a dp
C
                               variable use "call chtodp(ch8,dp)".
С
С
                               To unpack a dp variable into a char
С
                               use "call dptoch(dp,ch8)" )
С
                               Value is the data value that is to be stored for
                               "cNameIn" (arg 1). If "ctype=1" (arg 2) then this
C
С
                               value would be a "packed character" data from the
                               "chtodp" Ansys function.
C
C
   output arguments:
С
             (int,sc,out) - error flag (0=ok, 1=error)
С
C
                              kerr is an integer error flag that is
С
                               returned to the calling subroutine. Any
С
                               non zero number would indicate an error
                               was detected in subroutine "pardef"
C
 *** mpg pardef < parstore pardim ntableget rdsset<rdmac<rdcmd: define param
```

С

6.13. Miscellaneous Useful Functions

The ANSYS program has several miscellaneous functions you may find useful for working with UPFs:

- The erhandler subroutine displays output messages (notes, warnings, and errors).
- The RunCommand function lets you issue an ANSYS command from within a user subroutine.
- The GetStackDisp subroutine retrieves current displacement values.
- The /UNDO command writes an "undo" file at critical points as a user subroutine executes.
- The/HOLD command allows you to synchronize multiple tasks in ANSYS.
- The /TRACK command enables you to do program tracing and timing.

For further descriptions of erhandler and /TRACK, see Chapter 8, Subroutines for Users' Convenience (p. 285). For details about the GetStackDisp function, see Function GetStackDisp (Getting Current Displacement Values) (p. 201).

6.13.1. Using Function RunCommand

This function enables you to execute an ANSYS command from within a user subroutine. Inputs and outputs for RunCommand are as follows:

```
*deck.RunCommand
      function RunCommand (nChar, command)
c primary function:
                      Execute an ansys command
c object/library: upf
 *** Notice - This file contains ANSYS Confidential information ***
   input arguments:
C
      nChar
                                       - Length of the command string (8 min)
С
               (int,sc,in)
      command (ch*(nChar),sc,in)
С
                                      - A character string containing a
                                                      valid ANSYS command
C
  output arguments:
C
      RunCommand
                  (int.sc.out)
                                      - An internally defined value, ignore
```

6.13.2. Using the /UNDO Command

The "undo" file you create by issuing the **/UNDO** command is similar to the File.DB file created when you issue the **SAVE** command. The **/UNDO** command format is:

```
/UNDO,Action
```

Action

ON, to write the undo file
OFF, to prevent the undo file from being written
PROMPT, to have ANSYS ask permission before writing the file
STATUS, to restore the file as it existed after executing the last command issued before the /UNDO command.

6.13.3. Using the /HOLD command

Issue the /HOLD command to synchronize tasks in ANSYS. The ANSYS program can synchronize tasks at the end of each results file set.

/HOLD, Filename, TimeInterval, Timeout

Filename

The eight-character name of a message file. If the named file exists, the ANSYS program reads a command from the file and then deletes the file.

TimeInterval

The length of time, in seconds, that ANSYS waits before trying to read the message file again.

Timeout

The maximum length of time, in seconds, that ANSYS can wait between attempts to read the file.

Release 12.0 - © 2009 SAS IP, Inc. All rights reserved Contains proprie	etary and confidential information
of ANSYS. Inc. and its subsidiaries and af	ffiliator

Chapter 7: Accessing the ANSYS Database

This chapter describes how you can retrieve information in the ANSYS database (or store information in the database) by linking subroutines you create into the ANSYS program.

You can use the database access routines with any of the user-programmable features. For example, you can create your own ANSYS commands and use them to execute database access routines (or have a database access routine call a user-defined command).

Inputs and Outputs for Database Access Routines

The descriptions of the database access routines or functions within this chapter describe both the input arguments and output arguments. Argument information includes the argument's type, size and intent.

• Argument *type* is one of the following:

```
int - integer
```

dp - double precision

log - logical

chr - character

dcp - double precision complex

Argument size is one of the following:

```
sc - scalar variable
```

ar(n) - array variable of length n

func - functional return value

Argument intent is one of the following:

```
in - input argument
```

out - output argument

inout - both an input and an output argument

Types of Database Access Routines

The rest of this chapter describes the functions and subroutines available for accessing information in the ANSYS database. The function and subroutine descriptions are grouped into the following sections.

- 7.1. Routines for Selecting and Retrieving Nodes and Elements
- 7.2. Node Information Routines
- 7.3. Element Attribute Routines
- 7.4. Coupling and Constraint Routines
- 7.5. Nodal Loading Routines
- 7.6. Element Loading Routines
- 7.7. Results Information Routines

7.1. Routines for Selecting and Retrieving Nodes and Elements

7.1.1. ndnext Function (Getting the Next Node Number)

```
*deck,ndnext
     function ndnext (next)
c *** primary function: get the number of the next selected node
c *** Notice - This file contains ANSYS Confidential information ***
     input arguments:
C
        next
              (int,sc,in)
                                   - the last node number used
C
                                    = 0 - use for initial value
C
С
    output arguments:
      ndnext (int,func,out)
                                  - the next selected node number
C
                                     = 0 - no more nodes
```

7.1.2. ndprev Function (Getting the Number of the Previous Selected Node)

```
*deck,ndprev
     function ndprev (next)
c *** primary function:
                         get the number of the previous selected node
c *** Notice - This file contains ANSYS Confidential information ***
С
     typ=int,dp,log,chr,dcp siz=sc,ar(n),func intent=in,out,inout
C
С
С
     input arguments:
     variable (typ,siz,intent) description
C
                                 - the next node number used
                                    = 0 - use for initial value
С
С
С
     output arguments:
       ndprev (int,func,out)
                                  - the previous selected node number
С
                                     = 0 - no more nodes
C
```

7.1.3. ndnxdf Function (Getting the Number of the Next Defined Node)

```
*deck,ndnxdf
     function ndnxdf (next)
c *** primary function: get the number of the next defined node
C
c *** Notice - This file contains ANSYS Confidential information ***
     typ=int,dp,log,chr,dcp siz=sc,ar(n),func intent=in,out,inout
С
C
     input arguments:
С
                                description
C
     variable (typ,siz,intent)
                                  - the last node number used
С
       next
              (int,sc,in)
С
                                    = 0 - use for initial value
C
    output arguments:
       ndnxdf (int,func,out)
                                  - the next defined node number
С
С
                                    = 0 - no more nodes
```

7.1.4. ndsel Function (Selecting, Unselecting, Deleting, or Inverting a Node)

```
*deck,ndsel
     subroutine ndsel (ndmi, ksel)
c *** primary function: to select, unselect, delete, or invert a node.
c *** secondary functions: none.
c *** Notice - This file contains ANSYS Confidential information ***
      typ=int,dp,log,chr,dcp siz=sc,ar(n),func
                                                    intent=in,out,inout
C
      input arguments:
        variable (typ,siz,intent)
                                      description
C
                                    - node number
                  (int,sc,in)
                                      = 0 - all nodes
C
С
                                      < 0 - do not delete CPs and CEQNs
                                            (merge/offset/compress)
C
C
         ksel
                  (int.sc.in)
                                    - type of operation to be performed.
                                      ksel = 0 - delete node.
С
                                            = 1 - select node.
С
                                           =-1 - unselect node.
C
С
                                            = 2 - invert select status of node.
      output arguments:
C
         none.
```

7.1.5. elnext Function (Getting the Number of the Next Element)

```
*deck,elnext
     function elnext (next)
c *** primary function:
                         get the number of the next selected element
c *** Notice - This file contains ANSYS Confidential information ***
      input arguments:
C
                (int,sc,in)
                                   - the last element number used
С
С
                                     = 0 - use for initial value
      output arguments:
С
         elnext (int,func,out)
                                   - the next selected element
C
                                      = 0 - no more elements
C
```

7.1.6. elprev Function (Getting the Number of the Previous Selected Element)

```
*deck.elprev
     function elprev (prev)
c *** primary function:
                        get the number of the previous selected element
С
c *** Notice - This file contains ANSYS Confidential information ***
С
     typ=int,dp,log,chr,dcp siz=sc,ar(n),func
С
                                                  intent=in,out,inout
С
C
     input arguments:
C
     variable (typ,siz,intent)
                                   description
С
                 (int,sc,in)
                                   - the last element used
С
                                     = 0 - use for initial value
C
С
     output arguments:
С
        elprev (int,func,out)
                                   - the previous selected element
С
                                     = 0 - no more elements
```

7.1.7. elnxdf Function (Getting the Number of the Next Defined Element)

```
*deck,elnxdf
     function elnxdf (next)
c *** primary function:
                         get the number of the next defined element
c *** Notice - This file contains ANSYS Confidential information ***
C
С
      typ=int,dp,log,chr,dcp siz=sc,ar(n),func
                                                   intent=in,out,inout
C
      input arguments:
С
С
     variable (typ, siz, intent) description
              (int,sc,in)
                                  - the last element used
        next
C
                                     = 0 - use for initial value
С
С
С
     output arguments:
        elnxdf (int,func,out)
                                   - the next defined element
С
                                     = 0 - no more elements
C
```

7.1.8. elsel Subroutine (Selecting, Unselecting, Deleting, or Inverting an Element)

```
*deck,elsel
      subroutine elsel (ielei, ksel)
c *** primary function:
                         to select, unselect, delete, or invert an element.
c *** Notice - This file contains ANSYS Confidential information ***
С
      input arguments:
        ielei (int,sc,in)
                                    - element number
C
С
                                      = 0 - all elements
                 (int,sc,in)
                                    - type of operation to be performed.
С
        ksel
                                      = 0 - delete element.
C
С
                                      = 1 - select element.
                                      =-1 - unselect element.
C
C
                                      = 2 - invert select status for element
C
      output arguments: none
```

7.2. Node Information Routines

7.2.1. ndingr Function (Getting Information About a Node)

The primary function of ndingr is getting information about a node. This function also sets the current node pointer to this node.

Note

Some of the database commands in the input file shown below are in the common block ansysdef.inc, which must be included in the subroutine.

```
С
      input arguments:
С
         node
                  (int,sc,in)
                                     - node number
                                        Should be 0 for key=11, DB_NUMDEFINED,
С
                                        DB_NUMSELECTED, DB_MAXDEFINED, and
С
C
                                        DB_MAXRECLENG
         key
                   (dp,sc,in)
                                     - key as to information needed about
С
                                       the node.
                  = DB SELECTED
                                    - return select status:
С
                      ndingr = 0 - node is undefined.
С
                               =-1 - node is unselected.
С
                               = 1 - node is selected.
C
                  = DB_NUMDEFINED - return number of defined nodes
C
С
                  = DB_NUMSELECTED - return number of selected nodes
                  = DB_MAXDEFINED - return highest node number defined
C
                  = DB_MAXRECLENG - return maximum record length (dp words)
С
C
                  = 2, return length (dp words)
                     3,
C
С
                  = 4, pointer to first data word
                  = 11, return void percent (integer)
C
                  = 17, pointer to start of index
С
С
                  = -1.
                  = -2, superelement flag
C
С
                  = -3, master dof bit pattern
С
                  = -4, active dof bit pattern
C
                  = -5, solid model attachment
C
                  = -6, pack nodal line parametric value
                  = -7, constraint bit pattern
С
                  = -8, force bit pattern
С
                  = -9, body force bit pattern
С
С
                  = -10, internal node flag
С
                  = -11, orientation node flag =1 is =0 isnot
С
                   = -11, contact node flag <0
                  = -12, constraint bit pattern (for DSYM)
C
С
                  = -13, if dof constraint written to file.k (for LSDYNA only)
                  = -14, nodal coordinate system number (set by NROTATE)
С
                  =-101, pointer to node data record
C
                  =-102, pointer to angle record
С
C
                  =-103.
                  =-104, pointer to attached couplings
C
                  =-105, pointer to attacted constraint equations
С
C
                  =-106, pointer to nodal stresses
С
                  =-107, pointer to specified disp'S
С
                  =-108, pointer to specified forces
                  =-109, pointer to x/y/z record
C
                  =-110,
С
С
                  =-111.
                  =-112, pointer to nodal temperatures
C
                  =-113, pointer to nodal heat generations
C
С
C
                  =-115, pointer to calculated displacements
С
      output arguments:
C
                 (int,func,out)
                                    - the returned value of ndingr is based on
                                          setting of key.
```

7.2.2. getnod Function (Getting a Nodal Point)

```
= 0 - print no message if node is unselected
С
С
                                          or undefined
С
                                   = 1 - print message if node is undefined
                                    = 2 - print message if node is undefined
С
С
                                          or unselected
С
     kcrot
               (int,sc,in)
                                  - output coordinates in this coordinate system.
                                     if kcrot is negative, output theta and
C
                                     phi coordinates in radians
  output arguments:
С
               (dp,ar(6),out)
                                 - Coordinates (first 3 values) and rotation
С
                                     angles (last 3 values)
C
С
               (int,sc,inout)
                                  - select status
С
                                   = 0 - node is selected
                                    = 1 - node is not defined
C
                                    =-1 - node is unselected
```

7.2.3. putnod Function (Storing a Node)

```
*deck,putnod
     subroutine putnod (node, vctn, kcrot)
c *** primary function:
                          store a node
c *** secondary functions: display node if in immediate mode.
c *** Notice - This file contains ANSYS Confidential information ***
С
      input arguments:
        node
               (int,sc,in)
                                    - node number
C
                                    - array of 3 nodal coordinates and
         vctn
                  (dp,ar(6),in)
C
С
                                               3 nodal rotation angles.
                  (int,sc,in)
                                    - local coordinate system in which the nodal
C
        kcrot
                                       coordinates and angles are defined
С
     output arguments: none.
```

7.2.4. ndgall Function (Getting the XYZ/Rotation Coordinates Vector for a Node)

```
*deck,ndgall
     function ndgall (node,xyz)
c *** primary function:
                         get x,y,z,rotx,roty,rotz vector for a node.
c *** Notice - This file contains ANSYS Confidential information ***
     input arguments:
С
        node
               (int,sc,in)
                                   - node number for operation.
C
С
     output arguments:
                                    - status of node.
C
        ndgall (int,sc,out)
                                        0=node is undefined.
C
                                       -1=node is unselected.
С
                                        1=node is selected.
        xyz
                 (dp,ar(6),out)
                                    - vector containing x,y,z,rotx,roty,rotz
```

7.2.5. ndspgt Subroutine (Getting the Nodal Solution for a Node of an Element)

```
c *** Notice - This file contains ANSYS Confidential information ***
С
      input arguments:
                                  - The node number
С
         node
                  (int,sc,in)
                  (int,ar(DOFBITLENG),in) - The dofs to retrieve for the node.
C
         dofs
                                     dof = degree of freedom
C
                                     The dofs array should be zeroed out,
C
                                     except for the needed parts.
                                     dofs is a bit pattern with true bits
С
С
                                     representing the GLOBAL Dof set desired.
                                     That is, dofs(1) is used for UX to SP06,
С
                                     and dofs(2) is used for TBOT to TTOP.
C
                                     See ECHPRM for details. For example,
C
С
                                           dofs(1) = UX + TEMP
                                           dofs(2) = TE3
C
С
                                     TTOP is a special case. If you want
C
                                     TTOP alone, use:
                                           dofs(2) = ibset(0,TTOP)
C
                                     If TBOT and TTOP are desired, you must use:
                                           dofs(2) = TBOT
С
                                           dofs(2) = ibset(dofs(2), TTOP)
C
         ndof
                  (int,sc,in)
                                  - The number of node dofs (1, 2 or 3).
C
                  (int,sc,in)
                                  - Key to rotate dofs from nodal to global
         nrot
C
C
                                     coordinate systems.
                                     if 0, none. if 2, 2-d. if 3, 3-d
C
C
                                     if > 0, dof set must include and only
C
                                     include all terms of the vector (e.g.
C
                                     UX,UY,UZ, or AX,AY,AZ).
                  (dp,ar(6),in) - The xyz virgin node coordinates
C
         xyzang
                                      (including angles). Not used if
С
C
                                      nrot = 0 or ndof < 2.
         nuvect
                  (int,sc,in)
                                  - Number of vectors to retrieve. Can vary
C
C
                                     between 1 and 5. Normally 1 is what is
                                     wanted. Other vectors include previous
C
С
                                     values and/or velocities. See elucom for
                                     all possibilites. Contents are analysis
С
                                     type dependent.
C
С
      output arguments:
                  (dp,ar(ndof,nuvect),out) - Element nodal solution vectors in
C
         unode
                                                the global coordinate system.
```

7.3. Element Attribute Routines

7.3.1. elmiqr Function (Getting Information About an Element)

```
*deck,elmiqr
      function elmiqr (ielem, key)
c *** primary function:
                         get information about an element.
c *** secondary functions: set current element pointer to this element.
c *** Notice - This file contains ANSYS Confidential information ***
      input arguments:
C
C
         ielem
                  (int,sc,in)
                                     - element number
                                       should be zero for key=11, DB_NUMDEFINED,
C
                                         DB_NUMSELECTED, or DB_MAXDEFINED
C
                  (int,sc,in)
                                     - information flag.
                  = DB_SELECTED
                                    - return select status:
                                                                            (1)
С
                       elmigr = 0 - element is undefined.
С
                                =-1 - element is unselected.
С
                                = 1 - element is selected.
C
                  = DB_NUMDEFINED - return number of defined elements
C
С
                  = DB_NUMSELECTED - return number of selected elements
                                                                            (13)
                  = DB_MAXDEFINED - return maximum element number used
                                                                            (14)
C
С
                  = DB_MAXRECLENG
                                   - return maximum record length
                                       (int words)
```

```
= 2 - return length (int words)
C
С
                  = 3 - return layer number
С
                        (for cross reference files return number of entities)
                  = 4 - return address of first data word
С
С
                  = 5 - return length (in record type units)
C
                  = 6 - return compressed record number.
                  = 11 - return void percent (integer)
C
                  = 16 - return location of next record
C
                         (this increments the next record count)
С
                  = 17 - pointer to start of index
                  = 18 - return type of file.
С
                      elmigr = 0 - integer
C
                              = 1 - double precision
С
С
                              = 2 - real
                              = 3 - complex
C
С
                              = 4 - character*8
                              = 7 - index
С
                  = 19 - return virtual type of file.
C
                      elmiqr = 0 - fixed length (4.4 form)
                              = 1 - indexed variable length (layer data)
C
                              = 2 - xref data tables
С
С
                              = 3 - bitmap data (for 32 data item packed records)
                              = 4 - data tables (three dimensional arrays)
C
С
                  = -1 - material number etc. (see elmcmx)
С
                  =-101 - pointer to element integers etc.
                              (see elmcmx with elmilg and 1 instead of -101)
C
С
      output arguments:
C
         elmiqr (int,sc,out) - the returned value of elmiqr is based on
С
С
                                        setting of key.
С
c *** mpg elmiqr < el117,edgrde,edgrecc,edgmul: elem inquire
```

7.3.2. elmget Function (Getting an Element's Attributes and Nodes)

7.3.3. elmput Subroutine (Storing an Element)

```
*deck,elmput
      subroutine elmput (ielem,elmdat,nnod,nodes)
c *** primary function:
                          store element attributes and node numbers.
c *** secondary functions: set current element pointer to this element.
c *** Notice - This file contains ANSYS Confidential information ***
c *** NOTICE - The user is also responsible for defining the centroid for the
C
               element using the elmpct subroutine. Calling the elmput
               subroutine will NULL the element centroid previously defined.
С
      input arguments:
С
         ielem
                  (int,sc,in)
C

    element number

                  (int,ar(EL_DIM),in) - element attributes.
С
C
                              elmdat(EL_MAT) - material number
                                    (EL_TYPE) - element type
C
                                    (EL_REAL) - real constant number
С
С
                                    (EL_SECT) - section number
С
                                    (EL_CSYS) - coordinate system number
```

```
(EL_DEAD) - death flag (bit 0)
C
                                             if clear - alive
C
С
                                             if set - dead
                                     (EL_SOLID) - solid model reference
С
                                     (EL_SHAPE) - 100*shape + specific shape
С
С
                                     (EL_OBJOPTIONS) - reserved
                                     (EL_PEXCLUDE) - p element include flag
C
                                             (bit 0)
                                             if clear - include
C
С
                                             if set

    exclude

                                             For LSDYNA, it means part ID
C
                                             in regular ANSYS, it is never part ID
C
                                     - number of nodes for this element.
C
C
         nodes
                  (int,ar(*),in)
                                     - node numbers for this element.
      output arguments: none.
```

7.3.4. etyiqr Function (Getting a Data Item About an Element Type)

```
*deck,etyiqr
      function etyiqr (itype,key)
c *** primary function:
                          get information about an element type.
c *** Notice - This file contains ANSYS Confidential information ***
      input arguments:
C
С
                  (int,sc,in)
                                     - element type number
                                        Should be 0 for key=11, DB_NUMDEFINED,
C
                                        DB_NUMSELECTED, DB_MAXDEFINED, and
C
C
                                       DB_MAXRECLENG
         key
                                    - information flag.
C
                  (int,sc,in)
                  = DB_SELECTED
                                   - return select status:
С
                       etyiqr = 0 - element type is undefined.
C
                               =-1 - element type is unselected.
C
C
                               = 1 - element type is selected.
С
                  = DB_NUMDEFINED - return number of defined element types
C
                  = DB_NUMSELECTED - return number of selected element types
С
                  = DB_MAXDEFINED - return highest element type number defined
                  = DB_MAXRECLENG - return maximum record length (int words)
C
                 = -n, return element characteristic n from etycom for element
C
                        type itype.
С
                       n is correlated to the parameter names in echprm.
                       see elccmt for definitions of element characteristics.
С
C
                       note- this will not overwrite the current setting of
                        etycom.
C
С
      output arguments:
                                    - the returned value of etyiqr is based on
         etyigr (int, func, out)
C
                                        setting of key.
```

7.3.5. etyget Function (Getting Information About an Element Type)

```
*deck,etyget
      function etyget (itype,ielx)
c *** primary function:
                          get element type data.
c *** Notice - This file contains ANSYS Confidential information ***
      input arguments:
C
         itype
                  (int,sc,in)
                                    - element type number
C
      output arguments:
                                    - status of element type.
C
                  (int,func,out)
С
                                       = 0 - element type is undefined.
                                       < 0 - number of data items on unselected
С
```

7.3.6. etyput Subroutine (Storing Element Type Data)

```
*deck,etyput
      subroutine etyput (itype,n,ielx)
c *** primary function:
                          store element type data.
c *** Notice - This file contains ANSYS Confidential information ***
      input arguments:
        itype
                 (int,sc,in)
                                    - element type number for operation.
C
                  (int,sc,in)
                                    - length of data vector to store.
                  (int,ar(*),in)
         ielx
                                    - element type data. see elccmt for
C
С
                                      description.
C
      output arguments: none
c 2007 nov 5
c *** mpg etyput<etymod etydef dirasmdft dasupd dasdft:update elem type active
```

7.3.7. echrtr Subroutine (Getting Information About Element Characteristics)

```
*deck
      subroutine echrtr (iott,elcdn,ielc,kerr)
   primary function: collect all element characteristics based on
C
                                                       ityp, jtyp, and keyopts
c *** Notice - This file contains ANSYS Confidential information ***
С
С
      typ=int,dp,log,chr,dcp siz=sc,ar(n),func
                                                     intent=in.out.inout
С
   input arguments:
С
     variable (typ,siz,intent)
                                      description
C
С
             (int,sc,in)
                                       - printout file
               (int,ar(IELCSZ),inout) - input element characteristics
С
                                        in positions 1 to 20.
С
                                        (itype, jstif, keyopts, etc.)
С
C
  output arguments:
С
С
     elcdn
               (chr,sc,out)
                                       - element descriptive name as character
C
                                          string
С
               (int,ar(IELCSZ),inout) - input element characteristics
                                          in positions 21 to 150.
С
                                          (kdim, ishap, idegen, etc.)
C
                                          see elccmt for a full list
С
С
     kerr
               (int,sc,out)
                                       - error flag
                                         = 0 - no errors
= 1 - errors
С
С
С
```

7.3.8. etysel Subroutine (Selecting, Unselecting, Deleting, or Inverting an Element Type)

```
*deck,etysel
      subroutine etysel (itypi,ksel)
c *** primary function:
                           to select, unselect, delete, or invert an
                            element type.
 *** secondary functions: none.
C
 *** Notice - This file contains ANSYS Confidential information ***
С
C
      typ=int,dp,log,chr,dcp siz=sc,ar(n),func
                                                      intent=in.out.inout
C
C
С
      input arguments:
         variable (typ,siz,intent)
                                       description
C
         itypi
                  (int,sc,in)
                                     - element type number
C
                                       = 0 - all element types
C
C
         ksel
                  (int.sc.in)
                                     - type of operation to be performed.
                                       = 0 - delete element type.
C
                                       = 1 - select element type.
C
                                       =-1 - unselect element type.
C
                                       = 2 - invert element type.
C
C
C
      output arguments:
С
         none.
```

7.3.9. mpingr Function (Getting Information About a Material Property)

```
*deck,mpingr
     function mpinqr (mat,iprop,key)
c *** primary function:
                           get information about a material property.
c *** Notice - This file contains ANSYS Confidential information ***
C
      input arguments:
С
         mat.
                  (int,sc,in)
                                     - material number
                                        should be 0 for key=11,
C
                                        DB NUMDEFINED(12),
C
С
                                        DB_MAXDEFINED(14), and
C
                                        DB MAXRECLENG(15)
С
                  (int,sc,in)
                                     - property reference number:
С
          if iprop = 0, test for existence of any material property with this
С
                   material number (with key = DB_SELECTED(1))
C
         ---- MP command labels -----
C
С
         EX = 1, EY = 2, EZ = 3, NUXY = 4, NUYZ = 5, NUXZ = 6, GXY = 7, GYZ = 8
C
         GXZ = 9, ALPX=10, ALPY=11, ALPZ=12, DENS=13, MU =14, DAMP=15, KXX =16
         KYY =17, KZZ =18, RSVX=19, RSVY=20, RSVZ=21, C =22, HF =23, VISC=24
C
         EMIS=25, ENTH=26, LSST=27, PRXY=28, PRYZ=29, PRXZ=30, MURX=31, MURY=32
С
         MURZ=33, PERX=34, PERY=35, PERZ=36, MGXX=37, MGYY=38, MGZZ=39, EGXX=40
С
C
         EGYY=41, EGZZ=42, SBKX=43, SBKY=44, SBKZ=45, SONC=46, SLIM=47, ELIM=48
         USR1=49, USR2=50, USR3=51, USR4=51, FLUI=53, ORTH=54, CABL=55, RIGI=56
С
         HGLS=57, BM =58, QRAT=59, REFT=60, CTEX=61, CTEY=62, CTEZ=63, THSX=64,
C
         THSY=65, THSZ=66, DMPR=67, LSSM=68,
                                                 =69,
                                                           =79,
                                                                    =71,
                                                                             =72.
С
C
             =73,
                      =74,
                               =75,
                                         =76,
                                                  =77,
                                                           =78,
                                                                             =80
С
                (see mpinit for uncommented code and for TB command information)
С
         kev
C
                  (int.sc.in)
                                     - key as to the information needed
С
                                      about material property.
С
              = DB_SELECTED(1) - return select status:
                         mpinqr = 0 - material prop is undefined.
C
                                 = 1 - material prop is selected.
              = DB_NUMDEFINED(12) - number of defined material properties
C
              = DB_MAXDEFINED(14) - highest material property number defined
С
```

7.3.10. mpget Function (Getting a Material Property Table)

```
*deck, mpget
      function mpget (mat,iprop,temp,prop)
c *** primary function:
                             get a material property table.
c *** Notice - This file contains ANSYS Confidential information ***
      typ=int,dp,log,chr,dcp siz=sc,ar(n),func
                                                      intent=in.out.inout
С
      input arguments:
С
         variable (typ,siz,intent)
                                       description
                                       - material number
С
                   (int,sc,in)
                                      - property reference number:
C
         iprop
                   (int,sc,in)
         ---- MP command labels -----
C
         EX = 1, EY = 2, EZ = 3, NUXY = 4, NUYZ = 5, NUXZ = 6, GXY = 7, GYZ = 8
         \texttt{GXZ} \ = \ 9 \,, \ \texttt{ALPX=10} \,, \ \texttt{ALPY=11} \,, \ \texttt{ALPZ=12} \,, \ \texttt{DENS=13} \,, \ \texttt{MU} \quad = 14 \,, \ \texttt{DAMP=15} \,, \ \texttt{KXX} \ = 16 \,
C
         KYY =17, KZZ =18, RSVX=19, RSVY=20, RSVZ=21, C
С
                                                               =22, HF
         EMIS=25, ENTH=26, LSST=27, PRXY=28, PRYZ=29, PRXZ=30, MURX=31, MURY=32
С
         MURZ=33, PERX=34, PERY=35, PERZ=36, MGXX=37, MGYY=38, MGZZ=39, EGXX=40
C
         EGYY=41, EGZZ=42, SBKX=43, SBKY=44, SBKZ=45, SONC=46, SLIM=47, ELIM=48
         USR1=49, USR2=50, USR3=51, USR4=51, FLUI=53, ORTH=54, CABL=55, RIGI=56
С
         HGLS=57, BM =58, QRAT=59, REFT=60, CTEX=61, CTEY=62, CTEZ=63, THSX=64,
С
         THSY=65, THSZ=66, DMPR=67, LSSM=68,
                                                     =69,
                                                               =79,
С
                                                                         =71,
              =73.
                       =74.
                                 =75.
                                          =76.
                                                     =77.
                                                               =78.
                                                                         =79.
                                                                                   =80
C
C
                      (see mpinit for uncommented code and TB command information)
C
С
      output arguments:
         mpget
                 (int,func,out)
                                       - number of temperature values
C
                  (dp,ar(mpget),out) - vector of the temperature values
С
         temp
                  (dp,ar(mpget),out) - vector of the property values
```

7.3.11. mpput Subroutine (Storing a Material Property Table)

```
*deck, mpput
     subroutine mpput (mat,iprop,ntab,temp,prop)
c *** primary function:
                        store material property tables.
c *** Notice - This file contains ANSYS Confidential information ***
      input arguments:
C
                                   - material number.
С
        mat (int,sc,in)
С
        iprop
                 (int,sc,in)
                                   - property reference number:
         ---- MP command labels -----
C
С
         EX = 1, EY = 2, EZ = 3, NUXY = 4, NUYZ = 5, NUXZ = 6, GXY = 7, GYZ = 8
        GXZ = 9, ALPX=10, ALPY=11, ALPZ=12, DENS=13, MU =14, DAMP=15, KXX =16
C
        KYY =17, KZZ =18, RSVX=19, RSVY=20, RSVZ=21, C =22, HF =23, VISC=24
С
        EMIS=25, ENTH=26, LSST=27, PRXY=28, PRYZ=29, PRXZ=30, MURX=31, MURY=32
        \texttt{MURZ=33, PERX=34, PERY=35, PERZ=36, MGXX=37, MGYY=38, MGZZ=39, EGXX=40}
C
         EGYY=41, EGZZ=42, SBKX=43, SBKY=44, SBKZ=45, SONC=46, SLIM=47, ELIM=48
C
        USR1=49, USR2=50, USR3=51, USR4=51, FLUI=53, ORTH=54, CABL=55, RIGI=56
C
        HGLS=57, BM =58, QRAT=59, REFT=60, CTEX=61, CTEY=62, CTEZ=63, THSX=64,
C
        THSY=65, THSZ=66, DMPR=67, LSSM=68, =69,
                                                         =79,
                                                                   =71,
С
С
             =73.
                      =74,
                               =75.
                                        =76.
                                                 =77,
                                                          =78.
                                                                   =79.
                                                                            =80
С
```

```
(see mpinit for uncommented code and TB command information)
C
С
         ntab
                  (int,sc,in)
                                     - number of entries in the table
                                        (1 to 100)
C
C
                 (dp,ar(ntab),in) - temperature vector (ascending)
С
                 (dp,ar(ntab),in) - property vector
C
      output arguments:
C
         none.
```

7.3.12. mpdel Subroutine (Deleting a Material Property Table)

```
*deck,mpdel
                        subroutine mpdel (mat,iprop)
c *** primary function:
                                                                                                       delete material property tables.
c *** Notice - This file contains ANSYS Confidential information ***
C
                        input arguments:
C
                                  mat
                                                                      (int,sc,in)
                                                                                                                                               - material number.
С
                                   iprop
                                                                        (int,sc,in)
                                                                                                                                               - property reference number:
                                                                                                                                                          (0 = all properties)
                                    ---- MP command labels -----
C
                                   EX = 1, EY = 2, EZ = 3, NUXY = 4, NUYZ = 5, NUXZ = 6, GXY = 7, GYZ = 8
C
                                   \texttt{GXZ} = 9, \texttt{ALPX} = 10, \texttt{ALPY} = 11, \texttt{ALPZ} = 12, \texttt{DENS} = 13, \texttt{MU} = 14, \texttt{DAMP} = 15, \texttt{KXX} = 16
 C
                                    \texttt{KYY} = 17 \,, \; \texttt{KZZ} = 18 \,, \; \texttt{RSVX} = 19 \,, \; \texttt{RSVY} = 20 \,, \; \texttt{RSVZ} = 21 \,, \; \texttt{C} \qquad = 22 \,, \; \texttt{HF} \qquad = 23 \,, \; \texttt{VISC} = 24 \,, \; \texttt{C} = 24 \,,
C
                                    EMIS=25, ENTH=26, LSST=27, PRXY=28, PRYZ=29, PRXZ=30, MURX=31, MURY=32
С
                                   MURZ=33, PERX=34, PERY=35, PERZ=36, MGXX=37, MGYY=38, MGZZ=39, EGXX=40
 C
                                   EGYY=41, EGZZ=42, SBKX=43, SBKY=44, SBKZ=45, SONC=46, SLIM=47, ELIM=48
C
 C
                                   USR1=49, USR2=50, USR3=51, USR4=51, FLUI=53, ORTH=54, CABL=55, RIGI=56
                                   HGLS=57, BM =58, QRAT=59, REFT=60, CTEX=61, CTEY=62, CTEZ=63, THSX=64,
C
                                   THSY=65, THSZ=66, DMPR=67, LSSM=68,
                                                                                                                                                                                                  =69,
                                                                                                                                                                                                                                      =79.
                                                                                                                                                                                                                                                                          =71,
                                                                                                                                                                                                                                                                                                              =72,
С
                                                                                       =74,
                                                                                                                           =75,
                                                                                                                                                               =76,
                                                                                                                                                                                                   =77,
                                                                                                                                                                                                                                       =78.
 C
C
C
                                                                (see mpinit for uncommented code and for TB command information)
                       output arguments: none.
C
```

7.3.13. rlingr Function (Getting Information About a Real Constant Set)

```
*deck,rlingr
     function rlingr (nreal, key)
c *** primary function:
                        get information about a real constant set
c *** secondary functions: none
c *** Notice - This file contains ANSYS Confidential information ***
      typ=int,dp,log,chr,dcp siz=sc,ar(n),func
                                                    intent=in,out,inout
C
   input arguments:
C
         variable (typ,siz,intent)
                                      description
C
                                   - real constant table number
С
         nreal (int.sc.in)
С
                                       should be 0 for key=11, DB_NUMDEFINED,
                                       DB_NUMSELECTED, DB_MAXDEFINED, and
C
C
                                       DB_MAXRECLENG
                                    - information flag.
C
         key
                  (int,sc,in)
                               - return number of values stored for nreal
C
              = DB SELECTED
                               - return select status
C
                       rlinqr = 0 - real constant table is undefined.
C
C
                              =-1 - real constant table is unselected.
C
                              = 1 - real constant table is selected
              = DB_NUMDEFINED - return number of defined real constant tables
C
              = DB_NUMSELECTED - return number of selected real constant tables
C
              = DB_MAXDEFINED - return highest real constant table defined
              = DB_MAXRECLENG - return maximum record length (dp words)
C
```

7.3.14. riget Function (Getting Real Constant Data)

```
*deck,rlget
     function rlget (nreal, rtable)
c *** primary function: get real constant data
c *** Notice - This file contains ANSYS Confidential information ***
C
     input arguments:
                                   - real constant table number
С
        nreal
               (int,sc,in)
     output arguments:
C
       rlget (int,func,out)
                                  - number of real constant data obtained
                                  - real constant data obtained
C
        rtable (dp,ar(*),out)
```

7.3.15. rlsel Subroutine (Selecting or Deleting a Real Constant Set)

```
*deck,rlsel
      subroutine rlsel (nreai,ksel)
c *** primary function: select or delete a real constant set
c *** secondary functions: none
c *** Notice - This file contains ANSYS Confidential information ***
С
      typ=int,dp,log,chr,dcp siz=sc,ar(n),func intent=in,out,inout
С
С
   input arguments:
     variable (typ,siz,intent) description
C
     nreai (int,sc,in)
                                - real constant table
С
С
                                  = 0 - all real constant tables
C
    ksel
             (int.sc.in)
                                - type of operation to be performed.
С
                                  = 0 - delete real constant table.
С
                                  = 1 - select real constant table.
                                  =-1 - unselect real constant table.
C
                                  = 2 - invert real constant table.
С
c output arguments:
С
С
```

7.3.16. csyiqr Function (Getting Information About a Coordinate System)

```
*deck.csvigr
     function csyiqr (ncsy, key)
c *** primary function: get information about a coordinate system
c *** Notice - This file contains ANSYS Confidential information ***
   input arguments:
С
С
              (int,sc,in)
                                - coordinate system reference number
С
                                    should be zero for key= DB NUMDEFINED
С
                                   or DB_MAXDEFINED
              (int,sc,in)
                                 - information flag.
С
                               - return status:
C
               = DB SELECTED
С
                            csyiqr = 0 - coordinate system is not defined
                                     -1 - coordinate system is not selected
С
```

```
c = DB_NUMDEFINED - number of defined coordinate systems
c = DB_MAXDEFINED - maximum coordinate system reference
c number used.

c output arguments:
c csyiqr (int,func,out) - the returned value of csyiqr is based on setting of key.
```

7.3.17. csyget Function (Getting a Coordinate System)

```
*deck,csyget
     function csyget (ncsy,csydpx,csyinx)
c *** primary function: get a coordinate system
c *** secondary functions: none
c *** Notice - This file contains ANSYS Confidential information ***
 NOTE: As a time-saving device, this routine will not fetch the coordinate
          system data from the database (an expensive operation)
C
С
          if ncsy = csyinx(4), as this would indicate that the data is current.
          If you wish to force this routine to fetch coordinate system data (in
C
          the case of loading a local array, for example), you MUST set
C
          ncsy != csyinx(4) before function call.
      typ=int,dp,log,chr,dcp siz=sc,ar(n),func
                                                    intent=in.out.inout
C
  input arguments:
C
     variable (typ,siz,intent) description
                                                                csvcom name
C
С
               (int,sc,in)
                                - coordinate system number
     csyinx(4) (int,sc,inout) - coordinate system number
C
                                                                csyact
   output arguments:
С
      csydpx (dp,ar(18),out)
C
С
                          csydpx(1-9) - transformation matrix
                                (10-12) - origin (XC, YC, ZC)
С
                                (13-14) - coordinate system parameters cparm
C
С
                                                                        cparm2
                                (15)
C
                                       - spare
                                (16-18) - defining angles
C
     csyinx
               (int,ar(6),out)
C
                          csyinx(1-2)
                                      - theta, phi singularity keys
С
                                       - coordinate system type
С
                                (3)
                                                                        icdsys
С
          (csyinx(4) is inout)
                                (4)
                                       - coordinate system number
                                                                        csyact
                                       - spare
                                (5)
C
С
                                (6)
                                        - spare
С
     csyget
              (int,func,out)
                                   - status of coordinate system
                                    = 0 - coordinate system exists
C
                                    = 1 - coordinate system doesn't exist
```

7.3.18. csyput Subroutine (Storing a Coordinate System)

```
*deck,csyput
     subroutine csyput (ncsy,csydpx,csyinx)
c *** primary function:
                           store a coordinate system
c *** Notice - This file contains ANSYS Confidential information ***
   input arguments:
C
                                  - coordinate system number
С
      ncsv
               (int,sc,in)
C
      csvdpx
               (dp,ar(18),out)
                           csydpx(1-9)
C

    transformation matrix

                                 (10-12) - origin (XC, YC, ZC)
C
С
                                 (13-14) - coordinate system parameters
                                                                             cparm
С
                                                                            cparm2
```

```
(15)
                                           - spare
C
                                  (16-18) - defining angles
С
С
      csyinx
                (int,ar(6),out)
                            \operatorname{csyinx}(1-2) - theta, phi singularity keys
С
                                                                                icdsys
С
                                        - coordinate system type
                                        - coordinate system number
C
                                  (4)
                                                                                csyact
                                  (5)
                                       - spare
C
                                  (6) - spare
   output arguments: none
```

7.3.19. csydel Subroutine (Deleting a Coordinate System)

```
*deck,csydel
     subroutine csydel (ncsy)
c *** primary function: delete a coordinate system
c *** secondary functions: none
С
     typ=int,dp,log,chr,dcp
                            siz=sc,ar(n),func
                                                  intent=in.out.inout
С
c *** Notice - This file contains ANSYS Confidential information ***
C
  input arguments:
С
    variable (typ, siz, intent) description
С
     ncsy (int,sc,in)
                                - coordinate system number
C
С
  output arguments:
     none
C
```

7.3.20. userac Subroutine (Demonstrates Use of Element Attribute Routines)

See Subroutine userac (Accessing Element Information) (p. 132) for an example that demonstrates how to use the userac subroutine to extract information about an element type and element real constants from the ANSYS database. You can find this subroutine on your ANSYS distribution media.

7.4. Coupling and Constraint Routines

7.4.1. cpinqr Function (Getting Information About a Coupled Set)

```
*deck,cpingr
     function cpingr (nce, key)
c *** primary function: get information about a coupled set
c *** secondary functions: none
С
c *** Notice - This file contains ANSYS Confidential information ***
C
      typ=int,dp,log,chr,dcp siz=sc,ar(n),func intent=in,out,inout
С
С
С
   input arguments:
С
     variable (typ,siz,intent)
                                  description
               (int,sc,in) - coupled set number
С
     nce
С
     key
               (int.sc.in)
                               - inquiry key:
                                  should be zero for key=11, DB_NUMDEFINED,
С
                                   DB_NUMSELECTED, DB_MAXDEFINED, and
C
С
                                   DB_MAXRECLENG
                 = DB_SELECTED - return select status
C
                            cpinqr = 1 - coupled set is selected
C
С
                                   = 0 - coupled set in undefined
С
                                   =-1 - coupled set in unseleted
                 = DB_NUMDEFINED - return number of defined coupled sets
C
```

```
= DB_NUMSELECTED - return number of selected coupled sets
                 = DB_MAXDEFINED - return the number of the highest numbered
С
                                     coupled set
                 = DB_MAXRECLENG - return length of largest coupled set record
С
С
                                     (max record length)
                                  - return length (data units)
С
                 = 3
                                  - return layer number
C
                                  - return address of first data word
                 = 11
                                  - return void percent (integer)
C
С
                 = 16
                                  - return location of next record
                 = -1
                                  - return master node for this eqn (this is
C
                                    currently only used by solution DB object)
C
C
С
  output arguments:
             (int,func,out)
                                 - the returned value of cpingr is based on
C
      cpingr
С
                                        setting of key
```

7.4.2. cpget Function (Getting a Coupled Set)

```
*deck,cpget
      function cpget (ncp,ieqn)
c *** primary function:
                         get a coupled set
c *** Notice - This file contains ANSYS Confidential information ***
  input arguments:
C
С
     ncp
              (int,sc,in)
                                            - coupled set number
C
  output arguments:
С
                   (int,func,out)
                                          - number of nodes in list
                   (int,ar(cpget+2),out) - coupled set info:
С
                              ieqn(1:cpget) - list of coupled nodes
С
                              ieqn(cpget+1) - set degree of freedom
С
                              ieqn(cpget+2) - number of nodes in list
C
                                               (copy of return value)
```

7.4.3. cpput Subroutine (Storing a Coupled Set)

```
*deck,cpput
      subroutine cpput (ncp,n,ieqn)
c *** primary function: store a coupling set
c *** Notice - This file contains ANSYS Confidential information ***
  input arguments:
С
C
     ncp
           (int,sc,in)
                                    - coupled set number
                (int,sc,in)
                                    - number of nodes in coupled set
С
                (int,ar(n+2),in) - info for storage
C
                             ieqn(1:n) - list of coupled nodes
ieqn(n+1) - degree of freedom label for set
С
С
     (ieqn(n+2) is inout) ieqn(n+2) - number of nodes in coupled set
С
                                           (copy of n)
c output arguments:
С
      ieqn(n+2) (int,sc,inout)
                                     - number of nodes in coupled set
                                        (another copy of n)
```

7.4.4. cpsel Subroutine (Selecting or Deleting a Coupled Set)

```
*deck,cpsel subroutine cpsel (ncpi,ksel)
```

```
c *** primary function:
                          select or delete a coupled set
c *** secondary functions: none
c *** Notice - This file contains ANSYS Confidential information ***
С
С
      typ=int,dp,log,chr,dcp siz=sc,ar(n),func
                                                 intent=in,out,inout
C
  input arguments:
     variable (typ,siz,intent) description
C
С
     ncpi
               (int,sc,in)
                                 - coupled set number
                                 - select/delete flag
С
     ksel
               (int,sc,in)
                                   = 0 - delete coupled set
C
                                   = 1 - select coupled set
С
  output arguments:
С
C
     none
С
```

7.4.5. ceingr Function (Getting Information About a Constraint Equation Set)

```
*deck,ceingr
      function ceinqr (nce, key)
c *** primary function:
                         get information about a constraint equation set
c *** Notice - This file contains ANSYS Confidential information ***
      input arguments:
C
            (int,sc,in)
                                    - constraint equation number
C
      nce
С
      key
                  (int,sc,in)
                                    - inquiry key:
                                      should be zero for key=11, DB_NUMDEFINED,
C
                                      DB_NUMSELECTED, DB_MAXDEFINED, and
C
С
                                      DB_MAXRECLENG
             = DB SELECTED
                              - return select status
C
                                    ceinqr = 1 - equation is selected
С
                                           = 0 - equation is undefined
С
                                           =-1 - equation is unselected
C
             = DB_NUMDEFINED - return number of defined contraint equations
С
С
             = DB_NUMSELECTED - return number of selected contraint equations
C
             = DB_MAXDEFINED - return number of highest numbered constraint
С
                                equation defined
             = DB\_MAXRECLENG - return length of longest contraint equation set
С
                                (max record length)
C
             = 2
                             - return length (data units)
С
             = 3
С
                             - return layer number
                              - address of first data word
С
             = 4
С
             = 11
                             - return void percent (integer)
                             - return location of next record
C
             = 16
С
                              - return master dof for this eqn
     output arguments:
С
         ceinqr (int,func,out)
                                    - the returned value of ceingr is based on
С
                                        setting of key
```

7.4.6. ceget Function (Getting an Constraint Equation)

```
*deck, ceget
     function ceget (nce, ieqn, deqn)
c *** primary function:
                         get a constraint equation
c *** Notice - This file contains ANSYS Confidential information ***
C
  input arguments:
                     (int,sc,in)
                                        - constraint equation number
  output arguments:
C
                  (int,func,out)
                                            - number of dof in equation
С
С
                   (int,ar(ceget+1),out)
                                           - integer info
                              ieqn(1:ceget) - list of node*32+dof
С
```

7.4.7. ceput Subroutine (Storing a Constraint Equation)

```
*deck,ceput
     subroutine ceput (nce,n,ieqn,deqn)
c *** primary function:
                          store a constraint equation
c *** Notice - This file contains ANSYS Confidential information ***
  input arguments:
           (int,sc,in)
                                  - constraint equation set number
                (int,sc,in)
                                  - number of degrees of freedom in set
С
                (int,ar(n+1),in) - integer info
С
     ieqn
                            ieqn(1:n) - node*32+dof for each dof in set
C
                            ieqn(n+1) - number of dof in set (copy of n above)
C
С
                                      - negative means internal CE
                                 - dp info
С
     deqn
                 (dp,ar(n+1),in)
                            \text{deqn}(1:n) - coefficients of each dof in set
C
                            deqn(n+1) - constant term
С
C
  output arguments: none
```

7.4.8. cesel Subroutine (Deleting or Selecting a Constraint Equation)

```
*deck,cesel
     subroutine cesel (ncei, ksel)
                        select or delete a constraint equation
c *** primary function:
c *** Notice - This file contains ANSYS Confidential information ***
С
  input arguments:
              (int,sc,in)
                                     - constraint equation number
C
     ncei
С
                  (int,sc,in)
                                     - select/delete flag
                                       = 0 - delete equation
С
                                       = 1 - select equation
С
c output arguments: none
c *** mpg cesel cedele < pr7rst, edgung solvcl - delete ce
```

7.5. Nodal Loading Routines

7.5.1. disigr Function (Getting a Information About Constraints)

```
*deck,disiqr
    function disiqr (node,key)
c *** primary function: get information about constraints

c *** Notice - This file contains ANSYS Confidential information ***
c input arguments:
c node (int,sc,in) - node number for inquire.
```

```
- key as to the information needed
С
         key
                 (int,sc,in)
                           = 1
С
                                             - return constraint mask
С
                           = DB_MAXDEFINED,
                             DB NUMDEFINED - return number of nodal constraints
С
                                               NOTE: both DB_MAXDEFINED and
С
С
                                               DB_NUMDEFINED produce the same
                                               functionality
C
C
     output arguments:
С
        disiqr (int,func,out)
                                  - the returned value of disigr is based on
                                     setting of key.
```

7.5.2. disget Function (Getting a Constraint from the Database)

```
*deck,disget
      function disget (inode,idf,value)
c *** primary function:
                          get a constraint from the data base (in raw form)
c *** Notice - This file contains ANSYS Confidential information ***
С
      input arguments:
С
      variable (typ,siz,intent)
                                    description
         inode
                  (int,sc,in)
                                    - node number (negative value for no
С
                                                   partabeval)
C
        idf
                 (int,sc,in)
                                   - reference number for the DOF: (1-32)
C
    UX = 1, UY = 2, UZ = 3, ROTX = 4, ROTY = 5, ROTZ = 6, AX = 7, AY = 8
C
     AZ = 9, VX = 10, VY = 11, VZ = 12
C
     PRES=19, TEMP=20, VOLT=21, MAG =22, ENKE=23, ENDS=24
С
     EMF =25, CURR=26 SP01=27, SP02=28, SP03=29, SP04=30, SP05=31, SP06=32
С
                                  (missing entries are spares)
C
С
     output arguments:
                 (int,func,out)
                                    - status of constraint.
С
         disget
                                      = 0 - no constraint on this node
С
                                             for this DOF
C
C
                                      = 4 - this node has a constraint
С
                                             defined for this DOF
C
                                      = -4 - this node has a pseudo-support
С
                                             defined for this DOF
                                    - constraint values
С
         value
                  (dp,ar(4),out)
                          value(1-2) - (real,imag) values of present settings
C
                          value(3-4) - (real,imag) values of previous settings
```

7.5.3. disput Subroutine (Storing a Constraint at a Node)

```
*deck.disput
     subroutine disput (node,idf,value)
c *** primary function:
                          store a constraint at a node.
c *** Notice - This file contains ANSYS Confidential information ***
      input arguments:
C
С
        node
                 (int,sc,in)
                                    - node number
        idf
                                    - reference number of DOF: (1-32)
С
                 (int,sc,in)
         UX = 1, UY = 2, UZ = 3, ROTX = 4, ROTY = 5, ROTZ = 6, AX = 7, AY = 8
C
C
         AZ = 9, VX = 10, VY = 11, VZ = 12
                            PRES=19, TEMP=20, VOLT=21, MAG =22, ENKE=23, ENDS=24
C
         EMF = 25, CURR = 26
                                    (missing entries are spares)
C
         value
                  (dp,ar(2),in)
                                    - (real, imag) values for constraint
C
      output arguments: none.
```

7.5.4. disdel Subroutine (Deleting a Constraint at a Node)

```
*deck,disdel
     subroutine disdel (node,idf)
c *** primary function:
                          delete a constraint at a node
c *** Notice - This file contains ANSYS Confidential information ***
C
      input arguments:
                                   - node number.
        node
              (int,sc,in)
C
                                   - reference number of DOF: (1-32)
C
                  (int,sc,in)
         UX = 1, UY = 2, UZ = 3, ROTX= 4, ROTY= 5, ROTZ= 6, AX = 7, AY = 8
C
         AZ = 9, VX = 10, VY = 11, VZ = 12
C
                           PRES=19, TEMP=20, VOLT=21, MAG =22, ENKE=23, ENDS=24
         EMF = 25. CURR= 26
С
                                    (missing entries are spares)
     output arguments: none.
```

7.5.5. forigr Function (Getting Information About Nodal Loads)

```
*deck,forigr
     function forigr (node, key)
c *** primary function: get information about nodal loads.
c *** Notice - This file contains ANSYS Confidential information ***
C
      input arguments:
         node
                  (int,sc,in)
                                    - number of node being inquired about.
C
                                       should be 0 for key=DB_MAXDEFINED or
C
                                       DB_NUMDEFINED
С
         key
                  (dp,sc,in)
                                     - key as to information needed
C
                               = 1
                                                - return force mask for node
C
C
                               = DB MAXDEFINED,
С
                                DB_NUMDEFINED - return number of nodal loadings
                                                   in model
C
                                      NOTE: both DB_MAXDEFINED and DB_NUMDEFINED
C
C
                                      produce the same functionality
      output arguments:
C
         forigr (int, func, out)
                                    - the returned value of forigr is based on
C
                                       setting of key.
C
```

7.5.6. forget Function (Getting a Constraint from the Database)

```
*deck,forget
     function forget (inode, idf, value)
c *** primary function:
                         get a force from the data base (in raw form)
c *** Notice - This file contains ANSYS Confidential information ***
      input arguments:
C
         inode
                 (int,sc,in)
                                    - node number (negative value for no
С
C
                                                   partabeval)
C
         idf
                  (int,sc,in)
                                    - reference number for the DOF: (1-32)
С
                                      (see echprm.inc)
     output arguments:
C
С
         forget (int,func,out) - status of constraint.
С
                                    = 0 - no loading on this node for this DOF
С
                                    = 4 - this node has a loading for this DOF
         value
                  (dp,ar(4),out)
C
                          value(1-2) - (real,imag) values of present settings
C
                          value(3-4) - (real, imag) values of previous settings
С
```

7.5.7. forput Subroutine (Storing a Nodal Load at a Node)

```
*deck,forput
     subroutine forput (node,idf,value)
                        store a nodal load at a node
c *** primary function:
c *** Notice - This file contains ANSYS Confidential information ***
     input arguments:
C
        node
                 (int,sc,in)
                                   - node number
C
        idf
                                   - reference number for the DOF: (1-32)
C
                 (int,sc,in)
         FX = 1, FY = 2, FZ = 3, MX = 4, MY = 5, MZ = 6, CSGX= 7, CSGY= 8
C
С
         CSGZ= 9, VFX =10, VFY =11, VFZ =12
                           FLOW=19, HEAT=20, AMPS=21, FLUX=22, NPKE=23, NPDS=24
C
         CURT=25, VLTG=26
                                    (missing entries are spares)
С
        value
                                   - (real, imag) values of force
                 (dp,ar(2),in)
C
C
     output arguments: none.
```

7.5.8. fordel Subroutine (Deleting a Nodal Load at a Node)

```
*deck,fordel
     subroutine fordel (node,idf)
c *** primary function: delete a nodal load at a node
c *** secondary functions: none.
c *** Notice - This file contains ANSYS Confidential information ***
     typ=int,dp,log,chr,dcp siz=sc,ar(n),func
C
                                                 intent=in.out.inout
С
     input arguments:
     variable (typ,siz,intent)
C
                                   description
                (int,sc,in)
                                   - node number
С
С
                 (int,sc,in)
                                   - reference number for the DOF: (1-32)
         FX = 1, FY = 2, FZ = 3, MX = 4, MY = 5, MZ = 6, CSGX = 7, CSGY = 8
C
         CSGZ= 9, VFX =10, VFY =11, VFZ =12
С
                           FLOW=19, HEAT=20, AMPS=21, FLUX=22, NPKE=23, NPDS=24
         CURT=25, VLTG=26
                                    (missing entries are spares)
C
С
    output arguments:
С
        none.
```

7.5.9. ntpiqr Function (Getting Information About a Nodal Temperature)

```
*deck,ntpiqr
     function ntpigr (node, key)
                          get information about a nodal temperature
c *** primary function:
c *** Notice - This file contains ANSYS Confidential information ***
С
     typ=int,dp,log,chr,dcp siz=sc,ar(n),func
                                                 intent=in,out,inout
     input arguments:
С
С
     variable (typ,siz,intent)
                                   description
С
                (int,sc,in)
                                   - node number
                                     should be zero for key=2
C
        kev
                 (int,sc,in)
                                    - key for operation
C
С
                                      = 1 - return temperature status
                                      ntpiqr = 0 - node has no temperature
С
```

```
constraint defined
C
                                               = 1 - node has a temperature
С
С
                                                      constraint defined
                                       = 2 - return total number of nodal
С
                                              temperatures defined in model
С
     output arguments:
C
        ndingr (int, func, out)
                                    - the returned value of ndingr is based on
                                        setting of key.
C
```

7.5.10. ntpget Function (Getting a Specified Nodal Temperature)

```
*deck,ntpget
     function ntpget (node, tmp)
c *** primary function:
                        get specified nodal heat generation (in raw form)
c *** Notice - This file contains ANSYS Confidential information ***
     input arguments:
        node
                (int,sc,in)
                                  - node number
С
     output arguments:
        ntpget (int,func,out)
                                   - heat generation status of node.
                                    = 0 - nodal heat generation undefined
C
                                    = 1 - nodal heat generation is defined
С
        tmp
                  (dp,ar(2),out) - the nodal heat generation (new,old).
```

7.5.11. ntpput Subroutine (Storing a Nodal Temperature)

7.5.12. ntpdel Subroutine (Deleting a Nodal Temperature)

```
*deck,ntpdel
     subroutine ntpdel (node)
c *** primary function: delete node temperatures.
c *** Notice - This file contains ANSYS Confidential information ***
C
     typ=int,dp,log,chr,dcp siz=sc,ar(n),func
С
                                                  intent=in,out,inout
С
     input arguments:
C
     variable (typ,siz,intent)
                               description
       node (int,sc,in)
                                 - node number
С
С
     output arguments:
С
        none.
```

7.5.13. nhgiqr Function (Getting Information About Nodal Heat Generations)

```
*deck,nhgiqr
     function nhgiqr (node, key)
                          get information about nodal heat generations
c *** primary function:
c *** Notice - This file contains ANSYS Confidential information ***
      typ=int,dp,log,chr,dcp siz=sc,ar(n),func
                                                    intent=in,out,inout
      input arguments:
С
C
      variable (typ,siz,intent)
                                   description
        node
                 (int,sc,in)
                                   - node number
C
                                     should be 0 for key=2
С
                                   - key for operation
С
        key
                  (int,sc,in)
                           = 1 - return whether node has a heat generation rate
С
                                   defined
C
                                nhgiqr = 0 - no heat generation defined for node
C
                                      = 1 - heat generation is defined for node
С
С
                            = 2 - return total number of nodal heat generation
                                   rates defined in model
C
С
      output arguments:
                                    - the returned value of nhgigr is based on
        nhgiqr (int,func,out)
C
                                       setting of key.
С
```

7.5.14. nhgget Function (Getting a Nodal Heat Generation)

```
*deck,nhgget
     function nhgget (node,hg)
                         get specified nodal heat generation (in raw form)
c *** primary function:
c *** Notice - This file contains ANSYS Confidential information ***
     typ=int,dp,log,chr,dcp siz=sc,ar(n),func intent=in,out,inout
С
     input arguments:
С
     variable (typ,siz,intent)
                                   description
С
        node
                 (int,sc,in)
                                   - node number
     output arguments:
C
        nhgget (int,func,out)
                                   - heat generation status of node.
С
                                     = 0 - nodal heat generation undefined
                                     = 1 - nodal heat generation is defined
C
                 (dp,ar(2),out)
                                   - the nodal heat generation (new,old).
        hg
```

7.5.15. nhgput Subroutine (Storing Nodal Heat Generation)

7.5.16. nhgdel Subroutine (Deleting a Nodal Heat Generation)

```
*deck,nhgdel
     subroutine nhgdel (node)
c *** primary function:
                          delete nodal heat generations.
c *** Notice - This file contains ANSYS Confidential information ***
C
С
      typ=int,dp,log,chr,dcp siz=sc,ar(n),func
                                                   intent=in,out,inout
C
     input arguments:
C
     variable (typ,siz,intent)
C
                                description
        node
              (int.sc.in)
                                 - node number
C
С
     output arguments:
С
        none.
```

7.5.17. nfuigr Function (Getting Information About Nodal Fluences)

```
*deck,nfuigr
     function nfuiqr (node,key)
c *** primary function:
                         get information about nodal fluences
c *** Notice - This file contains ANSYS Confidential information ***
      typ=int,dp,log,chr,dcp siz=sc,ar(n),func
С
                                                  intent=in,out,inout
С
      input arguments:
C
     variable (typ,siz,intent)
                                  description
С
                 (int,sc,in)
                                   - node number
С
                                     should be zero for key=2
                                   - key for operation
                  (int,sc,in)
С
        key
                          = 1 - return status:
С
С
                           nfuigr = 0 - node does not have a fluence constraint
                                  = 1 - node has a fluence constraint
C
                          = 2 - return total number of nodal fluences defined on
С
С
                                 model
      output arguments:
C
С
        nfuigr (int,func,out)
                                    - the returned value of nfuigr is based on
                                       setting of key.
```

7.5.18. nfuget Function (Getting a Nodal Fluence)

```
*deck,nfuget
     function nfuget (node, fluen)
c *** primary function:
                        get specified nodal fluence.
c *** Notice - This file contains ANSYS Confidential information ***
      input arguments:
C
        node
                (int,sc,in)
                                   - node number
С
     output arguments:
C
                                   - fluence status of node.
С
        nfuget (int,func,out)
С
                                     = 0 - node has no fluence constraint
                                     = 1 - node has a fluence constaint
C
         fluen
                 (dp ,ar(2),out)
                                  - the nodal fluences (new,old).
```

7.5.19. nfuput Subroutine (Storing a Nodal Fluence)

```
*deck,nfuput
    subroutine nfuput (node,fluen)
c *** primary function: store nodal fluence.

c *** Notice - This file contains ANSYS Confidential information ***
c input arguments:
    node (int,sc,in) - node number
c fluen (dp,sc,in) - nodal fluence
c output arguments: none.
```

7.5.20. nfudel Subroutine (Deleting a Nodal Fluence)

```
*deck,nfudel
      subroutine nfudel (node)
c *** primary function:
                         delete node fluences.
c *** Notice - This file contains ANSYS Confidential information ***
С
      typ=int,dp,log,chr,dcp siz=sc,ar(n),func
С
                                                 intent=in,out,inout
C
      input arguments:
С
     variable (typ,siz,intent) description
С
                                  - node number
С
        node
               (int,sc,in)
С
С
     output arguments:
С
        none.
```

7.5.21. ndciqr Function (Getting Information About Nodal Current Densities)

```
*deck,ndciqr
     function ndciqr (node, key)
c *** primary function:
                          get information about nodel current densities
c *** Notice - This file contains ANSYS Confidential information ***
      typ=int,dp,log,chr,dcp siz=sc,ar(n),func
C
                                                  intent=in.out.inout
С
      input arguments:
      variable (typ,siz,intent)
                                    description
C
С
        node
                 (int,sc,in)
                                   - node number
C
                                     should be zero for key=2
                                 - key for operation
                  (int.sc.in)
        kev
C
                        = 1 - return nodal current status:
С
С
                           ndciqr = 0 - no current density defined for this node
C
                                  = 1 - node has a current density defined
                        = 2 - total number of nodal current densities defined
С
                               on model
C
      output arguments:
C
                 (int,func,out)
                                    - the returned value of ndciqr is based on
С
С
                                       setting of key.
```

7.5.22. ndcget Function (Getting a Nodal Current Density)

```
*deck,ndcget
     function ndcget (node, currd)
                        get specified nodal current density.
c *** primary function:
c *** Notice - This file contains ANSYS Confidential information ***
     input arguments:
        node
              (int,sc,in)

    node number

C
C
     output arguments:
       ndcget (int.func.out)
                                  - current density status of node.
C
                                    = 0 - node has no current density defined
                                    = 1 - node has a current density defined
С
        currd
               (dp,ar(4,2),out) - the node current density (new,old).
```

7.5.23. ndcput Subroutine (Storing a Nodal Current Density)

```
*deck,ndcput
    subroutine ndcput (node,currd)
c *** primary function: store nodal current density.

c *** Notice - This file contains ANSYS Confidential information ***

c input arguments:
    node (int,sc,in) - node number
    c currd (dp,ar(4),in) - nodal current densities

c output arguments: none.
```

7.5.24. ndcdel Subroutine (Deleting a Nodal Current Density)

```
*deck,ndcdel
     subroutine ndcdel (node)
c *** primary function: delete nodal current densities
c *** Notice - This file contains ANSYS Confidential information ***
С
     typ=int,dp,log,chr,dcp siz=sc,ar(n),func intent=in,out,inout
C
С
     input arguments:
С
    variable (typ, siz, intent) description
C
С
              (int,sc,in)
                                 - node number
C
    output arguments:
C
        none.
```

7.5.25. nvdiqr Function (Getting Information About Nodal Magnetic Virtual Displacements)

```
С
     input arguments:
С
     variable (typ,siz,intent) description
С
        node
              (int,sc,in)
                                  - node number
                                     should be zero for key=2
С
С
        key
                 (int,sc,in)
                                   - key for operation
                     = 1 - return magnetic virtual displacement status
С
                         nvdiqr = 0 - no mag. virt. disps defined for this node
                                = 1 - node has mag. virt. disps defined
C
С
                      = 2 - return total number of nodal magnetic virtual
                            displacements defined on model
С
С
     output arguments
С
        nvdiqr (int,func,out)
                                   - the returned value of nvdiqr is based on
                                      setting of key.
C
```

7.5.26. nvdget Function (Getting a Nodal Magnetic Virtual Displacement)

```
*deck,nvdget
     function nvdget (node, virtd)
c *** primary function: get specified nodal magnetic virtual displacement
c *** Notice - This file contains ANSYS Confidential information ***
     input arguments:
C
        node
                (int,sc,in)
                                   - node number
С
С
     output arguments:
        nvdget (int,func,out)
                                   - virtual disp status of node.
C
                                     = 0 - node has no magnetic virtual
C
                                            displacement
С
                                      = 1 - node has a magnetic virtual
                                            displacement
С
        virtd
                                    - the nodal virtual displacement value
                 (dp ,sc,out)
```

7.5.27. nvdput Subroutine (Storing a Nodal Virtual Displacement)

7.5.28. nvddel Subroutine (Deleting a Nodal Virtual Displacement)

```
*deck,nvddel
     subroutine nvddel (node)
c *** primary function:
                         delete nodal virtual displacements.
c *** Notice - This file contains ANSYS Confidential information ***
С
С
     typ=int,dp,log,chr,dcp siz=sc,ar(n),func intent=in,out,inout
C
     input arguments:
C
     variable (typ,siz,intent)
С
                                  description
        node
                (int,sc,in)
                                  - node number
С
```

```
c
c output arguments:
c none.
```

7.6. Element Loading Routines

7.6.1. epriqr Function (Getting Information About Element Pressure/Convection)

```
*deck,epriqr
     function epriqr (ielem, iface, key)
c *** primary function: get information about element pressure/convection
c *** Notice - This file contains ANSYS Confidential information ***
      input arguments:
C
С
         ielem
                  (int,sc,in)
                                    - element number
                                       should be zero for key=DB_NUMDEFINED or
С
С
                                       DB_MAXRECLENG
         iface
                  (int,sc,in)
                                     - face number for inquire (0-6)
                                       face number is needed for key=5. for
C
                                       other values of key, iface has different
C
                                       meaning (see below)
C
C
         kev
                  (int,sc,in)
                                    - key as to the information needed
С
                  = 1
                                    - return pressure mask for element
С
                  = 5
                                    - return number of pressures for this
                                        element face
C
C
                  = DB_NUMDEFINED,
                  = DB_MAXDEFINED - return value is based on setting of iface
C
                                       NOTE: both DB_NUMDEFINED and
С
                                       DB_MAXDEFINED produce the same
C
С
                                      functionality
C
                              iface = 0 - return number of surface loads defined
                                     = 1-6 - return number of pressure loads
C
C
                                       defined for this element.
С
                                        NOTE: only 1-6 is valid, but this
                                        routine simply checks that iface is in
С
                                       the range. The actual value of iface
C
                                       does not matter in this case.
C
                  = DB_MAXRECLENG - return the maximum number of element
C
                                      pressures on any element (max record
С
                                       length)
С
      output arguments:
С
         epriqr (int,func,out)
                                    - the returned value of eprigr is based on
                                        setting of key.
```

7.6.2. eprget Function (Getting an Element Face Pressure)

```
*deck,eprget
      function eprget (elem,iface,value)
c *** primary function:
                         get an element face pressure
c *** Notice - This file contains ANSYS Confidential information ***
      input arguments:
C
                  (int,sc,in)
                                    - element number (negative value for
C
                                        no partabeval)
C
С
         iface
                  (int,sc,in)
                                    - face number (1-68)
C
      output arguments:
С
         eprget
                  (int,func,out)
                                    - status of element.
С
                                      =-1 - element has no pressures
                                       = 0 - this element face has no pressures
С
```

```
c > 0 - number of values defined c value (dp ,ar(*),out) - the element pressures (real,imag) at each
```

7.6.3. eprput Subroutine (Storing an Element Face Pressure)

```
*deck,eprput
     subroutine eprput (ielem,iface,nval,value)
c *** primary function:
                         store an element face pressure.
c *** Notice - This file contains ANSYS Confidential information ***
     input arguments:
        ielem (int,sc,in)
                                  - element number for operation.
С
        iface (int,sc,in)
                                  - face number (1-68)
C
С
                 (int,sc,in)
                                  - number of values to put
С
        value
                 (dp ,ar(nval),in) - the element pressures (real,imag) at each
                                     face
C
C
     output arguments: none.
```

7.6.4. eprdel Subroutine (Deleting an Element Pressure/Convection)

```
*deck,eprdel
      subroutine eprdel (ielem, iface)
c *** primary function:
                          delete a pressure/convection on an element
c *** Notice - This file contains ANSYS Confidential information ***
      input arguments:
        ielem (int,sc,in)
C
                                     - element number
С
        iface
                  (int,sc,in)
                                     - face number
С
                                       = 0 - delete all pressures on this
C
                                              element
                                       = 1-6 - delete pressure on this face
C
C
     output arguments: none.
```

7.6.5. ecviqr Function (Getting Information About Element Convections)

```
*deck,ecviqr
      function ecviqr (ielem,iface,key)
c *** primary function: get information about element convections
c *** Notice - This file contains ANSYS Confidential information ***
      input arguments:
С
C
        ielem (int,sc,in)
                                    - element number for inquire
С
                                       should be zero for key=DB_NUMDEFINED or
                                       DB MAXRECLENG
С
        iface
                  (int,sc,in)
                                    - face number for inquire (0-6)
C
                                       face number is needed for key=5. for
C
                                       other values of key, iface has different
С
                                        meaning (see below)
С
         key
                  (int,sc,in)
                                    - key as to the information needed
С
                                   - return convection mask for element
С
                  = 1
С
                  = 5
                                   - return number of convections for this
                                      element face
С
C
                  = DB NUMDEFINED,
С
                  = DB_MAXDEFINED - return value is based on setting of iface
                                      NOTE: both DB_NUMDEFINED and
С
```

```
DB_MAXDEFINED produce the same
C
С
                                       functionality
С
                                iface = 0 - return number of surface loads
                                             defined (rec length)
С
                                      = 1-6 - return number of convection loads
С
С
                                         defined for this element.
                                         NOTE: only 1-6 is valid, but this
С
C
                                         routine simply checks that iface is in
                                         the range. The actual value of iface
С
С
                                         does not matter in this case.
                  = DB_MAXRECLENG
                                   - return the maximum number of convections
C
                                       on any element (max rec length)
C
С
      output arguments:
         ecviar
                 (int.func.out)
                                     - the returned value of ecviqr is based on
C
С
                                        setting of key.
```

7.6.6. ecvget Function (Getting an Element Face Convection)

```
*deck,ecvget
       function ecvget (elem,iface,value)
c *** primary function:
                           get an element face convection (in raw form)
c *** Notice - This file contains ANSYS Confidential information ***
С
      input arguments:
                  (int,sc,in)
                                     - element number
C
         elem
С
         iface
                  (int,sc,in)
                                     - face number (1-6)
      output arguments:
C
C
         ecvget
                  (int,func,out)
                                     - status of element.
С
                                       =-1 - element has no convections/heat
                                              fluxes
С
                                       = 0 - this element face has no
C
                                              convections/heat fluxes
C
C
                                       > 0 - number of values defined
         value
                  (dp ,ar(*),out)
                                     - the element convections
C
С
                                      NOTE: Two values at each node of an
                                       element face: if loading is a convection,
C
                                       the first first value is the film
C
                                       coefficient and the second value is the
С
                                       bulk temperature. If loading is a heat
                                       flux, the first value is the heat flux,
С
C
                                       and the second value is a large number
                                       (2**100)
C
```

7.6.7. ecvput Subroutine (Storing an Element Face Convection)

```
*deck,ecvput
      subroutine ecvput (ielem,iface,nval,value)
c *** primary function:
                           store an element face convection.
c *** Notice - This file contains ANSYS Confidential information ***
      typ=int,dp,log,chr,dcp siz=sc,ar(n),func intent=in,out,inout
C
      input arguments:
C
      variable (typ,siz,intent)
                                   description
C
         ielem
                 (int,sc,in)
                                    - element number
C
         iface
                  (int,sc,in)
                                    - face number (1-6)
C
                                    - number of values to put
C
         nval
                  (int,sc,in)
                  (dp ,ar(nval),in) - the element convections.
C
         value
                                      NOTE: Two values at each node of an
C
                                       element face: if loading is a convection,
C
С
                                       the first first value is the film
                                       coefficient and the second value is the
С
```

```
c bulk temperature. If loading is a heat flux, the first value is the heat flux, and the second value is a large number c (2**100)
c output arguments:
c none.
```

7.6.8. ecvdel Subroutine (Deleting a Convection on an Element)

```
*deck,ecvdel
      subroutine ecvdel (ielem, iface)
c *** primary function:
                          delete a convection on an element
c *** Notice - This file contains ANSYS Confidential information ***
      typ=int,dp,log,chr,dcp siz=sc,ar(n),func
      input arguments:
C
      variable (typ,siz,intent)
                                    description
        ielem
                  (int,sc,in)
С
                                     - element number.
С
                  (int,sc,in)
                                     - face number
                                       = 0 - delete all convections on this
                                              element
C
                                       = 1-6 - delete convections on this face
C
C
     output arguments:
С
        none.
```

7.6.9. etpiqr Function (Getting Information About Element Temperatures)

```
*deck,etpiqr
      function etpigr (ielem, key)
c *** primary function: get information about element temperatures.
c *** Notice - This file contains ANSYS Confidential information ***
C
      input arguments:
С
                 (int,sc,in)
                                     - element number
                                        Should be 0 for key=11, DB_NUMDEFINED,
C
                                        DB_MAXDEFINED, and DB_MAXRECLENG
C
С
         kev
                  (int,sc,in)
                                    - information flag.
                  = DB_SELECTED
С
                                   - return status:
                       etpiqr = 0 - element has no temperatures
С
                              = 1 - element has temperatures defined
С
                  = DB NUMDEFINED - return number of temperatures defined for
C
C
                                      this element (rec length)
                  = DB_MAXDEFINED - return number of temperatures defined in
C
C
                                      model
С
                  = DB_MAXRECLENG - return maximum number of temperatures
С
                                       defined for any element (max rec length)
                  = 2 - return length (dp words)
C
                  = 3 - return layer number (for cross reference files return
С
С
                              number of entities)
                  = 4 - return address of first data word
С
С
                  = 5 - return length (dp words)
                  = 6 - return compressed record number.
C
                  = 11 - return void percent (integer)
C
                  = 16 - return location of next record (this increments the
С
                              next record count)
C
                  = 18 - return type of file.
С
                           etpiqr = 0 - integer
С
                                   = 1 - double precision
C
                                   = 2 - real
С
С
                                   = 3 - complex
С
                                   = 4 - character*8
```

```
= 7 - index
C
                  = 19 - return virtual type of file.
С
С
                            etpiqr = 0 - fixed length (4.4 form)
                                   = 1 - indexed variable length
С
                                           (layer data)
С
                                   = 2 - xref data tables
С
                                   = 3 - bitmap data (for 32 data item packed
C
                                           records)
                                   = 4 - data tables (three dimensional arrays)
C
     output arguments:
C
                                     - the returned value of etpiqr is based on
С
         etpiqr (int,func,out)
С
                                        setting of key.
```

7.6.10. etpget Function (Getting an Element Temperature)

```
*deck,etpget
     function etpget (ielem,tem)
c *** primary function:
                          get element temperatures (in raw form)
c *** Notice - This file contains ANSYS Confidential information ***
C
      input arguments:
С
        ielem
                 (int,sc,in)
                                    - element number
      output arguments:
C
         etpget (int,func,out)
                                    - status of element.
С
С
                                      = 0 - this element has no element
С
                                             temperatures
C
                                      > 0 - number of element temperatures
C
                                             retrieved
                 (dp,ar(n,2),out) - the element temperatures (new,old).
С
         tem
     NOTE THAT TEM MUST DOUBLE THE NUMBER OF DESIRED
C
     TEMPERATURES IN THE CALLING ROUTINE!
С
                                      NOTE: If a value is not defined (i.e.,
C
С
                                            defaults to TUNIF), value will be a
                                            very small number (2**-100)
C
```

7.6.11. etpput Subroutine (Storing an Element Temperature)

```
*deck,etpput
     subroutine etpput (ielem,n,temp)
c *** primary function:
                        store element temperatures.
c *** Notice - This file contains ANSYS Confidential information ***
С
      input arguments:
                                  - element number
C
        ielem (int,sc,in)
                 (int,sc,in)
                                   - number of element temperature values
C
        n
                 (dp ,ar(n),in)
                                   - element temperatures.
     output arguments: none.
C
С
                                     NOTE: If a value is not defined (i.e.,
                                           defaults to TUNIF), a very small
С
                                           number should be used (2**-100)
```

7.6.12. etpdel Subroutine (Deleting an Element Temperature)

```
*deck,etpdel
```

```
subroutine etpdel (ielem)

c *** primary function: delete element temperatures.

c *** Notice - This file contains ANSYS Confidential information ***

c input arguments:
    ielem (int,sc,in) - element number

c output arguments: none.
```

7.6.13. ehgiqr Function (Getting Information About Element Heat Generation)

```
*deck,ehgiqr
      function ehgiqr (ielem,key)
c *** primary function: get information about element heat generations.
c *** Notice - This file contains ANSYS Confidential information ***
C
      input arguments:
С
         ielem
                (int,sc,in)
                                    - element number
С
                                        should be 0 for key=11, DB_NUMDEFINED,
                                        DB_MAXDEFINED, and DB_MAXRECLENG
С
                  (int,sc,in)
                                    - information flag.
C
         key
                  = DB_SELECTED
                                   - return status:
С
                       ehgiqr = 0 - heat generation is undefined
С
                              = 1 - heat generation is defined
C
                  = DB_NUMDEFINED - return number of defined heat generations
С
С
                                       in model
                  = DB_MAXRECLENG - return maximum number of heat generations
С
C
                                       on any element (max rec length)
                  = 2 - return length (dp words)
C
С
                  = 3 - return layer number (for cross reference files return
                                        number of entities)
С
                  = 4 - return address of first data word
С
С
                  = 5 - return length (record type units)
С
                  = 6 - return compressed record number.
C
                  = 11 - return void percent (integer)
С
                  = 16 - return location of next record (this increments the
С
                                       next record count)
                  = 18 - return type of file.
С
                       ehgiqr = 0 - integer
С
                              = 1 - double precision
С
                              = 2 - real
С
С
                              = 3 - complex
                              = 4 - character*8
С
                               = 7 - index
С
                  = 19 - return virtual type of file.
C
                       ehgiqr = 0 - fixed length (4.4 form)
C
                               = 1 - indexed variable length
С
С
                                                (layer data)
                              = 2 - xref data tables
C
                               = 3 - bitmap data (for 32 data
С
С
                                                item packed records)
С
                               = 4 - data tables (three
                                                dimensional arrays)
C
С
      output arguments:
         ehgiqr (int,func,out)
                                    - the returned value of ehgiqr is based on
C
                                        setting of key.
С
```

7.6.14. ehgget Function (Getting an Element Heat Generation)

```
*deck,ehgget function ehgget (ielem,ggen)
```

```
c *** primary function:
                          get element heat generations (in raw form)
c *** Notice - This file contains ANSYS Confidential information ***
С
      input arguments:
С
         ielem (int,sc,in)
                                    - element number
C
      output arguments:
         ehgget (int,func,out)
                                   - status of element.
C
С
                                      = 0 - heat generations undefined for this
                                             element
C
                                      > 0 - number of heat generations defined
C
С
                  (dp ,ar(*),out)
                                    - the element heat generations.
         ggen
С
                                      NOTE: If a value is not defined, it will
                                            be a very small number (2**-100)
```

7.6.15. ehgput Subroutine (Storing an Element Heat Generation)

```
*deck,ehgput
      subroutine ehgput (ielem,n,qgen)
c *** primary function:
                          store element heat generations
c *** Notice - This file contains ANSYS Confidential information ***
С
      input arguments:
        ielem (int,sc,in)
                                    - element number
C
С
                  (int,sc,in)
                                    - number of element heat generation values
                  (dp ,ar(n),in)
                                    - element heat generations
C
        qgen
С
     output arguments: none
                                      NOTE: If a value is not defined, a very
C
                                         small number should be used (2**-100)
С
```

7.6.16. ehgdel Subroutine (Deleting an Element Heat Generation)

```
*deck,ehgdel
     subroutine ehgdel (ielem)
c *** primary function:
                        delete element heat generations.
c *** Notice - This file contains ANSYS Confidential information ***
     typ=int,dp,log,chr,dcp siz=sc,ar(n),func
                                                  intent=in,out,inout
      input arguments:
     variable (typ,siz,intent)
                                 description
C
С
        ielem (int,sc,in)
                                   - element number
     output arguments:
C
        none
```

7.6.17. efuiqr Function (Getting Information About Element Fluences)

```
*deck,efuiqr
    function efuiqr (ielem,key)
c *** primary function: get information about element fluences

c *** Notice - This file contains ANSYS Confidential information ***

c input arguments:
    ielem (int,sc,in) - element number or zero (see below)
    key (int,sc,in) - key as to the information needed
```

```
= 1 or DB_MAXRECLENG - return element fluences info
С
                          for ielem > 0 - return number of fluences for this
С
С
                                           element (record length)
                                    = 0 - return maximum number of fluences
С
                                           defined for any element
С
С
                                           (max rec length)
                  = DB_NUMDEFINED,
C
                  = DB_MAXDEFINED - return number of defined fluences
C
                                      in model
С
                                     NOTE: both DB_NUMDEFINED and DB_MAXDEFINED
                                           produce the same functionality
С
С
      output arguments:
С
         efuigr (int,func,out)
                                    - the returned value of efuigr is based on
                                        setting of key
C
```

7.6.18. efuget Function (Getting an Element Fluence)

```
*deck,efuget
      function efuget (ielem, value)
c *** primary function:
                         get element fluences.
c *** Notice - This file contains ANSYS Confidential information ***
      input arguments:
C
        ielem
               (int,sc,in)
                                    - element number
C
      output arguments:
C
        efuget (int,func,out)
                                    - status of element.
C
С
                                       = 0 - element has no fluences defined
                                       > 0 - number of element fluences defined
С
        value
                  (dp,ar(*),out)
                                    - element fluences.
С
                                      NOTE: If a value is not defined, it will
C
C
                                            be a very small number (2**-100)
```

7.6.19. efuput Subroutine (Storing an Element Fluence)

```
*deck,efuput
     subroutine efuput (ielem,n,value)
c *** primary function:
                          store element fluences
c *** Notice - This file contains ANSYS Confidential information ***
     input arguments:
C
С
        ielem (int,sc,in)
                                    - element number
                 (int,sc,in)
                                    - the number of values to store
С
        value
                 (dp,ar(n),in)
                                     - element fluences.
C
С
     output arguments: none
                                      NOTE: If a value is not defined, a very
С
С
                                         small number should be used (2**-100)
```

7.6.20. efudel Subroutine (Deleting an Element Fluence)

```
*deck,efudel
    subroutine efudel (ielem)
c *** primary function:    delete element fluences
c *** Notice - This file contains ANSYS Confidential information ***
```

```
c variable (typ,siz,intent) description
c ielem (int,sc,in) - element number
c output arguments: none
```

7.6.21.edciqr Function (Getting Information About Element Current Densities)

```
*deck,edciqr
      function edcigr (ielem,key)
c *** primary function:
                          get information about element current densities
c *** Notice - This file contains ANSYS Confidential information ***
C
      input arguments:
С
                  (int,sc,in)
                                    - element number or zero (see below)
С
         key
                  (int,sc,in)
                                    - key as to the information needed
                  = 1 or DB MAXRECLENG - return element densities info
C
                          for ielem > 0 - number of current densities for this
C
                                           element (rec length)
С
                                    = 0 - maximum number of current densities
                                           defined for any element
                                            (max rec length)
C
                  = DB_NUMDEFINED,
C
С
                  = DB_MAXDEFINED - return total number of current densities
C
                                       defined in model
                                     NOTE: both DB_NUMDEFINED and DB_MAXDEFINED
С
                                           produce the same functionality
C
С
      output arguments:
         edciqr (int,func,out)
                                    - the returned value of edcigr is based on
C
                                        setting of key
С
```

7.6.22. edcget Function (Getting Element Current Densities)

```
*deck,edcget
      function edcget (ielem, value)
c *** primary function:
                         get element current densities
c *** Notice - This file contains ANSYS Confidential information ***
      typ=int,dp,log,chr,dcp siz=sc,ar(n),func intent=in,out,inout
                 (int,sc,in)
                                    - element number
         ielem
C
      output arguments:
                                      = 0 - element has no current densities
С
                                             defined
C
                                      > 0 - number of element current
C
С
                                             densities defined
                  (dp,ar(*),out)
                                    - element current densities
C
         value
                                      NOTE: If a value is not defined, it will
C
                                            be a very small number (2**-100)
```

7.6.23. edcput Subroutine (Storing an Element Current Density)

```
input arguments:
С
С
        ielem (int,sc,in)
                                    - element number
С
                  (int,sc,in)
                                    - the number of current densities to store
                 (dp,ar(n),in)
                                    - element current densities
С
        value
С
     output arguments:
C
        none
                                     NOTE: If a value is not defined, a very
                                        small number should be used (2**-100)
C
```

7.6.24. edcdel Subroutine (Deleting an Element Current Density)

```
*deck,edcdel
    subroutine edcdel (ielem)
c *** primary function:    delete element current densities

c *** Notice - This file contains ANSYS Confidential information ***
c    input arguments:
c    ielem (int,sc,in) - element number

c    output arguments: none
```

7.6.25. evdiqr Function (Getting Information About Element Virtual Displacements)

```
*deck.evdiar
     function evdigr (ielem,key)
c *** primary function: get information about element virt disps
c *** Notice - This file contains ANSYS Confidential information ***
      input arguments:
C
C
                (int,sc,in)
                                    - element number or zero (see below)
С
         key
                  (int,sc,in)
                                   - key as to the information needed
                  = 1 or DB_MAXRECLENG - return element virt disps info
C
                          for ielem > 0 - number of virt disps defined for this
C
                                           element (rec length)
С
                                    = 0 - maximum number of virt disps defined
C
                                           for any element (max rec length)
C
                  = DB NUMDEFINED,
С
                  = DB_MAXDEFINED - return total number of virt disps defined
С
                                      in model
                                     NOTE: both DB_NUMDEFINED and DB_MAXDEFINED
C
С
                                           produce the same functionality
С
      output arguments:
С
         evdiqr (int,func,out)
                                    - the returned value of evdiqr is based on
                                        setting of key
```

7.6.26. evdget Function (Getting an Element Virtual Displacement)

```
*deck,evdget
    function evdget (ielem,value)
c *** primary function: get element virtual displacements

c *** Notice - This file contains ANSYS Confidential information ***
c input arguments:
c ielem (int,sc,in) - element number
c
c output arguments:
```

```
- status of element.
C
         evdget
                  (int,func,out)
                                       = 0 - no virt disps defined for this
С
С
                                              element
                                       > 0 - number of element virtual
С
C
                                              displacements
С
         value
                  (dp,ar(*),out)
                                       element virtual displacements
С
                                       NOTE: If a value is not defined, it will
                                             be a very small number (2**-100)
```

7.6.27. evdput Subroutine (Storing an Element Virtual Displacement)

```
*deck,evdput
      subroutine evdput (ielem,n,value)
c *** primary function:
                           store element virtual displacements
c *** Notice - This file contains ANSYS Confidential information ***
      input arguments:
C
         ielem
                 (int,sc,in)
                                     - element number
                  (int,sc,in)
                                     - the total number of values
C
         n
С
         value
                  (dp,ar(n),in)
                                     - element virtual displacments
C
      output arguments: none
                                      NOTE: If a value is not defined, a very
С
С
                                         small number should be used (2**-100)
```

7.6.28. eimigr Function (Getting Information About Element Impedances)

```
*deck,eimigr
      function eimigr (ielem,iface,key)
c *** primary function: get information about element impedences
c *** Notice - This file contains ANSYS Confidential information ***
      input arguments:
C
                                     - element number for inquire.
         ielem
                  (int,sc,in)
C
С
                                         should be zero for key=DB_NUMDEFINED,
                                         DB_MAXDEFINED or DB_MAXRECLENG
C
C
         iface
                  (int,sc,in)
                                     - face number for inquire (0-6)
С
                                         face number is needed for key=5. for
                                         other values of key, iface has different
C
C
                                         meaning (see below)
                                    - key as to the information needed
C
         key
                  (int,sc,in)
                  = 1
                                    - return impedence mask for element
С
                     5
                                    - return number of impedences for this
C
                                        element face
C
C
                  = DB_NUMDEFINED,
                  = DB_MAXDEFINED
                                   - return value is based on setting of iface
C
                                       NOTE: both DB_NUMDEFINED and
C
С
                                       DB_MAXDEFINED produce the same
C
                                       functionality
                               iface = 0 - return number of surface loads defined
C
С
                                           in model
                                     = 1-6 - return number of pressure loads
C
                                         defined for this element. (rec length)
С
                                         NOTE: only 1-6 is valid, but this
С
                                         routine simply checks that iface is in
C
                                         the range. The actual value of iface
C
                                         does not matter in this case.
C
                                   - return the maximum number of element
                  = DB MAXRECLENG
C
                                       impedences defined for any element
C
C
                                       (max rec length)
      output arguments:
C
C
         eimigr
                  (int,func,out)
                                     - the returned value of eimigr is based on
```

C

setting of key.

7.6.29. eimget Function (Getting an Element Face Impedance)

```
*deck,eimget
     function eimget (ielem,iface,value)
c *** primary function: get an element face impedance
c *** Notice - This file contains ANSYS Confidential information ***
     input arguments:
С
        ielem
                 (int,sc,in)
                                   - element number
C
        iface
                 (int,sc,in)
                                   - face number (1-6)
C
С
     output arguments:
                                   - status of element.
        eimget (int,func,out)
C
                                     =-1 - element has no impedances
С
С
                                     = 0 - this element face has no impedances
                                     > 0 - number of values defined
C
        value
                 (dp ,ar(*),out)
                                  - the element impedances (real, imag)
```

7.6.30. eimput Subroutine (Storing an Element Impedance)

```
*deck,eimput
     subroutine eimput (ielem,iface,nval,value)
c *** primary function:
                         store an element face impedance.
c *** Notice - This file contains ANSYS Confidential information ***
     input arguments:
C
        ielem (int,sc,in)
                                  - element number
        iface (int,sc,in)
                                  - face number (1-6)
C
С
        nval
                 (int,sc,in)
                                  - number of values to put
        value
                 (dp ,ar(nval),in) - the element impedances (real,imag)
     output arguments: none
```

7.6.31. eimdel Subroutine (Deleting an Element Impedance)

```
*deck,eimdel
     subroutine eimdel (ielem,iface)
c *** primary function:
                          delete an impedance on a element
c *** Notice - This file contains ANSYS Confidential information ***
     input arguments:
С
        ielem (int,sc,in)
                                     - element number
C
С
        iface
                  (int,sc,in)
                                     - face number
C
                                       = 0 - delete all impedances on this
С
                                             element
С
                                       = 1-6 - delete impedance on this face
     output arguments: none
```

7.6.32. esfiqr Function (Getting Information About Element Surface Stress Data)

```
*deck,esfiqr
     function esfigr (ielem, key)
c *** primary function:
                          get information about element surface stress data
c *** Notice - This file contains ANSYS Confidential information ***
С
      input arguments:
        ielem
                                    - element number (or zero, see below)
                  (int,sc,in)
C
С
         key
                 (int,sc,in)
                                   - key as to the information needed
                  = 1 - return info about surface stress
C
С
                          ielem > 0 - return number of surface stresses on this
С
                                       element (rec length)
                                = 0 - return maximum number of surface stresses
C
                                      on any element (max rec length)
                  = DB_NUMDEFINED - return the number of surface stresses
C
                                    defined in model
C
С
      output arguments:
                                    - the returned value of esfigr is based on
С
         esfigr (int,func,out)
                                        setting of key
```

7.6.33. esfget Function (Getting Element Surface Stress Data)

```
*deck,esfget
      function esfget (ielem, value)
c *** primary function:
                         get element surface stress data.
c *** Notice - This file contains ANSYS Confidential information ***
С
      input arguments:
        ielem (int.sc.in)
                                    - element number
C
С
     output arguments:
                                    - status of element.
        esfget (int,func,out)
C
                                      = 0 - element undefined
                                      > 0 - number of values returned
С
        value
                  (dp,ar(*),out)
                                    - element surface stress data.
```

7.6.34. esfput Subroutine (Storing Element Surface Stress Data)

```
*deck,esfput
     subroutine esfput (ielem,nval,value)
c *** primary function:
                          store surface stresses for an element.
c *** Notice - This file contains ANSYS Confidential information ***
C
      input arguments:
        ielem (int.sc.in)
                                    - element number
C
                                     - the total number of values
                   (int,sc,in)
С
                                       (19 * number of stress faces)
                                       There is a max of 2 stress faces
C
        value
                  (dp,ar(nval),in)
                                    - the values
     output arguments: none
```

7.6.35. esfdel Subroutine (Deleting an Element's Surface Stress Data)

7.6.36. efsdel Subroutine (Deleting a Flagged Surface on an Element)

```
*deck,efsdel
     subroutine efsdel (ielem,iface)
c *** primary function:
                         delete a flagged surface on an element
c *** Notice - This file contains ANSYS Confidential information ***
С
     input arguments:
       ielem (int,sc,in)
                                    - element number
C
                                    - face number
С
                                      = 0 - all flagged surfaces
                                      = 1-6 - this flagged surface
C
C
     output arguments: none.
```

7.6.37. efsget function (Getting Element Face Flagged Surfaces)

```
*deck,efsget
     function efsget (ielem,iface,value)
c *** primary function: get element face flagged surfaces
c *** Notice - This file contains ANSYS Confidential information ***
С
     input arguments:
C
        ielem (int,sc,in)
                                   - element number
С
        iface
                 (int,sc,in)
                                   - face number (1-6)
     output arguments:
С
        efsget (int,func,out)
                                   - status of element.
С
C
                                     =-1 - no values for this element
                                     = 0 - zero flagged surfaces defined
C
                                     > 0 - number of values defined
        value
                 (dp ,ar(*),out) - the element flagged surfaces
```

7.6.38. efsiqr function (Getting Information About Flagged Surfaces)

```
*deck,efsiqr
    function efsiqr (ielem,iface,key)
c *** primary function: get information about flagged surfaces
c *** Notice - This file contains ANSYS Confidential information ***
c    input arguments:
c    ielem (int,sc,in) - element number for inquire.
```

```
should be zero for key=DB_NUMDEFINED,
C
                                         DB_MAXDEFINED or DB_MAXRECLENG
С
С
         iface
                  (int,sc,in)
                                     - face number for inquire (0-6)
                                         face number is needed for key=5. for
С
                                         other values of key, iface has different
С
C
                                         meaning (see below)
         key
                  (int,sc,in)
                                     - key as to the information needed
С
                  = 1
C
                                    - return flagged surfaces mask for element
                  = 5
                                    - return number of flagged surfaces for this
C
С
                                        element face
                  = DB_NUMDEFINED,
C
                  = DB_MAXDEFINED
                                   - return value is based on setting of iface
C
С
                                       NOTE: both DB_NUMDEFINED and
С
                                       DB_MAXDEFINED produce the same
                                       functionality
C
                               iface = 0 - return total number of pressures,
С
C
                                            convections, etc defined in model
                                     = 1-6 - return number of flagged surfaces
C
                                         defined for this element. (rec length)
                                         NOTE: only 1-6 is valid, but this
C
                                         routine simply checks that iface is in
С
С
                                         the range. The actual value of iface
                                         does not matter in this case.
C
                  = DB_MAXRECLENG - return maximum number of flagged surfaces
С
                                         for any element (max rec length)
C
C
      output arguments:
                                     - the returned value of efsigr is based on
C
         efsiqr (int,func,out)
                                        setting of key.
C
```

7.6.39. efsput Subroutine (Storing an Element Face Flagged Surface)

```
*deck,efsput
      subroutine efsput (ielem,iface,nval,value)
c *** primary function:
                          store an element face flagged surface.
c *** Notice - This file contains ANSYS Confidential information ***
      input arguments:
C
                 (int,sc,in)
                                    - element number
        ielem
C
         iface
                  (int,sc,in)
                                    - face number (1-6)
С
         nval
                  (int,sc,in)
                                    - number of values to put
                  (dp ,ar(nval),in) - the element flagged surface values
C
      output arguments: none.
```

7.7. Results Information Routines

7.7.1. dspiqr Function (Getting Information About Nodal Results)

```
*deck,dspigr
      function dspiqr (node, key)
c *** primary function: get information about nodal results
c *** Notice - This file contains ANSYS Confidential information ***
С
      input arguments:
C
         node
                (int,sc,in)
                                    - node number
                                     > 0 - return result mask at this node
C
                                      = 0 - return number of calculated
С
                                             displacements in model
C
         key
                 (int,sc,in)
                                    - key as to the information needed
                                      At this time, key should always = 1
```

```
c output arguments:
c dspiqr (int,func,out) - the returned value of dspiqr is based on
c setting of key
```

7.7.2. dspget Function (Getting a Nodal Result from the Database)

```
*deck,dspget
     function dspget (node,ndf,idf,value)
c *** primary function:
                         get a nodal result from the data base
c *** Notice - This file contains ANSYS Confidential information ***
С
     input arguments:
        node
                                   - node number
C
                (int,sc,in)
        ndf
С
                 (int,sc,in)
                                   - number of results requested
                 (int,ary(ndf),in) - reference number for the DOF: (1-32)
C
        idf
    UX = 1, UY = 2, UZ = 3, ROTX = 4, ROTY = 5, ROTZ = 6, AX = 7, AY = 8
C
    AZ = 9, VX = 10, VY = 11, VZ = 12
                      PRES=19, TEMP=20, VOLT=21, MAG =22, ENKE=23, ENDS=24
С
С
    EMF = 25, CURR=26 SP01=27, SP02=28, SP03=29, SP04=30, SP05=31, SP06=32
                                  (missing entries are spares)
С
     output arguments:
        value
               (dp,ar(ndf),out) - result values
С
```

7.7.3. dspput Subroutine (Storing a Result at a Node)

```
*deck,dspput
     subroutine dspput (node,ndf,idf,value)
c *** primary function: store a result at a node.
c *** Notice - This file contains ANSYS Confidential information ***
С
     input arguments:
        node
              (int,sc,in)
                                   - node number
C
        ndf
C
                 (int.sc.in)
                                   - number of results to be stored
                 (int,ary(ndf),in) - reference number for the DOF: (1-32)
С
        idf
        value
                 (dp,ar(ndf),in)

    displacement values

     output arguments: none
```

7.7.4. dspdel Subroutine (Deleting a Result at a Node)

```
*deck,dspdel
     subroutine dspdel (node,ndf,idf)
c *** primary function:
                          delete a result at a node
c *** Notice - This file contains ANSYS Confidential information ***
     input arguments:
C
                 (int,sc,in)
                                   - node number. (0 to delete DOF at all
С
                                      nodes)
        ndf
                 (int,sc,in)
                                    - number of DOFs to delete (0 to delete
С
                                      all DOFs)
                                   - reference number for the DOF: (1-32)
С
        idf
                (int,ar(*),in)
         UX = 1, UY = 2, UZ = 3, ROTX = 4, ROTY = 5, ROTZ = 6, AX = 7, AY = 8
C
         AZ = 9, VX =10, VY =11, VZ =12
С
                           PRES=19, TEMP=20, VOLT=21, MAG =22, ENKE=23, ENDS=24
C
С
         EMF = 25, CURR = 26
                                     (missing entries are spares)
```

```
c output arguments: none
```

7.7.5. emsiqr Function (Getting Information About an Element's Miscellaneous Summable Data)

```
*deck,emsiqr
      function emsigr (ielem, key)
c *** primary function:
                         get information about element misc summable data
c *** Notice - This file contains ANSYS Confidential information ***
С
      input arguments:
         ielem (int,sc,in)
С
                                    - element number (or zero, see below)
                                    - key as to the information needed
С
         key
                 (int,sc,in)
C
                      = 1 - return info about misc summed data records
                                    ielem > 0 - return number of misc summed
С
                                                 data items for this element
С
                                                 (record length)
C
                                          = 0 - return maximum number of misc
С
С
                                                 summed data items on any
                                                 element (max record length)
C
                      = DB_NUMDEFINED - return total number of misc summed data
C
                                         items defined in model
C
С
      output arguments:
         emsiqr (int,func,out)
                                    - the returned value of emsigr is based on
С
                                        setting of key
C
```

7.7.6. emsget Function (Getting an Element's Miscellaneous Summable Data)

```
*deck,emsget
     function emsget (ielem, value)
c *** primary function:
                         get element misc summable data.
c *** Notice - This file contains ANSYS Confidential information ***
     input arguments:
C
        ielem
                                  - element number
                (int.sc.in)
C
С
     output arguments:
С
        emsget (int,func,out)
                                   - status of element.
С
                                     = 0 - element is undefined
                                     > 0 - number of data items returned
C
        value
                  (dp,ar(*),out)
                                   - element misc summed data.
С
                                   NOTE: the contents of this record is element
                                        dependent. See SMISC on ETABLE command
```

7.7.7. emsput Subroutine (Storing an Element's Miscellaneous Summable Data)

```
*deck,emsput
    subroutine emsput (ielem,nval,value)
c *** primary function:    store misc. summable data for an element.
c *** Notice - This file contains ANSYS Confidential information ***
c    input arguments:
c    ielem    (int,sc,in) - element number
```

7.7.8. emsdel Subroutine (Deleting an Element's Miscellaneous Summable Data)

7.7.9. enfiqr Function (Getting Information About Element Nodal Forces)

```
*deck,enfiqr
      function enfiqr (ielem, key)
c *** primary function: get information about element nodal forces
c *** Notice - This file contains ANSYS Confidential information ***
С
      input arguments:
                                    - element number (or zero, see below)
        ielem
                 (int,sc,in)
C
С
                 (int,sc,in)
                                    - key as to the information needed
С
                      = 1 - return info about element nodal forces
C
                                    ielem > 0 - return number of element nodal
                                                  forces for this element
C
                                                  (record length)
С
                                           = 0 - return maximum number of element
C
                                                  nodal forces on any element
C
                                                  (max record length)
                      = DB_NUMDEFINED - return total number of element nodal
С
С
                                         forces defined in model
С
      output arguments:
         enfiqr (int,func,out)
                                    - the returned value of enfiqr is based on
С
С
                                        setting of key
```

7.7.10. enfget Function (Getting an Element's Nodal Forces)

```
*deck,enfget
    function enfget (ielem,value)
c *** primary function:    get element nodal forces.

c *** Notice - This file contains ANSYS Confidential information ***
c    input arguments:
    ielem (int,sc,in) - element number

c    output arguments:
    enfget (int,func,out) - status of element.
```

```
c = 0 - element has no nodal forces
c > 0 - number of nodal forces returned
c value (dp,ar(*),out) - element nodal forces
```

7.7.11. enfput Subroutine (Storing an Element's Nodal Forces)

```
*deck,enfput
     subroutine enfput (ielem,nval,value)
c *** primary function:
                          store nodal force results at an element.
c *** Notice - This file contains ANSYS Confidential information ***
     input arguments:
С
С
        ielem (int,sc,in)
                                    - element number
        nval
                  (int,sc,in)
                                    - the total number of values
C
                                      NOTE: There may be a maximum of 3 sets of
C
С
                                      nodal forces in the record: static
                                     forces, inertia forces, and damping forces
C
        value
                  (dp,ar(nval),in) - nodal force results
     output arguments: none
```

7.7.12. enfdel Subroutine (Deleting an Element's Nodal Forces)

7.7.13. ensigr Function (Getting Information About an Element's Nodal Stresses)

```
*deck,ensigr
      function ensigr (ielem,key)
c *** primary function: get information about element nodal stresses
c *** Notice - This file contains ANSYS Confidential information ***
      input arguments:
C
                                    - element number (or zero, see below)
С
         ielem
                  (int,sc,in)
                                    - key as to the information needed
C
         kev
                 (int,sc,in)
                      = 1 - return info about element nodal stresses
C
C
                                    ielem > 0 - return number of element nodal
C
                                                  stresses for this element
                                                  (record length)
С
С
                                          = 0 - return maximum number of element
                                                  nodal stresses on any element
C
С
                                                  (max record length)
                      = DB_NUMDEFINED - return total number of element
С
                                        nodal stresses defined in model
C
C
      output arguments:
         ensiqr
                (int,func,out)
                                    - the returned value of ensigr is based on
C
```

С

setting of key

7.7.14. ensget Function (Getting an Element's Nodal Stresses)

```
*deck,ensget
      function ensget (ielem, value)
c *** primary function:
                          get element nodal stresses.
c *** Notice - This file contains ANSYS Confidential information ***
      input arguments:
                                    - element number
C
        ielem (int,sc,in)
С
      output arguments:
         ensget (int,func,out)
                                    - status of element.
C
С
                                      = 0 - element undefined
С
                                      > 0 - number of nodal stresses
                                             returned
C
         value
                  (dp,ar(*),out)
                                    - element nodal stresses
С
                                NOTE: Stresses at each corner node in the order
                                         X, Y, Z, XY, YZ, XZ, S1, S2, S3, SI, SE
С
                                      For solid elements, stresses at each
C
                                         corner node
C
С
                                      For shell elements, stresses at each
                                         corner node (first top durface, then
C
С
С
                                      For layered elements (w/KEYOPT(8)=0),
                                         stresses for "first" layer at each
C
С
                                         corner node (first at the bottom
                                         surface of the bottom layer, then the
С
                                          top surface of the top layer).
С
                                         Stresses for "second" layer at each
С
                                         corner node (first the bottom surface,
C
C
                                         then the top surface for the layer with
С
                                         the largest failure criteria).
                                         The second layer is not present if
C
С
                                          failure criteria were not used or are
C
                                         not appropriate
                                      For layered elements (w/KEYOPT(8)=1),
C
                                          stresses for each layer at each corner
С
                                         node (first at the bottom surface, then
                                         the top surface)
С
C
                                       For beam elements, the contents of this
                                         record is element depenent. See LS
C
                                          item of ETABLE command.
```

7.7.15. ensput Subroutine (Storing Nodal Stresses at an Element)

```
*deck,ensput
     subroutine ensput (ielem,nval,value)
c *** primary function:
                          store nodal stresses at an element.
c *** Notice - This file contains ANSYS Confidential information ***
     input arguments:
C
        ielem (int,sc,in)
                                    - element number
C
С
        nval
                  (int,sc,in)
                                    - the total number of values
                                      (6*nnod*nface)
C
        value
                 (dp,ar(nval),in)
                                   - the stress values
     output arguments: none
C
С
                                NOTE: Stresses at each corner node in the order
                                        X, Y, Z, XY, YZ, XZ, S1, S2, S3, SI, SE
С
```

```
For solid elements, stresses at each
C
С
                                          corner node
С
                                       For shell elements, stresses at each
                                          corner node (first top durface, then
С
С
С
                                       For layered elements (w/KEYOPT(8)=0),
                                          stresses for "first" layer at each
C
C
                                          corner node (first at the bottom
                                          surface of the bottom layer, then the
C
С
                                          top surface of the top layer).
                                          Stresses for "second" layer at each
C
                                          corner node (first the bottom surface,
C
С
                                          then the top surface for the layer with
С
                                          the largest failure criteria).
                                          The second layer is not present if
C
С
                                          failure criteria were not used or are
C
                                          not appropriate
                                       For layered elements (w/KEYOPT(8)=1),
C
                                          stresses for each layer at each corner
                                          node (first at the bottom surface, then
C
                                          the top surface)
С
С
                                       For beam elements, the contents of this
                                          record is element depenent. See LS
C
                                          item of ETABLE command.
```

7.7.16. ensdel Subroutine (Deleting an Element's Nodal Stresses)

7.7.17. engiqr Function (Getting Information About an Element's Energies)

```
*deck,engigr
      function engigr (ielem,key)
c *** primary function: get information about element energies
c *** Notice - This file contains ANSYS Confidential information ***
      input arguments:
C
С
         ielem
                   (int,sc,in)
                                    - element number (or zero, see below)
                                   - key as to the information needed
C
         key
                 (int,sc,in)
                  = 1 - return info about element energies
C
                              ielem > 0 - return number of element energies on
C
                                           this element (rec length)
                                    = 0 - return maximum number of element
С
С
                                            energies on any element
                                            (max rec length)
C
                  = DB_NUMDEFINED - return the number of element energies
C
                                     defined in model
C
      output arguments:
С
                                    - the returned value of engiqr is based on
C
         engiar
                (int,func,out)
                                        setting of key
```

7.7.18. engget Function (Getting an Element's Energies)

```
*deck,engget
     function engget (ielem, value)
c *** primary function:
                         get element energies.
c *** Notice - This file contains ANSYS Confidential information ***
С
      input arguments:
         ielem (int,sc,in)
                                    - element number
C
С
      output arguments:
         engget (int,func,out)
                                    - status of element.
C
                                      = 0 - element undefined
С
                                      = 11 - energies returned
С
         value
                  (dp,ar(6),out)
С
                                value(1) = volume of element
С
                                     (2) = strain energy
C
                                     (3) = dissipation energy
С
                                     (4) = kinetic energy
C
                                      (5) = plastic energy
C
С
                                      (6) = creep energy
С
                                     (7) = stabilization energy
                                     (8) = spares
C
                                     (9) = thermal energy
С
                                  (10-11) = spares
```

7.7.19. engput Subroutine (Storing an Element's Energies and Volume)

```
*deck, engput
      subroutine engput (ielem,nval,value)
c *** primary function: store volume and energies for an element.
c *** Notice - This file contains ANSYS Confidential information ***
      input arguments:
С
С
        ielem (int,sc,in)
                                    - element number
С
        nval
                  (int,sc,in)
                                     - the total number of values to be stored
                                       Must be 11!
С
         value
                  (dp,ar(6),in)
                                     - volume and energies
C
                                value(1) = volume of element
С
                                     (2) = strain energy
                                     (3) = dissipation energy
C
                                     (4) = kinetic energy
С
С
                                     (5) = plastic energy
                                     (6) = creep energy
C
С
                                     (7) = stabilization energy
C
                                     (8) = spares
С
                                     (9) = thermal energy
С
                                 (10-11) = spares
      output arguments: none
```

7.7.20. engdel Subroutine (Deleting an Element's Energies)

```
*deck,engdel
    subroutine engdel (ielem)
c *** primary function:    delete element energies

c *** Notice - This file contains ANSYS Confidential information ***

c    input arguments:
    ielem (int,sc,in) - element number
```

```
c = 0 - delete for all defined elements c output arguments: none.
```

7.7.21. egriqr Function (Getting Information About an Element's Nodal Gradients)

```
*deck,egriqr
      function egriqr (ielem, key)
c *** primary function: get information about element nodal gradients
c *** Notice - This file contains ANSYS Confidential information ***
      input arguments:
С
                   (int,sc,in) - element number (or 2010, not so in) - key as to the information needed
C
         ielem
                                     - element number (or zero, see below)
C
         kev
                  (int,sc,in)
                   = 1 - return info about nodal gradients
С
                           for ielem > 0 - return number of nodal gradients on
C
                                              this element (record length)
С
С
                                      = 0 - return maximum number of nodal
                                              gradients on any element
C
                                              (maximum record length)
                   = DB_NUMDEFINED - return the number of nodal gradients defined
С
C
                                      in model
С
      output arguments:
        egriqr (int,func,out)
                                     - the returned value of egriqr is based on
С
                                          setting of key
```

7.7.22. egrget Function (Getting an Element's Nodal Gradients)

```
*deck,egrget
      function egrget (ielem, value)
c *** primary function:
                         get element nodal gradients.
c *** Notice - This file contains ANSYS Confidential information ***
      input arguments:
C
                 (int,sc,in)
         ielem
                                    - element number
С
      output arguments:
С
         egrget (int,func,out)
                                    - status of element.
                                      = 0 - element undefined
C
С
                                      > 0 - number of nodal gradients
С
                                              returned
С
         value
                  (dp,ar(*),out)
                                    - element nodal gradients
                                     Note: If a coupled field, a set of
C
                                       gradients are stored in the following
С
                                       order (as available): fluid, thermal,
                                       electric, magnetic
c *** mpg egrget < pagend, magget < hsnget2: get elem gradient, H,
```

7.7.23. egrput Subroutine (Storing an Element's Nodal Gradients)

```
c *** Notice - This file contains ANSYS Confidential information ***
С
     input arguments:
        ielem (int,sc,in)
                                    - element number
C
С
                  (int,sc,in)
                                    - the total number of values
C
                                      (ndir*nnod*nscalr)
        value
               (dp,ar(nval),in)
                                   - the gradient values
C
                                     Note: If a coupled field, a set of
C
С
                                      gradients are stored in the following
                                       order (as appropriate): fluid, thermal,
С
                                      electric, magnetic
C
     output arguments: none
```

7.7.24. egrdel Subroutine (Deleting an Element's Nodal Gradients)

7.7.25. eeliqr Function (Getting Information About an Element's Nodal Elastic Strains)

```
*deck,eeligr
     function eeligr (ielem,key)
c *** primary function: get information about element nodal elastic strains
c *** Notice - This file contains ANSYS Confidential information ***
      input arguments:
C
        ielem
                   (int,sc,in)
                                    - element number (or zero, see below)
C
         key
                 (int,sc,in)
                                    - key as to the information needed
                  = 1 - return info about elastic strains
С
С
                              ielem > 0 - return number of nodal elasic strains
                                           on this element (rec length)
C
С
                                    = 0 - return maximum number of nodal elastic
С
                                           strains on any element
С
                                           (max rec length)
С
                  = DB_NUMDEFINED - return the number of nodal elastic strains
                            defined in model
С
      output arguments:
С
                                   - the returned value of eeligr is based on
C
         eeligr (int,func,out)
                                        setting of key
```

7.7.26. eelget Function (Getting an Element's Nodal Elastic Strains)

```
input arguments:
C
С
         ielem
                  (int,sc,in)
                                     - element number
С
      output arguments:
С
         eelget
                  (int,func,out)
                                     - status of element.
С
                                       = 0 - element undefined
                                       > 0 - number of nodal elastic strains
С
                                              returned
         value
                  (dp,ar(*),out)
                                     - element nodal elastic strains
C
                                 NOTE: Strains at each corner node in the order
C
С
                                          X, Y, Z, XY, YZ, XZ
С
                                       For solid elements, strains at each
C
                                          corner node
С
                                       For shell elements, strains at each
C
                                          corner node (first top durface, then
                                          bottom)
C
                                       For layered elements (w/KEYOPT(8)=0),
                                          strains for "first" layer at each
C
                                          corner node (first at the bottom
С
С
                                          surface of the bottom layer, then the
                                          top surface of the top layer).
C
                                          Strains for "second" layer at each
С
С
                                          corner node (first the bottom surface,
C
                                          then the top surface for the layer with
С
                                          the largest failure criteria).
                                          The second layer is not present if
C
                                          failure criteria were not used or are
C
                                          not appropriate
                                       For layered elements (w/KEYOPT(8)=1),
C
                                          strains for each layer at each corner
C
C
                                          node (first at the bottom surface, then
                                          the top surface)
C
С
                                       For beam elements, the contents of this
                                          record is element depenent. See LEPEL
C
                                          item of ETABLE command.
C
```

7.7.27. eelput Subroutine (Storing an Element's Nodal Elastic Strains)

```
*deck,eelput
      subroutine eelput (ielem,nval,value)
c *** primary function:
                           store nodal elastic strains at an element.
c *** Notice - This file contains ANSYS Confidential information ***
      input arguments:
C
         ielem
                   (int,sc,in)
                                      - element number
C
C
         nval
                   (int,sc,in)
                                      - the total number of values
                                        (6*nnod*nface)
C
                  (dp,ar(nval),in)
C
         value
                                      - nval strain values
С
      output arguments: none
                                 NOTE: Strains at each corner node in the order
C
                                          X, Y, Z, XY, YZ, XZ
С
                                       For solid elements, strains at each
C
                                          corner node
С
                                       For shell elements, strains at each
С
                                          corner node (first top durface, then
С
                                          bottom)
                                       For layered elements (w/KEYOPT(8)=0),
С
                                          strains for "first" layer at each
C
                                          corner node (first at the bottom
C
                                          surface of the bottom layer, then the
С
                                          top surface of the top layer).
C
                                          Strains for "second" layer at each
C
```

```
corner node (first the bottom surface,
С
С
                                          then the top surface for the layer with
С
                                          the largest failure criteria).
                                          The second layer is not present if
С
С
                                          failure criteria were not used or are
C
                                          not appropriate
                                       For layered elements (w/KEYOPT(8)=1),
C
                                          strains for each layer at each corner
                                          node (first at the bottom surface, then
C
С
                                          the top surface)
                                       For beam elements, the contents of this
С
                                          record is element depenent. See LEPEL
C
                                          item of ETABLE command.
```

7.7.28. eeldel Subroutine (Deleting an Element's Nodal Elastic Strains)

7.7.29. epliqr Function (Getting Information About an Element's Nodal Plastic Strains)

```
*deck,epliqr
     function epligr (ielem, key)
c *** primary function: get information about element nodal plastic strains
c *** Notice - This file contains ANSYS Confidential information ***
С
     input arguments:
        ielem (int,sc,in)
                                   - element number (or zero, see below)
C
                 (int,sc,in)
                                   - key as to the information needed
                 = 1 - return info about plastic strains
C
С
                              ielem > 0 - return number of nodal plastic strains
С
                                            on this element
                                            (record length)
C
С
                                    = 0 - return maximum number of nodal plastic
С
                                            strains on any element
С
                                            (max record length)
С
                  = DB_NUMDEFINED - return the number of nodal plastic strains
С
                                    defined in model
     output arguments:
С
        epliqr (int,func,out)
                                   - the returned value of epligr is based on
C
С
                                        setting of key
```

7.7.30. eplget Function (Getting an Element's Nodal Plastic Strains)

```
input arguments:
С
С
         ielem
                  (int,sc,in)
                                     - element number
С
      output arguments:
C
         eplget
                  (int,func,out)
                                     - status of element.
                                       = 0 - element undefined
С
C
                                       > 0 - number of nodal plastic strains
С
                                              returned
С
         value
                  (dp,ar(*),out)
                                     - element nodal plastic strains
                                 NOTE: Strains at each corner node in the order
C
С
                                          X, Y, Z, XY, YZ, XZ
С
                                       For solid elements, strains at each
C
                                          corner node
С
                                       For shell elements, strains at each
C
                                          corner node (first top durface, then
                                          bottom)
C
                                       For layered elements (w/KEYOPT(8)=0),
                                          strains for "first" layer at each
С
                                          corner node (first at the bottom
С
С
                                          surface of the bottom layer, then the
                                          top surface of the top layer).
C
                                          Strains for "second" layer at each
С
С
                                          corner node (first the bottom surface,
                                          then the top surface for the layer with
C
C
                                          the largest failure criteria).
                                          The second layer is not present if
C
                                          failure criteria were not used or are
C
                                          not appropriate
                                       For layered elements (w/KEYOPT(8)=1),
C
                                          strains for each layer at each corner
C
C
                                          node (first at the bottom surface, then
                                          the top surface)
C
С
                                       For beam elements, the contents of this
                                          record is element depenent. See LEPPL
С
                                          item of ETABLE command.
C
```

7.7.31. eplput Subroutine (Storing an Element's Nodal Plastic Strains)

```
*deck,eplput
      subroutine eplput (ielem, nval, value)
c *** primary function:
                            store nodal plastic strains at a element.
c *** Notice - This file contains ANSYS Confidential information ***
      input arguments:
C
                                      - element number
         ielem
                   (int,sc,in)
C
                                      - the total number of values
С
         nval
                    (int.sc.in)
С
                                        (6*nnod*nface)
                  (dp,ar(nval),in)
                                     - the strain values
C
         value
С
      output arguments: none
                                 NOTE: Strains at each corner node in the order
C
                                          X, Y, Z, XY, YZ, XZ
C
                                       For solid elements, strains at each
C
                                          corner node
С
С
                                       For shell elements, strains at each
                                          corner node (first top durface, then
C
С
                                          bottom)
                                       For layered elements (w/KEYOPT(8)=0),
С
С
                                          strains for "first" layer at each
                                          corner node (first at the bottom
C
                                          surface of the bottom layer, then the
C
                                          top surface of the top layer).
С
                                          Strains for "second" layer at each
С
                                          corner node (first the bottom surface,
C
                                          then the top surface for the layer with
C
```

```
the largest failure criteria).
С
                                          The second layer is not present if
С
С
                                          failure criteria were not used or are
С
                                          not appropriate
С
                                       For layered elements (w/KEYOPT(8)=1),
C
                                          strains for each layer at each corner
                                          node (first at the bottom surface, then
C
                                          the top surface)
                                       For beam elements, the contents of this
C
С
                                          record is element depenent. See LEPPL
                                          item of ETABLE command.
```

7.7.32. epidel Subroutine (Deleting an Element's Nodal Plastic Strains)

7.7.33. ecriqr Function (Getting Information About an Element's Nodal Creep Strains)

```
*deck,ecriqr
      function ecriqr (ielem, key)
c *** primary function: get information about element nodal creep strains
c *** Notice - This file contains ANSYS Confidential information ***
      input arguments:
С
                                    - element number (or zero, see below)
С
        ielem
                 (int,sc,in)
                                   - key as to the information needed
         kev
                 (int,sc,in)
C
                  = 1 - return info about creep strains
C
                              ielem > 0 - return number of nodal creep strains
С
                                            on this element
С
                                            (record length)
                                    = 0 - return maximum number of nodal creep
C
С
                                            strains on any element
С
                                            (max record length)
С
                  = DB_NUMDEFINED - return the number of nodal creep strains
С
                                     defined in model
     output arguments:
С
                                    - the returned value of ecriqr is based on
С
C
                                        setting of kev
```

7.7.34. ecrget Function (Getting an Element's Nodal Creep Strains)

```
*deck,ecrget
    function ecrget (ielem,value)
c *** primary function: get element nodal creep strains.

c *** Notice - This file contains ANSYS Confidential information ***
c input arguments:
```

```
ielem
                  (int,sc,in)
                                     - element number
C
С
      output arguments:
С
         ecrqet
                  (int, func, out)
                                     - status of element.
                                       = 0 - element undefined
C
                                       > 0 - number of nodal creep strains
С
                                              returned
C
         value
                  (dp,ar(*),out)
                                     - element nodal creep strains
С
                                 NOTE: Strains at each corner node in the order
                                          X, Y, Z, XY, YZ, XZ
С
                                       For solid elements, strains at each
C
С
                                          corner node
С
                                       For shell elements, strains at each
                                          corner node (first top durface, then
C
С
                                          bottom)
C
                                       For layered elements (w/KEYOPT(8)=0),
                                          strains for "first" layer at each
C
                                          corner node (first at the bottom
                                          surface of the bottom layer, then the
C
                                          top surface of the top layer).
С
                                          Strains for "second" layer at each
С
                                          corner node (first the bottom surface,
C
С
                                          then the top surface for the layer with
С
                                          the largest failure criteria).
                                          The second layer is not present if
C
C
                                          failure criteria were not used or are
C
                                          not appropriate
                                       For layered elements (w/KEYOPT(8)=1),
C
                                          strains for each layer at each corner
C
                                          node (first at the bottom surface, then
С
                                          the top surface)
C
                                       For beam elements, the contents of this
                                          record is element depenent. See LEPCR
C
C
                                          item of ETABLE command.
```

7.7.35. ecrput Subroutine (Storing an Element's Nodal Creep Strains)

```
*deck,ecrput
      subroutine ecrput (ielem,nval,value)
c *** primary function:
                            store nodal creep strains at an element.
c *** Notice - This file contains ANSYS Confidential information ***
      input arguments:
C
С
         ielem
                   (int,sc,in)
                                      - element number
         nval
                    (int,sc,in)
                                      - the total number of values
C
                                        (6*nnod*nface)
C
                  (dp,ar(nval),in)
                                      - the strain values
C
         value
C
      output arguments: none
                                 NOTE: Strains at each corner node in the order
C
С
                                          X, Y, Z, XY, YZ, XZ
                                       For solid elements, strains at each
C
                                          corner node
C
                                       For shell elements, strains at each
C
                                          corner node (first top durface, then
С
С
                                          bottom)
                                       For layered elements (w/KEYOPT(8)=0),
C
                                          strains for "first" layer at each
С
                                          corner node (first at the bottom
С
С
                                          surface of the bottom layer, then the
                                          top surface of the top layer).
С
                                          Strains for "second" layer at each
C
                                          corner node (first the bottom surface,
C
                                          then the top surface for the layer with
С
                                          the largest failure criteria).
C
                                          The second layer is not present if
C
```

```
failure criteria were not used or are
С
С
                                          not appropriate
С
                                       For layered elements (w/KEYOPT(8)=1),
                                          strains for each layer at each corner
С
                                          node (first at the bottom surface, then
С
C
                                          the top surface)
                                       For beam elements, the contents of this
C
                                          record is element depenent. See LEPCR
                                          item of ETABLE command.
C
```

7.7.36. ecrdel Subroutine (Deleting an Element's Nodal Creep Strains)

7.7.37. ethiqr Function (Getting Information About an Element's Nodal Thermal Strains)

```
*deck,ethigr
     function ethiqr (ielem, key)
c *** primary function: get information about element nodal thermal strains
c *** Notice - This file contains ANSYS Confidential information ***
С
      input arguments:
        ielem
                  (int.sc.in)
                                    - element number (or zero, see below)
C
                                 - key as to the information needed
         key
С
                  = 1 - return info about thermal strains
C
                              ielem > 0 - return number of nodal thermal strains
C
                                            on this element
C
                                            (record length)
С
                                    = 0 - return maximum number of nodal thermal
С
                                            strains on any element
                                            (max record length)
C
С
                  = DB_NUMDEFINED - return the number of nodal thermal strains
С
                                     defined in model
С
      output arguments:
                                    - the returned value of ethigr is based on
С
         ethiqr (int,sc,out)
                                        setting of key
С
```

7.7.38. ethget Function (Getting an Element's Nodal Thermal Stresses)

```
C
      output arguments:
                                     - status of element.
С
         ethget
                  (int,func,out)
С
                                       = 0 - element undefined
                                       > 0 - number of nodal thermal strains
С
C
                                              returned
С
         value
                  (dp,ar(*),out)
                                     - element nodal thermal strains
C
                                 NOTE: Strains at each corner node in the order
                                          X, Y, Z, XY, YZ, XZ, epswel
C
С
                                       For solid elements, strains at each
С
                                          corner node
                                       For shell elements, strains at each
C
                                          corner node (first top durface, then
С
С
                                          bottom)
                                       For layered elements (w/KEYOPT(8)=0),
C
                                          strains for "first" layer at each
С
                                          corner node (first at the bottom
C
                                          surface of the bottom layer, then the
C
                                          top surface of the top layer).
                                          Strains for "second" layer at each
C
                                          corner node (first the bottom surface,
С
С
                                          then the top surface for the layer with
                                          the largest failure criteria).
C
С
                                          The second layer is not present if
С
                                          failure criteria were not used or are
C
                                          not appropriate
C
                                       For layered elements (w/KEYOPT(8)=1),
C
                                          strains for each layer at each corner
                                          node (first at the bottom surface, then
C
                                          the top surface)
C
                                       For beam elements, the contents of this
С
                                          record is element depenent. See LEPTH
C
                                          item of ETABLE command.
```

7.7.39. ethput Subroutine (Storing an Element's Nodal Thermal Stresses)

```
*deck,ethput
      subroutine ethput (ielem,nval,value)
c *** primary function:
                           store nodal thermal strains at an element.
                            also the volumetric swelling strain
c *** Notice - This file contains ANSYS Confidential information ***
C
      input arguments:
                                      - element number
         ielem
                   (int.sc.in)
C
С
         nval
                   (int,sc,in)
                                      - the total number of values
                                        (6*nnod*nface)
C
                  (dp,ar(nval),in)
         value
                                     - the strain values
C
С
      output arguments: none
                                NOTE: Strains at each corner node in the order
C
                                          X, Y, Z, XY, YZ, XZ, epswel
C
С
                                       For solid elements, strains at each
С
                                          corner node
                                       For shell elements, strains at each
C
                                          corner node (first top durface, then
C
С
С
                                       For layered elements (w/KEYOPT(8)=0),
                                          strains for "first" layer at each
C
                                          corner node (first at the bottom
С
С
                                          surface of the bottom layer, then the
С
                                          top surface of the top layer).
                                          Strains for "second" layer at each
С
                                          corner node (first the bottom surface,
C
С
                                          then the top surface for the layer with
                                          the largest failure criteria).
С
C
                                          The second layer is not present if
                                          failure criteria were not used or are
C
```

```
c not appropriate
c For layered elements (w/KEYOPT(8)=1),
c strains for each layer at each corner
c node (first at the bottom surface, then
c the top surface)
c For beam elements, the contents of this
c record is element depenent. See LEPTH
c item of ETABLE command.
```

7.7.40. ethdel Subroutine (Deleting an Element's Thermal, Initial, and Swelling Strains)

7.7.41.euliqr Function (Getting Information About an Element's Euler Angles)

```
*deck,euligr
     function euligr (ielem, key)
c *** primary function: get information about element euler angles
c *** Notice - This file contains ANSYS Confidential information ***
      input arguments:
C
         ielem
                (int,sc,in)
                                    - element number (or zero, see below)
С
                                   - key as to the information needed
С
         kev
                  (int,sc,in)
                  = 1 - return info about element euler angles
C
                              ielem > 0 - return number of euler angles on this
C
                                            element
С
                                            (record length)
С
                                    = 0 - return maximum number of euler angles
                                            on any element
C
С
                                            (max record length)
                  = DB_NUMDEFINED - return the number of element euler angles
С
С
                                    defined in model
С
      output arguments:
                                    - the returned value of euligr is based on
С
        euliqr (int,func,out)
                                        setting of key
```

7.7.42. eulget Function (Getting an Element's Nodal Euler Angles)

```
*deck,eulget
    function eulget (ielem,value)
c *** primary function: get element nodal euler angles.

c *** Notice - This file contains ANSYS Confidential information ***
c input arguments:
c ielem (int,sc,in) - element number
```

```
С
      output arguments:
С
         eulget
                  (int,func,out)
                                     - status of element.
                                       = 0 - element undefined
С
                                       > 0 - number of euler angle values
С
                                              returned
С
         value
                  (dp,ar(*),out)
                                     - element euler angles
С
                                   NOTE: For lower-ordered elements, rotations
С
С
                                           at centroid
                                         For higher-ordered elements, rotations
C
                                           at each corner node
C
С
                                         For layered shells, rotations at each
С
                                           corner node, plus layer rotation angle
                                           for each layer (real constant THETA)
C
С
                                         For layered solids, rotation angles at
C
                                           centroid, plus layer rotation angle
                                           for each layer (real constant THETA)
C
```

7.7.43. eulput Subroutine (Storing an Element's Euler Angles)

```
*deck,eulput
      subroutine eulput (ielem,nval,value)
c *** primary function:
                           store nodal euler angles for an element.
c *** Notice - This file contains ANSYS Confidential information ***
С
      input arguments:
         ielem
                   (int,sc,in)
                                     - element number
C
C
         nval
                   (int,sc,in)
                                     - the total number of values
                                       (3 * number of display nodes)
C
                  (dp,ar(nval),in)
                                     - the euler angle values
С
         value
C
      output arguments: none
                                  NOTE: For lower-ordered elements, rotations
С
С
                                          at centroid
C
                                        For higher-ordered elements, rotations
С
                                          at each corner node
C
                                         For layered shells, rotations at each
                                          corner node, plus layer rotation angle
C
                                           for each layer (real constant THETA)
С
                                         For layered solids, rotation angles at
                                           centroid, plus layer rotation angle
С
                                           for each layer (real constant THETA)
```

7.7.44. euldel Subroutine (Deleting an Element's Euler Angles)

7.7.45. efxigr Function (Getting Information About Element Fluxes)

```
*deck,efxiqr
     function efxiqr (ielem, key)
                         get information about element fluxes
c *** primary function:
c *** Notice - This file contains ANSYS Confidential information ***
С
      input arguments:
        ielem
                (int,sc,in)
                                    - element number (or zero, see below)
C
                                   - key as to the information needed
С
         kev
                  (int,sc,in)
                  = 1 - return info about element fluxes
C
                              ielem > 0 - return number of fluxes on this
C
С
                                            element
С
                                            (record length)
С
                                    = 0 - return maximum number of fluxes
                                            on any element
C
C
                                            (max record length)
                  = DB_NUMDEFINED - return the number of element fluxes defined
С
                                     in model
C
С
      output arguments:
                                    - the returned value of efxiqr is based on
С
         efxiqr (int,func,out)
                                       setting of key
```

7.7.46. efxget Function (Getting an Element Flux)

```
*deck,efxget
      function efxget (ielem, value)
c *** primary function:
                         get element nodal fluxes.
c *** Notice - This file contains ANSYS Confidential information ***
С
      input arguments:
С
         ielem
               (int,sc,in)
                                    - element number
С
      output arguments:
С
         efxget (int,func,out)
                                    - status of element.
С
                                      = 0 - element undefined
                                      > 0 - number of nodal fluxes returned
         value
                  (dp,ar(*),out)
                                    - element nodal fluxes
C
                                    Note: If a coupled field, a set of fluxes is
С
                                           stored in the following order (as
C
                                           available): fluid, thermal,
С
                                           electric, magnetic
c *** mpg efxget<pagend<paberrwb,edgzzx,panavg,papres,paterr: get ele nd flx, B
```

7.7.47. efxput Subroutine (Storing an Element's Fluxes)

```
*deck,efxput
     subroutine efxput (ielem,nval,value)
c *** primary function:
                          store nodal fluxes at an element.
c *** Notice - This file contains ANSYS Confidential information ***
      input arguments:
C
С
        ielem
                                     - element number
        nval
                                     - the total number of values
C
С
                                       (ndir*nnod*nscalr)
        value
                  (dp,ar(nval),in)
                                    - the flux values
C
      output arguments: none
С
```

7.7.48. efxdel Subroutine (Deleting Element Fluxes)

7.7.49. elfiqr Function (Getting Information About Element Local Forces)

```
*deck,elfiqr
      function elfiqr (ielem,key)
c *** primary function: get information about elem local forces
c *** Notice - This file contains ANSYS Confidential information ***
С
      input arguments:
                                    - element number (or zero, see below)
         ielem
                   (int.sc.in)
C
                   (int,sc,in)
                                    - key as to the information needed
                  = 1 - return info about element local forces
C
С
                              ielem > 0 - return number of local forces on this
C
                                             element.
                                             (record length)
C
                                    = 0 - return maximum number of local forces
С
С
                                             on any element
C
                                             (max record length)
                  = DB_NUMDEFINED - return the number of element local forces
C
С
                                     defined in model
С
      output arguments:
         elfigr (int,func,out)
                                    - the returned value of elfiqr is based on
С
                                        setting of key
С
```

7.7.50. elfget Function (Getting an Element Local Force)

```
*deck,elfget
     function elfget (ielem, value)
c *** primary function:
                          get element local nodal forces.
c *** Notice - This file contains ANSYS Confidential information ***
С
      input arguments:
         ielem
                (int,sc,in)
                                    - element number
С
С
      output arguments:
                                    - status of element.
С
         elfget
                 (int,func,out)
С
                                      = 0 - element has no local nodal forces
C
                                      > 0 - number of nodal forces returned
                                    - element local nodal forces.
         value
                  (dp,ar(*),out)
```

```
c *** mpg elfget<pagend<paberrwb,edgzzx,panavg,papres,paterr: get ele nd frc, F
```

7.7.51. elfput Subroutine (Storing an Element's Local Forces)

```
*deck,elfput
     subroutine elfput (ielem,nval,value)
c *** primary function:
                         store element local nodal forces.
c *** Notice - This file contains ANSYS Confidential information ***
     input arguments:
     variable (typ,siz,intent)
                                 description
C
        ielem (int,sc,in)
                                    - element number
C
С
        nval
                  (int,sc,in)
                                    - the total number of values
                                      NOTE: There may be a maximum of 3 sets of
C
                                     nodal forces in the record: static
С
С
                                     forces, inertia forces, and damping forces
                 (dp,ar(nval),in)
                                   - element local nodal forces
        value
C
C
     output arguments: none
```

7.7.52. elfdel Subroutine (Deleting Element Local Forces)

7.7.53. emniqr Function (Getting Information About Element Miscellaneous Non-summable Data)

```
*deck,emnigr
      function emniqr (ielem,key)
c *** primary function:
                           get information about element misc non-summable
                           data
c *** Notice - This file contains ANSYS Confidential information ***
      input arguments:
C
С
         ielem
                  (int,sc,in)
                                    - element number (or zero, see below)
                                    - key as to the information needed
С
         key
                   (int,sc,in)
                  = 1 - return info about element misc non-summed data
C
                              ielem > 0 - return number of data items on this
C
                                            element
С
                                            (record length)
С
                                    = 0 - return maximum number of data items
                                            on any element
С
С
                                            (max record length)
С
                  = DB_NUMDEFINED - return the number of element misc non-summed
                                     data items defined in model
C
С
      output arguments:
         emnigr (int, func, out)
                                    - the returned value of emnigr is based on
C
```

setting of key

C

7.7.54. emnget Function (Getting an Element's Miscellaneous Non-summable Data)

```
*deck,emnget
     function emnget (ielem, value)
c *** primary function:
                         get misc non-summable data.
c *** Notice - This file contains ANSYS Confidential information ***
С
     input arguments:
                (int,sc,in)
                                   - element number
С
        ielem
C
     output arguments:
        emnget (int,func,out)
                                   - status of element.
C
С
                                      = 0 - no non-summed misc data at this
C
                                             element
                                      > 0 - number of data items returned
С
                  (dp,ar(*),out)
С
        value
                                    - element misc non-summed data.
                                    NOTE: the contents of this record is element
C
C
                                        dependent. See NMISC on ETABLE command
```

7.7.55. emnput Subroutine (Storing an Element's Miscellaneous Non-summable Data)

```
*deck,emnput
     subroutine emnput (ielem,nval,value)
c *** primary function:
                        store misc. non-summable data for an element.
c *** Notice - This file contains ANSYS Confidential information ***
      input arguments:
        ielem (int,sc,in) - element number
C
        nval
                  (int,sc,in)
                                   - the total number of values
С
С
        value (dp,ar(nval),in) - the misc. non-summed data items
С
     output arguments: none
                                   NOTE: the contents of this record is element
С
                                      dependent. See NMISC on ETABLE command
C
```

7.7.56. emndel Subroutine (Deleting an Element's Miscellaneous Non-summable Data)

7.7.57. ecdigr Function (Getting Information About Element Current Densities)

```
*deck,ecdiqr
     function ecdiqr (ielem, key)
c *** primary function:
                          get information about element current densities
c *** Notice - This file contains ANSYS Confidential information ***
С
      input arguments:
        ielem
                (int,sc,in)
                                    - element number (or zero, see below)
C
                                   - key as to the information needed
С
         kev
                  (int,sc,in)
                  = 1 - return info about element current densities
C
                              ielem > 0 - return number of current densities on
C
                                            this element
С
С
                                            (record length)
С
                                    = 0 - return maximum number of current
                                            densities on any element
C
C
                                            (max record length)
                  = DB_NUMDEFINED - return the number of element current
С
                                     densities defined in model
C
С
      output arguments:
                                    - the returned value of ecdiqr is based on
С
         ecdiqr (int,func,out)
                                        setting of key
C
```

7.7.58. ecdget Function (Getting an Element Current Density)

```
*deck,ecdget
      function ecdget (ielem, value)
c *** primary function:
                          get calculated element current densities.
c *** Notice - This file contains ANSYS Confidential information ***
С
      input arguments:
        ielem
                (int,sc,in)
                                    - element number
C
С
      output arguments:
С
         ecdget (int,func,out)
                                    - status of element.
                                      = 0 - element has no current densities
C
                                      > 0 - number of calculated element
C
                                             current densities
C
         value
                  (dp,ar(*),out)
                                    - calculated element current densities.
                                    NOTE: current densities are in the order
С
С
                                           X, Y, Z
```

7.7.59. ecdput Subroutine (Storing an Element's Current Densities)

```
*deck.ecdput
     subroutine ecdput (ielem,nval,value)
c *** primary function:
                         store calculated element current densities
c *** Notice - This file contains ANSYS Confidential information ***
С
     input arguments:
С
        ielem
                (int,sc,in)
                                    - element number
                                    - the total number of values
        nval
                 (int,sc,in)
C
                 (dp,ar(nval),in)

    calculated element current densities.

C
С
     output arguments:
                         none
                                   NOTE: current densities are in the order
C
                                          X, Y, Z
С
```

7.7.60. ecddel Subroutine (Deleting Element Current Densities)

7.7.61.enliqr Function (Getting Information About Element Nonlinear Tables)

```
*deck,enligr
      function enliqr (ielem, key)
c *** primary function: get information about element nonlinear tables
c *** Notice - This file contains ANSYS Confidential information ***
      input arguments:
С
С
         ielem
                   (int,sc,in)
                                    - element number (or zero, see below)
                   (int,sc,in)
                                    - key as to the information needed
C
                  = 1 - return info about element nonlinear tables
С
С
                              ielem > 0 - return number of nonlinear tables for
                                             this element
C
                                             (record length)
                                    = 0 - return maximum number of nonlinear
C
С
                                             tables for any element
C
                                             (max record length)
                  = DB_NUMDEFINED - return the number of element nonlinear
C
С
                                     tables defined in model
C
      output arguments:
                                    - the returned value of enligr is based on
С
         enligr
                (int,func,out)
                                        setting of key
С
```

7.7.62. enlget Function (Getting Element Nonlinear Tables)

```
*deck,enlget
      function enlget (ielem, value)
c *** primary function:
                          get element nonlinear tables.
c *** Notice - This file contains ANSYS Confidential information ***
C
      input arguments:
С
         ielem
                  (int,sc,in)
                                    - element number
      output arguments:
C
                 (int,func,out)
                                    - status of element.
         enlaet
                                      = 0 - nonlinear tables undefined
С
                                       > 0 - number of nonlinear tables defined
С
         value
                  (dp ,ar(n),out)
                                    - the element nonlinear tables.
                                    NOTE: Nonlinear data at each node are in the
С
С
                                           order SEPL, SRAT, HPRES, EPEQ, PSV,
C
                                           PLWK, and 4 spares
С
                                          For beam elements, the contents and
С
                                           number of information is element
```

```
c dependent. See NLIN on ETABLE c command
```

7.7.63. enlput Subroutine (Storing an Element's Nonlinear Tables)

```
*deck,enlput
      subroutine enlput (ielem,n,temp)
                          store element nonlinear tables
c *** primary function:
c *** Notice - This file contains ANSYS Confidential information ***
      input arguments:
C
        ielem
                 (int,sc,in)
                                    - element number
C
С
        n
                  (int,sc,in)
                                    - number of element nonlinear table values
                  (dp ,ar(6),in)
                                    - element nonlinear table, etc.
C
        temp
С
      output arguments: none.
                                    NOTE: Nonlinear data at each node are in the
C
                                           order SEPL, SRAT, HPRES, EPEQ, PSV,
С
                                           PLWK, and 4 spares
С
                                          For beam elements, the contents and
                                           number of information is element
С
                                           dependent. See NLIN on ETABLE
C
```

7.7.64. enIdel Subroutine (Deleting Element Nonlinear Tables)

7.7.65. ehciqr Function (Getting Information About Calculated Element Heat Generations)

```
*deck,ehciqr
     function ehciqr (ielem,key)
c *** primary function: get information about calculated elem heat generations
c *** Notice - This file contains ANSYS Confidential information ***
      input arguments:
С
        ielem
                  (int,sc,in)
                                    - element number (or zero, see below)
C
                   (int,sc,in)

    key as to the information needed

                  = 1 - return info about calculated element heat gens
С
С
                          for ielem > 0 - return number of heat gens for
                                             this element
C
С
                                             (record length)
С
                                     = 0 - return maximum number of heat gens
C
                                             for any element
                                             (max record length)
C
                  = DB_NUMDEFINED - return the number of calculated element heat
С
                                     generations defined in model
С
```

```
c output arguments:
c ehciqr (int,func,out) - the returned value of ehciqr is based on
c setting of key
```

7.7.66. ehcget Function (Getting a Calculated Element Heat Generation)

```
*deck,ehcget
      function ehcget (ielem, value)
c *** primary function:
                         get calculated element heat generations.
c *** Notice - This file contains ANSYS Confidential information ***
С
      input arguments:
        ielem (int,sc,in)
С

    element number

С
      output arguments:
         ehcget (int,func,out)
                                    - status of element.
C
С
                                      = 0 - element undefined
                                      > 0 - number of calculated element
C
С
                                             heat generations
         value
                  (dp,ar(*),out)
                                    - calculated element heat generations.
```

7.7.67. ehcput Subroutine (Storing an Element's Calculated Heat Generations)

```
*deck,ehcput
     subroutine ehcput (ielem,nval,value)
c *** primary function:
                         store calculated element heat generations
c *** Notice - This file contains ANSYS Confidential information ***
     input arguments:
С
C
        ielem
                 (int,sc,in)
                                    - element number
                                    - the total number of values
C
        nval
                  (int,sc,in)
                 (dp,ar(nval),in) - calculated element heat generations.
C
     output arguments: none
```

7.7.68. ehcdel Subroutine (Deleting Element Calculated Heat Generations)

Release 12.0 - © 2009 SAS IP.Inc. All rights reserved	 Contains proprietary and confidential information
	ubsidiaries and affiliates

Chapter 8: Subroutines for Users' Convenience

This chapter describes ANSYS routines available to you for use in programming. Using these routines isn't required, but may make your life easier. These routines include a set of general routines that perform utility-type functions, a set of routines supporting vector functions, a set of routines supporting matrix functions, and routines supporting message processing options.

The following topics are discussed in this chapter:

8.1. Input and Output Abbreviations

8.2. General Subroutines

8.3. Vector Functions

8.4. Matrix Subroutines

8.1. Input and Output Abbreviations

The descriptions of inputs and outputs for the routines discussed in this chapter use the following abbreviations:

Argument type is one of the following:

int - integer

dp - double precision

log - logical

chr - character

dcp - double precision complex

Argument size is one of the following:

sc - scalar variable

ar(n) - array variable of length n

func - functional return value

Argument intent is one of the following:

in - input argumentout - output argumentinout - both an input and an output argument

8.2. General Subroutines

8.2.1. dptoch Subroutine (Retrieve Eight Characters From a Double Precision Variable)

```
c !!! NOTICE to progammers: this routine does not convert from a !!!
c !!! machine-independent format! Use dpexttoch if this dp word !!!
c !!! came from a common or non-char database record !!!
c input arguments:
c dp8 (dp,sc,in) - dp variable containing characters
c output arguments:
c ch8 (ch*8,sc,out) - characters retreived from the dp word
```

8.2.2. wringr Function (Obtain Information About Output)

```
*deck,wringr
      function wringr (key)
c *** primary function: obtain information about output
c *** Notice - This file contains ANSYS Confidential information ***
c --- caution: the following variables are "saved/resumed".
C ---
              key=WR_COLINTER thru WR_SUPCOLMAX in "wrinqr/wrinfo"
C ---
               (data for "/fmt,/page,/header" commands).
               note that the whole common cannot be "saved/resumed". cwa
     typ=int,dp,log,chr,dcp siz=sc,ar(n),func intent=in,out,inout
C
C
  input arguments:
С
      variable (typ,siz,intent)
                                  description
                                                                     wrcom name
              (int,sc,in)
C
                  = WR_PRINT
                                   - print flag (kprint)
                                                                         prtkev
C
С
                       wringr = 0 - no output
C
                              = 1 - print
                  = WR_OUTPUT
                                   - current output unit number (iott) outfil
С
                                  - master output file
                  = WR_MASTEROUT
С
                  = WR_COLINTER
                                   - interactive columns per page
С
                                                                        intcol
С
                  = WR_COLBATCH
                                   - batch columns per page
                                                                        batcol
С
                  = WR_LINEINTER - interactive lines per page
                                                                        intlin
C
                  = WR_LINEBATCH - batch lines per page
                                                                        batlin
С
                  = WR_CHARITEM
                                   - characters per output item
                                                                        chrper
                                                                        chrdec
С
                  = WR_CHARDECIMAL - characters past decimal
                  = WR_CHARINTEGER - characters in leading integer
                                                                        chrint
С
                  = WR_CHARTYPE
                                                                        chrtyp
С
                        wringr = 1 - using E format in output
                               = 2 - using F format in output
С
С
                               = 3 - using G format in output
                  = WR_SUPTITLE - tlabel supress key
                                                                        kevhed
С
С
                   = WR_SUPSUBTITLE - subtitle supress key
                                                                        keytit
                  = WR_SUPLSITER - ls,iter id supress key
                                                                        keyid
                                 - note line supress key
                  = WR NOTELINE
                                                                        keynot
C
                  = WR_SUPCOLHEADER - column header supress key
С
                                                                        keylab
С
                  = WR_SUPCOLMAX - column maximum supress key
                                                                        kevsum

    ListOpt from /output command

C
                  = WR LISTOPT
                                                                       ListOpt
   output arguments:
С
      wringr
             (int,func,out)
                                  - the value corresponding to key
```

8.2.3. eringr Subroutine (Obtaining Information from the Errors Common)

```
1=keyerr, 2=errfil,
                                                           3=numnot, 4=numwrn,
C
С
                                    5=numerr, 6=numfat,
                                                          7=maxmsg, 8=lvlerr,
С
                                    9=mxpcmd, 10=nercmd, 11=nertim,12=nomore,
                                    13=eropen,14=ikserr, 15=kystat,16=mxr4r5,
С
                                                          19=opterr, 20=flowrn,
C
                                    17=mshkey,
C
                                    21=errhpi, 22=noreport, 23=pdserr, 24=mxpcmdw
                                    25=kystop, 26=icloads, 27=ifkey
С
  ---- below definitions copied from errcom 7/92 for user information
C
C
                             *** key number= .....
C
                     (see ansysdef for parameter definitions)
C
С
C
co keyerr - master error flag
                                                                  (ER ERRORFLAG)
co errfil - errors file unit number co numnot - total number of notes displayed
                                                                  (ER_ERRORFILE)
                                                                  (ER_NUMNOTE)
co numwrn - total number of warnings displayed
                                                                  (ER_NUMWARNING)
co numerr - total number of errors displayed
                                                                  (ER_NUMERROR)
co numfat - total number of fatals displayed
                                                                  (ER_NUMFATAL)
co maxmsg - max allowed number of displayed messages before abort(ER_MAXMESSAGE)
co lvlerr - used basicly in solution (from cnvr command.)
                                                                  (ER_ERRORLEVEL)
            -1=do not set keyerr for notes/errors/warnings.
CO
             -2=same as -1 but do not display message either.
co mxpcmd - maximum number of messages allowed per command
                                                                  (ER_MAXCOMMAND)
co nercmd - number of messages displayed for any one command
                                                                  (ER_NUMCOMMAND)
co nertim - key as to how message cleared from u/i pop-up
                                                                  (ER_UICLEAR)
            (as per rsg/pft 5/1/92 - only for "info" calls
             -1=message is timed before removal
CO
             0=message needs pick or keyboard before removal
              1=message stays up untill replaced by another message
          display any more messages
                                                                  (ER_NOMOREMSG)
co nomore
             0=display messages
            1=display discontinue message and stop displaying
CO
co eropen - 0=errors file is closed
                                                                  (ER_FILEOPEN)
            1=errors file is opened
co ikserr - 0=if interactive do not set keyerr
                                                                  (ER INTERERROR)
         - 1=if interactive set keyerr (used by mesher and tessalation)
co kystat - flag to bypass keyopt tests in the elcxx routines
                                                                  (ER KEYOPTTEST)
            associated with status/panel info inquiries.
C
             0=do not bypass keyopt tests
             1=perform all keyopt tests
C
С
            also flag to bypass setting of _STATUS upon resume
co mxr4r5 - mixed rev4-rev5 input logic (*do,*if,*go,*if-go)
                                                                  (ER_MIXEDREV)
             (used in chkmix called from rdmac)
C
             1=rev5 found (*do,*fi-then-*endif)
С
С
             2=rev4 found (*go,:xxx,*if,...,:xxx)
             3=warning printed. do not issue any more.
C
co mshkey - cpu intensive meshing etc. this will cause
            "nertim (11)" to be set to -1 for "notes", 1 for "warnings",
С
C
            and 0 for "errors". checking of this key is done in "anserr".
             0=not meshing or cpu intensive
             1=yes, meshing or cpu intensive
co syerro - systop error code. read by anserr if set.
                                                                  (18)
co opterr - 0=no error in main ansys during opt looping
                                                                  (ER_OPTLOOPING)
           1=an error has happened in main ansys during opt looping
co flowrn - flag used by "floqa" as to list floqa.ans
            0=list "floqa.ans"
С
            1="floqa.ans" has been listed. do not list again.
С
co noreport- used in GUI for turning off errors due to strsub calls (22)
С
            0=process errors as usual
С
            1=do NOT report errors
co pdserr - 0=no error in main ansys during pds looping
                                                                  (ER_PDSLOOPING)
            1=an error has happened in main ansys during pds looping
co mxpcmdw- number of messages written to file.err for any one
            command
CO
C
            0=write all errors to file.err
           1=only write displayed errors to file.err
co icloads - key to forbid the iclist command from listing solution
                                                                      (26)
             data instead of the input data.
C
            0=iclist is OK
С
            1=do not permit iclist
```

```
- key on whether or not to abort during /input on error
co ifkey
                                                                       (27)
           0=do not abort
С
           1=abort
С
co espare - spare integer variables
   --- end of information from errcom
С
c output arguments:
С
     eringr
             (int,sc,out)
                                - value corresponding to key
c *** mpg eringr < el117,el115,el126,el109,el53,el96,el97,edg?: get error stat
```

8.2.4. TrackBegin Subroutine (Beginning Tracking for a Subroutine Call)

8.2.5. TrackEnd Subroutine (Ending Tracking for a Subroutine Call)

8.2.6. erhandler Subroutine (Displaying ANSYS Errors)

```
*deck,erhandler
      subroutine erhandler (filein,msgid,msglvl,lngstrng,dperr,cherr)
c *** primary function:
                           Display ANSYS error messages
c *** Notice - This file contains ANSYS Confidential information ***
С
   input arguments:
      filein (ch*40,sc,in)
                                 - Filename used for character portion of
С
                                   message ID (this is the file name of the
С
                                   file which contains the source for this
C
                                   routine)
C
C
                                   if 'ErrorMessageProbe', then error was
С
                                     generated on another processor (distributed
С
                                     ANSYS). In that case, dperr contains the
```

```
message already made ASCII and expanded
C
С
С
      msgid
               (int,sc,in)
                                  - Numeric portion of the message ID
                                    1 - 9999, unique for each erhandler
С
С
                                    call in the FILE. Recommend using
С
                                    a sequence, similar to format conventions,
                                    i.e., 5000, 5010, 5020
С
C
                                    if filein='ErrorMessageProbe', this is the
                                      CPU # that originally generated the error
С
С
      msglvl
               (int,sc,in)
                                  - level of error (same as lngerr)
                                    0=no label (used for u/i pop-ups)
С
                                   -1=no label (used for u/i pop-ups) timed
C
С
                                      as a note message
С
                                    1=note, 2=warning, 3=error, 4=fatal
C
                                                       -3=error w/tech supp note
С
                                                       -4=fatal w/tech supp note
C
                                       (see lngerr.F for text of tech supp note)
      lngstrng (ch*(*),sc,in)
                                  - error message to display. use keywords
С
C
                                    of %i %g %c %/ for formating (same as
С
                                    lngerr)
               (dp,ar(*),in)
                                   - vector of data to display. contains both
С
      dperr
С
                                    integer and double precision data.
                                    (same as lngerr)
C
                                      if filein='ErrorMessageProbe', dperr
С
С
                                      contains the unpacked message and lngstrng
C
                                      and cherr are ignored
C
      cherr
               (ch*(*),ar(*),in) - vector of character data to display
C
                                    max length of character data is 32
                                    characters
C
```

8.2.7. intrp Subroutine (Doing Single Interpolation)

```
*deck,intrp
      subroutine intrp (klog,kppx,kstpz,xval,ax,ay,yval,nmax,kyoff)
c *** primary function: **** subroutine for single interpolation ****
С
                         (if double interpolation is needed, see intrpt)
C
С
  *** Notice - This file contains ANSYS Confidential information ***
C
      typ=int,dp,log,chr,dcp siz=sc,ar(n),func
                                                     intent=in.out.inout
C
C
С
   input arguments:
   variable (typ,siz,intent)
С
                                   description
С
      klog
                (int,sc,in)
                                   - interpolation type
                                     = 0 - use linear interpolation
C
С
                                     = 1 - use log-log interpolation
С
                                       -- note: there is no option yet for
                                                 lin-log or log-lin
C
                (int,sc,in)
                                   - X value end of table signal
С
      kppx
С
                                     = 0 - a repeated x-value will signal the end
                                           of the table
С
                                     = 1 - a repeated x-value will not signal the
C
С
                                           end of the table
С
                                           (only known use = c evaluation)
      kstpz
                (int,sc,in)
                                   - Y value end of table signal
C
                                     = 0 - a yval of zero will not signal the end
C
С
                                           of the table (e.g. stress fitting)
C
                                     = 1 - a yval of zero will signal the end of
                                           the table (in general, material
C
С
                                           properties (exception: alpx))
С
С
                           NOTE: the end of the table will be signaled thru
                                 either of the above conditions, or more
C
                                 commonly, that nmax values have been processed,
C
                                 or that the present x table entry is less than
С
                                 the previous one (ax(i) .lt. ax(i-1)).
С
                                 evaluations done after the end of the table are
C
                                 evaluated as if they were at the end of the
C
```

```
table. similarly, evaluations done before the
С
С
                                beginning of the table are done as if they were
С
                                done at the beginning of the table.
С
                                  - value of x with which to go into the table
С
     xval
               (dp,sc,in)
С
               (dp,ar(*),in)
                                 - table of x values, in ascending order
               (dp,ar(*),in)
                                  - table of y values
C
     av
                                  - maximum table size allowed
C
С
   output arguments:
               (dp,sc,out)
                                  - value of y which comes back from the table
С
     yval
      kyoff
                                  - xval status flag
С
               (int.sc.out)
С
                                    = 0 - xval in x range
С
                                    = 1 - xval less than minimum x
                                    = 2 - xval greater than maximum x
C
```

8.2.8. tranx3 Subroutine (Processing Geometry for 3-D Line Elements)

```
*deck,tranx3
     subroutine tranx3 (nnod,xyz,nx,tr)
c *** primary function: geometric processor for 3-d line elements
                        with or without a 3rd node
C
 *** Notice - This file contains ANSYS Confidential information ***
С
C
C
  input arguments:
С
     nnod
              (int,sc,in)
                                 - number of nodes (2 or 3)
               (dp,ar(nx,*),in) - coordinates (x,y,z down)
C
                                 - row dimension of xyz array
              (int.sc.in)
C
     nx
С
  output arguments:
             (dp,ar(3,3),in)
С

    transformation matrix

     tr
```

8.2.9. systop Subroutine (Stopping an ANSYS Program Run)

```
*deck,systop
     subroutine systop (icode)
c *** primary function:
                          stop an ansys run
c *** secondary functions: pass an error code to the system
c *** Notice - This file contains ANSYS Confidential information ***
С
С
   input arguments:
      icode
                                  - stop error code (0<icode<127)
С
               (int,sc,in)
                                    0 - normal exit
С
С
                                     1 - stack overflow error
                                     2 - stack level overflow
C
                                     3 - stack pop below zero
C
С
                                     4 - names do not match in stkpxp
                                     5 - command line argument error
С
                                     6 - unused (was: accounting file error)
C
                                     7 - licensing failure
С
С
                                     8 - indicated error or end-of-run
                                    11 - error in user routine
С
                                    12 - macro stop command
C
                                    13 - job already running
С
                                    14 - untrapped xox error
C
                                    15 - anserr fatal error
C
                                    16 - possible full disk
C
                                    17 - possible corrupted or missing file
С
                                    18 - Error in VM routines (corrupt db?)
С
                                    21 - unauthorized code section entered
C
                                    25 - unable to open x11 server
С
С
                                    30 - quit signal
                                    31 - failure to get signal in max time
С
```

```
c (syhold)
c >32 - system dependent error
c 35 - fatal error on another process
c (distributed ANSYS)
c
c output arguments: none
```

8.3. Vector Functions

8.3.1. vdot Function (Computing the Dot Product of Two Vectors)

```
*deck, vdot
     function vdot (v1,v2,n)
c *** primary function: compute dot product of vectors v1 and v2
c *** Notice - This file contains ANSYS Confidential information ***
c input arguments:
С
              (dp,ar(n),in)
                               - vector v1
С
     v2
              (dp,ar(n),in)
                               - vector v2
C
              (int,sc,in)
                               - length of vectors v1 and v2
c output arguments:
     vdot (dp,sc,out)
                              - dot product of v1 and v2
C
C
```

8.3.2. vidot Function (Computing the Dot Product of Two Vectors with Increments)

```
*deck, vidot
     function vidot (v1,inc1,v2,inc2,n)
c *** primary function: compute the dot product of vectors v1 and v2
                       (with increments)
С
c *** Notice - This file contains ANSYS Confidential information ***
С
С
  input arguments:
             (dp,ar(incl,n),in) - vector v1
     v1
C
С
     inc1
             (int,sc,in) - increment (first dimension) of vector v1
             (dp,ar(inc2,n,)in) - vector v2
С
     v2
     inc2
              (int,sc,in)
                                  - increment (first dimension) of vector v2
С
С
              (int,sc,in)
                                  - length (second dimension) of vectors
С
                                    v1, v2
C
c output arguments:
                                  - dot product
С
    vidot
            (dp,sc,out)
```

8.3.3. vsum Function (Summing Vector Components)

```
c output arguments:
c vsum (dp,sc,out) - vector sum
```

8.3.4. vmax Function (Retrieving the Maximum Vector Value at a Given Location)

```
*deck, vmax
     function vmax (v,n,locmax)
c *** primary function: return the max value and location in a vector
c *** Notice - This file contains ANSYS Confidential information ***
С
 input arguments:
              (dp,ar(n),in)
С
                                - vector v
                                - length of vector v
C
              (int,sc,in)
c output arguments:
   locmax (int,sc,out)
                                - location of max value in vector v
С
              (dp,sc,out)
                                - max value in vector v
С
     vmax
```

8.3.5. lastv Function (Retrieving the Position of the Last Nonzero Term in a Double Precision Vector)

```
*deck,lastv
     function lastv (v,n)
c ******* find position of last non-zero term in a d.p. vector ********
```

8.3.6. izero Function (Setting an Integer Vector to Zero)

8.3.7. imove Function (Assigning Equal Values to Two Integer Vectors)

8.3.8. vzero Subroutine (Initializing a Vector to Zero)

8.3.9. vmove Subroutine (Moving One Vector into Another)

```
*deck,vmove
     subroutine vmove (v1,v2,n)
c *** primary function: copy a vector into another vector
c *** Notice - This file contains ANSYS Confidential information ***
c *** Note: This function can have unpredictable results if v1 and v2
           reference the same array and overlap.
С
С
  input arguments:
C
    v1 (dp,ar(*),in) - vector v1
                               - length of vectors v1, v2
С
              (int,sc,in)
С
  output arguments:
С
         (dp,ar(*),out)
                              - vector v2
C
   v2
С
c *** mpg V2 = V1 vector move, see vimove for incremental move
```

8.3.10. vimove Subroutine (Moving One Vector into Another Incrementally)

```
*deck.vimove
     subroutine vimove (v1,inc1,v2,inc2,n)
c *** primary function: move one vector into another
c *** Notice - This file contains ANSYS Confidential information ***
С
С
 input arguments:
     v1
С
           (dp,ar(incl,n),in) - vector v1
C
     inc1
             (int,sc,in) - increment on vector v1
                                 - increment on vector v2
С
              (int,sc,in)
С
              (int,sc,in)
                                - number of items to be moved
С
c output arguments:
             (dp,ar(inc2,n),in) - vector v2
С
c *** mpg V2(1 \ldots inc2 * i) = V1(1 \ldots inc1 * i) vector incremental move
```

8.3.11. vinit Subroutine (Assigning a Scalar Constant to a Vector)

С

8.3.12. viinit Subroutine (Assigning a Scalar Constant to a Vector Incrementally)

```
*deck, viinit
     subroutine viinit (v,inc,n,const)
c *** primary function: set the components of vector v to const by increments
C
c *** Notice - This file contains ANSYS Confidential information ***
С
С
  input arguments:
              (int,sc,in)
                                 - increment (first dimension) of vector v
C
     inc
                                - length (second dimension) of vector v
              (int,sc,in)
С
     n
            (dp,sc,in)
                                  - constant to set components of vector v to
С
     const
c output arguments:
              (dp,ar(inc,n),out) - vector v
С
```

8.3.13. vapb Subroutine (Setting a Vector to Sum of Two Vectors)

```
*deck, vapb
     subroutine vapb (a,b,c,n)
c *** primary function: add vector a to vector b to get vector c
c *** Notice - This file contains ANSYS Confidential information ***
c input arguments:
С
     а
              (dp,ar(n),in)
                               - a vector
                             - b vector
С
     b
              (dp,ar(n),in)
                              - length of vectors a,b,c
     n
              (int,sc,in)
C
c output arguments:
C
     С
             (dp,ar(n),out)
                                - c vector
c *** mpg C = A + B vector add
```

8.3.14. vapb1 Subroutine (Combining Two Vectors in One)

8.3.15. vapcb1 Subroutine (Multiplying a Vector to a Constant)

8.3.16. vamb Subroutine (Gets a Third Vector by Subtracting One Vector from Another)

```
*deck, vamb
     subroutine vamb (a,b,c,n)
c *** primary function: subtract vector b from vector a to get vector c
c *** Notice - This file contains ANSYS Confidential information ***
c input arguments:
              (dp,ar(n),in) - vector a
(dp,ar(n),in) - vector b
           (dp,ar(n),in)
C
С
      h
              (int,sc,in)
                                 - length of vectors a,b,c
C
     n
c output arguments:
           (dp,ar(n),out)
                               - vector c
С
c *** mpg C = A - B vector substruct
```

8.3.17. vamb1 Subroutine (Subtracting One Vector from Another)

```
*deck,vamb1
      subroutine vamb1 (a,b,n)
c *** primary function: subtract vector b from vector a and save in vector a
c *** Notice - This file contains ANSYS Confidential information ***
c input arguments:
         (dp,ar(n),inout) - a vector
C
     а
               (dp,ar(n),in) - b vector
(int,sc,in) - length of vectors a,b
     b
              (int,sc,in)
С
     n
c output arguments:
            (dp,ar(n),inout) - a vector
C
   а
c *** mpg A = A - B vector substruct
```

8.3.18. vmult Subroutine (Multiplying a Vector by a Constant)

8.3.19. vmult1 Subroutine (Multiplying a Vector by a Constant)

```
*deck,vmult1
```

```
subroutine vmult1 (v1,n,const)
c *** primary function: multiply a vector by a constant
c *** Notice - This file contains ANSYS Confidential information ***
  input arguments:
            (dp,ar(n),inout) - vector v1
    v1
C
              (int,sc,in) - length of vector n
            (dp,sc,in)
                               - constant to multiply v1
C
     const
  output arguments:
          (dp,ar(n),inout) - vector v1
C
    v1
С
```

8.3.20. vcross Subroutine (Defining a Vector via a Cross Product)

```
*deck, vcross
      subroutine vcross (a,b,c)
c primary function: calculate c = a x b
c *** Notice - This file contains ANSYS Confidential information ***
      typ=int,dp,log,chr,dcp siz=sc,ar(n)
                                              intent=in,out,inout
C
  input arguments:
С
               (dp,ar(3),in)
                                 - first vector to be cross-multiplied
     b
               (dp,ar(3),in)
                                - second vector to be cross-multiplied
С
  output arguments:
             (dp,ar(3),out)
                                - resulting vector
C
С
```

8.3.21. vnorme Subroutine (Normalizing a Three-Component Vector)

```
*deck, vnorme
     subroutine vnorme (iel,v)
c primary function: normalize a vector to unit length
 this routine is to be called only from the elements.
                                                         it is only
c for a three component vector(i.e. processing geometry).
c this routine also differs from vnorm in that an error message is called
c if the vector length is zero.
c *** Notice - This file contains ANSYS Confidential information ***
   input arguments:
     iel
            (int,sc,inout)
                                - element number
C
C
              (dp,ar(3),inout) - vector to be normalized
  output arguments:
C
            (int,sc,inout)
                                - if 0, vector has zero length
С
              (dp,ar(3),inout) - normalized vector
С
```

8.3.22. vnorm Subroutine (Normalizing a Vector to Unit Length)

```
c input arguments:
c    v    (dp,ar(n),inout) - vector v
c    n    (int,sc,inout) - dimension length of vector v
c output arguments:
c    v    (dp,ar(n),inout) - normalized vector v
c    n    (int,sc,inout) - n = 0 if error in operation
```

8.3.23. ndgxyz Function (Getting the X,Y,Z Vector for a Node)

```
*deck,ndgxyz
     function ndgxyz (node,xyz)
c *** primary function:
                         get x,y,z vector for a node.
c *** Notice - This file contains ANSYS Confidential information ***
     input arguments:
        node
                (int,sc,in)
                                   - node number for operation.
С
     output arguments:
        ndgxyz (int,sc,out)
                                   - status of node.
                                       0=node is undefined.
C
                                      -1=node is unselected.
С
                                       1=node is selected.
                  (dp,ar(3),out)
                                   - vector containing x,y,z
        XVZ
c *** mpg ndgxyz<elmngt,elmnxt,esurf,edgcntae,edgrde,edgrdn,
         biolp0,biolp2,biot,biotlp : get node xyz
```

8.3.24. ndpxyz Subroutine (Storing X,Y,Z for a Node)

8.4. Matrix Subroutines

8.4.1. maxv Subroutine (Multiplying a Vector by a Matrix)

```
c output arguments:
c     w     (dp,ar(*),out) - product vector w
c
c *** mpg w = A v : A(nr,nc) : matrix vector product
```

8.4.2. maxv1 Subroutine (Multiplying a Vector by a Matrix)

```
*deck,maxv1
     subroutine maxv1 (a,v, nr,nc)
c *** primary function: multiply a vector by a matrix
c *** Notice - This file contains ANSYS Confidential information ***
c input arguments:
              (dp,ar(nr,nc),in) - matrix a
С
     а
С
              (dp,ar(nc),inout) - vector v
                             - number of rows in matrix a
              (int,sc,in)
C
     nr
                                  *** nr limited to 60 ***
                                - number of columns to multiply in matrix a
С
             (int,sc,in)
  output arguments:
             (dp,ar(nr),inout) - product, stored in vector v
C
C
c *** mpg v = A v : A(nr,nc) : matrix vector product, max 60 rows
```

8.4.3. matxv Subroutine (Multiplying a Vector by a Full Transposed Matrix)

```
*deck.matxv
     subroutine matxv (a,v,w, nr,nc)
c *** primary function: multiply vector by full transposed matrix
c *** Notice - This file contains ANSYS Confidential information ***
   input arguments:
              (dp,ar(nr,*),in) - matrix a (first dimension must = nr)
C
                                 - vector v (nv must be greater or equal
C
               (dp,ar(nv),in)
С
С
     nr
               (int,sc,in)
                                  - first dimension and number of active
                                       rows of the untransposed matrix a
C
                                       (also the number of active rows
С
                                        of vector v)
               (int,sc,in)
                                  - number of columns of the untransposed
С
     nc
С
                                       matrix a
                                       (also the number of computed items
C
С
                                       in the product vector w)
                                       if negative, accumulate
С
С
   output arguments:
              (dp,ar(na,*),out) - product vector w
С
С
c *** mpg A(nr,nc) : matrix transpose vector product
         w = A + v : if nr > 0
C
С
          w = w + A + v : if nr < 0
C
```

8.4.4. matxv1 Subroutine (Multiplying a Vector by a Full Transposed Matrix)

```
c *** Notice - This file contains ANSYS Confidential information ***
С
  input arguments:
С
              (dp,ar(nr,*),in)
                                - matrix a
              (dp,ar(nr),inout) - vector v
С
              (int,sc,in)
                                 - number of rows in matrix (un-transposed)
С
     nr
C
              (int.sc.in)
                                 - number of columns in matrix (un-transposed)
                                   *** nc limited to 60 ***
C
  output arguments:
              (dp,ar(nc),inout) - product, stored in vector v
C
c *** mpg A(nr,nc) : matrix transpose vector product
         v = A + v : max 60 nc
```

8.4.5. matxb Subroutine (Transposing a matrix)

```
*deck,matxb
     subroutine matxb (a,b,c, na,nb,nc, n1,n2,n3)
c *** primary function: (a)t * (b) = (c)
c *** Notice - This file contains ANSYS Confidential information ***
c input arguments:
С
              (dp,ar(na,*),in)
                                - matrix a
     а
              (dp,ar(nb,*),in) - matrix b
С
     b
              (int,sc,in)
                                - number of rows in matrix a
C
     na
С
              (int,sc,in)
                                - number of rows in matrix b
              (int,sc,in)
C
     nc
                                 - number of rows in matrix c
     n1
              (int,sc,in)
                                 - number of rows in matrix c to fill
С
     n2
              (int,sc,in)
                                 - number of columns in matrix c to fill
С
                                 - number of rows in matrix a and
     n3
              (int.sc.in)
С
С
                                  number of rows of matrix b
С
                                   to work with (the two need
C
                                   to be the same for the inner product)
С
                                   if n3 is negative, accumulate results in c
  output arguments:
С
             (dp,ar(nc,*),out) - product matrix c
 *** mpg C =
                A+ B
                          if n3 > 0
C
                         if n3 < 0
         C = C + A + B
C
         A(na,*) B(nb,*) C(nc,*) C:minor n1 * n2 n3: dot length
C
```

8.4.6. maat Subroutine (Changing a Matrix Value via Addition, Multiplication, and Transposition)

```
*deck, maat
     subroutine maat(a,c, nc,n, con)
c primary function: does con*a*at and sums the result onto c (a is a vector)
c *** Notice - This file contains ANSYS Confidential information ***
      typ=int,dp,log,chr,dcp siz=sc,ar(n) intent=in,out,inout
С
  input arguments:
              (dp,ar(*),in)
                                   - vector to be multiplied by itself to
С
                                     generate an nxn square matrix
C
                                      (a by a-transposed)
C
               (dp,ar(nc,*),inout) - matrix to be accumulated onto
C
     C
               (int,sc,in)
                                  - number of rows in the c matrix
С
     nc
               (int,sc,in)
                                  - size of square matrix
     n
```

```
c con (dp,sc,in) - multiplier on above square matrix
c output arguments:
c c (dp,ar(nc,*),inout) - matrix to be accumulated onto
c only the lower triangular matrix is done
c Note: this routine is usually followed by matsym,
c to do the complete matrix
```

8.4.7. matba Subroutine (Updating Matrix Value via Transposition, Multiplications, and Addition)

```
*deck, matba
     subroutine matba (a,b,c,na,nb,nc,n1,n2,work,con)
c primary function:
                      does con(at*b*a) and sums the result
c *** Notice - This file contains ANSYS Confidential information ***
 input arguments:
С
              (dp,ar(na,*),in)
                                - matrix a
С
              (dp,ar(nb,*),in) - matrix b (must be square,
С
     h
C
                                          and maximum dimension is (13,13)
              (dp,ar(nc,*),inout) - matrix c (see output)
С
              (int,sc,in) - number of rows in matrix a
С
    na
     nb
              (int,sc,in)
                                - number of rows in matrix b
C
                                - number of rows in matrix c
С
     nc
              (int,sc,in)
                                - number of rows in matrix c to fill
С
     n1
              (int,sc,in)
    n2
              (int,sc,in)
                                - number of columns in matrix c to fill
С
                                - multiplier on product added to sum
              (dp,sc,in)
c output arguments:
С
     C
              (dp,ar(nc,*),inout)-c=c+con*at*b*a
              (dp,ar(n2,*),out) - at*b (this byproduct is occasionally useful)
C
     work
c *** mpg C = C + con A+ B A A(na,*) B(nb,*) C(nc,*) C:minor n1 * n2
     see matbabd for block diagonal
С
```

8.4.8. matsym Subroutine (Filling the Upper Triangle from the Lower Triangle)

```
*deck, matsym
      subroutine matsym (a,nd,n)
c primary function:
                    fill upper triangle from lower triangle
c *** Notice - This file contains ANSYS Confidential information ***
      typ=int,dp,log,chr,dcp siz=sc,ar(n) intent=in,out,inout
C
  input arguments:
             (dp,ar(nd,*),inout) - matrix to have its lower triangular part
C
                                     copied to its upper triangular part
C
     nd
               (int.sc.in)
                                  - number of rows of the a matrix
C
               (int,sc,in)
                                   - size of matrix to be processed
 output arguments:
C
С
             (dp,ar(nd,*),inout) - matrix that has its lower triangular part
С
                                     copied to its upper triangular part
С
```

8.4.9. mctac Subroutine (Transposing a symmetric matrix)

```
*deck, mctac
```

```
subroutine mctac (a,na,c,nc,nold,nnew)
   **** function: do a = c(transpose) * a * c, where a is symmetric **
c *** Notice - This file contains ANSYS Confidential information ***
  input arguments:
C
С
               (dp,ar(na,na),inout) matrix to be pre and post multiplied
                                      (part operated on must be
C
С
                                      square(nold x nold) and symmetric)
                                    first dimension of the a matrix
C
               (dp,ar(nc,nnew),in) matrix to pre and post multiply a by
C
С
                                     (part used may be rectangular(nold x nnew))
С
      nc
               (int,sc,in)
                                    first dimension of the c matrix
C
     nold
               (int,sc,in)
                                    size of part of 'A' matrix that is
С
                                     to be processed(input size).
                                                                   maximum = 64
                                    size of part of 'A' matrix that
C
      nnew
               (int.sc.in)
                                      results from this operation(output size).
С
  output arguments:
C
С
               (dp,ar(na,na),inout) resulting matrix
                                      (still square(nnew x nnew) and symmetric).
C
```

8.4.10. tran Subroutine (Transposing a matrix)

```
*deck,tran
      subroutine tran (zs,tr,nz,ntr,nrow,irot)
c primary function: perform tr-transpose * zs * tr *********
c *** Notice - This file contains ANSYS Confidential information ***
  input arguments:
     variable (typ,siz,intent)
                                  description
C
C
               (dp,ar(nz,nz),inout) - matrix to be transformed
               (dp,ar(ntr,ntr),in) - transformation matrix
C
                                   - dimensioned size of zs matrix
               (int,sc,in)
C
     nz
С
     ntr
               (int,sc,in)
                                   - dimensioned size of tr matrix
                                   - number of rows of zs matrix to transform
С
     nrow
               (int,sc,in)
              (int,sc,in)
                                   - block size to transform(size of tr matrix)
     irot
С
c output arguments:
     variable (typ,siz,intent)
                                  description
C
               (dp,ar(nz,nz),inout) - transformed matrix
```

8.4.11. symeqn Subroutine (Solving Simultaneous Linear Equations)

```
*deck.symean
      function symeqn (a,nd,n,nc,defFlag)
c primary function: solve a set of simultaneous linear equations
c secondary functions: invert a matrix
C
C
           NOTE: this routine assumes that the matrix to be solved or
                  inverted is positive or negative definite. This routine
C
                  also assumes that the diagonals are all non-zero. If
C
                  this assumption is not true, use isimeq.F.
C
C
  *** Notice - This file contains ANSYS Confidential information ***
С
C
  input arguments:
C
      variable (typ,siz,intent)
С
                                    description
С
               (dp,ar(nd,*),inout) - matrix to be solved or inverted
С
                                        second dimension must be at least:
```

```
С
                                                                  n + abs(nc)
               (int,sc,in)
                                    - first dimension of the a matrix
С
      nd
                                    - number of equations
С
      n
               (int,sc,in)
               (int,sc,in)
                                    - number of additional columns.
С
      nc
                                        if nc = +n or -n, invert n \times n matrix and
С
                                        put result in the n+1 to 2xn columns.
С
                                        if nc is 0 or negative, nc will be reset to
С
С
                                        n and then symeqn will set up identity
                                        matrix after the input matrix, where the
C
С
                                        result of the inversion will be put.
С
                                        if nc is positive and less than n, do a
                                        partial inversion. see example 1 below.
C
      defFlag (int,sc,in)
                                    - flag indicating that incoming matrix MUST be:
С
С
                                        -1 - negative definite
                                         0 - positive or negative definite
С
                                         1 - positive definite
С
С
   output arguments:
C
С
      variable (typ,siz,intent)
                                    description
              (in,sc,out)
                                    - 0 - non-singular matrix
С
      symean
                                      1 - singular matrix
С
                                      2 - near-singular matrix
С
               (dp,ar(nd,*),inout) - results or inverted matrix.
С
                                        starts in column n+1.
С
С
                                        note: original information is destroyed.
С
С
    example 1:
                 Solve three simultaneous linear equations:
                       i = symeqn (a(1,1),3,3,1)
С
                    calling routine has a dimensioned as a(3,4)
С
С
                    each equation has its 3 coefficents in the first 3 columns,
                     and the constant term is in the fourth column.
С
                    solution is in fourth column.
С
С
                 Invert a 3x3 matrix:
    example 2:
С
С
                       i = symeqn (a(1,1),3,3,-3)
С
                    calling routine has a dimensioned as a(3,6)
С
                    input matrix was input in first 3 columns
                    output matrix in ouput in last 3 columns
С
```

Appendix A. Creating External Commands in UNIX

External commands allow you to add your own customized extensions to ANSYS without relinking the program. You can create custom routines in C that access any of the ANSYS API functions, link them into shared libraries using the supplied utilities, and execute the routines via the "external command" feature within ANSYS. In addition, ANSYS provides special commands that list all available external commands and allow you to reset all currently referenced external commands.

External command capability is supported on all UNIX platforms. Refer to your ANSYS, Inc. UNIX/Linux Installation Guide for currently supported compilers; the following instructions assume the presence of compatible compilers and linkers.

A.1. Tasks in Creating an External Command

To create a functional external command, you will need to complete the following general steps:

- Create compilable source code.
- Create a shared library. This is facilitated by the gen_share utility and your system's make capability.
- Create an external table file (ans_ext.tbl), listing the various shared libraries, functions, and the related command.
- Set an environment variable pointing to the directory that holds the external table file.

The following sections detail each of these tasks.

A.1.1. Creating Compatible Code

You can create your functions using any of the API functions described in //ansys_inc/v120/ansys/cus-tomize/include/cAnsInterface.h, cAnsQuery.h, and cAnsPick.h. The following code segment demonstrates, at a minimal level, how to create functions that can be used as an entry point into a custom coded shared library.

The most important point in the following example it that the C program interface is an integer function that has one argument (a char pointer).

```
Output
Return Value:
    The return value is ignored by the calling function;
  ----- End Function Description ------
int extfnc(char* uecmd)
   /* Note: uecmd is the entire command given to invoke this function */
   char* cmdsend = {"/COM, COMMAND SENT FROM EXTERNAL COMMAND"};
   char* querystr = {"NODE,,NUM,MAX"};
   char strrtn[32];
   int i, itype;
   double dblrtn;
   /* Send a simple command to be executed */
   i = cAnsSendCommand(cmdsend);
   /* Perform a simple query */
   i = cAnsGetValue(querystr,&dblrtn,strrtn,&itype);
   /* Display the value retrieved */
   cAns_printf("Max Node Number = %g\n",dblrtn);
   return (i);
```

A.1.2. Creating a Shared Library

Once you have written the source code for your functions, you can create a Makefile (using the **gen_share** utility) to build a shared library. The utility creates the Makefile in the current directory. The Makefile incorporates all the interdependencies of the C source files it encounters in that current directory. The **gen_share** utility is meant to setup the basic build. The user may need to make modifications to the Makefile depending on the situation.

The **gen_share** utility has the following syntax:

```
where

-h
Produces command help.

-64
Configures the Makefile to use the -mips4 option for IRIX64.

shared_object_name
Is the name that will be given to the shared library.
```

As gen_share is executing, you may see one or more "No match" messages. This is normal. The script is searching for .c, .f, and .F file types in the current directory and returns this message if it cannot locate any files matching one of those types.

To create a shared library called mylibrary.so, you would issue the following command:

```
% gen_share mylibrary.so
```

The utility will produce a Makefile in the current directory. You will be able to generate the shared library by issuing the following command:

make

For example, to create the shared library for mylibrary.so, you would issue the following command:

```
% make
```

You will then find the specified shared library file in the current directory. You may also see warnings from the make process, and you may need to modify the Makefile or your source code.

A.1.3. Creating an External Table File

The external table file (ans_ext.tbl) can reside in any directory (but you must specify that directory in the **ANSYS_EXTERNAL_PATH** environment variable). The file contains an entry for each shared library function you wish to allow ANSYS to access. There is no limit to the number of entries. The file entries have the following format:

```
/shared/library/path/library.so ~cm_name function_name
```

where

```
/shared/library/path/library.so
```

Is the path to the directory that contains the shared library file. (Remotely mounted file systems are not recommended.)

```
~cm_name
```

Is the command used to invoke the function within ANSYS. The command name must begin with a tilde (~) and each command name must be unique within the first four characters. The command name must be eight characters or less, including the tilde (~).

```
function_name
```

Is the name of the function that is referenced by the specified command name. (This must be unique within the first four characters if multiple external commands are specified.)

For example, the following entry references the /home/mydir/mylibs/myobject.so shared library and the myobject_function. It specifies ~myobj as the related command:

```
/home/mydir/mylibs/myobject.so ~myobj myobject_
```

ANSYS also makes use of external commands, and places its own shared libraries and the associated external table file in the /ansys_inc/v120/ansys/lib/<platform> directory (where <platform> is the directory specific to your computing platform, such as /sgi64 or /hppa8000-64).

ANSYS loads external commands in the following order:

- ANSYS first checks the ans_ext.tbl file in the /ansys_inc/v120/ansys/lib/<platform> directory and loads any external commands referenced there.
- ANSYS then loads external commands referenced by the external table file in the directory designated
 with the ANSYS_EXTERNAL_PATH environment variable (see section Setting the ANSYS_EXTERNAL_PATH
 Environment Variable (p. 306)).

If you designate a command name that has the same first four characters as a command listed in the /ansys_inc/v120/ansys/lib/<platform>/ans_ext.tbl file, you will not be able to access your command. Therefore, it is a good practice to check the ANSYS external table file to make sure you have no external command name conflicts. Do not modify the /ansys_inc/v120/ansys/lib/<plat-

form>/ans_ext.tbl file. You can also use the ~DEBUG command to verify that no external command
name conflicts exist.

Note

The shared library must be consistent with the computer type and OS level on which ANSYS will be executed.

A.1.4. Setting the ANSYS_EXTERNAL_PATH Environment Variable

Before launching ANSYS, you must first set the **ANSYS_EXTERNAL_PATH** to point to the directory containing the external table file. (For convenience, if you distribute your new functionality to other users they should set their .login or .cshrc files so that it is persistent from session to session.) For example, the following sets the environment variable to point to the /home/mydir directory.

```
setenv ANSYS_EXTERNAL_PATH /home/mydir
```

A.1.5. Using External Commands

To call an external command, enter it as you would any other ANSYS command. You can also call external commands through either an APDL macro or UIDL script.

Note

Avoid recursive external commands; that is, avoid situations where an external command calls another external command.

A.1.6. Checking External Command Status

You can check what shared libraries are currently accessible by entering the **~DEBUG** command in the command input window. The following figure shows an example of **~DEBUG** command output.

In this example, the output lists the command, the related shared library, the function, and if the command has been accessed.

A.1.7. Resetting External Commands

You can

- Close all shared libraries
- · Free memory associated with external commands

by issuing the **~RESET** command. The command issues the following message to confirm that the reset operation was complete.

```
~RESET was processed: The external command buffers have been cleared.
```

Note

The /CLEAR command also closes/resets all external command shared libraries.

Release 12.0 - © 2009 SAS IP, Inc. All rights rese	erved - Contains n	roprietary and co	nfidential information
			macmammommation
of ANSYS. Inc. and	d its subsidiaries a	and affiliates.	

Appendix B. Creating External Commands in Windows

This section describes the steps required to create external commands on Windows platforms.

B.1. Tasks in Creating an External Command

To create a functional external command, you will need to complete the following general steps:

- · Create compatible C source code.
- Create an external definition file (projname.def).
- Create a new project in Microsoft Developer Studio.
- Create a shared library.
- Create an external table file (ans_ext.tbl), listing the various shared libraries, each function and the
 related command.
- Set the ANSYS_EXTERNAL_PATH environment variable

The following sections detail each of these tasks.

B.1.1. Creating Compatible Code

You can create your functions using any of the API functions described in Program Files\Ansys Inc\V120\customize\include\cAnsInterface.h, cAnsQuery.h, and cAnspick.h. You can then execute these functions via the "external command" feature within ANSYS. In addition, ANSYS provides special commands that list all available external commands and allow you to reset all currently referenced external commands. The following code segment demonstrates, at a minimal level, how to create functions that can be used as an entry point into a custom coded shared library.

The most important point in the following example is:

• The C program interface is an integer function that has one argument (a char pointer).

```
Output
Return Value:
    The return value is ignored by the calling function;
    ----- End Function Description ------
int extfnc(char* uecmd)
   /* Note: uecmd is the entire command given to invoke this function */
   char* cmdsend = {"/COM, COMMAND SENT FROM EXTERNAL COMMAND"};
   char* querystr = {"NODE,,NUM,MAX"};
       char strrtn[32];
   int i, itype;
   double dblrtn;
   /* Send a simple command to be executed */
   i = cAnsSendCommand(cmdsend);
   /* Perform a simple query */
   i = cAnsGetValue(querystr,&dblrtn,strrtn,&itype);
   /* Display the value retrieved */
   cAns_printf("Max Node Number = %g\n",dblrtn);
   return (i);
   }
```

B.1.2. Creating a New Project

To build and link your code, you will first need to create a project in the Microsoft Developer Studio.

- After launching Developer Studio, press Ctrl+N (or choose New from the FILE menu). In the New
 window, click on the Projects tab. Select Win 32 Dynamic-Link Library, and specify your project name
 and directory.
- In the Workspace frame, click on the **File View** tab. Right click on the sub category **projname files** and press **F** or select **Add Files to Project**). Choose your source and definition files, ansys.lib, cAnsInterface.h and any other necessary header files. The import library file ansys.lib is in *Program Files\Ansys Inc\V120\ANSYS\custom\lib\platform* and the header file cAnsysInterface.h is in \Program Files\Ansys Inc\V120\ANSYS\customize\include.

B.1.3. Creating an External Definition File

For each external function, you must declare it in the external definition file. The naming convention for this file is the name of your project with the .def extension; it must be located in your project directory. This file consists of the word *EXPORTS* on the first line, and the name(s) of the functions to be exported on each successive line. For the example function above:

```
EXPORTS extfunc
```

B.1.4. Creating a Shared Library

Once all of the necessary files have been incorporated into your project, simply compile (Ctrl+F7) and build (F7) the project. In your project directory, Developer Studio will create a Debug directory and will place the library in that directory (projname.dll).

B.1.5. Creating an External Table File

The external table file (ans_ext.tbl) can reside in any directory (but you must specify that directory in the ANSYS_EXTERNAL_PATH environment variable). The file contains an entry for each shared library function you wish ANSYS to access. There is no limit to the number of entries. The file entries have the following format:

C:\shared\library\path\projname.dll ~cm_name function_name

where:

C:\shared\library\path\projname.dll is the path to the directory that contains the shared library file. (Remotely mounted file systems are not recommended.)

~cm_name Is the command used to invoke the function within ANSYS. The command name must begin with a tilde (~) and the first four characters of each command name must be unique.

function_name Is the name of the function that is referenced by the specified command name. (This must be unique within the first four characters if multiple external commands are specified.)

For example, the following entry references the C:\home\mydir\mylibs\myobject.dll shared library and the myobject function, and specifies ~myobj as the related command:

```
C:\home\mydir\mylibs\myobject.dll ~myobj myobject
```

ANSYS also makes use of external commands, and places its own shared libraries and the associated external table file in the C:\Program Files\Ansys Inc\V120\lib\platform directory (where platform is the directory specific to your computing platform, such as \Intel). ANSYS loads external commands in the following order:

- ANSYS first checks the ans_ext.tbl file in the C:\Program Files\Ansys
 Inc\V120\lib\platform directory and loads any external commands referenced there.
- ANSYS then loads external commands referenced by the external table file in the directory designated
 with the ANSYS_EXTERNAL_PATH environment variable (see Setting the ANSYS_EXTERNAL_PATH Environment Variable (p. 311)).

If you designate a command name that has the same first four characters as a command listed in the C:\Program Files\Ansys Inc\V120\lib\platform file, you will not be able to access your command. Therefore, it is a good practice to check the ANSYS external table file to make sure you have no external command name conflicts. Do not modify the C:\Program Files\Ansys Inc\V120\lib\platform\ans_ext.tbl file. You can also use the ~DEBUG command to verify that no external command name conflicts exist.

Note

The shared library must be consistent with the computer type and OS level on which ANSYS will be executed.

B.1.6. Setting the ANSYS_EXTERNAL_PATH Environment Variable

Before launching ANSYS, you must first set the **ANSYS_EXTERNAL_PATH** to point to the directory containing the external table file. In Windows NT, the environment variables are in System Properties, which can be

accessed through the Control Panel. For example, the following string sets the environment variable to point to the C:\home\mydir directory.

```
set ANSYS_EXTERNAL_PATH=C:\home\mydir
```

B.1.7. Using External Commands

To call an external command, enter it as you would any other ANSYS command in the ANSYS command window. You can also call external commands through either an APDL macro or UIDL routine.

Note

Avoid recursive external commands; that is, avoid situations where an external command calls another external command.

B.1.8. Checking External Command Status

You can check what shared libraries are currently accessible by entering the **~DEBUG** command in the command input window. The following figure shows an example of **~DEBUG** command output.

Note that the output lists the command, the related shared library, the function, and whether or not the command has been accessed.

B.1.9. Resetting External Commands

You can

- Close all shared libraries
- Free memory associated with external commands

by issuing the **~RESET** command. This command issues the following message to confirm that the reset operation is complete.

```
~RESET was processed: The external command buffers have been cleared.
```

Note

The /CLEAR command also closes/resets all external command shared libraries.

Appendix C. User Material (UserMat) Subroutine Example

This example of a simple bilinear plasticity material model (identical to **TB**,BISO) demonstrates the user material subroutine UserMat, described in *Subroutine UserMat* (*Creating Your Own Material Model*) (p. 156).

C.1. UserMat Example Description

The example subroutine defines a 3-D material with plane strain and axisymmetric stress states. The analysis uses the 3-D solid element SOLID185. Comparison is made with the prediction by ANSYS **TB**,BISO material option.

The example is a two-element test case under simple tension. Element 1 has material defined using the **TB**,USER option, while Element 2 has material defined using the **TB**,BISO option. A 100-percent deformation is applied to both elements. Finite deformation (**NLGEOM**, ON) is considered. The POST26 processor results of stress components (Sxx, Syy) and plastic strain components (EPxx, EPyy) are printed for both elements. They are expected to be the same.

C.2. UserMat Example Input Data

```
/batch,list
/title, mvpl-um01, gal, usermat.F test case
/com.
/com, This is a single element test case for testing usermat.F
/com, usermat.F is the user materials subroutine
/com for current-technology elements.
/com, The material subroutine provided as the example
/com, is the same as the TB, BISO.
/com, A side by side comparison is made for two 185 elements,
/com, among which one is defined by TB,BISO, and another
/com, is defined as TB,USER. They are expected to produce
/com, the same results.
/com, uniaxial tension stress, large deformation.
/com.
/nopr
/nolist
/prep7
ele1=185
ele2=185
mat1=1
mat 2=2
et,1,ele1
keyopt,1,2,1
mat,mat1
block, 0, 1, 0, 1, 0, 1
esize,.1
vmesh,1
mat, mat2
block,0,1,0,1,0,1
esize,,1
vmesh,2
elist
```

```
! define material 1 by tb,biso
mp,ex ,mat1,20e5
mp,nuxy,mat1,0.3
tb, biso, mat1, 2, 4
tbtemp,1.0
tbdata, 1, 1e3, 100,
tbtemp,2.0
tbdata, 1, 2e3, 100,
! define material 2 by tb,user
tb,user,mat2,2,4
tbtemp,1.0
                                      ! first temp.
tbdata,1,19e5, 0.3, 1e3,100,
                                     ! E, posn, sigy, H
tbtemp,2.0
tbdata,1,21e5, 0.3, 2e3,100,
                                      ! define 8 state variables
tb, state, mat2,,8
! boundary condition
nsel,s,loc,x
d,all,ux
nall
nsel,s,loc,y
d,all,uy
nall
nsel,s,loc,z
d,all,uz
nall
fini
/solu
tunif.1.5
nlgeom, on
nsel,s,loc,y,1
nsubst,20,100,1
d,all,uy,1.0
time,1
nall
outres,,-10
outpr,all,-10
solv
fini
/post26
esol,2,1,,s,x,SX\_BISO
esol, 3, 2, s, x, SX\_USER
esol, 4, 1, , s, y, SY_BISO
esol,5,2,,s,y,SY_USER
esol,6,1,,eppl,x,EPX_BISO
esol,7,2,,eppl,x,EPX_USER
esol,8,1,,eppl,y,EPY_BISO
esol,9,2,,eppl,y,EPY_USER
prvar, 2, 3, 4, 5
prvar,6,7,8,9
fini
/exit,no save
```

C.3. UserMat Example POST26 Output Results

***** ANSYS POST26 VARIABLE LISTING *****

TIME	1 S X	2 S X	1 S Y	2 S Y
	SX_BISO	SX_USER	SY_BISO	SY_USER
0.10000	-0.188102E-02	-0.188102E-02	1509.45	1509.45
0.28750	-0.110968	-0.110968	1525.07	1525.07

0.45625 0.66204 0.89592 1.0000	-0.814415 -1.73160 -1.86240 -0.176924E-01	-0.814415 -1.73160 -1.86240 -0.176924E-01	1536.67 1548.95 1561.97 1569.16	1536.67 1548.95 1561.97 1569.16
****	ANSYS POST26	VARIABLE LISTING	G ****	
TIME	1 EPPLX EPX_BISO	2 EPPLX EPX_USER	1 EPPLY EPY_BISO	2 EPPLY EPY_USER
0.10000	-0.472687E-01	-0.472687E-01	0.945374E-01	0.945374E-01
0.28750	-0.125917	-0.125917	0.251834	0.251834
0.45625	-0.187417	-0.187417	0.374835	0.374835
0.66204	-0.253409	-0.253409	0.506818	0.506818
0.89592	-0.319141	-0.319141	0.638282	0.638282
1.0000	-0.345853	-0.345853	0.691707	0.691707

C.4. USERMAT.F List File for This Example

```
subroutine usermat(
                        matId, elemId,kDomIntPt, kLayer, kSectPt,
     δ
     &
                        ldstep, isubst, keycut,
                        nDirect, nShear, ncomp, nStatev, nProp,
     &
                        Time, dTime, Temp, dTemp,
                         stress, statev, dsdePl, sedEl, sedPl, epseq,
                        Strain, dStrain, epsPl, prop, coords,
                        var0, defGrad_t, defGrad,
                         tsstif, epsZZ,
     æ
                        var1, var2, var3, var4, var5,
                        var6, var7, var8)
      *** primary function ***
C
С
            user defined material constitutive model
С
С
      Attention:
C
           User must define material constitutive law properly
С
            according to the stress state such as 3-D, plain strain
C
            and axisymmetry, plane stress and beam.
C
C
С
            a 3-D material constitutive model can be used for
С
            plain strain and axisymmetric cases.
C
С
            When using shell elements, the plane stress algorithm
С
           must be used.
C
        The following demonstrates a USERMAT subroutine for
С
        a plasticity model in 3-D stress state. The plasticity
С
C
        model is the same as TB, BISO.
        See "ANSYS user material subroutine USERMAT" for detailed
C
        description of how to write a USERMAT routine.
C
С
C
С
      input arguments
С
      _____
      matId
С
                (int,sc,i)
                                         material #
С
       elemId
                 (int,sc,i)
                                         element #
      kDomIntPt (int,sc,i)
                                         "k"th domain integration point
C
C
      kLayer
                (int,sc,i)
                                         "k"th layer
С
      kSectPt
                (int,sc,i)
                                         "k"th Section point
      ldstep
                (int,sc,i)
                                         load step number
С
С
       isubst
                 (int,sc,i)
                                         substep number
С
      nDirect
                 (int,sc,in)
                                         # of direct components
С
      nShear
                (int.sc.in)
                                         # of shear components
                                         nDirect + nShear
С
      ncomp
                (int,sc,in)
С
      nstatev
                (int,sc,1)
                                         Number of state variables
C
                (int,sc,l)
                                         Number of material ocnstants
      nProp
C
      Temp
                (dp,sc,in)
                                         temperature at beginning of
```

```
time increment
С
С
       dTemp
                 (dp,sc,in)
                                           temperature increment
С
       Time
                 (dp,sc,in)
                                           current time
       dTime
                 (dp,sc,in)
                                           current time increment
С
С
С
       Strain
                (dp,ar(ncomp),i)
                                           Strain at beginning of time increment
       dStrain (dp,ar(ncomp),i)
                                           Strain increment
С
C
                (dp,ar(nprop),i)
                                           Material constants defined by TB, USER
       coords
                (dp,ar(3),i)
                                           current coordinates
C
С
      defGrad_t(dp,ar(3,3),i)
                                           Deformation gradient at time t
       defGrad (dp,ar(3,3),i)
                                           Deformation gradient at time t+dt
С
C
С
      input output arguments
С
      ______
C
      stress
                (dp,ar(nTesn),io)
                                           stress
С
       statev
                (dp,ar(nstatev),io)
                                           statev
                                           elastic work
С
       sedEl
                (dp,sc,io)
                (dp,sc,io)
      sedPl
                                           plastic work
C
С
       epseq
                (dp,sc,io)
                                           equivalent plastic strain
C
      var?
                (dp,sc,io)
                                           not used, they are reserved arguments
                                           for further development
С
С
      output arguments
С
С
      ===========
С
      keycut
               (int,sc,io)
                                           loading bisect/cut control
С
                                           0 - no bisect/cut
                                           1 - bisect/cut
С
                                           (factor will be determined by ANSYS solution control)
C
                                           material jacobian matrix
       dsdePl
                (dp,ar(ncomp,ncomp),io)
C
C*3
С
С
       ncomp
                   for 3-D
                   for plane strain/stress, axisymmetric
               4
C
       ncomp
С
      ncomp
                   for 1-D
C
       stresss and strains, plastic strain vectors
C
           11, 22, 33, 12, 23, 13
                                     for 3-D
С
С
           11, 22, 33, 12
                                      for Plane strain/stress, axisymmetry
          11
                                      for 1D
C
C
      material jacobian matrix
C
С
         3-D
С
            dsdePl
                         1111
                                1122
                                       1133
                                               1112
                                                      1123
                                                             1113
            dsdePl
                                2222
                                        2233
                                                      2223
                                                             2213
                         2211
                                               2212
C
            dsdePl
                         3311
                                3322
                                        3333
                                               3312
                                                      3323
                                                             3313
С
С
            dsdePl
                         1211
                                1222
                                        1233
                                               1212
                                                      1223
                                                             1213
            dsdePl
                         2311
                                2322
                                        2333
                                               2312
                                                      2323
                                                             2313
C
            dsdePl
                         1311
                                1322
                                        1333
                                               1312
                                                      1323
                                                             1313
С
С
         plane strain/stress, axisymmetric
            dsdePl
                      1111
                                1122
                                        1133
                                               1112
C
            dsdePl
                         2211
                                2222
                                        2233
                                               2212
            dsdePl
                         3311
                                3322
                                        3333
                                               3312
С
С
                         1211
                                1222
                                        1233
                                               1212
С
            for plane stress entities in third colum/row are zero
C
С
                      | 1111 |
C**
#include "impcom.inc"
      INTEGER
                       matId, elemId,
     &
                       kDomIntPt, kLayer, kSectPt,
     &
                       ldstep, isubst, keycut,
                       nDirect, nShear, ncomp, nStatev, nProp
     DOUBLE PRECISION
                       Time,
                                dTime,
                                          Temp,
                                                   dTemp,
     &
                       sedEl,
                                sedPl,
                                          epseq,
                                                   epsZZ
     DOUBLE PRECISION
     &
                       stress
                                (ncomp ), statev (nStatev),
     8
                       dsdePl (ncomp, ncomp),
```

```
Strain (ncomp ), dStrain (ncomp ),
    &
     &
                               (ncomp ), prop
    æ
                       coords (3),
                       defGrad_t(3,3),
                                         defGrad(3,3),
    &
                       tsstif (2)
c --- parameters
С
                      NEWTON, mcomp
     DOUBLE PRECISION HALF, THIRD, ONE, TWO, SMALL, ONEHALF,
                      ZERO, TWOTHIRD, ONEDM02, ONEDM05, sqTiny
     PARAMETER
                      (ZERO
                                 = 0.d0,
    δ
                      HALF
                                  = 0.5d0,
                       THIRD
                                 = 1.d0/3.d0,
    &
    &
                      ONE
                                 = 1.d0,
                       TWO
                                  = 2.d0,
                       SMALL
                                  = 1.d-08,
                       sqTiny
                                 = 1.d-20,
                       ONEDM02
                                 = 1.d-02,
     &
                       ONEDM05
                                  = 1.d-05
                       ONEHALE
                                  = 1.5d0.
                                 = 2.0d0/3.0d0,
                       TWOTHIRD
                       NEWTON
                                 = 10,
                                  = 6
    δ
                       mcomp
     &
C
  --- local variables
С
С
       sigElp
                (dp,ar(6 ),1)
                                         trial stress
С
                (dp,ar(6,6),1)
С
       dsdeEl
                                         elastic moduli
С
       sigDev
                (dp,ar(6),1)
                                         deviatoric stress tensor
       dfds
                (dp,ar(6),1)
                                         derivative of the yield function
C
С
                (dp,ar(6,6),1)
                                         2D matrix for a 4 order tensor
С
      pEl
                (dp,sc
                          ,1)
                                         hydrostatic pressure stress
                          ,1)
С
       qEl
                (dp,sc
                                         von-mises stress
      pleq_t
                (dp,sc
                                         equivalent plastic strain at
С
                          ,1)
С
                                         beginnig of time increment
                (dp,sc
                                         equivalent plastic strain at end
С
      pleq
                          ,1)
                                         of time increment
С
       dpleq
                           ,1)
                                         incremental equivalent plastic strain
C
                (dp,sc
                          ,1)
                                         correction of incremental
С
       cpleq
                (dp,sc
С
                                         equivalent plastic strain
                                         yield stress at beginnig of time increments
       sigy_t
                (dp,sc
                           ,1)
C
С
                (dp,sc
                          ,1)
                                         yield stress at end of time
       sigy
С
                                         increment
                          ,1)
                                         Young's modulus
C
                (dp,sc
       young
                (dp,sc
                          ,1)
                                         Poiss's ratio
С
       posn
                                         initial yield stress
С
       sigy0
                (dp,sc
                           ,1)
                          ,1)
                                         plastic slope
C
       dsigdep
                (dp,sc
С
       twoG
                (dp,sc
                          ,1)
                                         two time of shear moduli
                          ,1)
       threeG
                (dp,sc
                                         three time of shear moduli
C
                                         nonlinear function to be solved
С
       funcf
                (dp,sc
                           ,1)
С
                                         for dpleq
                                         derivative of nonlinear function
                          ,1)
C
       dFdep
                (dp,sc
С
                                         over dpleq
С
   -- temporary variables for solution purpose
С
С
       i, j
       threeOv2qEl, oneOv3G, qElOv3G, con1, con2, fratio
С
C
     DOUBLE PRECISION sigElp(mcomp), dsdeEl(mcomp,mcomp), G(mcomp),
                      sigDev(mcomp), JM (mcomp,mcomp), dfds(mcomp)
    &
     DOUBLE PRECISION var0, var1, var2, var3, var4, var5,
                      var6, var7, var8
     DATA G/1.0D0,1.0D0,1.0D0,0.0D0,0.0D0,0.0D0/
C
                       i, j
     DOUBLE PRECISION pEl,
                             qEl,
                                      pleq_t, sigy_t , sigy,
```

```
cpleq, dpleq,
     &
                                        pleq,
     &
                       young, posn,
                                        sigy0,
                                                 dsigdep,
     &
                       elast1, elast2,
                       twoG, threeG,
     &
                                       oneOv3G, qElOv3G, threeOv2qEl,
                       funcf, dFdep,
     &
                                        fratio, con1,
С
      keycut
               = 0
      dsigdep = ZERO
      pleg t
               = statev(1)
     pleq
               = pleq t
     get Young's modulus and Poisson's ratio, initial yield stress and others
              = prop(1)
     young
      posn
               = prop(2)
      siqy0
               = prop(3)
c *** calculate the plastic slope
      dsigdep = young*prop(4)/(young-prop(4))
               = young / (ONE+posn)
      twoG
               = ONEHALF * twoG
C
 *** calculate elastic stiffness matrix (3-D)
С
С
C
      elast1=young*posn/((1.0D0+posn)*(1.0D0-TW0*posn))
      elast2=young/(TWO*(1.0D0+posn))
      dsdeEl(1,1) = (elast1+TWO*elast2)*G(1)*G(1)
      dsdeEl(1,2)=elast1*G(1)*G(2)+elast2*TWO*G(4)*G(4)
      dsdeEl(1,3)=elast1*G(1)*G(3)+elast2*TWO*G(5)*G(5)
      dsdeEl(1,4)=elast1*G(1)*G(4)+elast2*TWO*G(1)*G(4)
      dsdeEl(1,5)=elast1*G(1)*G(5)+elast2*TWO*G(1)*G(5)
      dsdeEl(1,6) = elast1*G(1)*G(6) + elast2*TWO*G(4)*G(5)
      dsdeEl(2,2) = (elast1+TWO*elast2)*G(2)*G(2)
      dsdeEl(2,3)=elast1*G(2)*G(3)+elast2*TWO*G(6)*G(6)
      dsdeEl(2,4)=elast1*G(2)*G(4)+elast2*TWO*G(1)*G(4)
      dsdeEl(2,5) = elast1*G(2)*G(5) + elast2*TWO*G(1)*G(5)
      dsdeEl(2,6)=elast1*G(2)*G(6)+elast2*TWO*G(2)*G(6)
      dsdeE1(3,3) = (elast1+TWO*elast2)*G(3)*G(3)
      dsdeEl(3,4)=elast1*G(3)*G(4)+elast2*TWO*G(5)*G(6)
      dsdeE1(3,5) = elast1*G(3)*G(5) + elast2*TWO*G(5)*G(3)
      dsdeEl(3,6)=elast1*G(3)*G(6)+elast2*TWO*G(6)*G(3)
      dsdeEl(4,4) = elast1*G(4)*G(4) + elast2*(G(1)*G(2)+G(4)*G(4))
      dsdeEl(4,5)=elast1*G(4)*G(5)+elast2*(G(1)*G(6)+G(5)*G(4))
      dsdeE1(4,6) = elast1*G(4)*G(6) + elast2*(G(4)*G(6)+G(5)*G(2))
      dsdeE1(5,5)=elast1*G(5)*G(5)+elast2*(G(1)*G(3)+G(5)*G(5))
      dsdeEl(5,6)=elast1*G(5)*G(6)+elast2*(G(4)*G(3)+G(5)*G(6))
      dsdeEl(6,6)=elast1*G(6)*G(6)+elast2*(G(2)*G(3)+G(6)*G(6))
      do i=1,ncomp-1
        do j=i+1,ncomp
          dsdeEl(j,i)=dsdeEl(i,j)
        end do
      end do
C
c *** calculate the trial stress and
      copy elastic moduli dsdeEl to material Jacobian matrix
C
      do i=1,ncomp
         sigElp(i) = stress(i)
         do j=1,ncomp
            dsdePl(j,i) = dsdeEl(j,i)
            sigElp(i) = sigElp(i)+dsdeEl(j,i)*dStrain(j)
         end do
      end do
c *** hydrostatic pressure stress
     pEl = -THIRD * (sigElp(1) + sigElp(2) + sigElp(3))
c *** compute the deviatoric stress tensor
      sigDev(1) = sigElp(1) + pEl
      sigDev(2) = sigElp(2) + pEl
      sigDev(3) = sigElp(3) + pEl
      sigDev(4) = sigElp(4)
      sigDev(5) = sigElp(5)
      sigDev(6) = sigElp(6)
c *** compute von-mises stress
      qEl =
```

```
& sigDev(1) * sigDev(1) + sigDev(2) * sigDev(2) +
     & sigDev(3) * sigDev(3)+
     & TWO*(sigDev(4) * sigDev(4)+ sigDev(5) * sigDev(5)+
     & sigDev(6) * sigDev(6))
     qEl = sqrt( ONEHALF * qEl)
c *** compute current yield stress
              = sigy0 + dsigdep * pleq
      siav
С
      fratio = qEl / sigy - ONE
c *** check for yielding
      IF (sigy .LE. ZERO.or.fratio .LE. -SMALL) GO TO 500
C
      sigy_t = sigy
      threeOv2qEl = ONEHALF / qEl
c *** compute derivative of the yield function
      DO i=1, ncomp
         dfds(i) = threeOv2qEl * sigDev(i)
      END DO
      oneOv3G = ONE / threeG
      qElOv3G = qEl * oneOv3G
c *** initial guess of incremental equivalent plastic strain
              = (qEl - sigy) * oneOv3G
      dpleq
               = pleq_t + dpleq
      pleq
c *** Newton-Raphosn procedure for return mapping iteration
      DO i = 1, NEWTON
         sigy
               = sigy0 + dsigdep * pleq
         funcf = qElOv3G - dpleq - sigy * oneOv3G
         dFdep = - ONE - dsigdep * oneOv3G
         cpleq = -funcf / dFdep
         dpleq = dpleq + cpleq
         --- avoid negative equivalent plastic strain
С
         dpleq = max (dpleq, sqTiny)
         pleq = pleq_t + dpleq
         fratio = funcf/qElOv3G
c ***
         check covergence
         IF (((abs(fratio) .LT. ONEDM05
                                              ) .AND.
             (abs(cpleq ) .LT. ONEDM02 * dpleq)) .OR.
((abs(fratio) .LT. ONEDM05 ) .AND
     &
                                               ) .AND.
              (abs(dpleq ) .LE. sqTiny
                                                ))) GO TO 100
     &
      END DO
c *** Uncovergence, set keycut to 1 for bisect/cut
      keycut
      GO TO 990
100 CONTINUE
С
c *** update stresses
      con1 = twoG * dpleq
      DO i = 1 , ncomp
         stress(i) = sigElp(i) - con1 * dfds(i)
      END DO
C
c *** update plastic strains
      DO i = 1 , nDirect
        epsPl(i) = epsPl(i) + dfds(i) * dpleq
      DO i = nDirect + 1 , ncomp
         epsPl(i) = epsPl(i) + TWO * dfds(i) * dpleq
      END DO
      epseq = pleq
c *** Update state variables
      statev(1) = pleq
c *** Update plastic work
      sedPl = sedPl + HALF * (sigy_t+sigy)*dpleq
c *** Material Jcobian matrix
С
      IF (qEl.LT.sqTiny) THEN
         con1 = ZERO
      ELSE
         con1 = threeG * dpleq / qEl
```

```
END IF
      con2 = threeG/(threeG+dsigdep) - con1
      con2 = TWOTHIRD * con2
      DO i=1,ncomp
         DO j=1,ncomp
            JM(j,i) = ZERO
         END DO
      END DO
      DO i=1,nDirect
        DO j=1,nDirect
           JM(i,j) = -THIRD
         END DO
        JM(i,i) = JM(i,i) + ONE
      END DO
      DO i=nDirect + 1.ncomp
         JM(i,i) = HALF
      END DO
      DO i=1,ncomp
        DO j=1,ncomp
            dsdePl(i,j) =
                             dsdeEl(i,j) - twoG
                 * ( con2 * dfds(i) * dfds(j) + con1 * JM(i,j))
         END DO
      END DO
      goto 600
 500 continue
c *** Update stress in case of elastic/unloading
      do i=1,ncomp
         stress(i) = sigElp(i)
     end do
  600 continue
c *** Claculate elastic work
      sedEl = ZERO
      DO i = 1 , ncomp
         sedEl = sedEl + stress(i)*(Strain(i)+dStrain(i)-epsPl(i))
              = sedEl * HALF
      sedEl
990
     CONTINUE
C
      return
      end
```

C.5. Accessing Solution and Material Data

These APIs are provided for your convenience to help you access solution and material data easily.

```
c *** subroutine get_ElmInfo(inquire, value)
С
C
         description
С
            function to inquire element and solution information
С
         definition
            inquire - query argument (string)
C
            value
                   - value of query argument
С
         variables
С
            inquire
                           - value
С
            'LDSTEP'
                           - load step number
            'ISUBST'
                           - substep step number
C
            'IEQITR'
С
                           - current interation number
С
            'NUMINTG'
                           - number of gauss integration
            'ELEMID'
                           - element number
С
С
            'MATID'
                           - material number of current element
                           - number of state variable for current material at
С
          'NSVAR'
                             gauss intg.
C
            'NCOMP'
                           - number of vector components, such as stresses
c *** subroutine get_ElmData (inquire, elemId, kIntg, nvect, vect)
С
```

```
description
C
            function to get/inquire solution dependent variables
С
С
            such as stress, strains at gauss intg. point.
         definition
С
            inquire
                           - query argument (string)
С
                           - element number
С
            elemId
            kIntg
                           - gauss intg. number
С
С
            nvect
                           - number of vector to be inquired
                           - vector to be inquired
            vect
C
С
         variables
С
           'SIG '
                           - stress vector
           'EPTO'
                           - Total strain vector (EPEL+EPPL+EPCR+EPTH)
C
                           - plastic strain vector
С
           'EPPL'
С
           'EPCR'
                           - creep strain vector
C
           'EPTH'
                           - thermal strain vector
                           - Initial stress vector
С
           'ISIG'
                           - accumulated equivalent plastic strain
C
           'PLEQ'
           'CREQ'
                           - accumulated equivalent creep strain
C
           'SVAR'
                           - State variables (define by tb, state)
C
 *** subroutine put_ElmData (inquire, elemId, kIntg, nvect, vect)
С
         description
C
            function to put solution dependent variables
С
С
            such as stress, strains at gauss intg. point.
         !! Use this in caution, it overides ansys database. Usually
C
С
         !! you should only write user defined state variables,
         !! SVAR
C
         definition
С
            inquire
                           - query argument (string)
С
С
            elemId
                           - element number
                           - gauss intg. number
С
            kIntg
С
            nvect
                           - number of vector to be inquired
                           - vector to be inquired
С
            vect
С
С
         variables
С
           'SIG '
                           - stress vector
           'EPTO'
                           - Total strain vector (EPEL+EPPL+EPCR+EPTH)
С
           'EPPL'
                           - plastic strain vector
С
                           - creep strain vector
           'EPCR'
С
           'EPTH'
                           - thermal strain vector
С
C
           'ISIG'
                           - Initial stress vector
С
           'PLEQ'
                           - accumulated equivalent plastic strain
С
           'CREQ'
                           - accumulated equivalent creep
                           - State variables (define by tb,state)
           'SVAR'
C
```

Release 12.0 - © 2009 SAS IP, Inc. All rights reserved Contains proprietary and confidential information
of ANSYS Inc. and its subsidiaries and affiliates

binset function, 57 Index bintfo subroutine, 57 bintrd subroutine, 62 bintst program, 62 **Symbols** bintwr subroutine, 63 3-D line elements, 290 binwrt8 subroutine, 59 3-D nodal coordinates, 146 Buffered binary I/O systems initializing, 55 Α **Buffered file Abbreviations** reading data from, 59 file access routines, 55 writing data to, 59 acelcm.inc file, 102 activating UPFs, 107 C **ANSYS** C, 99, 101 customized version of, 104 Fortran shell for, 101 executing command from a user subroutine, 212 .c extension, 104 linking calculating optimization program to, 204 rotation caused by internal pressure, 183 **UPFs to, 103** Calculating parameters, 102 dot product of two vectors, 291 running as a subroutine, 193 dot product of two vectors with increments, 291 standard file header, 57 sum of vector components, 291 start-up **CDWRITE** accessing program logic at, 203 using the command, 71 stopping, 290 CE, 73 tailoring with UPFs, 99 ceget, 232 UPFs for, 99 ceingr, 232 ansyscust file, 105 ceput, 233 ansysdef.inc file, 102 cesel, 233 arc-length debugging, 109 Character string auto time stepping debugging, 109 converting to integer string, 60 /AUX2 command, 55 Characters retrieved from a double precision variable, AUX2 process, 55 285 CMBLOCK, 74 cmopt common block, 205 basic element pass debugging, 110 cmopt.inc file, 102 Beam cross sections cnvget subroutine, 148 reading, 85 COMBIN7 and COMBIN37 parameters, 151 Binary files commands accessing, 53 creating, 194 characteristics of, 54 common blocks, 102 closing or deleting a blocked binary file, 61 cmopt, 205 Converting Two Integers into a Pointer, 61 component entities copying contents, 63 defining, 74 opening a blocked binary file, 57 constitutive material models viewing contents, 55 creating, 156 binclo subroutine, 61 Constraint equation binhed subroutine, 58 defining constant term in, 73 binini subroutine, 55 Constraint equations binigr function, 56 deleting or selecting, 233 binlib library, 53 getting, 232 binrd8 subroutine, 59 getting information about, 232

storing, 233	displacement and coordinate debugging, 110
Constraints	displacement values, 201
deleting at a node, 235	displacement vector debugging, 109
getting from the database, 234–235	disput, 234
getting information, 233	DOF pointers, 102
storing at a node, 234, 258	Double precision vectors, 292
Contact	dptoch subroutine, 285
defining, 76	dspdel, 258
convection loads	dspget, 258
defining, 148	dspiqr, 257
converting nodal data to fixed format data blocks, 111	dspput, 258
Coordinate system	
deleting a coordinate system, 230	E
getting a coordinate system, 229	
getting information about a coordinate system, 228	ecddel, 281
storing a coordinate system, 229	ecdget, 280
Coordinate Systems	ecdiqr, 280
defining, 78	ecdput, 280
Coupled nodes	echprm.inc file, 102
defining, 73	echrtr, 224
Coupled set	ecrdel, 272
getting a coupled set, 231	ecrget, 270
getting a coupled set, 251 getting information about, 230	ecriqr, 270
selecting or deleting, 231	ecrput, 271
	ecvdel, 246
storing a coupled set, 231	ecvget, 245
Coupling and constraint routines, 230	ecviqr, 244
CP,73	ecvput, 245
cpget, 231	EDCADAPT, 75
cpinqr, 230	edcdel, 252
cpput, 231	EDCGEN, 76
cpsel, 231	edcget, 251
creep1 subroutine, 141	edciqr, 251
creep3 subroutine, 142	edcput, 251
csydel, 230	EDCURVE,77
csyget, 229	EDDRELAX, 77
csyiqr, 228	EDLCS, 78
csyput, 229	EDLOAD, 78
CURVE	EDPREAD, 79
defining, 77	EDWELD, 80
custom optimization subroutine, 205	eeldel, 268
	eelget, 266
D	eeligr, 266
debugging	eelput, 267
elements, 110	efsdel, 256
solutions, 109	efsget, 256
decks, 102	efsigr, 256
Degree of freedom labels	efsput, 257
customizing, 71	efudel, 250
Demonstration routines, 61	efuget, 250
/DFLAB, 71	efuiger, 249
disdel, 235	efuput, 250
disget, 233–234	efxdel, 277
uisgel, 233-237	CIAUCI, ZII

efxget, 276	getting, 264
efxiqr, 276	getting information about, 263
efxput, 276	storing, 264
egen function, 178	Element euler angle
egrdel, 266	deleting, 275
egrget, 265	getting, 274
egriqr, 265	storing, 275
egrput, 265	Element euler angles
ehcdel, 283	getting information about, 274
ehcget, 283	Element face flagged surface
ehciqr, 282	deleting, 256
ehcput, 283	getting, 256
ehgdel, 249	getting information about, 256
ehgget, 248	storing, 257
ehgiqr, 248	Element face impedance
ehgput, 249	getting, 254
eimdel, 254	Element face pressure
eimget, 254	getting, 243
eimigr, 253	storing, 244
eimput, 254	Element fluence
elccmt subroutine, 128	deleting, 250
elccmt.inc file, 102	getting, 250
eldwrnL subroutine, 145	getting, 250 getting information about, 249
eldwrtL subroutine, 145	storing, 250
elecom.inc file, 102	Element fluxes
element array data pointers, 102	deleting, 277
Element Attribute Routines, 221	getting information about, 276
	storing, 276
Element calculated heat generation	Element Fluxes
deleting, 283	
getting, 283	getting, 276 Element heat generation
getting information about, 282	<u> </u>
storing, 283	deleting, 249
Element convection	getting, 248
deleting, 246	getting information about, 248
getting, 245	storing, 249
getting information about, 244	Element impedance
storing, 245	deleting, 254
Element current densities	getting information about, 253
deleting, 281	storing, 254
getting, 280	Element loading routines, 243
getting information about, 280	Element local force
storing, 280	deleting, 278
Element current density	getting, 277
deleting, 252	storing, 278
getting, 251	Element local forces
getting information about, 251	getting information about, 277
storing, 251	element matrices, 130
element data	accessing, 152
converting to fixed format blocks, 111	reusing, 135
writing to a file, 145	Element miscellaneous non-summable data
Element energy	deleting, 279
deleting, 264	getting, 279

getting information about, 278 storing, 279	deleting, 256 getting, 255
Element miscellaneous summable data	
	getting information about, 255
deleting, 260	storing, 255
getting, 259	Element temperature
getting information about, 259	deleting, 247
storing, 259	getting, 247
Element nodal creep strain	getting information about, 246
deleting, 272	storing, 247
getting, 270	Element thermal, initial, and swelling strain
getting information about, 270	deleting, 274
storing, 271	Element virtual displacement
Element nodal elastic strain	getting, 252
deleting, 268	getting information about, 252
getting, 266	storing, 253
getting information about, 266	Elements
storing, 267	defining, 81
Element nodal force	example of element attribute routines, 230
getting, 260	getting information about an element, 221
getting information about, 260	getting information about an element type, 223
storing, 261	getting information about element characteristics,
Element nodal forces	224
deleting, 261	getting the number of the next defined element, 218
Element nodal gradients	getting the number of the next element, 217
deleting, 266	getting the number of the previous selected element,
getting, 265	217
getting information about, 265	routines for selecting and retrieving, 216
storing, 265	selecting, unselecting, deleting, or inverting an ele-
Element nodal plastic strain	ment, 218
deleting, 270	selecting, unselecting, deleting, or inverting an ele-
getting, 268	ment type, 225
getting information about, 268	storing an element, 222
storing, 269	storing element type data, 224
Element nodal stress	elements, 99
deleting, 263	accessing information, 132
getting, 262	changing face
getting information about, 261	charge density surface information, 182
storing, 262	convection surface information, 180
Element nodal thermal strain	heat flux surface information, 181
getting information about, 272	changing pressure information, 179
Element nodal thermal stress	debugging, 110
getting, 272	evaluating material properties, 138
storing, 273	finding coordinates, 202
Element nonlinear tables	load vectors, 137
deleting, 282	matrix debugging, 110
_	modifying, 149
getting, 281	
getting information about, 281	monitoring, 149
storing, 282	non-summable miscellaneous data
Element pressure/convection	writing to results file, 145
deleting, 244	overriding characteristic defaults, 130
getting informatin about, 243	parameters, 102
Element surface stress data	particle velocities and accelerations

due to waves and current, 187	engdel, 264
returning reference names, 133	engget, 264
rotation vectors	engiqr, 263
computing, 146	engput, 264
rotational pseudovector	enldel, 282
updating, 147	enlget, 281
storing	enliqr, 281
user-supplied output, 150	enlput, 282
subroutines, 102	ensdel, 263
for element characteristics, 128	ensget, 262
for element creation, 133	ensigr, 261
transformation matrix	ensput, 262
computing, 146	epidel, 270
types, 128	eplget, 268
updating	epligr, 268
3-D nodal coordinates, 146	eplput, 269
creep history, 141	eprdel, 244
plastic history, 140	eprget, 243
swelling history, 142	eprigr, 243
updating rotational pseudovector, 147	eprput, 244
user-defined: creating, 113	erhandler subroutine, 288
writing saved variable set, 134	eringr subroutine, 286
ElemGetMat subroutine, 125	Errors
elfdel, 278	displaying, 288
elfget, 277	obtaining information, 286
elfiqr, 277	esfdel, 256
elfput, 278	esfget, 255
Ellnterp subroutine, 202	esfigr, 255
elLenPsvrBuf function, 143	esfput, 255
elmiqr, 221	ethdel, 274
elmput, 222	ethget, 272
elnext, 217	ethiqr, 272
elnxdf, 218	ethput, 273
elparm.inc file, 102	etpdel, 247
elprev, 217	etpget, 247
ElResultGet subroutine, 202	etpiqr, 246
ElResultStrt subroutine, 202	etpput, 247
elsel, 218	etycom.inc file, 102
elucom.inc file, 102	etyget, 223
emndel, 279	etyiqr, 223
emnget, 279	etyput, 224
emniqr, 278	etysel, 225
emnput, 279	euldel, 275
emsdel, 260	eulget, 274
emsget, 259	euligr, 274
emsigr, 259	eulput, 275
emsput, 259	evaluating material properties for 1-D elements, 139
EN, 81	evdget, 252
enfdel, 261	evdigr, 252
enfget, 260	evdput, 253
enfigr, 260	exinc4 subroutine, 60
enfput, 261	• • • • • • • • • • • • • • • • • • • •
•	

r	1
.F extension, 104	I/O systems, 55
faces	imove function, 292
convection surface information, 180	impcom.inc file, 102
fAnsMemAlloc function, 208	files
fAnsMemFree subroutine, 208	.inc, 102
File access routines	.inc extension, 102
abbreviations, 55	include files, 102
file headers, 111	inexc4 subroutine, 60
File status	Initializing paging space, 57
retrieving, 55	Integer string
Files	converting to character string, 60
ansyscust, 105	Interpolation, 289
binary	intrp subroutine, 289
accessing, 53	izero function, 292
characteristics of, 54	
files, 104–105	J
START.ANS, 194	
/FILNAM, 71	Jobname.cdb
fordel, 236	defining, 79
forget, 235	_
foriqr, 235	L
forput, 236	large data files, 111
Fortran, 99–101	largeIntGet subroutine, 61
Full transposed matrix, 298	lastv function, 292
functions	Libraries, 53
miscellaneous, 212	linking UPFs to ANSYS, 99, 103
Functions	load data
	getting from analysis results, 202
vector, 291	reading for a substructure generation run, 136
	load steps
G	accessing program logic, 203
Geometry	containing information about, 102
relaxing, 77	load vectors, 130
getnod, 219	accessing, 152
GetRForce function, 200	Loads
GetStackDisp function, 201	defining, 78
/GFILE, 87	loads, 99
Graphics files	computing complex load vector, 182
neutral graphics file format, 88	convection, 148
pixmap fromat for graphic display files , 87	customizing, 178
z-buffered graphics, 87	debugging, 110
	heat generation, 148
H	pressure, 148
heat generation	temperature, 147
defining loads, 148	LOCAL, 81
hgnget subroutine, 148	Local coordinate system
/HOLD command, 213	defining, 81
hyperelastic material models	denining, or
* *	8.6
creating, 164	M
	M, 82
	maat subroutine, 299

,89
,00

nfuiqr, 239	storing, 242
nfuput, 240	Node information routines, 218
nhgdel, 239	Nodes
nhgget, 238	defining, 82
nhgiqr, 238	getting a nodal point, 219
nhgput, 238	getting information about a node, 218
nlget function, 144	getting the next node number, 216
nminfo subroutine, 133	getting the nodal solution for a node of an element,
nmisc record	220
storing data in, 144	getting the number of the next defined node, 216
nodal coordinate system transformation debugging,	getting the number of the previous selected node,
111	216
Nodal current density	getting the xyz/rotation coordinates vector for a
deleting, 241	node, 220
getting, 241	getting XYZ vector for, 297
getting information about, 240	routines for selecting and retrieving, 216
storing, 241	selecting, unselecting, deleting, or inverting a node,
nodal data conversion to fixed format data blocks, 111	217
Nodal fluence	storing a node, 220
deleting, 240	storing XYZ vector for, 297
Nodal fluences	nonlinear debugging, 110
getting, 239	ntpdel, 237
getting information about, 239	ntpget, 237
storing, 240	ntpiqr, 236
Nodal heat generation	ntpput, 237
deleting, 239	nvddel, 242
getting, 238	nvdget, 242
getting information, 238	nvdigr, 241
storing, 238	nvdput, 242
Nodal load	op 114 = 11
deleting at a node, 236	0
storing at a node, 236	
Nodal loading routines, 233	.o extension, 104
Nodal loads	.obj extension, 105
getting information about, 235	object files, 104–105
<u> </u>	optimization program
Nodal magnetic virtual displacement	creating, 204
getting, 242	linking to ANSYS, 204
getting information about, 241	optimization variables, 102
nodal orientation matrix	optimizing ANSYS file reads, 111
adjusting, 132	outpcm.inc file, 102
nodal reaction force values	output
getting, 200	generating, 111
Nodal results	Output
getting from the database, 258	obtaining information about, 286
getting information about, 257	output control, 102
Nodal temperature	
deleting, 237	Р
getting, 237	
getting information about, 236	parallel processing and UPFs, 100
storing, 237	Parameters
Nodal virtual displacement	retrieving system-dependent parameters, 56
deleting, 242	parameters
<i>5</i> ,	adding, 210

ANSYS, 102	ResRdElem function, 68
creating a dimensioned parameter, 210	ResRdEstr function, 70
defining in subroutines, 128	ResRdFix function, 69
element characteristics, 102	ResRdForc function, 69
finding and evaluating, 210	ResRdGeomBegin function, 67
performing user operations on COMBIN7 and COM-	ResRdNode function, 68
BIN37, 151	ResRdReal function, 67
subroutines to process, 210	ResRdRfor function, 69
pardef subroutine, 210	ResRdSolBeginl function, 68
pardim subroutine, 210	ResRdType function, 67
plast1 subroutine, 140	restrictions on UPF creation, 99
plast3 subroutine, 140	Results
plasticity laws	deleting at a node, 258
	Results file
creating, 171	
pointers to element data array, 102	opening the file and retrieving global information,
POST1 postprocessor, 128	
pressure loads	retrieving applied nodal constraints, 69
defining, 148	retrieving applied nodal loads solution, 69
printer output, 131	retrieving coordinate systems, 67
Printing file contents, 62	retrieving element solutions, 70
program logic	retrieving element types, 67
accessing, 203	retrieving elements, 68
tracking, 109	retrieving geometry information, 67
programmer debugging, 109	retrieving nodal coordinates, 68
programming UPFs, 101	retrieving nodal solution, 69
prope1 subroutine, 139	retrieving reaction solution, 69
propev subroutine, 138	retrieving real constants, 67
prsget subroutine, 148	retrieving result set location, 68
pstev1 subroutine, 139	Results files
putnod, 220	retrieving information from, 65
	Results information routines, 257
R	results values
R, 84	getting at selected points, 202
rdfull, 64	ResWrDemo, 65
rdsubs subroutine, 64	Retrieving
Reading a results file, 65	maximum vector value for a location, 292
Reading an ANSYS substructure file, 64	position of last nonzero term in a vector, 292
Reading and reformatting the .FULL file, 64	reusing element matrices, 135
reading large data files rapidly, 111	RLBLOCK, 84
Real constant set	rlget, 228
defining, 84	rlingr, 227
real constants	rlsel, 228
debugging, 110	rottr subroutine, 146
fetching, 137	RunCommand function, 212
Real constants	rvrget subroutine, 137
getting information about a real constant set, 227	
getting real constant data, 228	S
selecting or deleting a real constant set, 228	saved variables
ResRdBegin function, 66	debugging, 110
ResRdCsys function, 67	fetching, 134
ResRdDemo, 65	Scalar constants
ResRdDisp function, 69	assigning to vectors, 293

scalar fields to user-defined values, 178	defining, 147
SECBLOCK, 85	Temperature table
SFBEAM, 85	defining, 82
SFE, 86	tmpget subroutine, 147
Simultaneous linear equations, 301	TrackBegin subroutine, 109, 288
solution control debugging, 109	TrackEnd subroutine, 109, 288
solutions	tran subroutine, 301
accessing program logic, 203	transient debugging, 109
debugging, 109	Transposing a matrix, 299
soptcm.inc file, 102	tranx3 subroutine, 290
source files, 104	trrot subroutine, 146
stack storage definition, 102	·
stack.inc file, 102	U
Standard ANSYS file header, 57–58	_
START.ANS file, 194	UAnBeg subroutine, 203
stepcm.inc file, 102	UAnFin subroutine, 203
Stopping ANSYS, 290	uec100 subroutine, 128–129
storing	uel100 subroutine, 128, 130
user-supplied element output, 150	uel101 subroutine, 102
subrd subroutine, 136	UElMatx subroutine, 152
Subroutines	uep100 subroutine, 128, 131
accessing results files, 65	uex100 subroutine, 128, 130
matrix, 297	UltBeg subroutine, 203
tracking calls to, 288	UltFin subroutine, 203
subroutines	ULdBeg subroutine, 203
	ULdFin subroutine, 203
for elements, 102	/UNDO command, 212
for memory management, 208	UNIX systems
for processing parameters, 210	compiling and linking UPFs on, 104
running ANSYS as, 193	Fortran programming on, 101
supporting for user-created elements, 133	UPFs (see user programmable functions (UPFs))
supporting user-defined commands, 200	user programmable features (UPFs), 99
subwrt subroutine, 137	activating, 107
Surface load	and loads, 99
defining, 86	and material properties, 99
defining as beams, 85	before you begin using, 100
svgidx subroutine, 133	capabilities of, 99
svpidx subroutine, 135	compiling and linking
svrget subroutine, 134	on UNIX systems, 104
svrput subroutine, 134	on Windows systems, 105
swell1 subroutine, 142	considerations for using, 100
swell3 subroutine, 143	debugging, 108
swelling laws, 173	definition of, 99
symeqn subroutine, 301	development strategy, 102
sysiqr function, 55	include files, 102
system dependencies, 101	linking subroutines to user routines, 103
systop subroutine, 290	planning, 101
	running your executable, 107
Т	understanding, 101
table data	verifying your routines, 108
retrieving, 139	user subroutines
tbuser subroutine, 139	for creating new elements, 113, 128
temperature loads	for creating new materials, 156

for defining material linear elastic properties, 168	V
User subroutines, 128	vamb subroutine, 295
user-defined commands, 194	vamb1 subroutine, 295
making available to all ANSYS sessions, 194	vapb subroutine, 294
user-defined elements, 113	vapb1 subroutine, 294
user-defined materials, 156	vapcb1 subroutine, 294
User-defined variable storage, 102	variables
user-programmable features	defining additional, 111
defining material linear elastic properties, 168	fetching the index for saved, 133
user-defined elements, 113	optimization, 102
user-defined materials, 156	writing saved element index, 135
user01 command, 194	Variables
user02 command, 195	retrieving characters from, 285
user03 command, 196	vcross subroutine, 296
USER300 user-defined element, 115	vdot function, 291
user_tbelastic subroutine, 168	Vector functions, 291
userac, 230	Vectors
userac subroutine, 128, 132	assigning equal values to two vectors, 292
useran subroutine, 151	assigning scalar constants to, 293
userch subroutine, 182	assigning scalar constants to incrementally, 294
userck subroutine, 174	combining two in one, 294
UserCR subroutine, 165	defining by a cross product, 296
UserCreep subroutine, 165	getting the difference of two, 295
usercv subroutine, 180	getting XYZ vector for a node, 297
UserElem subroutine, 116	initializing to zero, 292
usereo subroutine, 144	moving one into another, 293
userfd subroutine, 182	moving one into another incrementally, 293
userfx subroutine, 181	•
UserHyper subroutine, 164	multiplying a matrix by, 298
UserMat subroutine, 156	multiplying by a full transposed matrix, 298 multiplying one by a constant, 295
userop subroutine, 99, 205	, , , , , , , , , , , , , , , , , , , ,
userou subroutine, 150	multiplying to a constant, 294 normalizing a three-component vector, 296
userPartVelAcc subroutine, 187	,
userpe function, 183	normalizing to unit length, 296
UserPL subroutine, 171	setting integer vector to zero, 292
userpr subroutine, 179	setting one to be the sum of two vectors, 294
userrc subroutine, 151	storing XYZ vector for a node, 297
usersw subroutine, 173	subtracting one from another, 295
usertr subroutine, 128, 132	velocity information, 102
UserVisLaw subroutine, 174	vidot function, 291
USolBeg subroutine, 203	viinit subroutine, 294
USolFin subroutine, 203	vimove subroutine, 293
USRCAL command, 178, 203	vinit subroutine, 293
usrefl subroutine, 178	viscoplastic/creep material behavior, 165
UsrFictive subroutine, 153	viscosity laws
usrsurf116 subroutine, 184	defining, 174
UsrViscEl subroutine, 184	vmax function, 292
USsBeg subroutine, 203	vmove subroutine, 293
USsFin subroutine, 203	vmult subroutine, 295
usvrcm.inc file, 102	vmult1 subroutine, 295
UTHICK subroutine, 152	vnorm subroutine, 296
oen Subroudine, 192	vnorme subroutine, 296
	vsum function, 291

vzero subroutine, 292

W

wavefront reordering debugging, 111
Weld
defining, 80
Windows systems
compiling and linking UPFs on, 105
wringr function, 286
Writing an ANSYS substructure file, 63
wrtsub subroutine, 63

X

xyzup3 subroutine, 146

Z

Z-buffered graphics, 87