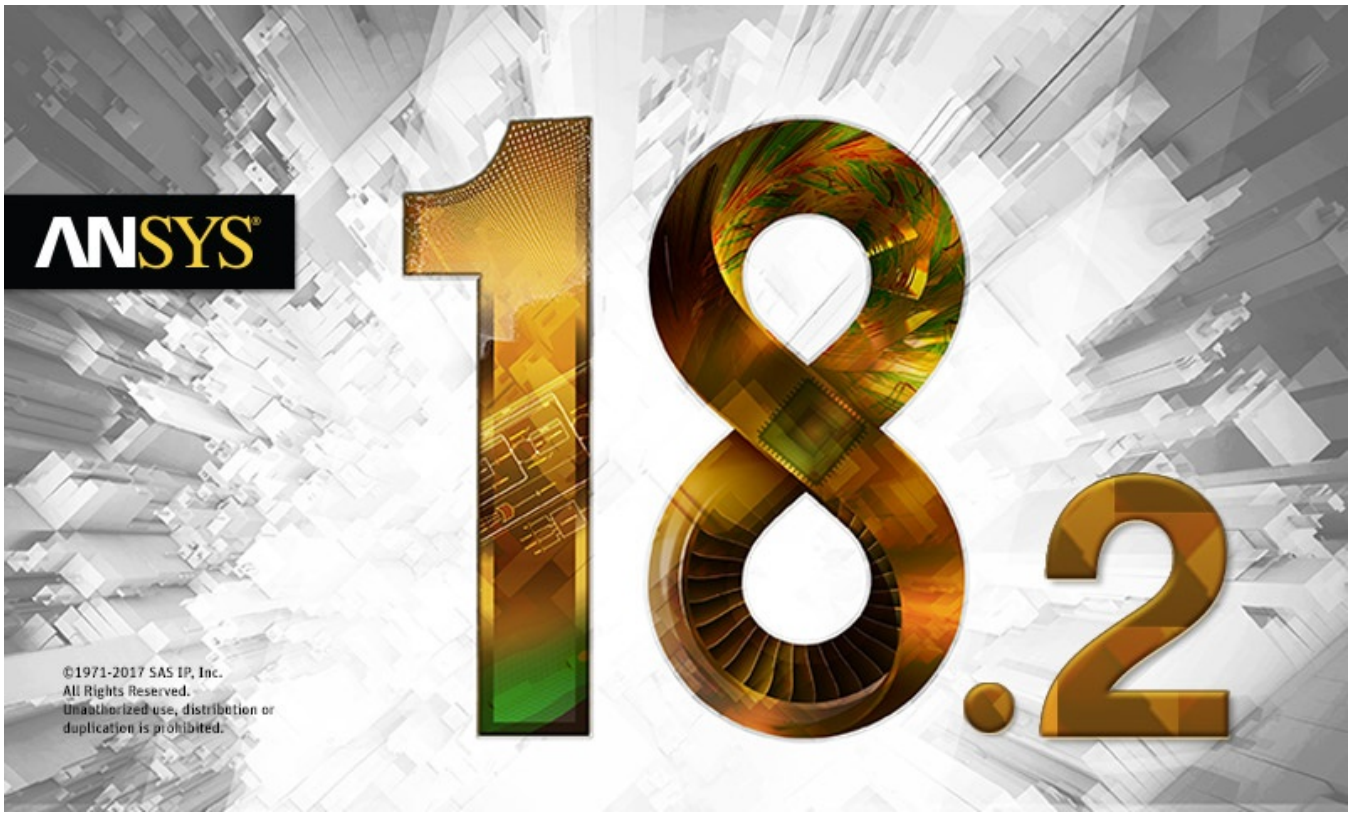


ANSYS Mechanical APDL Programmer's Reference

ANSYS, Inc.
Southpointe
2600 ANSYS Drive
Canonsburg, PA 15317
ansysinfo@ansys.com
http://www.ansys.com
(T) 724-746-3304
(F) 724-514-9494

Release 18.2
August 2017
002328

ANSYS, Inc. and ANSYS Europe, Ltd. are UL registered ISO 9001: 2008 companies.

Copyright and Trademark Information

© 2017 ANSYS, Inc. Unauthorized use, distribution or duplication is prohibited.

ANSYS, ANSYS Workbench, AUTODYN, CFX, FLUENT 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 located 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. FLEXlm and FLEXnet are trademarks of Flexera Software LLC.

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. and ANSYS Europe, Ltd. are UL registered ISO 9001: 2008 companies.

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, contact ANSYS, Inc.

Published in the U.S.A.

Table of Contents

Preface xix
I. 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 4
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 6
1.2.3. Results File Format 6
1.3. Description of the Reduced Displacement File 27
1.3.1. Standard ANSYS File Header 27
1.3.2. RDSP File Format 27
1.4. Description of the Reduced Complex Displacement File 31
1.4.1. Standard ANSYS File Header 31
1.4.2. RFRQ File Format 31
1.5. Description of the Modal Results File 35
1.5.1. Standard ANSYS File Header 35
1.5.2. MODE File Format 35
1.6. Description of the Element Matrices File 40
1.6.1. Standard ANSYS File Header 40
1.6.2. EMAT File Format 40
1.7. Description of the Substructure Matrices File 45
1.7.1. Standard ANSYS File Header 45
1.7.2. SUB File Format 45
1.8. Description of the Component Mode Synthesis Matrices (CMS) File 50
1.8.1. Standard ANSYS File Header 50
1.8.2. CMS File Format 50
1.8.3. TCMS File Format 52
1.9. Description of the Full Stiffness-Mass File 53
1.9.1. Standard ANSYS File Header 53
1.9.2. FULL File Format 53
1.10. Description of the Substructure Displacement File 60
1.10.1. Standard ANSYS File Header 60
1.10.2. DSUB File Format 60
2. Accessing Binary Data Files 63
2.1. Accessing ANSYS Binary Files 63
2.1.1. Access Routines to Results, Substructure, and Matrix Files 63
2.1.2. Characteristics of ANSYS Binary Files 64
2.1.3. Viewing Binary File Contents 65
2.1.4. Abbreviations 65
2.1.5. binini (Initializing Buffered Binary I/O Systems) 65
2.1.6. Function sysiqr (Retrieving the Status of a File) 66
2.1.7. Function biniqr8 (Retrieving System-Dependent Parameters) 66
2.1.8. Function binset (Opening a Blocked Binary File or Initializing Paging Space) 67
2.1.9. Subroutine bintfo (Defining Data for a Standard ANSYS File Header) 68
2.1.10. Subroutine binhed (Writing the Standard ANSYS File Header) 68
2.1.11. Subroutine binrd8 (Reading Data from a Buffered File) 69
2.1.12. Subroutine binwrt8 (Writing Data to a Buffered File) 70
2.1.13. Subroutine exinc4 (Decoding an Integer String into a Character String) 70

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

2.1.14. Subroutine inexc4 (Coding a Character String into an Integer String) 71
2.1.15. Subroutine binclo (Closing or Deleting a Blocked Binary File) 71
2.1.16. Subroutine largeIntGet (Converting Two Integers into a Pointer) 71
2.2. Demonstration Routines 72
2.2.1. Program bintst (Demonstrates Dumping a Binary File and Copying It for Comparison Purposes) 72
2.2.1.1. Common Variables: 72
2.2.2. Subroutine bintrd (Demonstrates Printing a Dump of File Contents) 72
2.2.3. Subroutine bintwr (Demonstrates Copying Binary File Contents) 73
2.2.4. Program wrtsub (Demonstrates Writing an ANSYS Substructure File) 73
2.2.5. Program rdsubs (Demonstrates Reading a Substructure File) 74
2.2.6. Program rdfull (Demonstrates Reading and Reformatting the .FULL File) 74
2.2.7. Program ResRdDemo (Demonstrates Reading a Results File) 75
2.2.8. Program ResWrDemo (Demonstrates Writing a Results File) 75
2.3. Results File Access Routines 75
2.3.1. Overview of the Routines 75
2.3.2. ResRdBegin (Opening the File and Retrieving Global Information) 76
2.3.3. ResRdGeomBegin (Retrieving Global Geometry Information) 77
2.3.4. ResRdType (Retrieving Element Types) 77
2.3.5. ResRdReal (Retrieving Real Constants) 77
2.3.6. ResRdCsys (Retrieving Coordinate Systems) 78
2.3.7. ResRdNode (Retrieving Nodal Coordinates) 78
2.3.8. ResRdElem (Retrieving Elements) 78
2.3.9. ResRdSectMatBegin (Retrieving Global Section and Material Information) 79
2.3.10. ResRdSect (Retrieving Section Data) 79
2.3.11. ResRdMat (Retrieving Material Data) 79
2.3.12. ResRdSolBegin (Retrieving Result Set Location) 80
2.3.13. ResRdDisp (Retrieving Nodal Solution) 80
2.3.14. ResRdRfor (Retrieving Reaction Solution) 80
2.3.15. ResRdFix (Retrieving Applied Nodal Constraints) 81
2.3.16. ResRdForc (Retrieving Applied Nodal Loads Solution) 81
2.3.17. ResRdEstr (Retrieving Element Solutions) 81
3. The CDWRITE (CDB) File Format 83
3.1. Using the CDWRITE Command 83
3.1.1. Customizing Degree of Freedom Labels: the /DFLAB Command 83
3.2. Coded Database File Commands 84
3.2.1. CE Command 85
3.2.2. CP Command 85
3.2.3. CMBLOCK Command 86
3.2.4. CYCLIC Command 86
3.2.5. EBLOCK Command 92
3.2.6. EDCADAPT Command 93
3.2.7. EDCGEN Command 94
3.2.8. EDCURVE Command 95
3.2.9. EDDRELAX Command 95
3.2.10. EDLCS Command 96
3.2.11. EDLOAD Command 96
3.2.12. EDPREAD Command 97
3.2.13. EDWELD Command 98
3.2.14. EN Command 99
3.2.15. LOCAL Command 100
3.2.16. M Command 100

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

3.2.17. MPDATA Command 100
3.2.18. MPTEMP Command 101
3.2.19. N Command 101
3.2.20. NBLOCK Command 102
3.2.21. *PREAD Command 103
3.2.22. R Command 103
3.2.23. RLBLOCK Command 103
3.2.24. SE Command 104
3.2.25. SECBLOCK Command 104
3.2.26. SFBEAM Command 105
3.2.27. SFE Command 106
4. ANSYS Graphics File Format 107
4.1. Pixmap Format for Graphic Display Files 107
4.2. Neutral Graphics File Format 108
4.2.1. Characters the Graphics File Uses 108
4.2.2. Graphics File Directives 108
4.2.2.1. Parameter Types for Graphics File Directives 109
4.2.2.2. Directive Descriptions 109
4.2.2.3. Color Specification 111
4.3. Decoding a Graphics File: an Example 112
4.3.1. The Example Command Stream 113
4.3.2. Example Graphics File Contents 113
II. Guide to User-Programmable Features 117
1. Understanding User Programmable Features (UPFs) 119
1.1. What Are User Programmable Features? 119
1.2. What You Should Know Before Using UPFs 120
1.3. Planning Your UPFs 121
1.4. Studying the Mechanical APDL User Routines 122
1.5. Programming in Languages Other than Fortran 122
1.6. Developing UPFs: a Suggested Strategy 122
1.7. Include Decks 123
1.8. Choosing a Linking Method 124
1.9. Compiling and Linking UPFs on Linux Systems 125
1.9.1. Using the /UPF Command 125
1.9.2. Creating a Shared Library 126
1.9.3. Using the ANS_ADMIN Utility 126
1.10. Compiling and Linking UPFs on Windows Systems 127
1.10.1. Using the /UPF Command 128
1.10.1.1. Using the /UPF Command on a Windows HPC Server System 129
1.10.2. Creating a Dynamic-link (DLL) Library 130
1.10.3. Using the ANS_ADMIN Utility 131
1.11. Activating UPFs 133
1.12. Running Your Custom Executable 134
1.13. Verifying Your Routines 135
1.14. Debugging Commands 135
1.14.1. Tracking the Path of Program Logic 135
1.14.2. Debugging Elements and Solutions 136
1.14.2.1. Solution Debug Format 136
1.14.2.2. Element Debug Format 137
1.14.2.3. General Debug Format 138
1.15. Other Useful Commands 138
1.16. Generating Output 138

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

1.17. Reading Large Data Files More Rapidly 138
2. UPF Subroutines and Functions 141
2.1. Creating a New Element 141
2.1.1. Input and Output Abbreviations 143
2.1.2. Creating a New Element via the User-Defined Element API 143
2.1.2.1. Subroutine UserElem (Writing Your Own Elements) 144
2.1.2.2. Subroutine ElemGetMat (Calling the Standard Structural Material Library) 155
2.1.3. Creating a New Element by Directly Accessing the Program Database 159
2.1.3.1. User Subroutines 159
2.1.3.2. Subroutine uec100 (Defining Characteristics of the usr100 Subroutine) 160
2.1.3.2.1. Subroutines uec101 through uec105 160
2.1.3.3. Subroutine uex100 (Overriding Element Characteristic Defaults) 160
2.1.3.3.1. Subroutines uex101 through uex105 161
2.1.3.4. Subroutine uel100 (Calculating Element Matrices, Load Vectors, and Results) 161
2.1.3.4.1. Subroutines uel101 through uel105 162
2.1.3.5. Subroutine uep100 (Printing Output for User Elements in POST1 via **PRESOL**,ELEM) 162
2.1.3.5.1. Subroutines uep101 through uep105 162
2.1.3.6. Subroutine usertr (Adjusting the Nodal Orientation Matrix) 163
2.1.3.7. Subroutine userac (Accessing Element Information) 163
2.2. Supporting Subroutines for Element Creation 163
2.2.1. Subroutine nminfo (Returning Element Reference Names) 164
2.2.2. Subroutine svgidx (Fetching the Index for Saved Variables) 164
2.2.3. Subroutine svrget (Fetching Saved Variable Data for an Element) 164
2.2.4. Subroutine svrput (Writing an Element's Saved Variable Set) 165
2.2.5. Subroutine svpidx (Writing the Saved Variable Element Index to a File) 165
2.2.6. Subroutine mreuse (Determining Which Element Matrices Can Be Reused) 166
2.2.7. Subroutine subrd (Reading Element Load Data for a Substructure Generation Run) 167
2.2.8. Subroutine subwrt (Writing an Element Load Vector to a File for a Substructure Generation Run) 167
2.2.9. Subroutine rvrget (Fetching Real Constants for an Element) 168
2.2.10. Subroutine propev (Evaluating a Group of Material Properties) 168
2.2.11. Subroutine prope1 (Evaluating One Material Property) 169
2.2.12. Subroutine pstev1 (Evaluating Material Properties for 1-D Elements) 170
2.2.13. Subroutine tbuser (Retrieving User Table Data) 170
2.2.14. Subroutine plast1 (Updating an Element's Plastic History) 170
2.2.15. Subroutine plast3 (Updating an Element's Plastic History, 4 or 6 components) 171
2.2.16. Subroutine creep1 (Updating an Element's Creep History) 172
2.2.17. Subroutine creep3 (Updating an Element's Creep History, 3-D Elements) 172
2.2.18. Subroutine swell1 (Updating an Element's Swelling History) 173
2.2.19. Subroutine swell3 (Updating an Element's Swelling History, 3-D Elements) 173
2.2.20. Function elLenPsvrBuf (Determining Additional ESAV Record for Plasticity) 174
2.2.21. Function nlget (Retrieving Material Nonlinear Property Information) 174
2.2.22. Subroutine usereo (Storing Data in the nmisc Record) 175
2.2.23. Subroutine eldwrtL (Writing Element Data to a File) 176
2.2.24. Subroutine eldwrnL (Writing Element Nonsummable Miscellaneous Data to the Results File) 176
2.2.25. Subroutine trrot (Calculating the Rotation Vector) 176
2.2.26. Subroutine rottr (Calculating the Transformation Matrix) 177
2.2.27. Subroutine xyzup3 (Updating an Element's 3-D Nodal Coordinates) 177
2.2.28. Subroutine updrot (Updating the Rotation Pseudovector) 177
2.2.29. Subroutine tmpget (Defining Current Temperature Loads) 178

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

2.2.30. Subroutine prsget (Defining Current Pressure Loads) 178
2.2.31. Subroutine cnvget (Defining Current Convection Loads) 179
2.2.32. Subroutine hgnget (Defining Current Heat Generation Loads) 179
2.2.33. Subroutine prinst (Calculating Principal Stress and Stress Intensity) 180
2.3. Subroutines for Modifying and Monitoring Existing Elements 180
2.3.1. Subroutine userou (Storing User-Provided Element Output) 180
2.3.2. Subroutine useran (Modifying Orientation of Material Properties) 181
2.3.3. Subroutine userrc (Performing User Operations on COMBIN37 Parameters) 182
2.3.4. Subroutine UElMatx (Accessing Element Matrices and Load Vectors) 182
2.3.5. Subroutine uthick (Getting User-Defined Initial Thickness) 183
2.3.6. Subroutine UsrFictive (Providing User-Defined Fictive Temperature Relationship) 184
2.3.7. Subroutine uflex (Calculating Flexibility Factors for PIPE288 and PIPE289) 184
2.3.8. Subroutine UsrShift (Calculating Pseudotime Time Increment) 185
2.3.9. Subroutine UTimeInc (Overriding the Program-Determined Time Step) 185
2.3.10. Subroutine UCnvrg (Overriding the Program-Determined Convergence) 185
2.4. Subroutines for Customizing Material Behavior 186
2.4.1. Subroutine UserMat (Creating Your Own Material Model) 187
2.4.1.1. UserMat Overview 187
2.4.1.2. Stress, Strain, and Material Jacobian Matrix 188
2.4.1.3. The UserMat API 189
2.4.1.4. UserMat Variables 191
2.4.1.5. Table (**TB**) Commands for UserMat 194
2.4.1.6. Material Constitutive Integration with UserMat 195
2.4.1.7. UserMat Restrictions 196
2.4.1.8. Accessing Material and Element Data for UserMat 196
2.4.1.9. Utility Functions for UserMat 197
2.4.2. Subroutine UserMatTh (Creating Your Own Thermal Material Model) 197
2.4.2.1. UserMatTh Element Support 197
2.4.2.2. The UserMatTh API 198
2.4.2.3. UserMatTh Variables 199
2.4.2.4. Table (**TB**) Commands for UserMatTh 200
2.4.2.5. UserMatTh Restrictions 201
2.4.2.6. Utility Functions for UserMatTh 202
2.4.3. Subroutine UserHyper (Writing Your Own Hyperelasticity Laws) 202
2.4.4. Subroutine UserCreep (Defining Creep Material Behavior) 203
2.4.5. Subroutine user_tbelastic (Defining Material Linear Elastic Properties) 205
2.4.5.1. Overview of the user_tbelastic Subroutine 205
2.4.5.2. Data Types Supported by user_tbelastic 205
2.4.5.3. Table (TB) Command for user_tbelastic 205
2.4.5.4. User Interface for user_tbelastic 206
2.4.5.5. The user_tbelastic API 206
2.4.5.6. Usage Example for user_tbelastic 207
2.4.6. Subroutine userfc (Defining Your Own Failure Criteria) 208
2.4.7. Subroutine userCZM (Defining Your Own Cohesive Zone Material) 209
2.4.8. Subroutine userswstrain (Defining Your Own Swelling Laws) 210
2.4.9. Subroutine userck (Checking User-Defined Material Data) 211
2.4.10. Supporting Function egen 211
2.4.11. Subroutine userfld (Update User-Defined Field Variables) 211
2.4.12. Subroutine userthstrain (Defining Your Own Thermal Strain) 213
2.5. Subroutines for Customizing Contact Interfacial Behavior 213
2.5.1. Subroutine usercnprop (Programming Your Own Contact Properties) 213
2.5.2. Subroutine userfric (Writing Your Own Friction Laws) 217

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

2.5.3. Subroutine userinter (Writing Your Own Contact Interactions) 220
2.5.4. Subroutine userwear (Writing Your Own Wear Law) 226
2.6. Subroutines for Customizing Loads 227
2.6.1. Subroutine usrefl (Changing Scalar Fields to User-Defined Values) 227
2.6.2. Subroutine userpr (Changing Element Pressure Information) 228
2.6.3. Subroutine usercv (Changing Element Face Convection Surface Information) 229
2.6.4. Subroutine userfx (Changing Element Face Heat Flux Surface Information) 230
2.6.5. Subroutine userch (Changing Element Face Charge Density Surface Information) 230
2.6.6. Subroutine userfd (Calculating the Complex Load Vector for Frequency Domain Logic) 231
2.6.7. Function userpe (Calculating Rotation Caused by Internal Pressure) 232
2.6.8. Subroutine usrsurf116 (Modifying SURF151 and SURF152 Film Coefficients and Bulk Temperatures) 232
2.6.9. Subroutine User116Cond (Calculating the Conductance Coefficient for FLUID116) 234
2.6.10. Subroutine User116Hf (Calculating the Film Coefficient for FLUID116) 234
2.6.11. Subroutine userPartVelAcc (Calculating Particle Velocities and Accelerations of Ocean Waves) 234
2.6.11.1. Subroutine userPartVelAccSetup (Initializing Data for Use by the userPartVelAcc Subroutine) 236
2.6.11.2. Subroutine userWavHt 236
2.6.11.3. Subroutine wvhybl 237
2.6.11.4. Subroutine wvargu 237
2.6.12. Subroutine userPanelHydFor (Calculating Panel Loads Caused by Ocean Loading) 238
2.6.12.1. Subroutine userOceanRead 238
2.7. Running Mechanical APDL as a Subroutine 239
2.8. Defining Your Own Commands 239
2.8.1. Function user01 240
2.8.2. Function user02 (Demonstrates Offsetting Selected Nodes) 241
2.8.3. Function user03 (Demonstrates Using Memory) 242
2.8.4. Function user04 244
2.8.5. Functions user05 through user10 246
2.9. Supporting Subroutines 246
2.9.1. Function GetRForce (Getting Nodal Reaction Force values) 246
2.9.2. Function GetStackDisp (Getting Current Displacement Values) 247
2.9.3. Subroutine ElResultStrt (Getting Load Data from Analysis Results) 248
2.9.4. Subroutine ElResultGet (Getting Results Values at Selected Points) 248
2.9.5. Subroutine ElInterp (Finding Element Coordinates) 248
2.10. Access at the Beginning and End of Various Operations 249
2.11. Memory Management Subroutines 250
2.11.1. Using the Memory-Management Subroutines 250
2.11.2. Function fAnsMemAlloc (Allocating Space and Returning a Pointer) 252
2.11.3. Subroutine fAnsMemFree (Deallocating Space) 252
2.12. Parameter-Processing Subroutines 252
2.12.1. Subroutine pardim (Creating a Dimensioned Parameter) 252
2.12.2. Function parevl (Finding and Evaluating a Parameter) 253
2.12.3. Subroutine pardef (Adding a Parameter) 253
2.13. Other Useful Functions 254
2.13.1. Using Function RunCommand 255
2.13.2. Using the **/UNDO** Command 255
2.13.3. Using the **/HOLD** command 255
3. Accessing the Mechanical APDL Database 257
3.1. Routines for Selecting and Retrieving Nodes and Elements 258
3.1.1. ndnext Function (Getting the Next Node Number) 258

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

3.1.2. ndprev Function (Getting the Number of the Previous Selected Node) 258
3.1.3. ndnxdf Function (Getting the Number of the Next Defined Node) 258
3.1.4. ndsel Function (Selecting, Unselecting, Deleting, or Inverting a Node) 259
3.1.5. elnext Function (Getting the Number of the Next Element) 259
3.1.6. elprev Function (Getting the Number of the Previous Selected Element) 259
3.1.7. elnxdf Function (Getting the Number of the Next Defined Element) 260
3.1.8. elsel Subroutine (Selecting, Unselecting, Deleting, or Inverting an Element) 260
3.2. Node Information Routines 260
3.2.1. ndinqr Function (Getting Information About a Node) 260
3.2.2. getnod Function (Getting a Nodal Point) 261
3.2.3. putnod Function (Storing a Node) 262
3.2.4. ndgall Function (Getting the XYZ/Rotation Coordinates Vector for a Node) 262
3.2.5. ndspgt Subroutine (Getting the Nodal Solution for a Node of an Element) 262
3.3. Element Attribute Routines 263
3.3.1. elmiqr Function (Getting Information About an Element) 263
3.3.2. elmget Function (Getting an Element's Attributes and Nodes) 264
3.3.3. elmput Subroutine (Storing an Element) 265
3.3.4. etyiqr Function (Getting a Data Item About an Element Type) 265
3.3.5. etyget Function (Getting Information About an Element Type) 266
3.3.6. etyput Subroutine (Storing Element Type Data) 266
3.3.7. echrtr Subroutine (Getting Information About Element Characteristics) 266
3.3.8. etysel Subroutine (Selecting, Unselecting, Deleting, or Inverting an Element Type) 267
3.3.9. mpinqr Function (Getting Information About a Material Property) 267
3.3.10. mpget Function (Getting a Material Property Table) 268
3.3.11. mpput Subroutine (Storing a Material Property Table) 268
3.3.12. mpdel Subroutine (Deleting a Material Property Table) 269
3.3.13. rlinqr Function (Getting Information About a Real Constant Set) 269
3.3.14. rlget Function (Getting Real Constant Data) 270
3.3.15. rlsel Subroutine (Selecting or Deleting a Real Constant Set) 270
3.3.16. csyiqr Function (Getting Information About a Coordinate System) 271
3.3.17. csyget Function (Getting a Coordinate System) 271
3.3.18. csyput Subroutine (Storing a Coordinate System) 272
3.3.19. csydel Subroutine (Deleting a Coordinate System) 272
3.3.20. userac Subroutine (Demonstrates Use of Element Attribute Routines) 272
3.4. Coupling and Constraint Routines 272
3.4.1. cpinqr Function (Getting Information About a Coupled Set) 272
3.4.2. cpget Function (Getting a Coupled Set) 273
3.4.3. cpput Subroutine (Storing a Coupled Set) 273
3.4.4. cpsel Subroutine (Selecting or Deleting a Coupled Set) 274
3.4.5. ceinqr Function (Getting Information About a Constraint Equation Set) 274
3.4.6. ceget Function (Getting a Constraint Equation) 275
3.4.7. ceput Subroutine (Storing a Constraint Equation) 275
3.4.8. cesel Subroutine (Deleting or Selecting a Constraint Equation) 275
3.5. Nodal Loading Routines 275
3.5.1. disiqr Function (Getting Information About Constraints) 275
3.5.2. disget Function (Getting a Constraint from the Database) 276
3.5.3. disput Subroutine (Storing a Constraint at a Node) 276
3.5.4. disdel Subroutine (Deleting a Constraint at a Node) 276
3.5.5. foriqr Function (Getting Information About Nodal Loads) 277
3.5.6. forget Function (Getting a Constraint from the Database) 277
3.5.7. forput Subroutine (Storing a Nodal Load at a Node) 278
3.5.8. fordel Subroutine (Deleting a Nodal Load at a Node) 278

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

3.5.9. ntpiqr Function (Getting Information About a Nodal Temperature) 278
3.5.10. ntpget Function (Getting a Specified Nodal Temperature) 279
3.5.11. ntpput Subroutine (Storing a Nodal Temperature) 279
3.5.12. ntpdel Subroutine (Deleting a Nodal Temperature) 279
3.5.13. nhgiqr Function (Getting Information About Nodal Heat Generations) 279
3.5.14. nhgget Function (Getting a Nodal Heat Generation) 280
3.5.15. nhgput Subroutine (Storing Nodal Heat Generation) 280
3.5.16. nhgdel Subroutine (Deleting a Nodal Heat Generation) 280
3.5.17. nfuiqr Function (Getting Information About Nodal Fluences) 281
3.5.18. nfuget Function (Getting a Nodal Fluence) 281
3.5.19. nfuput Subroutine (Storing a Nodal Fluence) 281
3.5.20. nfudel Subroutine (Deleting a Nodal Fluence) 282
3.5.21. ndciqr Function (Getting Information About Nodal Current Densities) 282
3.5.22. ndcget Function (Getting a Nodal Current Density) 282
3.5.23. ndcput Subroutine (Storing a Nodal Current Density) 283
3.5.24. ndcdel Subroutine (Deleting a Nodal Current Density) 283
3.5.25. nvdiqr Function (Getting Information About Nodal Magnetic Virtual Displacements) 283
3.5.26. nvdget Function (Getting a Nodal Magnetic Virtual Displacement) 284
3.5.27. nvdput Subroutine (Storing a Nodal Virtual Displacement) 284
3.5.28. nvddel Subroutine (Deleting a Nodal Virtual Displacement) 284
3.6. Element Loading Routines 285
3.6.1. epriqr Function (Getting Information About Element Pressure/Convection) 285
3.6.2. eprget Function (Getting an Element Face Pressure) 285
3.6.3. eprput Subroutine (Storing an Element Face Pressure) 286
3.6.4. eprdel Subroutine (Deleting an Element Pressure/Convection) 286
3.6.5. ecviqr Function (Getting Information About Element Convections) 286
3.6.6. ecvget Function (Getting an Element Face Convection) 287
3.6.7. ecvput Subroutine (Storing an Element Face Convection) 287
3.6.8. ecvdel Subroutine (Deleting a Convection on an Element) 288
3.6.9. etpiqr Function (Getting Information About Element Temperatures) 288
3.6.10. etpget Function (Getting an Element Temperature) 289
3.6.11. etpput Subroutine (Storing an Element Temperature) 289
3.6.12. etpdel Subroutine (Deleting an Element Temperature) 289
3.6.13. ehgiqr Function (Getting Information About Element Heat Generation) 290
3.6.14. ehgget Function (Getting an Element Heat Generation) 290
3.6.15. ehgput Subroutine (Storing an Element Heat Generation) 291
3.6.16. ehgdel Subroutine (Deleting an Element Heat Generation) 291
3.6.17. efuiqr Function (Getting Information About Element Fluences) 291
3.6.18. efuget Function (Getting an Element Fluence) 292
3.6.19. efuput Subroutine (Storing an Element Fluence) 292
3.6.20. efudel Subroutine (Deleting an Element Fluence) 292
3.6.21. edciqr Function (Getting Information About Element Current Densities) 293
3.6.22. edcget Function (Getting Element Current Densities) 293
3.6.23. edcput Subroutine (Storing an Element Current Density) 293
3.6.24. edcdel Subroutine (Deleting an Element Current Density) 294
3.6.25. evdiqr Function (Getting Information About Element Virtual Displacements) 294
3.6.26. evdget Function (Getting an Element Virtual Displacement) 294
3.6.27. evdput Subroutine (Storing an Element Virtual Displacement) 295
3.6.28. eimiqr Function (Getting Information About Element Impedances) 295
3.6.29. eimget Function (Getting an Element Face Impedance) 296
3.6.30. eimput Subroutine (Storing an Element Impedance) 296
3.6.31. eimdel Subroutine (Deleting an Element Impedance) 296

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

3.6.32. esfiqr Function (Getting Information About Element Surface Stress Data) 296
3.6.33. esfget Function (Getting Element Surface Stress Data) 297
3.6.34. esfput Subroutine (Storing Element Surface Stress Data) 297
3.6.35. esfdel Subroutine (Deleting an Element's Surface Stress Data) 297
3.6.36. efsdel Subroutine (Deleting a Flagged Surface on an Element) 298
3.6.37. efsget function (Getting Element Face Flagged Surfaces) 298
3.6.38. efsiqr function (Getting Information About Flagged Surfaces) 298
3.6.39. efsput Subroutine (Storing an Element Face Flagged Surface) 299
3.7. Results Information Routines 299
3.7.1. dspiqr Function (Getting Information About Nodal Results) 299
3.7.2. dspget Function (Getting a Nodal Result from the Database) 300
3.7.3. dspput Subroutine (Storing a Result at a Node) 300
3.7.4. dspdel Subroutine (Deleting a Result at a Node) 300
3.7.5. emsiqr Function (Getting Information About an Element's Miscellaneous Summable Data) 301
3.7.6. emsget Function (Getting an Element's Miscellaneous Summable Data) 301
3.7.7. emsput Subroutine (Storing an Element's Miscellaneous Summable Data) 301
3.7.8. emsdel Subroutine (Deleting an Element's Miscellaneous Summable Data) 302
3.7.9. enfiqr Function (Getting Information About Element Nodal Forces) 302
3.7.10. enfget Function (Getting an Element's Nodal Forces) 302
3.7.11. enfput Subroutine (Storing an Element's Nodal Forces) 303
3.7.12. enfdel Subroutine (Deleting an Element's Nodal Forces) 303
3.7.13. ensiqr Function (Getting Information About an Element's Nodal Stresses) 303
3.7.14. ensget Function (Getting an Element's Nodal Stresses) 304
3.7.15. ensput Subroutine (Storing Nodal Stresses at an Element) 304
3.7.16. ensdel Subroutine (Deleting an Element's Nodal Stresses) 305
3.7.17. engiqr Function (Getting Information About an Element's Energies) 305
3.7.18. engget Function (Getting an Element's Energies) 306
3.7.19. engput Subroutine (Storing an Element's Energies and Volume) 306
3.7.20. engdel Subroutine (Deleting an Element's Energies) 306
3.7.21. egriqr Function (Getting Information About an Element's Nodal Gradients) 307
3.7.22. egrget Function (Getting an Element's Nodal Gradients) 307
3.7.23. egrput Subroutine (Storing an Element's Nodal Gradients) 307
3.7.24. egrdel Subroutine (Deleting an Element's Nodal Gradients) 308
3.7.25. eeliqr Function (Getting Information About an Element's Nodal Elastic Strains) 308
3.7.26. eelget Function (Getting an Element's Nodal Elastic Strains) 308
3.7.27. eelput Subroutine (Storing an Element's Nodal Elastic Strains) 309
3.7.28. eeldel Subroutine (Deleting an Element's Nodal Elastic Strains) 310
3.7.29. epliqr Function (Getting Information About an Element's Nodal Plastic Strains) 310
3.7.30. eplget Function (Getting an Element's Nodal Plastic Strains) 310
3.7.31. eplput Subroutine (Storing an Element's Nodal Plastic Strains) 311
3.7.32. epldel Subroutine (Deleting an Element's Nodal Plastic Strains) 312
3.7.33. ecriqr Function (Getting Information About an Element's Nodal Creep Strains) 312
3.7.34. ecrget Function (Getting an Element's Nodal Creep Strains) 312
3.7.35. ecrput Subroutine (Storing an Element's Nodal Creep Strains) 313
3.7.36. ecrdel Subroutine (Deleting an Element's Nodal Creep Strains) 314
3.7.37. ethiqr Function (Getting Information About an Element's Nodal Thermal Strains) 314
3.7.38. ethget Function (Getting an Element's Nodal Thermal Strains) 314
3.7.39. ethput Subroutine (Storing an Element's Nodal Thermal Strains) 315
3.7.40. ethdel Subroutine (Deleting an Element's Thermal, Initial, and Swelling Strains) 316
3.7.41. euliqr Function (Getting Information About an Element's Euler Angles) 316
3.7.42. eulget Function (Getting an Element's Nodal Euler Angles) 316

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

3.7.43. eulput Subroutine (Storing an Element's Euler Angles) 317
3.7.44. euldel Subroutine (Deleting an Element's Euler Angles) 317
3.7.45. efxiqr Function (Getting Information About Element Fluxes) 318
3.7.46. efxget Function (Getting an Element Flux) 318
3.7.47. efxput Subroutine (Storing an Element's Fluxes) 318
3.7.48. efxdel Subroutine (Deleting Element Fluxes) 319
3.7.49. elfiqr Function (Getting Information About Element Local Forces) 319
3.7.50. elfget Function (Getting an Element Local Force) 319
3.7.51. elfput Subroutine (Storing an Element's Local Forces) 320
3.7.52. elfdel Subroutine (Deleting Element Local Forces) 320
3.7.53. emniqr Function (Getting Information About Element Miscellaneous Non-summable Data) 320
3.7.54. emnget Function (Getting an Element's Miscellaneous Non-summable Data) 321
3.7.55. emnput Subroutine (Storing an Element's Miscellaneous Non-summable Data) 321
3.7.56. emndel Subroutine (Deleting an Element's Miscellaneous Non-summable Data) 321
3.7.57. ecdiqr Function (Getting Information About Element Current Densities) 322
3.7.58. ecdget Function (Getting an Element Current Density) 322
3.7.59. ecdput Subroutine (Storing an Element's Current Densities) 322
3.7.60. ecddel Subroutine (Deleting Element Current Densities) 323
3.7.61. enliqr Function (Getting Information About Element Nonlinear Tables) 323
3.7.62. enlget Function (Getting Element Nonlinear Tables) 323
3.7.63. enlput Subroutine (Storing an Element's Nonlinear Tables) 324
3.7.64. enldel Subroutine (Deleting Element Nonlinear Tables) 324
3.7.65. ehciqr Function (Getting Information About Calculated Element Heat Generations) 324
3.7.66. ehcget Function (Getting a Calculated Element Heat Generation) 325
3.7.67. ehcput Subroutine (Storing an Element's Calculated Heat Generations) 325
3.7.68. ehcdel Subroutine (Deleting Element Calculated Heat Generations) 325
4. Subroutines for Your Convenience 327
4.1. Input and Output Abbreviations 327
4.2. General Subroutines 327
4.2.1. dptoch Subroutine (Retrieve Eight Characters From a Double Precision Variable) 327
4.2.2. wrinqr Function (Obtain Information About Output) 328
4.2.3. erinqr Subroutine (Obtaining Information from the Errors Common) 328
4.2.4. TrackBegin Subroutine (Beginning Tracking for a Subroutine Call) 330
4.2.5. TrackEnd Subroutine (Ending Tracking for a Subroutine Call) 330
4.2.6. erhandler Subroutine (Displaying Program Errors) 330
4.2.7. intrp Subroutine (Doing Single Interpolation) 331
4.2.8. tranx3 Subroutine (Processing Geometry for 3-D Line Elements) 332
4.2.9. systop Subroutine (Stopping a Program Run) 332
4.3. Vector Functions 333
4.3.1. vdot Function (Computing the Dot Product of Two Vectors) 333
4.3.2. vidot Function (Computing the Dot Product of Two Vectors with Increments) 333
4.3.3. vsum Function (Summing Vector Components) 333
4.3.4. vmax Function (Retrieving the Maximum Vector Value at a Given Location) 334
4.3.5. lastv Function (Retrieving the Position of the Last Nonzero Term in a Double Precision Vector) 334
4.3.6. izero Function (Setting an Integer Vector to Zero) 334
4.3.7. imove Function (Assigning Equal Values to Two Integer Vectors) 334
4.3.8. vzero Subroutine (Initializing a Vector to Zero) 334
4.3.9. vmove Subroutine (Moving One Vector into Another) 334
4.3.10. vimove Subroutine (Moving One Vector into Another Incrementally) 334
4.3.11. vinit Subroutine (Assigning a Scalar Constant to a Vector) 335

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

4.3.12. viinit Subroutine (Assigning a Scalar Constant to a Vector Incrementally) 335
4.3.13. vapb Subroutine (Setting a Vector to Sum of Two vectors) 335
4.3.14. vapb1 Subroutine (Combining Two Vectors in One) 335
4.3.15. vapcb1 Subroutine (Multiplying a Vector to a Constant) 335
4.3.16. vamb Subroutine (Gets a Third Vector by Subtracting One Vector from Another) 335
4.3.17. vamb1 Subroutine (Subtracting One Vector from Another) 335
4.3.18. vmult Subroutine (Multiplying a Vector by a Constant) 336
4.3.19. vmult1 Subroutine (Multiplying a Vector by a Constant) 336
4.3.20. vcross Subroutine (Defining a Vector via a Cross Product) 336
4.3.21. vnorme Subroutine (Normalizing a Three-Component Vector) 336
4.3.22. vnorm Subroutine (Normalizing a Vector to Unit Length) 336
4.3.23. ndgxyz Function (Getting the X,Y,Z Vector for a Node) 337
4.3.24. ndpxyz Subroutine (Storing X,Y,Z for a Node) 337
4.4. Matrix Subroutines 337
4.4.1. maxv Subroutine (Multiplying a Vector by a Matrix) 337
4.4.2. maxv1 Subroutine (Multiplying a Vector by a Matrix) 338
4.4.3. matxv Subroutine (Multiplying a Vector by a Full Transposed Matrix) 338
4.4.4. matxv1 Subroutine (Multiplying a Vector by a Full Transposed Matrix) 338
4.4.5. matxb Subroutine (Transposing a matrix) 339
4.4.6. maat Subroutine (Changing a Matrix Value via Addition, Multiplication, and Transposition) 339
4.4.7. matba Subroutine (Updating Matrix Value via Transposition, Multiplications, and Addition) 340
4.4.8. matsym Subroutine (Filling the Upper Triangle from the Lower Triangle) 340
4.4.9. mctac Subroutine (Transposing a symmetric matrix) 341
4.4.10. tran Subroutine (Transposing a matrix) 341
4.4.11. symeqn Subroutine (Solving Simultaneous Linear Equations) 341
A. Creating External Commands in Linux 343
A.1. Tasks in Creating an External Command 343
A.1.1. Creating Compatible Code 343
A.1.2. Creating a Shared Library 344
A.1.3. Creating an External Table File 345
A.1.4. Setting the ANSYS_EXTERNAL_PATH Environment Variable 346
A.1.5. Using External Commands 346
A.1.6. Checking External Command Status 346
A.1.7. Resetting External Commands 346
B. Creating External Commands in Windows 349
B.1. Tasks in Creating an External Command 349
B.1.1. Creating Compatible Code 349
B.1.2. Creating a Visual Studio Project 350
B.1.3. Creating an External Definition File 350
B.1.4. Creating a Shared Library 350
B.1.5. Creating an External Table File 350
B.1.6. Setting the ANSYS_EXTERNAL_PATH Environment Variable 351
B.1.7. Using External Commands 352
B.1.8. Checking External Command Status 352
B.1.9. Resetting External Commands 352
B.1.10. Example: Creating an External Command Using Visual Studio 2012 Professional 352
C. User Material (UserMat) Subroutine Example 355
C.1. UserMat Example Description 355
C.2. UserMat Example Input Data 355
C.3. UserMat Example POST26 Output Results 356

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

C.4. USERMAT.F List File for This Example 357
C.5. Accessing Solution and Material Data 363
D. Fully Coupled Wind Turbine Example in Mechanical APDL 365
D.1. Implementing a Fully Coupled Wind Turbine Analysis 365
D.2. Theory 366
D.3. Compiling a Custom Version of Mechanical APDL 367
D.4. Performing a Wind Coupled Analysis 368
D.4.1. The Wind Coupling Process 368
D.4.2. Data Exchange Routines 368
D.4.3. Important Analysis Notes 370
D.5. Example Analysis Using Provided "WindProg" Example for Aeroelastic Coupling 371
D.6. References 372
Index 375

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

List of Figures

4.1. Display Format for Z-buffered Graphics 107
4.2. Example Display of a Graphics File 113

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries and affiliates.

List of Tables

2.1. ANSYS Exit Codes 239

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries and affiliates.

Preface

About the *Programmer's Reference*

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

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries and affiliates.

Part I: Guide to Interfacing with ANSYS

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
1.10. Description of the Substructure Displacement 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 (a legacy results file)
- The *Jobname*.MODE file, storing data related to a modal analysis
- The *Jobname*.RDSP file, storing data related to a mode-superposition transient analysis.
- The *Jobname*.RFRQ file, storing data related to a 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 *Jobname*.DSUB file, storing displacements related to substructure matrices

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

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

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)

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

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
Item 29	The number of cores used with shared-memory parallel
Item 30	The number of cores used with distributed-memory parallel
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 (0 means the file will not split)
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
.rmg
.rst
.rth
.lnn
```

The ***XPL** command enables you to explore the contents of certain ANSYS binary files, including the results file. For more information, see Appendix B: Using APDL to List File Structure and Content in the *ANSYS Parametric Design Language Guide*.

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.

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

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 ANSYS.  All Rights Reserved.
c *** ansys, inc

c     **********  description of results file  **********
c     --- used for the following files:
c       .rfl
c       .rmg
c       .rst
c       .rth
c       .lnn(lxx)

      character*8  RSTNM
      parameter   (RSTNM='rst     ')

      LONGINT         resufpL, adrZipL, resuRfpL
      integer         resubk, resuut, resuRbk, resuRut
      common /fdresu/ resufpL, adrZipL, resubk, resuut,
     x                resuRfpL, resuRbk, resuRut

c   ********** common variable descriptions ***********
co  resufpL      file position on file resu
co  resubk       block number for file resu  (usually 6)
co  resuut       file unit for file resu     (0 if not open)  FUN12
c0  resuRxx      variables for remote modal RST file
c   See fddesc for documentation of how binary files are stored

c     **********  file format  **********

c        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.

c        type tells what kind of information is stored in this record:
c             i - integer
c             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    type    nrec    lrec     contents

c  ---      i       1      100      standard ANSYS file header (see binhed for
c                                   details of header contents)

c  ---      i       1       80      .RST FILE HEADER

c                                     fun12,    maxn,     nnod,   resmax,   numdof,
c                                      maxe,    nelm,      kan,    nsets,   ptrend,  (10)
c                                   ptrDSIl, ptrTIMl,  ptrLSPl,  ptrELMl,  ptrNODl,
c                                    ptrGEO, ptrCYCl,   CMSflg,    csEls,    units,  (20)
c                                   nSector,  csCord,  ptrEnd8,  ptrEnd8,  fsiflag,
c                                     pmeth,  noffst,   eoffst,   nTrans, ptrTRANl,  (30)
c                                   PrecKey,   csNds,   cpxrst,   extopt,   nlgeom,
c                                 AvailData,   mmass, kPerturb,  XfemKey,  rstsprs,  (40)
c                                   ptrDSIh, ptrTIMh,  ptrLSPh,  ptrCYCh,  ptrELMh,
c                                   ptrNODh, ptrGEOh, ptrTRANh,  Glbnnod, ptrGNODl,  (50)
c                                  ptrGNODh, qrDmpKy,  MSUPkey,   PSDkey,cycMSUPkey,
c                           XfemCrkPropTech,cycNoDup,decompMth,        0,        0,  (60)
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
c                                    0,       0,       0,       0,       0,
c                                    0,       0,       0,       0,       0,  (70)
c                                    0,       0,       0,       0,       0,
c                                    0,       0,       0,       0,       0   (80)

c                                  each item in header is described below:

c                                   fun12  - unit number (resu file is 12)
c                                   maxn   - maximum node number of the model
c                                   nnod   - the actual number of nodes used in
c                                            the solution phase
c                                   resmax - the maximum number of data sets
c                                            allowed on the file (defaults to
c                                            10000; minimum allowed is 10)
c                                   numdof - number of DOFs per node
c                                   maxe   - maximum element number of the
c                                            finite element model
c                                   nelm   - number of finite elements
c                                   kan    - analysis type
c                                   nsets  - number of data sets on the file
c                                   ptrend - pointer to the end of the file
c                                             (see ptrEnd8 in 23,24)
c                                ptrDSIl,h - 64 bit pointer to the data steps
c                                            index table
c                                ptrTIMl,h - 64 bit pointer to the table of time
c                                            values for a load step
c                                ptrLSPl,h - 64 bit pointer to the table of load
c                                            step, substep, and cumulative
c                                            iteration numbers
c                                ptrELMl,h - 64 bit pointer to the element equivalence
c                                            table (used when the mesh does not
c                                            change during solution)
c                                ptrNODl,h - 64 bit pointer to the nodal equivalence
c                                            table (used when the mesh does not
c                                            change during solution)
c                                ptrGEOl,h - 64 bit pointer to the beginning of
c                                            geometry information (used when the
c                                            mesh does not change during solution)
c                                ptrCYCl,h - 64 bit pointer to the table of cyc sym
c                                            nodal-diameters at each load step
c                                   CMSflg - CMS results flag: 0-non cms, >0-cms
c                                   csEls  - Cyclic sym # eles in master sector
c                                   units  - unit system used
c                                            = 0 - user defined units
c                                            = 1 - SI
c                                            = 2 - CSG
c                                            = 3 - U.S. Customary, using feet
c                                            = 4 - U.S. Customary, using inches
c                                            = 5 - MKS
c                                            = 6 - MPA
c                                            = 7 - uMKS
c                                  nSector - number of sectors for cyclic sym
c                                  csCord  - Cyclic symmetry coordinate system
c                                  ptrEnd8 - 64 bit file pointer to the end of
c                                            the file (i.e., length of file)
c                                  fsiflag - FSI analyis flag
c                                  pmeth   - p-method analyis flag
c                                  noffst  - node offset used in writing file
c                                  eoffst  - elem offset used in writing file
c                                  nTrans  - number of SE transformation vects
c                                ptrTRANl,h - 64 bit pointer to SE transformation vects
c                                  PrecKey - 0, double precision
c                                            1, single for element results only
c                                            2, single for all data
c                                  csNds   - Cyclic sym # nds in master sector
c                                  cpxrst  - complex results flag (0-no, 1-yes)
c                                  extopt  - mode extraction option
c                                  nlgeom  - NLGEOM key
c                                AvailData  - bits indicating available data any
c                                            where on the file; see resucm.inc
c                                  mmass   - number of missing mass resp. present
c                                 kPerturb - key for Linear Perturbation results
```

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
c                                  XfemKey - XFEM flag (set equal to Active_XfenId)
c                                                    (0=Inactive, 1=Active)
c                                  rstsprs - bitmask for suppressed items
c                                  Glbnnod - global number of nodes actually used
c                                            in the solution phase (== nnod unless
c                                            using Distributed Ansys)
c                               ptrGNODl,h - 64 bit pointer to the global nodal
c                                            equivalence table (only used with
c                                            Distributed ANSYS and when the mesh
c                                            does not change during solution)
c                                  qrDmpKy - QR damped calculations key
c                                  MSUPkey - MSUP results expanded with MSUPCombineModes
c                                  PSDkey  - PSD key
c                               cycMSUPkey - rst file format is for subsequent cyclic MSUP
c                                            (only base results on file)
c                           XfemCrkPropTech - XFEM crack propagation technology
c                                       (0=phantom node-based, 1=singularity-based)
c                                 cycNoDup - no duplicate sector created
c                                decompMth - domain decomposition method employed with DMP
c                                            (> 0 = MESH; -1 = FREQ, -2 = CYCHI)
c
c                                    Note: ptrXXX are relative to beginning of file

c   ---      i       1     numdof   Degrees of freedom per node
c                                    DOF reference numbers are:
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  *** 13-15 are spares *** , WARP=16
c         CONC=17, HDSP=18, PRES=19, TEMP=20, VOLT=21, MAG =22, ENKE=23, ENDS=24
c         EMF =25, CURR=26, SP01=27, SP02=28, SP03=29, SP04=30, SP05=31, SP06=32
c         TBOT=33, TE2 =34, TE3 =35, TE4 =36, TE5 =37, TE6 =38, TE7 =39, TE8 =40
c         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         TE25=57, TE26=58, TE27=59, TE28=60, TE29=61, TE30=62, TE31=63, TTOP=64
c          *** 65-128 are spares ***
c                                     (curdof(i),i=1,numdof)

c   NOD      i       1     nnod     Nodal equivalence table. This table equates
c                                    the number used for storage to the actual
c                                    node number
c                                     (Back(i),i=1,nnod)

c   ELM      i       1     nelm     Element equivalence table. The ANSYS program
c                                    stores all element data in the numerical
c                                    order that the SOLUTION processor solves the
c                                    elements.  This table equates the order
c                                    number used to the actual element number

c   GNOD     i       1    Glbnnod   Global nodal equivalence table. This table
c                                    equates the number used for storage to the
c                                    actual node number. Only written by the
c                                    master process in Distributed Ansys
c                                     (GlbBack(i),i=1,Glbnnod)

c   DSI      i       1    2*resmax  Data sets index table. This record contains
c                                    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                                    resmax records are the second 32 bits. To
c                                    create the 64 bit pointer, use:
c                                      LONGPTR = largeIntGet (first,second)
c                                    Read the solution data header as follows:
c                                      call bioBasePut (nblk,LONGPTR)
c                                      loc = bioiqr (nblk,12)
c                                      call biord (nblk,loc,...
c                                    The rest of the file reading continues to use
c                                    the ptrXXX's that are in the headers.

c   TIM     dp       1     resmax   Time/freq table. This record contains the
c                                    time (or frequency) values for each data
c                                    set.

c   LSP      i       1    3*resmax  Data set identifiers. This record contains
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries and affiliates.

```
c                                    the load step, substep, and cumulative
c                                    iteration numbers for each data set.

c   CYC      i       1     resmax   Cyclic symmetry harmonic index

c   TRAN     dp    nTran     25     Substructure transformation vectors

c   GEO      i       1       80     Geometry data header(was 20 in 32 bit vers)

c                                         0,  maxety,   maxrl,    nnod,    nelm,
c                                    maxcsy,  ptrETY,  ptrREL,  ptrLOC,  ptrCSY,  (10)
c                                    ptrEID,  maxsec,  secsiz,  maxmat,  matsiz,
c                                    ptrMAS,  csysiz,  elmsiz,  etysiz,   rlsiz,  (20)
c                                  ptrETYl, ptrETYh, ptrRELl, ptrRELh, ptrCSYl,
c                                  ptrCSYh, ptrLOCl, ptrLOCh, ptrEIDl, ptrEIDh,  (30)
c                                  ptrMASl, ptrMASh, ptrSECl, ptrSECh, ptrMATl,
c                                  ptrMATh, ptrCNTl, ptrCNTh, ptrNODl, ptrNODh,  (40)
c                                  ptrELMl, ptrELMh, Glblenb,ptrGNODl,ptrGNODh,
c                                     maxn,NodesUpd,  lenbac, maxcomp, compsiz,  (50)
c                                ptrCOMPl,ptrCOMPh,       0,       0,       0,
c                                       0,       0,       0,       0,       0,  (60)
c                                       0,       0,       0,       0,       0,
c                                       0,       0,       0,       0,       0,  (70)
c                                       0,       0,       0,       0,       0,
c                                       0,       0,       0,       0,       0   (80)

c                                   each item in header is described below:

c                                    0      - position not used
c                                    maxety - the maximum element type reference
c                                             number in the model
c                                    maxrl  - the maximum real constant reference
c                                             number in the model
c                                    nnod   - the number of defined nodes in the
c                                             model
c                                    nelm   - the number of defined elements in
c                                             the model
c                                    maxcsy - the maximum coordinate system
c                                             reference number in the model
c                                    ptrETY - pointer to the element type index
c                                             table
c                                    ptrREL - pointer to the real constant
c                                             index table
c                                    ptrLOC - pointer to the nodal point
c                                             locations
c                                    ptrCSY - pointer to the local coordinate
c                                             system index table
c                                    ptrEID - pointer to the element index
c                                             table
c                                    maxsec - the maximum section
c                                             reference number in the model
c                                    secsiz - the maximum size that any
c                                             section record may have
c                                    nummat - the number of materials
c                                             in the model
c                                    matsiz - the maximum size that any material
c                                             property or table may have
c                                    ptrMAS - pointer to the diagonal mass matrix
c                                    csysiz - the number of items describing a
c                                             local coordinate system (usually
c                                             24)
c                                    elmsiz - the maximum number of nodes that a
c                                             defined element may have
c                                    etysiz - the number of items describing an
c                                             element type(=IELCSZ from echprm.inc)
c                                    rlsiz  - the maximum number of items
c                                             defining a real constant (0, if no
c                                             real constants are defined)
c                                 ptrETYl,h - 64 bit pointer to element type data
c                                 ptrRELl,h - 64 bit pointer to real constant data
c                                 ptrCSYl,h - 64 bit pointer to coordinate system data
c                                 ptrLOCl,h - 64 bit pointer to nodal locations
```

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
c                                ptrEIDl,h - 64 bit pointer to element data
c                                ptrSECl,h - 64 bit pointer to section data
c                                ptrMATl,h - 64 bit pointer to material data
c                                ptrCNTl,h - 64 bit pointer to element centroids
c                                ptrNODl,h - 64 bit pointer to nodal equivalence table
c                                ptrELMl,h - 64 bit pointer to element equivalence table
c                                  Glblenb - global number of nodes actually used
c                                            in the solution phase (== lenbac unless
c                                            using Distributed Ansys)
c                                ptrGNODl,h- 64 bit pointer to the global nodal
c                                            equivalence table (only used with
c                                            Distributed ANSYS and when the mesh
c                                            does not change during solution)
c                                   maxn   - maximum node number of the model
c                                 NodesUpd - 1, node coords have been updated
c                                   lenbac - the actual number of nodes used in
c                                            the solution phase
c                                  maxcomp - the maximum component
c                                            reference number in the model
c                                  compsiz - the maximum size that any
c                                            component record may have
c                               ptrCOMPl,h - 64 bit pointer to packed component data
c
c                                   Note: ptrXXX are relative to beginning of file

c   ETY      i       1     maxety   The element types index table. This record
c                                   contains record pointers for each element
c                                   type description. (Relative to ptrETYPL
c                                   for 64 bit version)

c   ---      i     numety  etysiz   Element type description. Each of these
c                                   records is pointed to by a record pointer
c                                   given in the record labeled ETY. See
c                                   routines echprm and elccmt for a complete
c                                   description of the items stored here.

c                                   These items are typically stored into the
c                                   IELC array, and are used to determine the
c                                   element type characteristics at runtime.
c                                   The following items are typically of
c                                   interest:
c                                   * Item 1     - element type reference number
c                                   * Item 2     - element routine number
c                                   * Items 3-14 - element type option keys
c                                                  (keyopts)
c                                   * Item 34    - DOF/node for this element
c                                                  type.  This is a bit mapping
c                                                  of the DOF/node.
c                                   * Item 61    - number of nodes for this
c                                                  element type (nodelm)
c                                   * Item 63    - number of nodes per element
c                                                  having nodal forces, etc.
c                                                  (nodfor)
c                                   * Item 94    - number of nodes per element
c                                                  having nodal stresses, etc.
c                                                  (nodstr).  This number is the
c                                                  number of corner nodes for
c                                                  higher-ordered elements.

c   REL      i       1     maxrl    Real constants index table.  The record
c                                   contains record pointers for each real
c                                   constant set. (Relative to ptrRELL for
c                                   64 bit version)

c   ---      dp     numrl   varies   Element real constant data.  These records
c                                   contain real constant data used for the
c                                   elements.  (See the ANSYS Elements Reference
c                                   manual for values for a specific element.)
c                                   Each of these records is pointed to by a
c                                   record pointer given in the record labeled
c                                   REL. The length of these records varies for
c                                   each element type (actual length is returned
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries and affiliates.

```
c                                    from routine BINRD8).

c   CSY      i       1     maxcsy    Coordinate systems index table.  This record
c                                    contains the record pointers for each
c                                    coordinate system set. The ANSYS program
c                                    writes coordinate systems only if local
c                                    coordinate systems were defined.  If a local
c                                    system was defined, the predefined global
c                                    systems 1 to 2 also will be written.  The
c                                    global Cartesian system 0 will never be
c                                    written. (Relative to ptrCSYSL for 64
c                                    bit version)

c   ---      dp     numcsy  csysiz   Coordinate system description.  These
c                                    records contain coordinate system data for
c                                    each coordinate system defined.  Each of
c                                    these records is pointed to by a record
c                                    pointer given in the record labeled SYS.

c                                    The items stored in each record:

c                                    * Items 1-9  are the transformation matrix.
c                                    * Items 10-12 are the coordinate system
c                                      origin (XC,YC,ZC).
c                                    * Items 13-14 are the coordinate system
c                                      parameters (PAR1, PAR2).
c                                    * Items 16-18 are the angles used to define
c                                      the coordinate system.
c                                    * Items 19-20 are theta and phi singularity
c                                      keys.
c                                    * Item 21 is the coordinate system type
c                                      (0, 1, 2, or 3).
c                                    * Item 22 is the coordinate system reference
c                                      number.

c   LOC      dp       1     7*nnod    This group contains the node number and
c                                    coordinates (in the order
c                                      Node,X,Y,Z,THXY,THYZ,THZX) for each node.
c                                    (32 bit version)

c   LOC      dp     nnod      7       (64 bit version)
c                                    Node,X,Y,Z,THXY,THYZ,THZX for each node
c                                    Nodes are in node number order

c   EID      i       1     nelm      Element descriptions index table. This
c                                    record contains the record pointers for each
c                                    element description. (LONGINT (2*nelm) for
c                                    64 bit version, relative to ptrEIDL).
c                                    The order of the elements is the same as
c                                    the order in the element equivalence table.

c   ---      i     nelm  10+nodelm   Element descriptions. Each of these records
c                                    is pointed to by a record pointer given in
c                                    the record labeled EID. The length of these
c                                    records varies for each element (actual
c                                    length is returned from routine BINRD8).
c                                    nodelm shown here is the number of nodes for
c                                    this element. Its value is defined in the
c                                    element type description record.

c                                    The items stored in each record:
c                                       mat,  type,  real, secnum, esys,
c                                     death,solidm, shape, elnum,  baseeid,
c                                     nodes

c                                   each item is described below:

c                                     mat    - material reference number
c                                     type   - element type number
c                                     real   - real constant reference number
c                                     secnum - section number
c                                     esys   - element coordinate system
```

```
c                                    death  - death flag
c                                             = 0 - alive
c                                             = 1 - dead
c                                    solidm - solid model reference
c                                    shape  - coded shape key
c                                    elnum  - element number
c                                    baseeid- base element number
c                    (applicable to reinforcing elements only)
c                                    nodes  - node numbers defining the element
c                                             (See the ANSYS Elements Reference
c                                             for nodal order of an element)

c   CENT    dp     nelm      6      Centroid record for each element

c   MAS     dp       1  nnod*numdof Diagonal mass matrix

c   SEC      i       1     maxsec   Section index table.  The record
c                                   contains record pointers for each
c                                   section set.

c   ---     dp     numsec  varies   Element section data.  These records
c                                   contain section data used for the
c                                   elements.
c                                   Each of these records is pointed to by a
c                                   record pointer given in the record labeled
c                                   SEC. The length of these records varies for
c                                   each section type (actual length is returned
c                                   from routine BINRD8).

c   MAT      i     1  3+nummat*(158+1)+2
c                                   Total Sz = Header Data + (158+1)*Number of Materials + Tail
c                                   The 1st 3 integers contain the header information
c                                   1) Version number (-101 for differentiation from prev rst data)
c                                   2) Header size which is 3
c                                   3) Size of the index array
c
c                                   Ith material ID is stored at Data(3+((I-1)*(158+1))+1)
c                                     I varies from 1 to nummat
c
c                                   Material record pointers for the material ID at Ith position
c                                     is stored from location
c                                     3+(I-1)*(158+1)+2 to 3+(I-1)*(158+1)+1+158
c                                   The record includes MP pointers followed by TB pointers for a
c                                     single Material ID
c                                   Last 2 data contains the active table number and material ID

c   ---     dp     nummat  varies   Material property data.  These records
c                                   contain material property data used for the
c                                   elements.
c                                   Each of these records is pointed to by a
c                                   record pointer given in the record labeled
c                                   MAT. The length of these records varies for
c                                   each material property type (actual length
c                                   is returned from routine BINRD8).

c   NOD      i       1    lenbac    Nodal equivalence table. This table equates
c                                   the number used for storage to the actual
c                                   node number
c                                    (Back(i),i=1,lenbac)

c   ELM      i       1     nelm     Element equivalence table. The ANSYS program
c                                   stores all element data in the numerical
c                                   order that the SOLUTION processor solves the
c                                   elements.  This table equates the order
c                                   number used to the actual element number

c   GNOD     i       1    Glblenb   Global nodal equivalence table. This table
c                                   equates the number used for storage to the
c                                   actual node number. Only written by the
c                                   master process in Distributed Ansys
c                                    (GlbBack(i),i=1,Glblenb)
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries and affiliates.

```
c        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.

c        Each set of data is pointed to by a record pointer given in the record
c        labeled DSI.

c   ---      i       1       200     Solution data header. (was 100 in 32 bit)

c                                    pv3num,    nelm,    nnod,    mask,   itime,
c                                      iter,  ncumit,     nrf,  cs_LSC,   nmast,  (10)
c                                    ptrNSL,  ptrESL,   ptrRF,  ptrMST,   ptrBC,
c                                    rxtrap,    mode,    isym,  kcmplx,  numdof,  (20)
c                                      DOFS,    DOFS,    DOFS,    DOFS,    DOFS,
c                                      DOFS,    DOFS,    DOFS,    DOFS,    DOFS,  (30)
c                                      DOFS,    DOFS,    DOFS,    DOFS,    DOFS,
c                                      DOFS,    DOFS,    DOFS,    DOFS,    DOFS,  (40)
c                                      DOFS,    DOFS,    DOFS,    DOFS,    DOFS,
c                                      DOFS,    DOFS,    DOFS,    DOFS,    DOFS,  (50)
c                                     title,   title,   title,   title,   title,
c                                     title,   title,   title,   title,   title,  (60)
c                                     title,   title,   title,   title,   title,
c                                     title,   title,   title,   title,   title,  (70)
c                                    stitle,  stitle,  stitle,  stitle,  stitle,
c                                    stitle,  stitle,  stitle,  stitle,  stitle,  (80)
c                                    stitle,  stitle,  stitle,  stitle,  stitle,
c                                    stitle,  stitle,  stitle,  stitle,  stitle,  (90)
c                                    dbmtim,  dbmdat,  dbfncl,  soltim,  soldat,
c                                    ptrOND,  ptrOEL,  nfldof,  ptrEXA,  ptrEXT,  (100)
c                                  ptrEXAl, ptrEXAh, ptrEXTl, ptrEXTh, ptrNSLl,
c                                  ptrNSLh,  ptrRFl,  ptrRFh, ptrMSTl, ptrMSTh,  (110)
c                                   ptrBCl,  ptrBCh, ptrTRFl, ptrTRFh, ptrONDl,
c                                  ptrONDh, ptrOELl, ptrOELh, ptrESLl, ptrESLh,  (120)
c                                  ptrOSLl, ptrOSLh,sizeDEAD,ptrDEADl,ptrDEADh,
c                                  PrinKey, numvdof, numadof,       0,       0,  (130)
c                                  ptrVSLl, ptrVSLh, ptrASLl, ptrASLh,       0,
c                                        0,       0,       0,numRotCmp,      0,  (140)
c                                  ptrRCMl, ptrRCMh, nNodStr,       0,ptrNDSTRl,
c                                ptrNDSTRh,AvailData, geomID, ptrGEOl, ptrGEOh,  (150)
c                                        0,       0,       0,       0,       0,
c                                        0,       0,       0,       0,       0,  (160)
c                                        0,       0,       0,       0,       0,
c                                        0,       0,       0,       0,       0,  (170)
c                                        0,       0,       0,       0,       0,
c                                        0,       0,       0,       0,       0,  (180)
c                                        0,       0,       0,       0,       0,
c                                        0,       0,       0,       0,       0,  (190)
c                                        0,       0,       0,       0,       0,
c                                        0,       0,       0,       0,       0   (200)

c                                   each item in header is described below:

c                                    pv3num - current solu set number
c                                    nelm   - number of elements
c                                    nnod   - number of nodes
c                                    mask   - bitmask for the existence of
c                                             several records.  If a bit is set
c                                             here, it indicates that the
c                                             corresponding record exists on the
c                                             file.
c                                             The items in the bitmask that
c                                             correspond to each record are shown
c                                             in the record descriptions below.
c                                    itime  - loadstep
c                                    iter   - iteration number
c                                    ncumit - cumulative iteration number
c                                    nrf    - number of reaction forces
c                                    cs_LSC - cyclic symmetry count of the
c                                             load step for this SOLVE
c                                    nmast  - number of masters
c                                    ptrNSL - 32-bit pointer to nodal solution
```

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
c                                ptrESL - 32-bit pointer to element solution
c                                ptrRF  - 32-bit pointer to reaction forces
c                                ptrMST - 32-bit pointer to the masters
c                                ptrBC  - 32-bit pointer to the boundary conditions
c                                rxtrap - key to extrapolate integration
c                                         point results to nodes
c                                         = 0 - move
c                                         = 1 - extrapolate unless active
c                                               non-linear
c                                         = 2 - extrapolate always
c                                mode   - mode number of harmonic loading
c                                         (for cyclic symmetry: this is cs_LSF
c                                         = first load step for this SOLVE)
c                                isym   - symmetry for harmonic loading
c                                         (for cyclic symmetry: this is cs_LSL
c                                         = last load step for this SOLVE)
c                                kcmplx - complex key
c                                         = 0 - real
c                                         = 1 - imaginary
c                                numdof - number of DOFs/nodes for this data
c                                         set
c                                DOFS   - DOF/node reference numbers (numdof
c                                         values)
c                                title  - main title (in integer form)
c                               stitle1 - 1st subtitle (in integer form)
c                                dbmtim - time (in compact form) when the
c                                         database was last modified
c                                dbmdat - date (in compact form) when the
c                                         database was last modified
c                                dbfncl - number of times that the database
c                                         was modified
c                                soltim - time (in compact form) when the
c                                         solution for this data set was done
c                                soldat - date (in compact form) when the
c                                         solution for this data set was done
c                                ptrOND - 32-bit pointer to the ordered node
c                                         list (load case files only)
c                                ptrOEL - 32-bit pointer to the ordered element
c                                         list (load case files only)
c                                nfldof - number of extra Flotran DOFs/nodes
c                                         for this data set
c                                ptrEXA - 32-bit pointer to header extension
c                                         for FLOTRAN DOF/extra DOF list
c                                ptrEXT - 32-bit pointer to header extension
c                              ptrEXAl,h - 64-bit pointer to header extension
c                                         for FLOTRAN DOF/extra DOF list
c                              ptrEXTl,h - 64-bit pointer to header extension
c                              ptrNSLl,h - 64-bit pointer to nodal solution
c                              ptrRFl,h  - 64-bit pointer to reaction forces
c                              ptrMSTl,h - 64-bit pointer to the masters
c                              ptrBCl,h  - 64-bit pointer to the boundary conditions
c                              ptrTRFl,h - 64-bit pointer to the FLOTRAN
c                                         transient solution vector
c                              ptrONDl,h - 64-bit pointer to the ordered node
c                                         list (load case files only)
c                              ptrOELl,h - 64-bit pointer to the ordered element
c                                         list (load case files only)
c                              ptrESLl,h - 64-bit pointer to element solution
c                              ptrOSLl,h - 64-bit pointer to extra solution vector
c                                         (to be used later for rezoning project)
c                              sizeDEAD  - size of dead element list
c                              ptrDEADl,h- 64-bit pointer to dead element list
c                                PrinKey - principal stress key:
c                                          0, use rstsprs (if bit 26 set, no prin)
c                                          1, principals are written for this set
c                                         -1, no principals are written for this set
c                                numvdof - number of velocity dofs
c                                numadof - number of acceleration dofs
c                              ptrVSLl,h - 64-bit pointer to transient velocity
c                                         solution
c                              ptrASLl,h - 64-bit pointer to transient acceleration
c                                         solution
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries and affiliates.

```
c                                numRotCmp - number of rotating components
c                                ptrRCMl,h - 64-bit pointer to RCM
c                                  nNodStr - 0, no nodal component stresses
c                                            1, one set (TOP for shells)
c                                            2, two sets (TOP,BOT for shells)
c                                            3, three sets (TOP,BOT,MID)
c                              ptrNDSTRl,h - 64 bit pointer to nodal component str
c                                AvailData - bits indicating available data
c                                            in this data set; see resucm.inc
c                                   geomID - number identifying which geometry (mesh)
c                                            is used for this set of data (when mesh
c                                            does not change during solution this should
c                                            always be equal to 1)
c                                ptrGEOl,h - 64 bit pointer to geometry data (when mesh
c                                            does not change during solution this points
c                                            to first GEO record near start of file)
c                                        0 - position not used
c
c                                   Note: ptrXXX are relative to ptrDSI, except ptrGEO
c                                         which is relative to the beginning of the file

c   ---     dp       1        100   Solution header - double precision data

c                                    timfrq,  lfacto,  lfactn,  cptime,    tref,
c                                     tunif,   tbulk, volbase,   tstep, dsfpinc,  (10)
c                                     accel,   accel,   accel,   omega,   omega,
c                                     omega,   omega,   omega,   omega, omegacg,  (20)
c                                   omegacg, omegacg, omegacg, omegacg, omegacg,
c                                     cgcent,  cgcent,  cgcent, fatjack, fatjack,  (30)
c                                     dval1,   dval2,   dval3, pCnvVal, pCnvVal,
c                                   pCnvVal, pCnvVal, pCnvVal, pCnvVal, pCnvVal,  (40)
c                                   pCnvVal, pCnvVal, pCnvVal, pCnvVal, pCnvVal,
c                                   pCnvVal, pCnvVal, pCnvVal, pCnvVal, pCnvVal,  (50)
c                                    timdat,  timdat,  timdat,  timdat,  timdat,
c                                    timdat,  timdat,  timdat,  timdat,  timdat,  (60)
c                                    timdat,  timdat,  timdat,  timdat,  timdat,
c                                    timdat,  timdat,  timdat,  timdat,  timdat,  (70)
c                                    timdat,  timdat,  timdat,  timdat,  timdat,
c                                    timdat,  timdat,  timdat,  timdat,  timdat,  (80)
c                                    timdat,  timdat,  timdat,  timdat,  timdat,
c                                    timdat,  timdat,  timdat,  timdat,  timdat,  (90)
c                                    timdat,  timdat,  timdat,  timdat,  timdat,
c                                    timdat,  timdat,  timdat,  timdat,  timdat   (100)

c                                   each item is described below:

c                                    timfrq  - time value (or frequency value,
c                                              for a modal or harmonic analysis)
c                                    lfacto  - the "old" load factor (used in
c                                              ramping a load between old and new
c                                              values)
c                                    lfactn  - the "new" load factor
c                                    cptime  - elapsed cpu time (in seconds)
c                                    tref    - the reference temperature
c                                    tunif   - the uniform temperature
c                                    tbulk   - Bulk temp for FLOTRAN film coefs
c                                    VolBase - Initial total volume for VOF
c                                    tstep   - Time Step size for FLOTRAN analysis
c                                    dsfpinc - incident power of diffuse sound field
c                                    accel   - linear acceleration terms
c                                    omega   - angular velocity (first 3 terms) and
c                                              angular acceleration (second 3 terms)
c                                    omegacg - angular velocity (first 3 terms) and
c                                              angular acceleration (second 3 terms)
c                                              these velocity/acceleration terms are
c                                              computed about the center of gravity
c                                    cgcent  - (x,y,z) location of center of gravity
c                                    fatjack - FATJACK ocean wave data (wave height
c                                              and period)
c                                      dval1 - if pmeth=0: FATJACK ocean wave direction
c                                              if pmeth=1: p-method convergence values
c                                      dval2 - if pmeth=0: machine rpm
```

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries and affiliates.

```
c                                          if pmeth=1: p-method convergence values
c                                    dval3 - if pmeth=0: central frequency of tune band
c                                          if pmeth=1: p-method convergence values
c                                   pCnvVal - p-method convergence values
c                                    timdat - if pmeth=0: load data (slot 53 is
c                                          substep convergence key)
c                                          if pmeth=1: p-method convergence values

c   EXA      i       1         64    Header extension (if ptrEXA=ptrEXT, then
c                                                      ptrEXA is unused.)
c                                   positions   1-32  - current extra Flotran
c                                                       DOFs for this set
c                                   positions  33-64  - current extra Flotran
c                                                       DOF labels for this set

c                                   Extra Flotran DOF reference numbers are:
c         DENS= 1, VISC= 2, EVIS= 3, COND= 4, ECON= 5, LMD1= 6, LMD2= 7, LMD3= 8
c         LMD4= 9, LMD5=10, LMD6=11, EMD1=12, EMD2=13, EMD3=14, EMD4=15, EMD5=16
c         EMD6=17, PTOT=18, TTOT=19, PCOE=20, MACH=21, STRM=22, HFLU=23, HFLM=24
c         YPLU=25, TAUW=26, SPHT=27, CMUV=28
c         *************************** 29-32 are spares ************************

c   EXT      i       1        200    Header extension
c                                   positions   1-32  - current DOF for this
c                                                       result set
c                                   positions  33-64  - current DOF labels for
c                                                       this result set
c                                   positions  65-84  - The third title, in
c                                                       integer form
c                                   positions  85-104 - The fourth title, in
c                                                       integer form
c                                   positions 105-124 - The fifth title, in
c                                                       integer form
c                                   position 125 - ptrTRF- pointer to FLOTRAN
c                                                  previous time step DOF vals
c                                   position 126 - trnvar- #dof in FLOTRAN
c                                                  prev time st DOF vals
c                                                  (Note 2 old steps saved,
c                                                  thus #DP is 2*trnvar*nNode)
c                                   position 127 - numvdof, number of velocity
c                                                  items per node (ANSYS
c                                                  transient)
c                                   position 128 - numadof, number of
c                                                  acceleration items per
c                                                  node (ANSYS transient)
c                                   position 131-133 - position of velocity
c                                                      in DOF record
c                                                      (ANSYS transient)
c                                   position 134-136 - position of acceleration
c                                                      in DOF record
c                                                      (ANSYS transient)
c                                   position 137-142 - velocity and
c                                                      acceleration labels
c                                                      (ANSYS transient)
c                                   position 143 - number of stress items
c                                                  (6 or 11); a -11 indicates
c                                                  to use principals directly
c                                                  and not recompute (for PSD)
c                                   position 144-146 - position of rotational
c                                                      velocity in DOF record
c                                                      (ANSYS transient)
c                                   position 147-149 - position of rotational
c                                                      accel. in DOF record
c                                                      (ANSYS transient)
c                                   position 150-155 - rotational velocity and
c                                                      acceleration labels
c                                                      (ANSYS transient)
c                                   position 160 - ptrDMI (J Integral results)
c                                   position 161 - nContours
c                                   if pmeth=1:
c                                      positions 164-200 - p convergence specs
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries and affiliates.

```
c   GEO     ---      1     varies    Entire geometry record for this set of
c                                    results data.  Note, when the mesh does not
c                                    change during solution, this pointer will
c                                    simply point to the original GEO record
c                                    stored at the start of the file.  When the
c                                    mesh changes the new geoemtry data will be
c                                    stored here.  The geomID can be used to
c                                    determine when the mesh changes.

c * NSL     dp       1  nnod*Sumdof  The DOF solution for each node in the nodal
c                                    coordinate system.  The DOF order is the
c                                    same as shown above in the DOF number
c                                    reference table.  The nodal order is the
c                                    same order given above in the nodal
c                                    equivalence table.  If a DOF for a node
c                                    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                                    there is less than nnod*Sumdof items in the
c                                    record, then only a selected set of nodes
c                                    were output. Another record follows
c                                    (integer, less than nnod long) which
c                                    contains the list of nodes for which DOF
c                                    solutions are available.
c                                      (bit 10 (PDBN) in mask)

c   VSL      dp       1 nnod*numvdof  The velocity solution for each node in the
c                                    nodal coordinate system. The description
c                                    for the DOF solution above also applies
c                                    here.
c                                    ANSYS transient. (bit 27 (PDVEL) in mask)

c   ASL      dp       1 nnod*numadof  The acceleration solution for each node in
c                                    the nodal coordinate system. The
c                                    description for the DOF solution above
c                                    also applies here.
c                                    ANSYS transient. (bit 28 (PDACC) in mask)

c   OSL      dp       1  nnod*Sumdof  extra solution vector for element team
c                                    rezoning project.  To be used later.

c   RF     LONG       1       nrf     Reaction force DOFs.  This index is
c                                    calculated as (N-1)*numdof+DOF, where N is
c                                    the position number of the node in the
c                                    nodal equivalence table, and DOF is the DOF
c                                    reference number.
c                                      (bit 11 (PDBR) in mask)

c * ---     dp       1       nrf     Reaction forces.  The force values are
c                                    ordered according to the DOF order shown
c                                    above in the DOF number reference table.
c                                      (bit 11 (PDBR) in mask)

c   MST    LONG       1       nmast   Master DOF list.  This index is calculated
c                                    as (N-1)*numdof+DOF, where N is the
c                                    position number of the node in the nodal
c                                    equivalence table, and DOF is the DOF
c                                    reference number.
c                                      (bit 4 in mask)

c   BC       i        1        40     Boundary condition index table.
c                                      (bit 23 (PDBBC) in mask)
c                                    numdis,ptrDIX,ptrDIS,numfor,ptrFIX,
c                                    ptrFOR,format,     0,     0,     0,
c                                         0,     0,     0,     0,     0,
c                                         0,     0,     0,     0,     0,
c                                         0,     0,     0,     0,     0,
c                                         0,     0,     0,     0,     0,
c                                         0,     0,     0,     0,     0,
c                                         0,     0,     0,     0,     0

c                                  each item is described below:
```

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
c                                       numdis - number of nodal constraints
c                                       ptrDIX - pointer to the table of nodes
c                                                having nodal constraints
c                                       ptrDIS - pointer to nodal constraint values
c                                       numfor - number of nodal input force
c                                                loadings
c                                       ptrFIX - pointer to the table of nodes
c                                                having nodal forces
c                                       ptrFOR - pointer to nodal force values
c                                       format - key (0 or 1) denoting which format
c                                                is used for DIX/FIX data (see below)

c                                       positions 7-40 are unused

c   DIX      i       1      numdis      if format == 0 --> Nodal constraint DOF.
c                                       This index is calculated as N*32+DOF,
c                                       where N is the node number and DOF is
c                                       the DOF reference number.  Values are in
c                                       the same order as the DOF number reference
c                                       table.
c
c                                       if format == 1 --> Nodal constraint node
c                                       numbers.

c   ---      i       1      numdis      if format == 0 --> does not exist.
c
c                                       if format == 1 --> Nodal constraint DOF.
c                                       Values are in the same order as the DOF
c                                       number reference table.

c   DIS     dp       1    4*numdis      Nodal constraints.  This record contains
c                                       present and previous values (real and
c                                       imaginary) of the nodal constraints at each
c                                       DOF.

c   FIX      i       1      numfor      if format == 0 --> Nodal input force DOFs.
c                                       This index is calculated as N*32+DOF,
c                                       where N is the node number and DOF is
c                                       the DOF reference number.  Values are in
c                                       the same order as the DOF number reference
c                                       table.
c
c                                       if format == 1 --> Nodal input force node
c                                       numbers.

c   ---      i       1      numfor      if format == 0 --> does not exist.
c
c                                       if format == 1 --> Nodal input force DOF.
c                                       Values are in the same order as the DOF
c                                       number reference table.

c   FOR     dp       1    4*numfor      Nodal forces.  This record contains present
c                                       and previous values (real and imaginary) of
c                                       the nodal input force loadings at each DOF.

c   TRF     dp       1    28*nnod       Two displacement result sets for transient
c                                       solution in FLOTRAN
c                                         (bit 24 (PDTRFL) in mask)

c   OND      i       1       nnod       Ordered node list.  This record exists for
c                                       a load case file only.

c   OEL      i       1       nelm       Ordered element list.  This record exists
c                                       for a load case file only.

c   DED      i       1   sizeDead       List of dead elements (EKILL). Uses the
c                                       compressed CMBLOCK format

c   ESL      i       1     2*nelm       Element solutions index table.  This record
c                                       contains pointers to each element solution.
c                                       The order of the elements is the same as
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries and affiliates.

```
c                                    the order in the element equivalence table.
c                                      (bit 12 (PDBE) in mask)
c                                    (was nelm long in 32 bit version)

c   RCM     dp       1  6*numRotCmp  Angular velocities (3) and angular
c                                    accelerations (3) of components.

c   DMI     dp       1  3+nContours  Crack ID, Contour ID, TipNode, J Integral
c                                    values

c       The solution information for each individual element is stored starting
c       at this point in the file.  The next 25 records on the file are
c       repeated as a group nelm times (once for each element).  Item nelm is
c       defined in the file header.

c   ---      i       1       25      Individual element index table.

c                                    ptrEMS,ptrENF,ptrENS,ptrENG,ptrEGR,
c                                    ptrEEL,ptrEPL,ptrECR,ptrETH,ptrEUL,
c                                    ptrEFX,ptrELF,ptrEMN,ptrECD,ptrENL,
c                                    ptrEHC,ptrEPT,ptrESF,ptrEDI,ptrETB,
c                                    ptrECT,ptrEXY,ptrEBA,ptrESV,ptrMNL
c                                   (Relative to ptrESL for 64 bit version)

c                                  each item is described below:

c                                    ptrEMS - pointer to misc. data
c                                    ptrENF - pointer to nodal forces
c                                    ptrENS - pointer to nodal stresses
c                                    ptrENG - pointer to volume and energies
c                                    ptrEGR - pointer to nodal gradients
c                                    ptrEEL - pointer to elastic strains
c                                    ptrEPL - pointer to plastic strains
c                                    ptrECR - pointer to creep strains
c                                    ptrETH - pointer to thermal strains
c                                    ptrEUL - pointer to euler angles
c                                    ptrEFX - pointer to nodal fluxes
c                                    ptrELF - pointer to local forces
c                                    ptrEMN - pointer to misc. non-sum values
c                                    ptrECD - pointer to element current
c                                             densities
c                                    ptrENL - pointer to nodal nonlinear data
c                                    ptrEHC - pointer to calculated heat
c                                             generations
c                                    ptrEPT - pointer to element temperatures
c                                    ptrESF - pointer to element surface
c                                             stresses
c                                    ptrEDI - pointer to diffusion strains
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
c                                    ptrESV - pointer to state variables
c                                    ptrMNL - pointer to material nonlinear record

c                                    Note! If ptrXXX is negative, then all
c                                    |ptrXXX| items are zero and are not on
c                                    the file.

c   EMS     dp       1    varies     Element summable miscellaneous data.  The
c                                    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
c                                    ANSYS Commands Reference.

c   ENF     dp       1    varies     Element nodal forces.  This record contains
c                                    the forces at each node, in the same DOF
c                                    order as the DOF number reference table.
c                                    For static, damping, and inertia forces, a
c                                    set of forces will be repeated (as
```

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
c                                   appropriate).  Number of data items stored
c                                   in this record can be calculated as
c                                   follows: nodfor*NDOF*M, where NDOF is the
c                                   number of DOFs/node for this element,
c                                   nodfor is the number of nodes per element
c                                   having nodal forces (defined in element
c                                   type description record), and M may be 1,
c                                   2, or 3.  For a static analysis, M=1 only.
c                                   For a transient analysis, M can be 1, 2,
c                                   or 3.

c   ENS     dp       1    varies     Element nodal component stresses.  This
c                                   record contains the stresses at each corner
c                                   node, in the order SX,SY,SZ,SXY,SYZ,SXZ.
c                                   Nodal order corresponds to the connectivity
c                                   defined in the element description.
c                                   Stresses can be nodal values extrapolated
c                                   from the integration points or values at
c                                   the integration points moved to the nodes.
c                                   If an element is nonlinear, integration
c                                   point values always will be written.  (See
c                                   item rxtrap in the solution header for the
c                                   setting.)  An element is considered
c                                   nonlinear when either plastic, creep, or
c                                   swelling strains are present.
c
c                                   Definition of common terms referred here
c                                   and in subsequent EEL, EPL, ECR, ETH,
c                                   ENL, EUL EPT, and EDI sections:
c
c                                   nodstr - number of nodes per element
c                                            having stresses, strains, etc.
c                                            For higher-order elements, nodstr
c                                            equals to the number of corner
c                                            nodes (e.g., for 20-noded SOLID186,
c                                            nodstr = 8).
c                                   nodfor - number of nodes per element
c                                            having nodal forces, etc.
c                                   ncomp  - number of solution items per node
c                                            ncomp =   6 for ENS record
c                                                      7 for EEL record
c                                                      7 for EPL record
c                                                      7 for ECR record
c                                                      8 for ETH record
c                                                     10 for ENL record
c                                                      7 for EDI record
c                                   NL     - number of layers in layered
c                                            elements
c
c                                   Note: For result sets with NoPrin=0,
c                                   the ENS record will have ncomp=11 and include
c                                   the principal stresses S1,S2,S3,SI,SIGE.
c
c                                   * For solid elements or layered solid elements
c                                     with KEYOPT(8)=0, the record contains
c                                     stresses at each corner node, and
c                                     the number of items in this record is
c                                     nodstr*ncomp.
c                                   * For shell elements or layered shell elements
c                                     with KEYOPT(8)=0, the record contains
c                                     stresses at each corner node (first
c                                     at the bottom shell surface, then the top
c                                     surface), and the number of items in this
c                                     record is 2*nodstr*ncomp.
c                                   * For layered elements SHELL91, SHELL99,
c                                     SOLID46, and SOLID191 with KEYOPT(8) = 0,
c                                     if failure criteria were used, the record
c                                     contains additional stresses at each corner
c                                     nodes (first the bottom surface, then the
c                                     top surface) of the layer with the largest
c                                     failure criteria. Therefore, the total number
c                                     of items is 4*nodstr*ncomp for SHELL91 and
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries and affiliates.

```
c                                       SHELL99, and 2*nodstr*ncomp for SOLID46 and
c                                       SOLID191.
c                                     * For layered elements (with KEYOPT(8)=1),
c                                       stresses for each layer are at each
c                                       corner node (first at the bottom surface, then
c                                       at the top surface), and the number of
c                                       items in this record is NL*2*nodstr*ncomp for
c                                       layered shells and NL*nodstr*ncomp for
c                                       layered solid elements.
c                                     * For layered shell elements with KEYOPT(8)=2,
c                                       the record contains stresses for each layer
c                                       at each corner node  (first at the bottom
c                                       surface, then the top, and finally the middle
c                                       surface). Therefore, the number of items
c                                       in this record is NL*3*nodstr*ncomp.
c                                     * For layered membrane elements (SHELL181,
c                                       SHELL281, SHELL208, and SHELL209 with
c                                       KEYOPT(1)=1 and KEYOPT(8)=1), the record
c                                       contains stresses for each layer at each
c                                       corner node, and the number of items in
c                                       this record  is NL*nodstr*ncomp.
c                                     * For beam elements, the contents and number
c                                       of data items is element-dependent.  See
c                                       the Output Data section for the particular
c                                       element in the ANSYS Elements Reference.
c
c   ENG     dp       1      11       Element volume and energies.

c                                     volume,senergy,aenergy,kenergy,coenergy,
c                                     incenergy,0.0,0.0,thenergy,0.0,0.0

c                                    each item is described below:

c                                     volume   - element volume
c                                     senergy  - element energy associated with
c                                                the stiffness matrix
c                                     aenergy  - artificial hourglass energy
c                                     kenergy  - kinetic energy
c                                     coenergy - co-energy (magnetics)
c                                     incenergy- incremental energy (magnetics)
c                                     0.0      - position not used
c                                     0.0      - position not used
c                                     thenergy - thermal dissipation energy
c                                                (see ThermMat, shell131/132 only)
c                                     0.0      - position not used
c                                     0.0      - position not used

c   EGR     dp       1    varies     Element nodal field gradients.  This record
c                                     contains the gradients at each corner node
c                                     in the order X,Y,Z.  Nodal order
c                                     corresponds to the connectivity defined in
c                                     the element description.  If this is a
c                                     coupled-field analysis, the data is stored
c                                     in the following order (as available):
c                                     fluid, thermal (TEMP), electric (VOLT),
c                                     magnetic (AZ), and diffusion (CONC).
c                                     Gradients can be nodal values extrapolated
c                                     from the integration points or values at the
c                                     integration points moved to the nodes.  See
c                                     item rxtrap in the solution header for the
c                                     setting.  The number of items in this record
c                                     is nodstr*3*N, where N can be 1, 2, 3, or 4
c                                     (depending on the coupled-field
c                                     conditions).

c                                     NOTE: nodstr is defined in the element type
c                                     description record.

c   EEL     dp       1    varies     Element nodal component elastic strains.
c                                     This record contains strains in the order
c                                     X,Y,Z,XY,YZ,XZ,EQV. Elastic strains can be
c                                     can be nodal values extrapolated from the
```

```
c                                       integration points or values at the
c                                       integration points moved to the nodes.  If
c                                       an element is nonlinear, integration point
c                                       values always will be written.  See item
c                                       rxtrap in the solution header for the
c                                       setting.  An element is considered
c                                       nonlinear when either plastic, creep, or
c                                       swelling strains are present.  For beam
c                                       elements, see item LEPEL in the description
c                                       in the Output Data section for the
c                                       particular element in the ANSYS Elements
c                                       Reference.
c
c                                       NOTE: See ENS record section for more details
c                                       on record content and length.
c

c   EPL     dp      1    varies     Element nodal component plastic strains.
c                                   This record contains strains in the order
c                                   X,Y,Z,XY,YZ,XZ,EQV.
c                                   Plastic strains are always values at the
c                                   integration points moved to the nodes.  For
c                                   beam elements, see item LEPPL in the
c                                   Output Data section for the particular
c                                   element in the ANSYS Elements Reference.
c
c                                   NOTE: See ENS record section for more details
c                                   on record content and length.
c

c   ECR     dp      1    varies     Element nodal component creep strains.
c                                   This record contains strains in the order
c                                   X,Y,Z,XY,YZ,XZ,EQV.
c                                   Creep strains are always values at the
c                                   integration points moved to the nodes.  For
c                                   beam elements, see item LEPCR in the
c                                   Output Data section for the particular
c                                   element in the ANSYS Elements Reference.
c
c                                   NOTE: See ENS record section for more details
c                                   on record content and length.
c

c   ETH     dp      1    varies     Element nodal component thermal strains.
c                                   This record contains strains in the order
c                                   X,Y,Z,XY,YZ,XZ,EQV plus the element
c                                   swelling strain. Thermal
c                                   strains can be nodal values extrapolated
c                                   from the integration points or values at
c                                   the integration points moved to the nodes.
c                                   If the element in nonlinear, integration
c                                   point data always will be written. (An
c                                   element is considered nonlinear when either
c                                   plastic, creep, or swelling strains are
c                                   present.) See item rxtrap in the solution
c                                   header for the setting.  For beam elements,
c                                   see item LEPTH in the description of the
c                                   Output Data section for the particular
c                                   element in the ANSYS Elements Reference.
c
c                                   NOTE: See ENS record section for more details
c                                   on record content and length.
c

c   EUL     dp      1    varies     Element Euler angles.  This record contains
c                                   the Euler angles (THXY,THYZ,THZX). (No
c                                   attempt is made to make this information
c                                   complete for all cases of all element
c                                   types.  Programmers need to verify their
c                                   situations.)
c
c                                   ** FOR OLDER ELEMENTS **
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries and affiliates.

```
c                                    --For lower-order elements
c                                      (e.g. PLANE42 and SOLID45), angles are
c                                      at the centroid and the number of items
c                                      in this record is 3.
c                                    --For higher-order elements (e.g.
c                                      PLANE82, SOLID92, and SOLID95), angles
c                                      are at each corner node and the number of
c                                      items in this record is nodstr*3.
c                                      The above two categories are output if
c                                      ESYS is used, a material KEYOPT is used
c                                      (e.g. KEYOPT(1) for PLANE42), or if it is
c                                      from a large deflection superelement
c                                      stress pass
c
c                                    ** FOR NEW GENERATION SHELL ELEMENTS **
c                                    --For SHELL181/SHELL281:
c                                       For real constant input: the number of
c                                         items in this record is 12.
c                                       For section input:
c                                        IF KEYOPT(8) > 0:
c                                         the number of items in this record is
c                                                          12 + number of layers
c                                        IF KEYOPT(8) = 0:
c                                          IF regular shell  (KEYOPT(1) = 0)
c                                             the number of items in this record
c                                              is 14
c                                          IF membrane shell (KEYOPT(1) = 1):
c                                            IF number of layers = 1, the number
c                                              of items in this record is 13
c                                            IF number of layers > 1, the number
c                                              of items in this record is 14
c
c                                    ** FOR THE NEW GENERATION SOLID ELEMENTS **
c                                    --For uniform reduced integration lower-order
c                                      elements (e.g. PLANE182, KEYOPT(1)=1 and
c                                      SOLID185 KEYOPT(2)=1):
c                                      the angles are at the centroid and the number
c                                      of items is 3.
c                                    --For other formulations of lower-order
c                                      elements (e.g. PLANE182 and SOLID185) and
c                                      the higher-order elements
c                                      (e.g. PLANE183, SOLID186, and SOLID187):
c                                      The number of items in this record is
c                                      (nodstr*3).
c                                    --For layered solid elements, add NL values,
c                                      so that the number of items in this record
c                                      is (nodstr*3)+NL.
c
c                                    NOTE: See ENS record section for definition of
c                                          terms NL and nodstr.

c   EFX     dp       1    varies     Element nodal field fluxes.  This record
c                                    contains the fluxes at each corner node in
c                                    the order X,Y,Z. If this is a
c                                    coupled-field analysis, the flux data is
c                                    stored in the following order:  thermal,
c                                    electric, magnetic.  Nodal order
c                                    corresponds to the connectivity defined in
c                                    the element description.  Fluxes can be
c                                    nodal values extrapolated from the
c                                    integration points or values at the
c                                    integration points moved to the nodes.
c                                    See item rxtrap in the solution header for
c                                    the setting.  The number of items in this
c                                    record is nodstr*3*N, where N can be 1, 2,
c                                    or 3 depending on the coupled-field
c                                    conditions.

c                                    NOTE: nodstr is defined in the element type
c                                    description record.

c * ELF     dp       1    varies     Element nodal coupled-field forces.  This
```

```
c                                       record lists the forces at each node in the
c                                       order X,Y,Z.  For most elements, the number
c                                       of items in this record is nodfor*3.
c                                       However, for the PLANE53 element, the
c                                       number of items in this record is either
c                                       nodfor*3 or nodstr*3.  (See the description
c                                       of KEYOPT(7) for PLANE53 in the ANSYS
c                                       Elements Reference.)  NOTE: nodfor and
c                                       nodstr are defined in the element type
c                                       description record.

c                                       NOTE: nodstr is defined in the element type
c                                       description record.

c   EMN     dp       1    varies     Element nonsummable miscellaneous data.
c                                       The contents and number data items for this
c                                       record is element-dependent.   See the
c                                       description for item NMISC of the ETABLE
c                                       command in the ANSYS Commands Reference.

c * ECD     dp       1       3       Element current densities.  This record
c                                       contains the calculated current densities
c                                       in the order X,Y,Z.

c   ENL     dp       1    varies     Element nodal nonlinear data.  This record
c                                       stores nonlinear data at each corner node
c                                       in the order SEPL, SRAT, HPRES, EPEQ,
c                                       PSV or CREQ, PLWK, CRWK, and ELENG
c                                       followed by 2 spares.

c                                     each item is described below:
c                                       SEPL  - equivalent stress parameter
c                                       SRAT  - stress ratio
c                                       HPRES - hydrostatic pressure
c                                       EPEQ  - accumulated equivalent plastic
c                                               strain
c                                       PSV   - plastic state variable
c                                       CREQ  - accumulated equivalent creep
c                                               strain.  Applies to current
c                                               technology element types
c                                               180,181,182,183,185,186,
c                                               187,188,189,208,209,265,
c                                               281,288,289,290
c                                       PLWK  - plastic strain energy density(work)
c                                       CRWK  - creep strain energy density (work)
c                                       ELENG - elestic strain energy density

c                                       * See ENS record section for details on
c                                         solid and shell elements.
c                                       * For beam elements, the contents and
c                                         number of data items in this record is
c                                         element-dependent.  See the description
c                                         of item NLIN in the Output Data section
c                                         for the particular element in the ANSYS
c                                         Elements Reference.

c * EHC     dp       1       1       Element heat generation.  This record
c                                       stores the calculated heat generation.

c   EPT     dp       1    varies     Element structural nodal temperatures.

c                                       * For solid elements and SHELL41, the
c                                         record contains nodal temperatures at
c                                         each node and the number of items in this
c                                         record is nodfor.
c                                       * For shell elements, except SHELL41 and
c                                         SHELL91, the record contains nodal
c                                         temperatures at each corner node for the
c                                         top surface and the bottom surface.  The
c                                         number of items in this record is
c                                         nodstr*2.
c                                       * For SHELL91 and SOLID191, the record
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries and affiliates.

```
c                                       contains nodal temperatures at each
c                                       corner node for the bottom of the bottom
c                                       layer, and each succeeding interlayer
c                                       surface up to the top of the top layer.
c                                       The number of items in this record is
c                                       (NL+1)*nodstr.
c                                     * For layered shell elements SHELL181,
c                                       SHELL281, SHELL208, SHELL209, and layered
c                                       solid elements SOLID185, SOLID186,
c                                       and SOLSH190, the record contains
c                                       temperatures for each layer at each
c                                       corner node (first at the bottom layer
c                                       surface, then the top). Therefore, the number
c                                       of items in this record is NL*2*nodstr for
c                                       layered shells and NL*nodstr for layered
c                                       solid elements.
c                                     * For layered membrane elements (SHELL181,
c                                       SHELL281, SHELL208, and SHELL209 with
c                                       KEYOPT(1)=1), the record contains
c                                       temperatures for each layer at each
c                                       corner node. Therefore, the number of items
c                                       in this record is NL*nodstr.
c                                     * For beam elements, the contents and
c                                       number of data items in this record is
c                                       element-dependent.  See the description
c                                       of item LBFE in the Output Data section
c                                       for the particular element in the ANSYS
c                                       Elements Reference.

c                                     NOTE: See ENS record section for definition
c                                           of terms NL, nodstr, and nodfor.

c   ECT     dp       1    varies      Contact element results.  This record
c                                     stores contact results at each corner node
c                                     in the order STAT, PENE, PRES, SFRIC, STOT,
c                                     SLIDE, GAP, FLUX, CNOS, FPRS
c                                   each item is described below:
c                                     STAT  - Contact status
c                                     PENE  - Contact penetration
c                                     PRES  - Contact pressure
c                                     SFRIC - Contact friction stress
c                                     STOT  - Contact total stress (pressure plus friction)
c                                     SLIDE - Contact sliding distance
c                                     GAP   - Contact gap distance
c                                     FLUX  - Total heat flux at contact surface
c                                     CNOS  - Total number of contact status changes during substep
c                                     FPRS  - Actual applied fluid penetration pressure

c   ESF     dp       1    nsurf*19    Element surface stresses.  The
c                                     length of this record is nsurf*19 where
c                                     nsurf is the number of surfaces that have
c                                     surface stress information.  The stress
c                                     information is simply repeated in the
c                                     format shown below for each surface.

c                                     * For 2d elements:

c                                     facenm,  area,  temp, press, eppar,
c                                      epper,   epz, 0.0d0,  spar,  sper,
c                                         sz, 0.0d0, 0.0d0, 0.0d0,    s1,
c                                         s2,    s3,  sint,  seqv

c                                     * For 3d elements:

c                                     facenm,  area,  temp, press,   epx,
c                                        epy,   epz,  epxy,    sx,    sy,
c                                         sz,   sxy, 0.0d0, 0.0d0,    s1,
c                                         s2,    s3,  sint,  seqv

c                                     * For axisymmetric elements:

c                                     facenm,  area,  temp, press, eppar,
```

```
c                                    epper,   epz,  epsh,  spar,  sper,
c                                       sz, 0.0d0, 0.0d0,   ssh,    s1,
c                                       s2,    s3,  sint,  seqv

c                                  each item is described below:

c                                    facenm - face number
c                                    area   - face area
c                                    temp   - face temperature
c                                    press  - face pressure
c                                    epx    - strain parallel to face
c                                    epy    - strain parallel to face
c                                    epz    - strain perpendicular to face
c                                    epxy   - shear strain
c                                    eppar  - strain parallel to face
c                                    epper  - strain perpendicular to face
c                                    epsh   - torsion shear strain
c                                    sx     - stress parallel to face
c                                    sy     - stress parallel to face
c                                    sz     - stress perpendicular to face
c                                    sxy    - shear stress
c                                    spar   - stress parallel to face
c                                    sper   - stress perpendicular to face
c                                    ssh    - torsion shear stress
c                                    s1     - S(1)
c                                    s2     - S(2)
c                                    s3     - S(3)
c                                    sint   - S(INT)
c                                    seqv   - S(EQV)
c                                    0.0d0  - position not used

c   EDI     dp       1    varies     Element nodal component diffusion strains.
c                                    This record contains strains in the order
c                                    X,Y,Z,XY,YZ,XZ,EQV. Diffusion
c                                    strains can be nodal values extrapolated
c                                    from the integration points or values at
c                                    the integration points moved to the nodes.
c                                    See item rxtrap in the solution header for
c                                    the setting.
c
c                                    NOTE: See ENS record section for more details
c                                    on record content and length.
c

c   EXY     dp       1    varies     Element integration point coordinates
c                                    The length of the record is numint*3, where
c                                    numint is the number of integration points.
c                                    Even two-dimensional elements use the 3.
c                                    They are output only if requested with the
c                                    OUTRES,loci command.
c                                    Applicable only to legacy element types
c                                    2,42,45,82,92,95, and current technology
c                                    element types 180,181,182,183,185,186,187,
c                                    188,189,208,209,265,281,288,289,290

c   EBA     dp       1    varies     Element structural nodal back stresses
c                                    Record has the same form as the plastic
c                                    strains.  They are output if the form of
c                                    plasticity is kinematic hardening and the
c                                    plastic strains are requested.
c                                    Applicable only to legacy element types
c                                    2,42,45,82,92,95, and current technology
c                                    element types 180,181,182,183,185,186,187,
c                                    188,189,208,209,265,281,288,289,290

c   ESV     dp       1    varies     Element state variable record.  Exists only
c                                    if written by user in usermat or usercreep.

c   MNL     dp       1    varies     Material nonlinear record.

c   records marked with * to the left of the record id can be read and stored
c   into database with "ldread" command.
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries and affiliates.

```
c *** Nodal Component Stresses (unused)

c  NDSTR    dp       1    6*nnod     Nodal component stresses (TOP for shells)
c                                     (nNodStr > 0)
c           dp       1    6*nnod     BOT nodal component stresses for shells
c                                     (nNodStr > 1)
c           dp       1    6*nnod     MID nodal component stresses for shells
c                                     (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

```
*comdeck,fdrdsp

c *** Copyright ANSYS.  All Rights Reserved.
c *** ansys, inc


c     **********  description of reduced displacement file  **********
      character*8  RDSPNM
      parameter  (RDSPNM='rdsp    ')

      integer         RDSPHDLEN
      parameter      (RDSPHDLEN=80)

      LONGINT         rdspfpL, rdspfp
      integer         rdspbk,  rdsput
      common /fdrdsp/ rdspfpL, rdspbk, rdsput
      equivalence  (rdspfp,rdspfpL)

c  write:  lnfrcl,lnfrin,lnfrwr
c  write:  rdtrcl,rdtrin,rdtrwr
c  read:   rdtrs

c   ********** common variable descriptions ***********
co  rdspfpL      file position on file rdsp
co  rdspbk       block number for file rdsp
co  rdsput       file unit for file rdsp

c   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
c             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             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    type    nrec    lrec     contents

c  ---      i       1      100      standard ANSYS file header (see binhed for
c                                   details of header contents)

c  ---      i       1       80      .RDSP FILE HEADER
c
c                                fun10,    nmrow,   nmatrx,    nmode,   numdof,
c                                 maxn,    wfmax,   lenbac,    ngaps,   ncumit, (10)
c                                  kan,     nres,     ndva,    nvect,   DSPfmt,
c                               minmod,        0,  modlstp, ndefdval,  nEnfDof, (20)
c                               ptrDOF,   ptrDNC,   ptrSTF,   ptrMAS,   ptrDMP,
c                               ptrFRQ,   ptrDSP,  ptrSTFh,  ptrMASh,  ptrDMPh, (30)
c                             ptrFRQh,  ptrDSPh,   ptrDVA,  ptrDVAh,nrkeyPert,
c                            kPerturb,    keyVA,Glblenbac,        0,        0, (40)
c                                   0,        0,        0,        0,        0,
c                                   0,        0,        0,        0,        0, (50)
c                                   0,        0,        0,        0,        0,
c                                   0,        0,        0,        0,        0, (60)
c                                   0,        0,        0,        0,        0,
c                                   0,        0,        0,        0,        0, (70)
c                                   0,        0,        0,        0,        0,
c                                   0,        0,        0,        0,        0  (80)

c                                 each item in header is described below:

c                                   fun10  - unit number (rdsp file is 10)
c                                   nmrow  - number of rows/columns in matrices
c                                   nmatrx - number of reduced matrices on the
c                                            file
c                                   nmode  - number of modes extracted during
c                                            modal analysis (or nmrow if reduced
c                                            method)
c                                   numdof - number of dofs per node
c                                   maxn   - maximum node number
c                                   wfmax  - maximum wavefront
c                                   lenbac - number of nodes
c                                   ngaps  - number of gaps
c                                   ncumit - total number of iterations done
c                                            during analysis
c                                   kan    - analysis type
c                                            = 5 for reduced transient analysis
c                                   nres   - number of residual vectors used
c                                   modlstp- multiple load step key
c                                   ndva   - length of DVA (for restart)
c                                   nvect  - number of available load vectors
c                                   DSPfmt - 0,physical disps  .ne.0,modal coords
c                                   minmod - smallest mode number used
c                                   modlstp- multiple load step key
c                                   ndefdval- number of defined enforced motion
c                                   maxEnf - maximum enforced values
c                                   ptrDOF - pointer to degree of freedom set
c                                   ptrDNC - pointer to nodal constraints
c                                   ptrSTF - pointer to the reduced stiffness
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 step solution data:
c                                               - calculated displacements
c                                               - load vector scale factors
c                                               - gap restoring forces
c                                               - calculated velocities
c                                               - calculated accelerations
c                                   ptrSTFh- High part of reduced stiffness ptr
c                                   ptrMASh- High part of reduced mass ptr
c                                   ptrDMPh- High part of reduced damping ptr
c                                   ptrFRQh- High part of frequency ptr
c                                   ptrDSPh- High part of displacement ptr
c                                   ptrDVA - pointer to modal disp, velo and acc (for restart)
c                                   ptrDVAh- High part of modal disp, velo and acc (for restart)
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries and affiliates.

```
c                                  nrkeyPert - nrkey setting of base analysis (Linear Pertubation)
c                                  kPerturb - Linear Perturbation key
c                                  keyVA  - Key for velocities/accelerations on file
c                                  Glblenbac - global number of nodes (== 0 unless using Distributed Ansys)
c                                  0      - position not used

c  ---      i       1     numdof   Degrees of freedom per node
c                                   (curdof(i),i=1,numdof)
c                                  dof reference numbers are:
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  ****** 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
c                                   the number used for storage to the actual
c                                   node number
c                                  (Back(i),i=1,lenbac)

c  ---      i       1  Glblenbac  Global nodal equivalence table.  This
c                                   table equates the number used for storage
c                                   to the actual node number.  Only written
c                                   by the master process in Distributed Ansys
c                                  (GlbBack(i),i=1,Glblenbac)

c  ---      dp      1       10    Time information:
c                                  dtime,    0.0,    0.0,    0.0,    0.0,
c                                    0.0,    0.0,    0.0,    0.0, timend

c                                each item is described below:
c
c                                  dtime  - the time increment
c                                  timend - the final time of the analysis
c                                  0.0    - position not used

c  DOF      i       1     nmrow    Degree of freedom set used
c                                  The DOFs are calculated as (N-1)*numdof+DOF,
c                                  where N is the position number of the node in
c                                  the nodal equivalence table and DOF is the
c                                  DOF reference number given above.

c                                  If the analysis uses the reduced method, the
c                                  original DOF order (see next record) is
c                                  rearranged so that DOFs having nodal
c                                  constraints are listed first.
c
c                                  If the analysis uses the mode superposition
c                                  method (using the reduced mode extraction
c                                  technique), the DOF order is the same as the
c                                  original order (see next record).
c                                   (l(i),i=1,nmrow)

c  ---      i       1     nmrow+1  Original reduced set of DOFs used.
c                                  The DOFs are calculated as (N-1)*numdof+DOF,
c                                  where N is the position number of the node in
c                                  the nodal equivalence table and DOF is the
c                                  DOF reference number given above.

c                                  If the analysis uses the reduced method, the
c                                  original DOF order, plus the number of nodal
c                                  constraints (nbcdsp), is stored.

c                                  If the analysis uses the mode superposition
c                                  method (using the reduced mode extraction
c                                  technique), this record matches the previous
c                                  record.  The nmrow+1 entry will be zero.
c                                   (lorig(i),i=1,nmrow),nbcdsp

c  DNC      i       1     nbcdsp   This record is present only if the analysis
c                                  uses the reduced method and nbcdsp > 0 (see
c                                  record at ptrDOF). These numbers are the
c                                  positions in the previous record of dofs with
```

```
c                                   a nodal constraint. These are nodal
c                                   constraints only on nodes that also are
c                                   masters.
c                                    (na(i),i=1,nbcdsp)

c  STF      dp    nmrow   nmrow    Reduced stiffness matrix. Each row of the
c                                   matrix is stored as a record. The matrix is
c                                   present only if nmatrx > 0 and analysis is
c                                   not using mode superposition method Row
c                                   order is the same as the DOF order in record
c                                   at ptrDOF.
c                                    (ak(i,j),i=1,nmrow)

c  MAS      dp    nmrow   nmrow    Reduced mass matrix. Each row of the matrix
c                                   is stored as a record. The matrix is present
c                                   only if nmatrx > 1 and analysis is not using
c                                   mode superposition method. Row order is the same
c                                   as the DOF order in record at ptrDOF.
c                                    (am(i,j),i=1,nmrow)

c  DMP      dp    varies  varies   Reduced damping matrix or mode shapes.

c                                   If the analysis uses the reduced method,
c                                   each record will be nmrow items in length.
c                                   The reduced damping matrix is present only
c                                   if nmatrx > 2. There will be nmrow records of
c                                   this type stored here. Row order is the same
c                                   as the DOF order in record at ptrDOF.

c                                   If the analysis uses the mode superposition
c                                   method (using the reduced mode extraction
c                                   technique), each record will be nmode items
c                                   in length. These records contain mode shapes
c                                   (eigenvectors) of the frequencies
c                                   (eigenvalues) actually used in the harmonic
c                                   analysis. There will be nmode records of this
c                                   type stored here, with the first N records
c                                   containing the mode shapes and the other
c                                   records containing zeros, where N is the
c                                   number of modes actually used in the harmonic
c                                   analysis. Order corresponds to the DOF order
c                                   given in record at ptrDOF.

c                                   If the analysis uses the mode superposition
c                                   method, this record will not be present.
c                                     (psi(i,j),i=1,nmrow) (or ac)

c  FRQ      dp      1     nmrow    Frequencies extracted from the modal analysis.
c                                   This record is present only for analyses using
c                                   the mode superposition method.
c                                   The first nmode values are the frequencies
c                                   extracted from the modal analysis.  The
c                                   remaining values have no meaning.
c                                    (freq(i),i=1,nmrow)

c  *** The next 6 to 7 records are repeated (as a group) until the time value
c  *** equals the value of timend.  The number of iterations is stored as
c  *** ncumit. (see above records that deal with time)

c  DSP      dp      1     nmrow+6  Calculated displacements
c                                   The first nmrow entries are the displacements
c                                   in the same order as the original set of DOFs
c                                   (see record AFTER ptrDOF). If DSPfmt=0, these
c                                   are physical displacements, If DSPfmt!=0,
c                                   these are the nmode modal coordinates.
c
c                                   For the last six entries:
c                                    1. Time for these displacements
c                                    2. Load step number
c                                    3. Substep number
c                                    4. Cumulative iteration number
c                                    5. Scale factor (zero if the analysis uses
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries and affiliates.

```
c                                      the reduced method).
c                                    6. numdeflvs - number of scale factors
c
c  ---      i       1              Note: If, upon reading of this record, there
c                                  is less than nmrow+5 items in the record,
c                                  then only a selected set of nodes were
c                                  output. Another record follows (integer, less
c                                  than lenbac long) which contains the list of
c                                  nodes for which DOF solutions are available.

c  ---      i       1 numdeflvs    lvscal table scale factor IDs
c                                   (ilvscID(i),i=1,numdeflvs)

c  ---      dp      1 numdeflvs    lvscal table scale factor values
c                                   (dlvscVal(i),i=1,numdeflvs)

c  ---      dp      1     ngaps    Gap restoring forces. The order of these
c                                  forces corresponds to the node position order
c                                  given in record at ptrDNC. This record is
c                                  present only if ngaps > 0.
c                                   (fgaps(i),i=1,ngaps)

c  ---      dp      1     nmrow    Calculated velocities (present if keyVA = 1)

c  ---      dp      1     nmrow    Calculated accelerations (present if keyVA = 1)

c *** The next 3 records are kept for possible restart in mode superposition
c *** transient. They are overwritten upon restarting. They are written once (last
c *** loadstep).

c  DVA      dp      1     ndva+6  Calculated modal displacements
c                                   The first ndva entries are the modal
c                                   displacements. For the last six entries:
c                                    1. Time for these displacements
c                                    2. Load step number
c                                    3. Substep number
c                                    4. Cumulative iteration number
c                                    5. Scale factor (zero if the analysis uses
c                                       the reduced method).
c                                    6. numdeflvs - number of scale factors

c  ---      dp      1     ndva    Calculated modal velocities

c  ---      dp      1     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

```
*comdeck,fdrfrq

c *** Copyright ANSYS.  All Rights Reserved.
c *** ansys, inc


c     **********  description of reduced complex displacement file  **********
      character*8  RFRQNM
      parameter  (RFRQNM='rfrq    ')
```

```
      LONGINT         rfrqfpL, rfrqfp
      integer         rfrqbk,  rfrqut
      common /fdrfrq/ rfrqfpL, rfrqbk, rfrqut
      equivalence  (rfrqfp,rfrqfpL)

c   ********** common variable descriptions ***********
co  rfrqfpL      file position on file rfrq
co  rfrqbk       block number for file rfrq
co  rfrqut       file unit for file rfrq

c   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
c             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             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    type    nrec    lrec     contents

c  ---      i       1      100      standard ANSYS file header (see binhed for
c                                   details of header contents)

c  ---      i       1       40    .RFRQ FILE HEADER
c
c                                     fun10,    nmrow,   nmatrx,    nmode,   numdof,
c                                      maxn,    wfmax,   lenbac,   extopt,   ncumit, (10)
c                                        kan,     nres,   nmUsed,    nvect,   DSPfmt,
c                                    minmod,        0,  modlstp,        0,  nEnfdof, (20)
c                                    ptrDOF,   ptrDNC,   ptrSTF,   ptrMAS,   ptrDMP,
c                                    ptrFRQ,   ptrDSP,  ptrSTFh,  ptrMASh,  ptrDMPh, (30)
c                                   ptrFRQh,  ptrDSPh,nrkeyPert,  kPertrb,Glblenbac,
c                                    cpxmod,   SvCode,        0,        0,        0  (40)

c                                  each item in header is described below:

c                                    fun10  - unit number (rfrq file is 10)
c                                    nmrow  - number of rows/columns in matrices
c                                    nmatrx - number of reduced matrices on file
c                                    nmode  - number of modes extracted during
c                                             modal analysis (or nmrow if reduced
c                                             method)
c                                    numdof - number of dofs per node
c                                    maxn   - maximum node number
c                                    wfmax  - maximum wavefront
c                                    lenbac - number of nodes
c                                    extopt - mode extraction method
c                                              = 0 - reduced
c                                              = 3 - unsymmetric Lanczos
c                                              = 4 - damped Lanczos
c                                              = 6 - block Lanczos
c                                              = 8 - SuperNode
c                                              = 9 - PCG Lanczos
c                                    ncumit - total number of iterations done
c                                             during analysis
c                                    kan    - analysis type
c                                             = 6  - reduced harmonic
c                                    nmUsed - number of modes used in mode
c                                             superposition
c                                    nvect  - number of generated loads in .mlv
c                                    DSPfmt - 0,physical disps  .ne.0,modal coords
c                                    minmod - smallest mode number used
c                                    modlstp- multiple load step key
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries and affiliates.

```
c                                  maxEnf - maximum enforced values
c                                  ndval  - defined enforced values
c                                  ptrDOF - pointer to degree of freedom set
c                                           used in model
c                                  ptrDNC - pointer to nodal constraints
c                                  ptrSTF - pointer to the reduced stiffness
c                                           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
c                                           displacements
c                                  ptrSTFh- High part of STF pointer
c                                  ptrMASh- High part of MAS pointer
c                                  ptrDMPh- High part of DMP pointer
c                                  ptrFRQh- High part of FRQ pointer
c                                  ptrDSPh- High part of DSP pointer
c                                  nrkeyPert - nrkey setting of base analysis (Linear Pertubation)
c                                  kPertrb- Linear Perturbation key
c                                  Glblenbac - global number of nodes (== 0 unless using Distributed Ansys)
c                                  cpxmod   - key if complex modes were used
c                                  SvCode - Solver assembly code
c                                           = 0 Frontal assembly (SV_ANSYS)
c                                           = 1 Symbolic assembly (SV_CASI)
c                                  0      - position not used

c  ---      i       1     numdof   Degrees of freedom per node
c                                   (curdof(i),i=1,numdof)
c                                  dof reference numbers are:
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  ****** 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
c                                   the number used for storage to the actual
c                                   node number
c                                  (Back(i),i=1,lenbac)

c  ---      i       1  Glblenbac  Global nodal equivalence table.  This
c                                   table equates the number used for storage
c                                   to the actual node number.  Only written
c                                   by the master process in Distributed Ansys
c                                  (GlbBack(i),i=1,Glblenbac)

c  ---      dp      1       10    Unused record. contents:
c                                 1.0, 0.0, 0.0, 0.0, 0.0,
c                                 0.0, 0.0, 0.0, 0.0, 0.0

c  DOF      i       1     nmrow    Degree of freedom set used
c                                   The DOFs are calculated as (N-1)*numdof+DOF,
c                                   where N is the position number of the node in
c                                   the nodal equivalence table and DOF is the
c                                   DOF reference number given above.

c                                   If the analysis uses the reduced method, the
c                                   original DOF order (see next record) is
c                                   rearranged so that DOFs having nodal
c                                   constraints are listed first.
c
c                                   If the analysis uses the mode superposition
c                                   method (using the reduced mode extraction
c                                   technique), the DOF order is the same as the
c                                   original order (see next record).
c                                    (l(i),i=1,nmrow)

c  ---      i       1     nmrow+1  Original reduced set of DOFs used.
c                                   The DOFs are calculated as (N-1)*numdof+DOF,
c                                   where N is the position number of the node in
c                                   the nodal equivalence table and DOF is the
c                                   DOF reference number given above.
```

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
c                                   If the analysis uses the reduced method, the
c                                   original DOF order, plus the number of nodal
c                                   constraints (nbcdsp), is stored.

c                                   If the analysis uses the mode superposition
c                                   method (using the reduced mode extraction
c                                   technique), this record matches the previous
c                                   record.  The nmrow+1 entry will be zero.
c                                    (lorig(i),i=1,nmrow),nbcdsp

c  DNC      i       1     nbcdsp   This record is present only if the analysis
c                                   uses the reduced method and nbcdsp > 0 (see
c                                   record at ptrDOF). These numbers are the
c                                   positions in the previous record of dofs with
c                                   a nodal constraint. These are nodal
c                                   constraints only on nodes that also are
c                                   masters.
c                                    (na(i),i=1,nbcdsp)

c  STF      dp    nmrow   nmrow    Reduced stiffness matrix. Each row of the
c                                   matrix is stored as a record. The matrix is
c                                   present only if nmatrx > 0 and analysis is
c                                   not using mode superposition method Row
c                                   order is the same as the DOF order in record
c                                   at ptrDOF.
c                                    (ak(i,j),i=1,nmrow)

c  MAS      dp    nmrow   nmrow    Reduced mass matrix. Each row of the matrix
c                                   is stored as a record. The matrix is present
c                                   only if nmatrx > 1 and analysis is not using
c                                   mode superposition method. Row order is the same
c                                   as the DOF order in record at ptrDOF.
c                                    (am(i,j),i=1,nmrow)

c  DMP      dp    varies  varies   Reduced damping matrix or mode shapes.

c                                   If the analysis uses the reduced method,
c                                   each record will be nmrow items in length.
c                                   The reduced damping matrix is present only
c                                   if nmatrx > 2. There will be nmrow records of
c                                   this type stored here. Row order is the same
c                                   as the DOF order in record at ptrDOF.

c                                   If the analysis uses the mode superposition
c                                   method (using the reduced mode extraction
c                                   technique), each record will be nmode items
c                                   in length. These records contain mode shapes
c                                   (eigenvectors) of the frequencies
c                                   (eigenvalues) actually used in the harmonic
c                                   analysis. There will be nmode records of this
c                                   type stored here, with the first N records
c                                   containing the mode shapes and the other
c                                   records containing zeros, where N is the
c                                   number of modes actually used in the harmonic
c                                   analysis. Order corresponds to the DOF order
c                                   given in record at ptrDOF.

c                                   If the analysis uses the mode superposition
c                                   method, this record will not be present.
c                                     (psi(i,j),i=1,nmrow) (or ac)

c  FRQ      dp      1     nmrow    Frequencies extracted from the modal analysis.
c                                   This record is present only for analyses using
c                                   the mode superposition method. Frequencies are
c                                   complex if extopt=3 or 4.
c                                    (freq(i),i=1,nmrow)

c  *** The next 3 records are repeated (as a pair)
c  *** The number of iterations is stored as ncumit.

c  DSP      cmp  ncumit  nmrow+5   Calculated complex displacements
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries and affiliates.

```
c                                  The first nmrow entries are the displacements
c                                  in the same order as the original set of DOFs
c                                  (see record AFTER ptrDOF). If DSPfmt=0, these
c                                  are physical displacements, If DSPfmt!=0,

c                                  For the last five entries:
c                                     Real part             Imag part
c                                  1. frequency for these   frequency increment
c                                     values
c                                  2. load step number      substep number
c                                  3. cumulative iteration  zero
c                                     number
c                                  4. zero                  zero
c                                  5. scale factor          zero
c                                     (zero if the
c                                     analysis uses the
c                                     reduced method)
c                                         or
c                                  5. numdeflvs             zero
c                                       (cvs(i),i=1,nmrow),(freq,delf),
c                                       (itime,itter),(ncumit,0.0),(0.0,0.0),
c                                       (fscale,0.0)

c                                  Note: If, upon reading of this record, there
c                                  is less than nmrow+5 items in the record,
c                                  then only a selected set of nodes were
c                                  output. Another record follows (integer, less
c                                  than lenbac long) which contains the list of
c                                  nodes for which DOF solutions are available.

c  ---      i       1 numdeflvs    lvscal table scale factor IDs
c                                   (ilvscID(i),i=1,numdeflvs)

c  ---      dp      1 numdeflvs    lvscal table scale factor values
c                                   (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 ANSYS.  All Rights Reserved.
c *** ansys, inc.

c     **********  description of modal result file  **********

      character*8  MODENM
      parameter   (MODENM='mode    ')

      character*8  MODENM_LEFT
      parameter   (MODENM_LEFT='lmode   ')

c     *** NOTE: if this variable is changed in the future it should be
c     ***       updated in spdefines.h also for symbolic assembly (jrb)
      integer         MODEHDLEN
      parameter      (MODEHDLEN=100)
```

```
      LONGINT         modefpL
      integer         modebk, modeut

      LONGINT         modeLeftfpL
      integer         modeLeftbk, modeLeftut

      common /fdmode/ modefpL, modebk, modeut,
     x                modeLeftfpL, modeLeftbk, modeLeftut

c   ********** common variable descriptions ***********
co  modefpL      file position on file mode
co  modebk       block number for file mode
co  modeut       file unit for file mode
co  modeLeftfpL  file position on file .lmode
co  modeLeftbk   block number for file .lmode
co  modeLeftut   file unit for file .lmode


c   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
c             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             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    type    nrec    lrec     contents

c  ---      i       1      100      standard ANSYS file header (see binhed for
c                                   details of header contents)


c  ---      i       1      100      .MODE FILE HEADER
c
c                                     fun09,    nmrow,   nmatrx,    nmode,   numdof,  <  5
c                                      maxn,    wfmax,   lenbac,  nEnfGrp,    neqns,  < 10
c                                    lumpms,   extopt,   SvCode,      kan,   ldstep,  < 15
c                                    numitr,   expbeg,   expend,   nspect,   nSPdat,  < 20
c                                    ptrRDF,   ptrFRQ, kPerturb,   ptrSHP,   ptrLOD,  < 25
c                                    ptrSTF,   ptrMAS,   ptrDMP,nrkeyPert,   nrigid,  < 30
c                                    ptrLPM,   ptrSP1,  ptrSHPh,  ptrLODh,  ptrSTFh,  < 35
c                                   ptrMASh,  ptrDMPh,  ptrLPMh,  ptrSP1h, ptrIRHSl,  < 40
c                                  ptrIRHSh, PowerDyn,   ptrRES,  ptrRESh,Glblenbac,  < 45
c                                 KeyStress,   ptrELD,  ptrELDh,   ptrGBk,  ptrGBkh,  < 50
c                                   modlstp,    nresi,   ptrEf1,  ptrEf1h,    sstif,  < 55
c                                   ptrFSTA,   ptrEf2,  ptrEf2h,   ptrEf3,  ptrEf3h,  < 60
c                                 qrDampKey,cycMSUPkey,cycnmode,    ptrHI,  ptrKUNS,  < 65
c                                  ptrKUNSh, mrestart, LPrestls, LPrestss,   cpxmod,  < 70
c                                   keyLeft,    cpxlv,        0,        0,        0,  < 75
c                                         0,        0,        0,        0,        0,  < 80
c                                         0,        0,        0,        0,        0,  < 85
c                                         0,        0,        0,        0,        0,  < 90
c                                         0,        0,        0,        0,        0,  <100


c                                   each item in header is described below:

c                                    fun09  - unit number (mode file is 9)
c                                    nmrow  - number of rows/columns in matrices
c                                             (maxn*numdof). If extopt=0, nmrow
c                                             is the number of rows in the
c                                             reduced matrices and the number of
c                                             master degrees of freedom.
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries and affiliates.

```
c                              nmatrx - number of reduced matrices on the
c                                       file (applies only if extopt=0)
c                              nmode  - number of modes extracted
c                              numdof - number of dof per node
c                              maxn   - maximum node number (if extopt=3
c                                       or 4, the actual number of nodes is
c                                       referenced.)
c                              wfmax  - maximum wavefront (does not apply
c                                       if extopt=3 or 4)
c                              lenbac - number of nodes
c                             nEnfGrp - numbre of enforced group
c                              neqns  - number of equations on the .LN22
c                                       file (does not apply if extopt=0)
c                              lumpms - lumped mass key
c                                       = 0 - default matrix type
c                                       = 1 - lumped
c                                       (does not apply if extopt=3 or 4)
c                              extopt - mode extraction method
c                                       = 0 - reduced
c                                       = 3 - unsymmetric Lanczos
c                                       = 4 - damped Lanczos
c                                       = 6 - block Lanczos
c                                       = 8 - SuperNode
c                                       = 9 - PCG Lanczos
c                              SvCode - Solver assembly code
c                                       = 0 Frontal assembly (SV_ANSYS)
c                                       = 1 Symbolic assembly (SV_CASI)
c                              kan    - analysis type
c                                       = 1 - buckling
c                                       = 2 - modal
c                              ldstep - load step number - also number of load vectors
c                              numitr - total number of cumulative
c                                       iterations done during analysis
c                                       (does not apply if extopt=3 or 4)
c                              expbeg - beginning of the frequency range of
c                                       interest
c                              expend - end of the frequency range of
c                                       interest
c                              nspect - number of spectra; if -6, these are
c                                       the 6 default unit spectra
c                              nSPdat - number of data items per spectrum
c                              ptrRDF - pointer to reduced degree of
c                                       freedom set used in model
c                              ptrFRQ - pointer to the frequencies
c                            kPerturb - Linear Perturbation key
c                              ptrSHP - pointer to the mode shapes
c                                       (eigenvectors)
c                        ptrLOD, ptrLODh - pointer to the load vectors
c                              ptrSTF - pointer to the reduced stiffness
c                                       matrix
c                              ptrMAS - pointer to the reduced mass matrix
c                          ptrDMP, ptrDMPh - pointer to the reduced/modal damping
c                                       matrix
c                       ptrKUNS, ptrKUNSh - pointer to the modal stiffness
c                                       matrix (unsymmetric part)
c                             nrkeyPert - nrkey setting of base analysis (Linear Pertubation)
c                                nrigid - number of rigid body modes
c                                ptrLPM - pointer to the diagonal mass vector
c                                ptrSP1 - pointer to the the spectrum data
c                             ptrIRHSl,h - pointer to imaginary part of RHS vector
c                               PowerDyn - PowerDynamics key
c                                       = 0 non-PowerDynamics method
c                                       = 1 PowerDynamics method
c                           ptrRES,ptrRESh - pointer to residual vectors
c                                Glblenbac - global number of nodes (for D-ANSYS)
c                           ptrGBk,ptrGBkh - pointer to global nodal equivalence table
c                                  modlstp - multiple load step key
c                                    nresi - number of residual vectors in file
c                                KeyStress - key set if mode stresses on file
c                           ptrELD,ptrELDh - pointer to element records
c                           ptrEf1,ptrEf1h - pointer to enforced motion dof information
c                                    sstif - key denoting prestress effects are included,
```

```
c                                          this key is for internal usage only. SSTIF,on
c                                          or off is controlled by NLGEOM on or off now.
c                                 ptrFSTA - pointer to fstacm data
c                            ptrEf2,ptrEf2h - pointer to enforced motion modes
c                            ptrEf3,ptrEf3h - pointer to enforced motion force
c                                 qrDampKey - QR damped calcaulations key
c                                cycMSUPkey - mode file format is for subsequent cyclic MSUP
c                                             (only base results on file)
c                                  cycnmode - total number of cyclic modes extracted
c                                             (sum of all harmonic indices)
c                                     ptrHI - pointer to harmonic indices
c                                  mrestart - key for modal restart (=0 none, =1 modal restart)
c                                  LPrestls - restarted load step (from linear perturbation)
c                                  LPrestss - restarted substep (from linear perturbation)
c                                    cpxmod - key for complex frequencies/shapes (0=no 1=yes)
c                                   keyLeft - key for LMODE writing (0=no 1=yes)
c                                     cpxlv - key for RHS vector in complex form (0=no 1=yes)
c                                             = 0 no  (before version 17.0)
c                                             = 1 yes (since version 17.0) ; ptrIRHSl,h is 0
c                                         0 - position not used

c  ---      i       1     numdof    Degrees of freedom per node
c                                   DOF reference numbers are:
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  ****** 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                                    (curdof(i),i=1,numdof)

c  ---      i       1     lenbac    Nodal equivalence table
c                                   This table equates the number used for
c                                   storage to the actual node number.
c                                    (Back(i),i=1,lenbac)

c  GBK      i       1    Glblenbac  Global nodal equivalence table
c                                   This table equates the number used for
c                                   storage to the actual global node number.
c                                    (GlbBack(i),i=1,Glblenbac)

c  FSTA    dp       1     30        fstacm.inc information (mass and MofI)

c  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                                   where N is position number of the node in
c                                   the nodal equivalence table and DOF is the
c                                   DOF reference number given above
c                                    (l(i),i=1,nmrow) (if nmatrx>0)

c  HI       i       1     cycnmode  Signed harmonic index for each extracted frequency.
c                                   Only present if cycMSUPkey=1.

c  FRQ     dp/cmp   1     nf        Frequencies (eigenvalues).
c                                   Frequencies are complex if cpxmod=1 or qrdampKey=1.
c                                   If frequencies are real, numbers stored are
c                                      the squares of the natural circular
c                                      frequencies (w**2, where w=radians/time).
c                                      You can obtain the natural frequencies, f
c                                      (in cycles/time), using the equation f=w/2pi
c                                   If frequencies are complex, numbers stored are
c                                      the natural frequencies (Hz)
c                                   (freq(i),i=1,nf)
c                                       nf = nmode+nresi
c                                     if cycMSUPkey=1 then
c                                       nf = cycnmode


c  SHP     dp/cmp   ns    nmrow     Mode shapes (eigenvectors). Mode shapes are
c                                   complex if cpxmod=1. If extopt=0, the
c                                   mode shape order corresponds to the DOF list
c                                   stored at position ptrRDF.  If extopt does
c                                   not equal 0, the order corresponds to the
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries and affiliates.

```
c                                  nodal equivalence table
c                                   (psi(i,j),i=1,nmrow)
c                                      ns = nmode
c                                   if cycMSUPkey=1 then
c                                      ns = cycnmode

c  RES      dp    nresi   nmrow     Residual vectors

c  LOD     cmp   ldstep   nmrow     Load vectors in complex form (since version 17.0)
c                                   (f(i),i=1,nmrow)
c                                  Before version 17.0, records were (cpxlv=0):
c                                  LOD  dp ldstep nmrow Load vectors
c                                                        (f(i),i=1,nmrow)
c                                  IRHS dp ldstep nmrow Imaginary Load vectors
c                                                        (fimag(i),i=1,nmrow)

c  LPM      dp      1     nmrow     Lumped mass vector. This record is present
c                                  only if lumpms=1 and nmatrix=0. It is a
c                                  vector containing the mass at each node in
c                                  the system.
c                                   (mass(i),i=1,nmrow)

c  STF      dp    nmrow   nmrow     Reduced stiffness matrix. Each row of the
c                                  matrix is stored as a record. The matrix is
c                                  present only if nmatrx > 0. Row order is the
c                                  same as the DOF order stored at position
c                                  ptrRDF.
c                                   (ak(i,j),i=1,nmrow)

c  MAS      dp    nmrow   nmrow     Reduced mass matrix. Each row of the matrix
c                                  is stored as a record. The matrix is present
c                                  only if nmatrx > 1. Row order is the same as
c                                  the DOF order stored at position ptrRDF.
c                                   (am(i,j),i=1,nmrow)

c  DMP      dp    nmrow   nmrow     Reduced/modal damping matrix. Each row of the
c                                  matrix is stored as a record. If extopt=0,
c                                  the row order is the same as the DOF order
c                                  stored at position ptrRDF.
c                                   (ac(i,j),i=1,nmrow)

c  KUNS     dp    nmrow   nmrow     Modal unsymmetric stiffness matrix. Each row of the
c                                  matrix is stored as a record.
c                                   (aku(i,j),i=1,nmrow)

c  EF1     int      1    nEnfGrp    (groupID(i),i=1,nEnfGrp)
c          int      1    nEnfGrp    (grpdof(i),i=1,nEnfGrp)
c           dp  nEnfGrp  grpdof(i)  dofIndx(i,j) i=1,grpdof(j)
c                                  The above records contain information about each
c                                  enforced motion group.

c  EF2      dp  nEnfGrp   nmrow     Enforced static modes

c  EF3      dp  nEnfGrp   nmrow     Enforced forced vector

c for each spectrum (|nspect| records):
c  SP1      dp      1   nmode+nresi Participation factors for this spectra
c  ---      dp      1   nmode+nresi Mode coefficients for this spectra
c  ---      dp      1   nmode+nresi Modal damping values
c  ---      dp      1     613       svcom.inc (freqtb,...)
c  ---      dp      1     20        misc. spectra data

c  ELD      int     1     15        nelm,   mask, nItems, ptrELM, ptrERS,
c                                  ptrCER,ptrCERh, ptrESL,ptrESLh,    nRF
c                                   ptrFR, ptrRFh, PrecKey

c                                   each item in header is described below:

c                                     nelm - number of elements
c                                     mask - output mask (OUTRES)
c                                   nItems - number of element records (7, VOL
c                                            not included)
```

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
c                                 ptrELM - pointer to element equivalence table
c                                 ptrERS - pointer to element record sizes
c                                ptrCER,h - pointer to constant element records
c                                ptrESL,h - pointer to element index
c                                     nRF - number of reaction forces
c                                ptrRF,h - pointer to reaction forces
c                                PrecKey - 0, double precision 1, single
c                        above pointers are relative to ptrELD

c  ---      int     1   2*nItems   Total size of each element record (LONGINT)

c  ELM      int     1     nelm     Element equivalence table
c                                   This table equates the order number used to
c                                   the actual element number

c  ERS      int   nItems  nelm     Sizes of the nItem element results sets for
c                                   each element

c  CER      int     1     5       ptrVOL, ptrEPT, ptrEUL,      0,      0
c                        above pointers are relative to ptrCER

c     constant element records (do not vary by mode):
c   VOL     dp      1  nelm*1       Element volume
c   EPT     dp      1  nelm*size    Element structural nodal temperatures
c   EUL     dp      1  nelm*size    Element Euler angles

c   ESL     int     1     10      ptrENS, ptrEEL, ptrEMS, ptrENF, ptrENG,
c                                 ptrENSh,ptrEELh,ptrEMSh,ptrENFh,ptrENGh
c                        above pointers are relative to ptrESL

c     non-constant element records (do vary by mode):
c   ENS     dp   nelm  nmode*size   Element nodal component stresses
c   EEL     dp   nelm  nmode*size   Element nodal component elastic strains
c   EMS     dp   nelm  nmode*size   Element summable miscellaneous data
c   ENF     dp   nelm  nmode*size   Element nodal forces
c   ENG     dp   nelm  nmode*3      Element energies

c     see fdresu.inc for more information on the element results
```

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 ANSYS.  All Rights Reserved.
c *** ansys, inc.

c     **********  description of element matrix file  **********
c
c *** mpg fdemat.inc < eoelem elostr eofini outelm elfini EmatAssemble sffini
c          eqprep sfform elstrt slvstr: emat file description
c
      character*8  EMATNM
      parameter  (EMATNM='emat    ')

      LONGINT         ematfpL, ematfp
      integer         ematbk, ematut
      common /fdemat/ ematfpL, ematbk, ematut
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
      equivalence  (ematfp,ematfpL)

c   ********** common variable descriptions ***********
co  ematfpL      file position on file emat
co  ematbk       block number for file emat
co  ematut       file unit for file emat

c   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
c             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             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    type    nrec    lrec     contents

c  ---      i       1      100      standard ANSYS file header (see binhed for
c                                   details of header contents)

c  ---      i       1       40      .EMAT FILE HEADER
c
c                                    fun02,    nume,   numdof,    lenu,  lenbac,
c                                     maxn, nlgeEMA,  sstEMAT,  nodref,   lumpm,
c                                    kygst,    kygm,     kycd,   kygss,   kygaf,
c                                    kygrf,       0,Glblenbac, ptrGBkl, ptrGBkh,
c                                  ptrElmh, ptrFSTh,  ptrLSTh, ptrBITh, ptrEHDh,
c                                  ptrIDXh,   numCE,  maxLeng,  ptrCEl,  ptrCEh,
c                                   ptrDOF,  ptrBAC,  ptrELMl, ptrFSTl, ptrLSTl,
c                                  ptrBITl, ptrEHDl,  ptrIDXl, ptrendH, ptrendL

c                                each item in header is described below:

c                                   fun02  - unit number (emat file is 2)
c                                   nume   - number of elements
c                                   numdof - number of dofs per node
c                                   lenu   - total DOFs of model
c                                   lenbac - number of nodes
c                                   maxn   - maximum node number
c                                   nlgeEMA  = 0 - nlgeom is OFF the time this Emat file is created
c                                              1 - nlgeom is ON the time this Emat file is created
c                                   sstEMAT  = 0 - sstif key is OFF the time this Emat file is created
c                                              1 - sstif key is ON the time this Emat file is created
c                                                  this key is for internal use only
c                                   nodref - actual number of nodes referenced
c                                   lumpm  - lumped mass key
c                                            = 0 - default matrix type
c                                            = 1 - lumped
c                                   kygst  - global stiffness matrix calculate
c                                            key
c                                            = 0 - do not calculate
c                                            = 1 - calculate
c                                   kygm   - global mass matrix calculate key
c                                            = 0 - do not calculate
c                                            = 1 - calculate
c                                   kycd   - global damping matrix calculate key
c                                            = 0 - do not calculate
c                                            = 1 - calculate
c                                   kygss  - global stress stiffening matrix
c                                            calculate key
c                                            = 0 - do not calculate
c                                            = 1 - calculate
c                                   kygaf  - global applied force vector
```

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
c                                          calculate key
c                                          = 0 - do not calculate
c                                          = 1 - calculate
c                                    kygrf  - global restoring force vector
c                                          calculate key (Newton-Raphson only)
c                                          = 0 - do not calculate
c                                          = 1 - calculate
c                                    0      - position not used
c                                    Glblenbac - global global number of nodes (== lenbac unless using
c                                                Distributed Ansys)
c                                    ptrGBkl- low pointer to global nodal equivalence table
c                                    ptrGBkh- high pointer to global nodal equivalence table
c                                    ptrELMh- high pointer to element equivalence table
c                                    ptrFSTh- high pointer to first element at a
c                                          DOF table
c                                    ptrLSTh- high pointer to last element at a
c                                          DOF table
c                                    ptrBITh- high pointer to dof bits
c                                    ptrEHDh- high pointer to the start of the
c                                          element matrices
c                                    ptrIDXh- high pointer to element matrices
c                                          index table
c                                    numCE  - number of internal CEs
c                                    maxLeng- maximum length of any internal CE
c                                    ptrCEl - low pointer to internal CE list
c                                    ptrCEh - high pointer to internal CE list
c                                    ptrDOF - pointer to degrees of freedom per
c                                          node used in model
c                                    ptrBAC - pointer to nodal equivalence table

c                                    ptrELMl- Low pointer to element equivalence
c                                          table
c                                    ptrFSTl- Low pointer to first element at a
c                                          DOF table
c                                    ptrLSTl- Low pointer to last element at a
c                                          DOF table
c                                    ptrBITl- Low pointer to dof bits
c                                    ptrEHDl- Low pointer to the start of the
c                                          element matrices
c                                    ptrIDXl- Low pointer to element matrices
c                                          index table

c                                    ptrendH- High pointer to end of file
c                                    ptrendL- Low  pointer to end of file

c       Note: the analysis type sets the global calculate keys.

c  ---     dp       1       20      Time information
c
c                                    timval, timinc, frqval, timbeg, timend,
c                                       0.0,    0.0,    0.0,    0.0,    0.0,
c                                       0.0,    0.0,    0.0,    0.0,    0.0,
c                                       0.0,    0.0,    0.0,    0.0,    0.0,

c                                   each item is described below:

c                                    timval - the current time
c                                    timinc - the time increment
c                                    frqval - the current frequency (from a
c                                          harmonic analysis)
c                                    timbeg - the start time for the analysis
c                                    timend - the end time for the analysis
c                                    0.0    - position not used
c                                    0.0    - position not used
c                                    0.0    - position not used
c                                    0.0    - position not used
c                                    0.0    - position not used
c                                    0.0    - position not used
c                                    0.0    - position not used
c                                    0.0    - position not used
c                                    0.0    - position not used
c                                    0.0    - position not used
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries and affiliates.

```
c                                    0.0    - position not used
c                                    0.0    - position not used
c                                    0.0    - position not used
c                                    0.0    - position not used
c                                    0.0    - position not used

c  DOF      i       1    numdof     Degrees of freedom per node
c                                    DOF reference numbers are:
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  ****** 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                                     (curdof(i),i=1,numdof)

c  BAC      i       1    lenbac     Nodal equivalence table. This table equates
c                                    the number used for storage to the actual
c                                    node number
c                                     (Back(i),i=1,lenbac)

c  ELM      i       1     nume      Element equivalence table. The ANSYS program
c                                    stores all element data in the numerical
c                                    order that the SOLUTION processor solves the
c                                    elements.  This table equates the order
c                                    number used to the actual element number
c                                     (Order(i),i=1,nume)

c  GBK      i       1  Glblenbac    Global nodal equivalence table.  This
c                                    table equates the number used for storage
c                                    to the actual node number.  Only written
c                                    by the master process in Distributed Ansys
c                                     (GlbBack(i),i=1,Glblenbac)

c  FST      i       1     lenu      First element at a DOF table. This record
c                                    signifies the first element encountered at a
c                                    particular DOF.
c                                     (First(i),i=1,lenu)

c  LST      i       1     lenu      Last element at a DOF table. This record
c                                    signifies the last element encountered at a
c                                    particular DOF.
c                                     (Last(i),i=1,lenu)

c  BIT      i       1     lenu      Bits set at a DOF table. This record
c                                    has bits for constraints, forces, etc.
c                                     (DofBits(i),i=1,lenu) (added at 10.0)

c  IDX      i       1     2*nume    Element index table. This record specifies
c                                    the file location for the beginning of the
c                                    data for each element.
c                                     (index(i),i=1,nume) Low part of pointer
c                                     (index(i),i=1,nume) High part of pointer


c  The records at the end of the file store element information and get written
c  as a set for each element(nume sets of these records will appear on the file
c  at this point) ptrEHD indicates the beginning of the element data.

c  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
c  at this time, as it is likely to change in the future.


c  EHD      i       1      10       Element matrix header
c
c                                     stkey,   mkey,   dkey,  sskey, akey,
c                                     nrkey,   ikey,  kckey,      0, nmrow

c                                    each item in header is described below:

c                                     stkey  - stiffness matrix key
c                                              = 0 - matrix not present
c                                              = 1 - matrix present
```

```
c                                   mkey   - mass matrix key
c                                            = 0 - matirx not present
c                                            = 1 - matrix present
c                                   dkey   - damping matrix key
c                                            = 0 - matrix not present
c                                            = 1 - matrix present
c                                   sskey  - stress stiffening matrix key
c                                            = 0 - matrix not present
c                                            = 1 - matrix present
c                                   akey   - applied load vector key
c                                            = 0 - vector not used
c                                            = 1 - vector used
c                                   nrkey  - newton-raphson(restoring) load
c                                            vector key (for nonlinear analyses)
c                                            = 0 - vector not used
c                                            = 1 - vector used
c                                   ikey   - imaginary load vector key
c                                              (for complex analyses)
c                                            = 0 - vector not used
c                                            = 1 - vector used
c                                   kckey    = 0 or 1 or 2 position for internal use or not in use
c                                            = 3 complex number stiffness matrix key
c
c                                   0      - position not used
c                                   nmrow  - numbers/columns in matrices. If the
c                                            number is negative, the matrices
c                                            will be written in upper triangular
c                                            form.

c  ---      i       1    nmrow      DOF index table. This record specifies the
c                                   DOF locations of this element matrix in
c                                   relation to the global matrix. The index is
c                                   calculated as (N-1)*NUMDOF+DOF, where N is
c                                   the position number of the node in the nodal
c                                   equivalence table and DOF is the DOF
c                                   reference number given above

c  ---     dp    varies  varies     Element matrices. This record is repeated
c                                   for each stiffness, mass, damping,stress stiffening
c                                   and complex stiffness matrice. If the matrix is
c                                   diagonal, the length of the records will be
c                                   nmrow.  If the matrix is unsymmetric, the
c                                   length of the records will be nmrow*nmrow.
c                                   If the matrix is symmetric, only the upper
c                                   triangular terms are written and the length
c                                   of the records will be (nmrow)*(nmrow+1)/2.

c  ---     dp       1    2*nmrow    Element force vectors. This record contains
c                                   both the applied force vector and the
c                                   (restoring or imaginary) load vector.
c
c
c      *************** Internal CE information ***********************
c      The following records repeat numCE times... one for each internal
c      CE created during solution... these are stored here for the
c      usage of a prestressed modal analysis such as the linear perturbation analysis
c
c  CE      i       3     numCE      First part is the CE number, the second part is
c                                   the number of terms in this internal CE, and
c                                   the third part is the external element number
c                                   of the element that created this internal CE
c
c  ---     i     nTerms  numCE      integer info (list of node*32 + dof)
c
c  ---    dp     nTerms  numCE      dp info (list of coefficients including constant term)
c
c
c   kygst        global stiffness matrix calculate key
c   kygm         global mass matrix calculate key
c   kygd         global damping matrix calculate key
c   kygss        global stress stiffening matrix calculate key
c   kygaf        global applied force matrix calculate key
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries and affiliates.

```
c   kygrf        global restoring force matrix calculate key
```

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 ANSYS.  All Rights Reserved.
c *** ansys, inc

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

c  write:  matout
c  read:

c   ********** common variable descriptions **********
co  subfpL      file position on file sub
co  subbk       block number for file sub
co  subut       file unit for file sub
co  lenSubL     length of sub file (saved for slvdta.F)

c   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
c          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          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    type    nrec    lrec     contents

c  ---      i       1      100      standard ANSYS file header (see binhed
c                                   for details of header contents)

c  HED      i       1       60      .SUB FILE HEADER  (FULL MATRICES)
c
c                                      8,  nmrow, nmatrx,  nedge, numdof,
c                                   maxn,  wfmax, lenbac,   nnod, kunsym,  10
c                                   kstf,  kmass,  kdamp,    kss,  nvect,
c                                 nWorkL,  lenU1, sesort, lenlst,ptrLodL,  20
c                                 ntrans, ptrMtx, ptrXFM, ptrHED,  name1,
c                                  name2,  ptrCG,      0,  name3,  name4,  30
c                                 ptrDOF, ptrDST, ptrBAC, ptrTIT, ptrNOD,
c                                 ptrXYZ, ptrEDG, ptrGDF, thsubs, ptrPOS,  40
```

```
c                                ptrORG, stfmax,ptrLodH, nmodes, keydim,
c                              cmsMethod,  name5,  name6,  name7,  name8,  50
c                                nvnodes,ptrCTXM, nWorkH,      0,ptrTVAL,
c                               gyroDamp,kstress,      0,ptrEndL,ptrEndH   60

c  HED      i       1       60      .SUB FILE HEADER  (SPARSE MATRICES)
c
c                                     9,   nEqn, nmatrx,  ndege, numdof,
c                                   maxn,  wfmax, lenbac,   nnod, kunsym,  10
c                                   kstf,  kmass,  kdamp,       ,  nvect,
c                                 nTermL,       ,       , lenlst,ptrLodL,  20
c                                 ntrans,ptrMtxL, ptrXFM, ptrHED,  name1,
c                                  name2,  ptrCG,       ,  name3,  name4,  30
c                                 ptrDOF,       , ptrBAC, ptrTIT, ptrNOD,
c                                 ptrXYZ, ptrEdg, ptrGDF, thsubs,       ,  40
c                                       , stfmax,ptrLodH,       , keydim,
c                                       ,  name5,  name6,  name7,  name8,  50
c                                       ,ptrCTXM, nTermH,ptrMtxH,ptrColL,
c                                ptrColH,ptrCofL,ptrCofH,ptrEndL,ptrEndH   60

c                                 each item in header is described below:

c                                  fun08  - unit number (full sub file is 8)
c                                           (sparse substructure file is 9)
c                                  nmrow  - number of rows in matrices (also
c                                           number of dofs in substructure)
c                                  nmatrx - number of matrices on file
c                                  nedge  - number of edges for outline
c                                  numdof - number of dofs per node
c                                  maxn   - maximum node number of complete
c                                           model presently in database
c                                  wfmax  - maximum wavefront of substruct.
c                                           during generation pass
c                                  lenbac - number of nodes defining
c                                           substructure during the
c                                           generation pass
c                                  nnod   - number of unique nodes in the
c                                           substructure having DOFs, and
c                                           which define this substructure
c                                           during the use pass.  Also, the
c                                           number of nodes having master
c                                           DOFs.
c                                  kunsym - unsymmetric matrix key
c                                           = 0 - symmetric
c                                           = 1 - unsymmetric
c                                  kstf   - stiffness matrix present key
c                                           = 0 - matrix is not on file
c                                           = 1 - matrix is on file
c                                  kmass  - mass matrix present key
c                                           = 0 - matrix is not on file
c                                           = 1 - matrix is on file
c                                           =-1 - Lumped mass vestor (Sparse only)
c                                  kdamp  - damping matrix present key
c                                           = 0 - matrix is not on file
c                                           = 1 - matrix is on file
c                                  kss    - stress stiffening matrx present
c                                           = 0 - matrix is not on file
c                                           = 1 - matrix is on file
c                                  nvect  - number of load vectors
c                                           (at least 1 is required)
c                                  nWorkL,H  - BCS workspace length (only for
c                                              bacsub)
c                                  nTermL,H  - Number of terms in sparse matrix
c                                  lenU1  - length of intermediate transformation
c                                           vector
c                                  sesort - DOF set sort key
c                                           = 0 - numbers are not sorted
c                                           = 1 - numbers are sorted in
c                                                 ascending order
c                                  lenlst - maximum length of DOF set for
c                                           this substructure (maxn*numdof)
c                                  ptrLod - pointer to the start of the load
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries and affiliates.

```
c                                  vectors (see also ptrLodh)
c                          ntrans - transformed key
c                                   = 0 - substructure has not been
c                                         transformed
c                                   > 0 - substructure copied
c                                         from another substructure,
c                                         via either SESSYM or SETRAN
c                          ptrMtxL,H - pointer to the start of the
c                                      substructure matrices (iDiagL for
c                                      sparse matrices)
c                          ptrXFM - pointer to the substructure
c                                   transformations
c                          ptrHED - pointer to the SUB file header
c                          name1  - first four characters of the
c                                   substructure file name, in
c                                   integer form
c                          name2  - second four characters of the
c                                   substructure file name, in
c                                   integer form
c                          name3  - third four characters of the
c                                   substructure file name, in
c                                   integer form
c                          name4  - fourth four characters of the
c                                   substructure file name, in
c                                   integer form
c                          ptrDOF - pointer to the DOF/node list
c                          ptrDST - pointer to the local DOF set
c                          ptrBAC - pointer to the nodes comprising
c                                   the substructure
c                          ptrTIT - pointer to the title
c                          ptrNOD - pointer to the unique nodes
c                                   defining the substructure
c                          ptrXYZ - pointer to the coordinates of the
c                                   unique nodes
c                          ptrEDG - pointer to the substructure edges
c                          ptrGDF - pointer to the global DOF set
c                          ptrCG  - pointer to the element mass information
c                          thsubs - element type key
c                                   = 0 - structural
c                                   = 1 - 1st order non-structural
c                                         (generally from thermal)
c                                   = 2 - 2nd order non-structural
c                          ptrPOS - pointer to the sorted substructure
c                                   DOF set to the original
c                          ptrORG - pointer to the DOF set of the model
c                                   during the generation pass
c                          stfmax - maximum diagonal stiffness term
c                                   (packed into an integer)
c                          ptrLodh- High 32 bits of 64 bit pointer
c                          nmodes - number of modes used to generate
c                                   CMS s.e.
c                          keydim - dimensionality key
c                                   = 1 - axisymmetric
c                                   = 2 - 2-D
c                                   = 3 - 3-D
c                        cmsMethod - component mode synthesis method
c                          name5  - fifth four characters of the
c                                   substructure file name, in integer
c                                   form
c                          name6  - sixth four characters of the
c                                   substructure file name, in integer
c                                   form
c                          name7  - seventh four characters of the
c                                   substructure file name, in integer
c                                   form
c                          name8  - eighth four characters of the
c                                   substructure file name, in integer
c                                   form
c                         nvnodes - number of virtual nodes that contain
c                                   the modal coordinates
c                        gyroDamp - gyroscopic damping matrix key
c                                   = 0 - damping matrix is
```

```
c                                          not skew-symmetric if present
c                                       = 1 - damping matrix is
c                                             skew-symmetric, due to
c                                             gyroscopic effect
c                                kStress - = 1 if modal element results are
c                                          on the .cms file
c                                ptrCTXM - coordinate transformation
c                                ptrColL,H - pointer to the iCol sparse matrix
c                                            array
c                                ptrCofL,H - pointer to the of the
c                                            sparse matrix Sk(1:nTerm),
c                                            Sm(1:nTermL),Sc(1:nTermL),
c                                            Ss(1:nTermL)  Each matrix is a
c                                            single large record
c                                ptrEndL,H - Next location after end of file

c                               note: name1/2/3/4/5/6/7/8 are the
c                                     inexc4 representation of the
c                                     32 character filename.
c                                     name1/2/5/6/7/8 will be "0"
c                                     for pre rev 5.2 files - cwa

c   Note: If ntrans > 0, records from position ptrDOF to ptrGDF will be
c         identical to the data for the copied substructure.


c  XFM      dp      1      125   Substructure transformations (5*25 double
c                                precisions). This record has meaning only
c                                if ntrans > 0. You can define up to five
c                                levels of transformations, with 25 variables
c                                in each level.  Up to the first seven
c                                variables are used as follows:
c
c                                If the substructure was transferred (via the
c                                 SETRAN command):
c                                1st variable - 1.0
c                                2nd variable - nodal increment
c                                3rd variable - reference number of
c                                 coordinate system where substructure will
c                                 be transferred
c                                4th variable - reference number of
c                                 coordinate system where substructure is
c                                 presently defined
c                                5th variable - x coordinate increment
c                                6th variable - y coordinate increment
c                                7th variable - z coordinate increment

c                                If the substructure used symmetry (via the
c                                 SESYMM command):
c                                1st variable - 2.0
c                                2nd variable - nodal increment
c                                3rd variable - number of coordinate
c                                 component to be used in operation
c                                 = 1 - x coordinate
c                                 = 2 - y coordinate
c                                 = 3 - z coordinate
c                                4th variable - reference number of
c                                 coordinate system to be used for symmetry
c                                 operation
c  CTXM     dp      1      250   Substructure transformations

c  DOF      i       1     numdof    Degrees of freedom per node  (Global)
c                                     (curdof(i),i=1,numdof)
c                                   DOF reference numbers are:
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  ****** 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  DST      i       1     nmrow  This record contains degrees of freedom for
c                                this substructure of the unique nodes, as
c                                used with this substructure, in ascending
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries and affiliates.

```
c                                  order. This index is calculated as
c                                  (N-1)*numdof+DOF, where N is the node number
c                                  and DOF is the DOF reference number given
c                                  above
c                                   (lsort(i),i=1,nmrow)

c  POS      i       1     nmrow  This record stores the positions of the
c                                  local DOF set in relation to the generated
c                                  DOF set. (lposit(i),i=1,nmrow)

c  ORG      i       1     nmrow  DOF set of the model as defined during the
c                                  generation pass. This index is calculated as
c                                  (N-1)*NUMDOF+DOF, where N is the position
c                                  number of the node in the nodal equivalence
c                                  table and DOF is the DOF reference number
c                                  given above
c                                   (lorig(i),i=1,nmrow)

c  BAC      i       1     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                                  equal to the number of used or referenced
c                                  nodes. This table equates the number used
c                                  for storage to the actual node number.
c                                   (Back(i),i=1,lenbac)

c  TIT      i       1       20   Substructure title (converted to integers -
c                                  see inexc4)

c  NOD      i       1      nnod  This record describes unique nodes defining
c                                  the substructure for the use pass of the
c                                  analysis.  These are also the nodes having
c                                  master degrees of freedom.
c                                   (node(i),i=1,nnod)

c  XYZ     dp     nnod      6    This record describes the coordinates of a
c                                  unique node, in the order X, Y, Z, THXY,
c                                  THYZ, and THZX.  Nodal order corresponds to
c                                  that of the node list given above
c                                   (xyzang(j,i),j=1,6)

c  EDG     dp     nedge     6    This record contains beginning and ending
c                                  locations (X1,Y1,Z1,X2,Y2,Z2 coordinates) of
c                                  a straight line comprising an edge of the
c                                  substructure.

c  GDF    LONG      1     nmrow  This record describes global degrees of
c                                  freedom of the unique nodes in ascending
c                                  order, as used during the analysis use pass.
c                                  This index is calculated as (N-1)*32+DOF,
c                                  where N is the node number and DOF is the
c                                  DOF reference number given above
c                                   (l(i),i=1,nmrow) (sorted)
c                                  (Made LONGINT in rev 14.0)

c  CG      dp       1      10    total mass,CGx,CGy,CGz,6 moments of inertia

c  TVAL    dp       1     1000   current time value corresponds to each load
c        steps

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
c  matrix is written, then the second row of each matrix, etc. this pattern
c  continues until all nmrow rows of each matrix have been written to the file.

c  MAT      dp      1     nmrow   Row of the stiffness matrix, if nmatrx > 0.
c                                   (ak(i,j),i=1,nmrow)
c  ---      dp      1     nmrow   Row of the mass matrix, if nmatrx > 1.
c                                   (am(i,j),i=1,nmrow)
c  ---      dp      1     nmrow   Row of the damping matrix, if nmatrx > 2.
c                                   (ac(i,j),i=1,nmrow)
c  ---      dp      1     nmrow   Row of the stress stiffening matrix, if
```

```
c                                nmatrx > 3.
c                                 (gs(i,j),i=1,nmrow)

c  LOD      dp   nvect    nmrow   This record contains the load vectors.
c                                 (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 *** Copyright ANSYS.  All Rights Reserved.
c *** ansys, inc.

c     **********  description of CMS (component mode synthesis) transformation file  **********

      character*8  CMSNM
      parameter   (CMSNM='cms     ')

      LONGINT         cmsfpL, cmsfp
      integer         cmsbk, cmsut

      common /fdcms/  cmsfpL, cmsbk, cmsut
      equivalence (cmsfp,cmsfpL)

c   ********** common variable descriptions ***********
co  cmsfp       file position on file cms
co  cmsbk       block number for file cms
co  cmsut       file unit for file cms


c   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
c             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             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    type    nrec    lrec     contents

c  ---      i       1      100      standard ANSYS file header (see binhed for
c                                   details of header contents)

c  ---      i       1       40      .CMS FILE HEADER
c
c                                  fun45,    neqn,   nirfm,   nnorm,   ncstm,
c                                  nrsdm, cmsMeth, kStress,       0,       0,
c                                      0,       0,       0,       0,       0,
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
c                                      0,       0,       0,       0,       0,
c                                      0,       0,       0,       0,       0,
c                                      0,       0,       0,       0,       0,
c                                  ptrIRFl, ptrNORl, ptrCSTl, ptrRSDl, ptrIRFh,
c                                  ptrNORh, ptrCSTh, ptrRSDh, ptrELDl, ptrELDh

c                                    each item in header is described below:
c                                     fun45  - unit number
c                                      neqn  - number of equations (DOF)
c                                     nirfm  - number of inertia relief modes
c                                     nnorm  - number of normal modes
c                                     ncstm  - number of constraint modes
c                                            > 0 available in file
c                                            < 0 not available in file
c                                     nrsdm  - number of residual modes
c                                   cmsMeth  - CMS method key
c                                              0 = fix interface method
c                                              1 = free interface method
c                                              3 = residual-flexible free interface method
c                                              4 = user defined method
c                                    kStress - key if modal element results are on file
c                           ptrIRFl,ptrIRFh  - 64 bit pointer to inertia relief modes
c                           ptrNORl,ptrNORh  - 64 bit pointer to normal modes
c                           ptrCSTl,ptrCSTh  - 64 bit pointer to constraint modes
c                           ptrRSDl,ptrRSDh  - 64 bit pointer to residual modes
c                           ptrELDl,ptrELDh  - 64 bit pointer to element records
c                                     0      - position not used

c  ---       i       1    neqn      BCS to ANS mapping (lBCStoANS(i), i= 1,neqn)

c                                     Note: BCS to ANS mapping is not available for
c                                     residual-flexible free interface method.

c  NOR       dp   nnorm    neqn      Normal Modes
c
c  IRF       dp   nirfm    neqn      Inertia Relief Modes
c
c  CST       dp   ncstm    neqn      Constraint Modes
c
c  RSD       dp   nrsdm    neqn      Residual modes

c modal element results if kStress = 1 (fix interface method only):
c  ELD       int     1     15        nelm,   mask, nItems, ptrELM, ptrERS,
c                                  ptrCER,ptrCERh, ptrESL,ptrESLh,    nRF
c                                   ptrFR, ptrRFh, PrecKey

c                                   each item in header is described below:

c                                     nelm - number of elements
c                                     mask - output mask (OUTRES)
c                                   nItems - number of element records (7, VOL
c                                            not included)
c                                   ptrELM - pointer to element equivalence table
c                                   ptrERS - pointer to element record sizes
c                                  ptrCER,h - pointer to constant element records
c                                  ptrESL,h - pointer to element index
c                                       nRF - number of reaction forces
c                                   ptrRF,h - pointer to reaction forces
c                                   PrecKey - 0, double precision 1, single
c                        above pointers are relative to ptrELD

c  ---       int     1   2*nItems   Total size of each element record (LONGINT)

c  ELM       int     1     nelm     Element equivalence table
c                                    This table equates the order number used to
c                                    the actual element number

c  ERS       int   nItems  nelm     Sizes of the nItem element results sets for
c                                    each element

c  CER       int     1     5       ptrVOL, ptrEPT, ptrEUL,      0,      0
c                        above pointers are relative to ptrCER
```

```
c     constant element records (do not vary by mode):
c   VOL     dp      1  nelm*1       Element volume
c   EPT     dp      1  nelm*size    Element structural nodal temperatures
c   EUL     dp      1  nelm*size    Element Euler angles

c   ESL     int     1     10      ptrENS, ptrEEL, ptrEMS, ptrENF, ptrENG,
c                                 ptrENSh,ptrEELh,ptrEMSh,ptrENFh,ptrENGh
c                      above pointers are relative to ptrESL

c     non-constant element records (do vary by mode ; mode order is [NOR CST]):
c   ENS     dp   nelm  nmode*size   Element nodal component stresses
c   EEL     dp   nelm  nmode*size   Element nodal component elastic strains
c   EMS     dp   nelm  nmode*size   Element summable miscellaneous data
c   ENF     dp   nelm  nmode*size   Element nodal forces
c   ENG     dp   nelm  nmode*3      Element energies

c     see fdresu.inc for more information on the element results
```

1.8.3. TCMS File Format

```
*comdeck,fdtcms

c *** Copyright ANSYS.  All Rights Reserved.
c *** ansys, inc.

c     **********  description of CMS (component mode synthesis) transformation file  **********

      character*8  TCMSNM
      parameter   (TCMSNM='tcms     ')

      LONGINT         tcmsfpL, tcmsfp
      integer         tcmsbk, tcmsut

      common /fdtcms/ tcmsfpL, tcmsbk, tcmsut
      equivalence (tcmsfp,tcmsfpL)

c   ********** common variable descriptions ***********
co  tcmsfp       file position on file tcms
co  tcmsbk       block number for file tcms
co  tcmsut       file unit for file tcms


c   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
c             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             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    type    nrec    lrec     contents

c  ---      i       1      100      standard ANSYS file header (see binhed for
c                                   details of header contents)

c  ---      i       1       40      .TCMS FILE HEADER
c
c                                  fun48,  nNodes,    neqn,  numdof,  nirfm,
c                                  nnorm,   ncstm,       0,       0,       0,
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries and affiliates.

```
c                                ptrNORl, ptrCSTl, ptrIRFl,       0,       0,
c                                      0,       0,       0,       0,       0,
c                                ptrNORh, ptrCSTh, ptrIRFh,       0,       0,
c                                      0,       0,       0,       0,       0,
c                                      0,       0,       0,       0,       0,
c                                      0,       0,       0,       0,       0,
c                                      0,       0,       0,       0,       0

c                                  each item in header is described below:
c                                   fun48  - unit number
c                                  nNodes  - number of nodes in file
c                                    neqn  - number of equations (nNodes*numdof)
c                                  numdof  - number of dofs per node
c                                   nirfm  - number of inertia relief modes
c                                   nnorm  - number of normal modes
c                                   ncstm  - number of constraint modes
c                           ptrIRFl,ptrIRFh  - 64 bit pointer to inertia relief modes
c                           ptrNORl,ptrNORh  - 64 bit pointer to normal modes
c                           ptrCSTl,ptrCSTh  - 64 bit pointer to constraint modes
c                                     0      - position not used

c   ---      i       1  nNodes      Nodal equivalence table. This table equates
c                                   the number used for storage to the actual
c                                   node number

c   NOR     dp   nnorm    neqn      Normal Modes

c   IRF     dp   nirfm    neqn      Inertia Relief Modes

c   CST     dp   ncstm    neqn      Constraint Modes
```

1.9. Description of the Full Stiffness-Mass File

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 ANSYS.  All Rights Reserved.
c *** ansys, inc.

c     **********  description of full stiffness-mass file  **********

      character*8  FULLNM
      parameter  (FULLNM='full    ')

c     *** NOTE: if this variable is changed in the future it should be
c     ***       updated in spdefines.h also for symbolic assembly (jrb)
      integer         FULLHDLEN
      parameter      (FULLHDLEN=160)

      LONGINT         fullfpL, fullfp
      integer         fullbk,  fullut, wrLdstep, wrSbstep, wrEqiter,
     x                wrOption
      common /fdfull/ fullfpL, fullbk, fullut,
     x                wrLdstep,wrSbstep,wrEqiter,wrOption
      equivalence  (fullfp,fullfpL)

c   ********** common variable descriptions **********
```

```
co  fullfpL      file position on file full
co  fullbk       block number for file full
co  fullut       file unit for file full

c     **********  file format (except for extopt=3,4) **********

c   See fddesc for documentation of how binary files are stored.

c     **********  file format  **********

c        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 the second file header.

c        type tells what kind of information is stored in this record:
c             i - integer
c             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    type    nrec    lrec     contents

c  ---      i       1      100      standard ANSYS file header (see binhed for
c                                   details of header contents)

c  ---      i       1      160      .FULL FILE HEADER
c
c                                  fun04,     neqn,    nmrow,   nmatrx,      kan,
c                                  wfmax,   lenbac,   numdof,  ntermKl,  ntermKh,   (10)
c                                  lumpm,    nmrow,   ntermK,   keyuns,   extopt,
c                                  keyse,   sclstf,   nxrows,  ptrSTFl,  ptrSTFh,   (20)
c                                ncefull,  ntermMh,  ptrENDl,  ptrENDh, ptrIRHSl,
c                               ptrIRHSh,  ptrMASl,  ptrMASh,  ptrDMPl,  ptrDMPh,   (30)
c                                 ptrCEl,   ptrCEh,   nNodes,  ntermMl,  ntermDl,
c                                ptrDOFl,  ptrDOFh,  ptrRHSl,  ptrRHSh,  ntermDh,   (40)
c                                ngMaxNZ, ptrNGPHl, ptrNGPHh, minKdiag, maxKdiag,
c                               minMdiag, maxMdiag, minDdiag, maxDdiag,  ngTerml,   (50)
c                                ngTermh, ngTermCl, ngTermCh,ptrDIAGKl,ptrDIAGKh,
c                              ptrDIAGMl,ptrDIAGMh,ptrDIAGCl,ptrDIAGCh, ptrSCLKl,   (60)
c                               ptrSCLKh,  Glbneqn,  distKey, ngTermFl, ngTermFh,
c                              GlbnNodes, GlbnVars, GlbfAcCE,  lcAcLen,   GlbfCE,   (70)
c                                ptrGmtl,  ptrGmth,nceGprime,numA12A11,GnVirtBCs,
c                                ntermGl,  ntermGh,ptrDensel,ptrDenseh, nVirtBCs,   (80)
c                              ptrVrtBCl,ptrVrtBCh,  ptrMRKl,  ptrMRKh, ptrKclxl,
c                               ptrKclxh, ntermKCl, ntermKCh,minKCdiag,maxKCdiag,   (90)
c                                  ngChg,   ptrBCl,   ptrBCh, ptrPHYSl, ptrPHYSh,
c                                predKey,fullDistF, ptrGVBCl, ptrGVBCh,    nzRow,  (100)
c                           localNonlKey,        0,        0,        0,        0,
c                                     0,        0,        0,        0,        0,  (120)
c                                     0,        0,        0,        0,        0,
c                                     0,        0,        0,        0,        0,  (140)
c                                     0,        0,        0,        0,        0,
c                                     0,        0,        0,        0,        0   (160)


c                                  each item in header is described below:

c                                   fun04  - negative of the unit number (-4)
c                                            NOTE: if fun04 is > 0 it means that
c                                                  frontal assembly was used (which
c                                                  is longer documented here)
c                                   neqn   - number of equations on file
c                                   nmrow  - number of active DOF (neqn-BC)
c                                   nmatrx - number of matrices on file
c                                   kan    - analysis type
c                                   wfmax  - maximum row size
c                                   lenbac - number of nodes
c                                   numdof - number of dofs per node
c                                   ntermKl,ntermKh - number of terms in Stiffness
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries and affiliates.

```
c                                       matrix
c                                  lumpm  - lumped mass key
c                                           = 0 - default matrix type
c                                           = 1 - lumped
c                                  ntermK - pre-8.1 this is the number of terms
c                                           in Stiffness matrix (otherwise this
c                                           value must be 0 and ntermKl,ntermKh
c                                           must be used)
c                                  keyuns - unsymmetric key
c                                           = 0 - no unsymmetric matrices on
c                                                 file
c                                           = 1 - there is at least one
c                                                 unsymmetric matrix on file
c                                  extopt - mode extraction method
c                                           = 0 - reduced
c                                           = 1 - lumped
c                                           = 3 - unsymmetric Lanczos
c                                           = 4 - damped Lanczos
c                                           = 6 - block Lanczos
c                                           = 7 - QRdamped
c                                           = 8 - SuperNode
c                                           = 9 - PCG Lanczos
c                                  keyse  - superelement key; set if at least
c                                         one superelement
c                                  sclstf - maximum absolute stiffness matrix term
c                                  nxrows - the maximum rank for this solution
c                                  ptrSTFl,h  - pointer to Stiffness matrix
c                                  ncefull    - number of CE+CP equations
c                                  ptrENDl    - low part of 64 bit end of file ptr
c                                  ptrENDh    - high part of 64 bit end of file ptr
c                                  ptrIRHSl,h - pointer to imaginary RHS (F)
c                                  ptrMASl,h  - pointer to Mass matrix
c                                  ptrDMPl,h  - pointer to Damping matrix
c                                  ptrCEl,h   - pointer to Gt and g matrices
c                                  nNodes     - number of nodes considered by assembly
c                                  ntermMl,h  - number of terms in Mass matrix
c                                  ntermDl,h  - number of terms in Damping matrix
c                                  ptrDOFl,h  - pointer to DOF info
c                                  ptrRHSl,h  - pointer to RHS (F)
c                                  ngMaxNZ    - maximum number of nodes per nodal
c                                               block in nodal graph structure
c                                  ptrNGPHl,h - pointer to vectors needed for
c                                               nodal graph structure
c                                  minKdiag - minimum absolute stiffness matrix
c                                             diagonal term
c                                  maxKdiag - maximum absolute stiffness matrix
c                                             diagonal term
c                                  minMdiag - minimum absolute mass matrix
c                                             diagonal term
c                                  maxMdiag - maximum absolute mass matrix
c                                             diagonal term
c                                  minDdiag - minimum absolute damping matrix
c                                             diagonal term
c                                  maxDdiag - maximum absolute damping matrix
c                                             diagonal term
c                                  ngTerml,h  - total number of nonzeroes in nodal graph
c                                               (expanded graph based value, no BC applied)
c                                  ngTermCl,h - total number of nonzeroes in nodal graph
c                                               (compressed graph based value)
c                                  ptrDIAGKl,h - pointer to stiffness matrix DIAGONAL vector
c                                                (NOTE: this is a copy of the diagonal
c                                                       values stored in the full matrix)
c                                  ptrDIAGMl,h - pointer to mass matrix DIAGONAL vector
c                                                (NOTE: this is a copy of the diagonal
c                                                       values stored in the full matrix)
c                                  ptrDIAGCl,h - pointer to damping matrix DIAGONAL vector
c                                                (NOTE: this is a copy of the diagonal
c                                                       values stored in the full matrix)
c                                  ptrSCLKl,h  - pointer to stiffness matrix diagonal scaling
c                                                vector (may contain all 1.0's when not scaling)
c                                  Glbneqn  - global number of active DOF (this will match nmrow
c                                             unless we are writing distributed "local" FULL files
```

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
c                                          in Distributed ANSYS
c                                   distKey  - key denoting whether the FULL file is a single,
c                                          global FULL file (0) or multiple, local FULL file (1)
c                                   ngTermFl,h - total number of nonzeroes in nodal graph
c                                            as passed to the solver (after BC applied)
c                                   GlbnNodes - global number of nodes considered by assembly
c                                   GlbnVars  - global number of equations (will match neqn
c                                            unless we are writing distributed "local" FULL files
c                                            in Distributed ANSYS
c                                   GlbfAcCE  - total number of across CE
c                                   lcAcLen   - number of acrossCE where slaves are in my domain
c                                   GlbfCE    - total number of all the CE
c                                   ptrGmtl,h - pointer to G prime matrix for local nonlinearity
c                                   nceGprime - number of CE (or equations) in G prime local nonlinearity
c                                   numA12A11 - number of equations in local nonlinear matrix: excluding
c                                            equations from G prime
c                                   GnVirtBCs - global number of virtual constraints
c                                   ntermGl,ntermGh - total number of terms in total local nonlinear
c                                                matrix including A12,A11 and G: total sum
c                                   ptrDensel,ptrDenseh - dense matrix information in local nonlinear
c                                                  matrix
c                                   nVirtBCs  - number of virtual constraints
c                                   ptrVrtBCl,ptrVrtBCh - pointer to the virtual constraint DOF data
c                                   ptrMRKl,h - pointer to the DOF marker array
c                                   0         - position not used
c                                   ptrKclxl,h - pointer to K complex (the 4th matrix) matrix
c                                            full case is: K, M, C, Kcplx: existing at same time
c                                   ntermKCl,h - number of terms in Complex Stiffness matrix
c                                   minKCdiag - minimum absolute complex stiffness matrix
c                                            diagonal term
c                                   maxKCdiag - maximum absolute complex stiffness matrix
c                                            diagonal term
c                                   ngChg     - key denoting whether or not the nodal graph written
c                                            to this FULL file differs from the previous FULL file
c                                   ptrBCl,h  - pointer to boundary condition data
c                                   ptrPHYSl,h - pointer to the physics marker array
c                                   predKey   - nonlinear analysis predictor key
c                                   fullDistF - indicate the global profiling of distributed full file
c                                   ptrGVBCl,ptrGVBCh - pointer to the global virtual constraint DOF data
c                                   nzRow - number of non-zero rows of the matrics for the full file under
c                                          global full file, -1 means not applicable
c                                   localNonlKey - local nonlinearity speedup key


c  ---      i       1      numdof   Degrees of freedom per node
c                                   DOF reference numbers are:
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  ****** 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
c                                   the number used for storage to the actual
c                                   node number

c  ---      i       1   Glblenbac   Global nodal equivalence table.  This record
c                                   EXISTS ONLY for distributed .full files.
c
c Stiffness Matrix. The next two records are repeated as a group neqn times.
c The pair of records will repeat GlbnVars times when model has across CEs.
c And row indices are global in the case.

c  STF      i       1      varies   Matrix row indices. The last item
c                                   corresponds to the diagonal. The
c                                   length of this record will vary (actual
c                                   length is returned from routine BINRD8)

c  ---   dp/cmp     1      varies   Matrix terms

c                                   If keyuns=0, this record will contain the
c                                   terms before the diagonal.
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries and affiliates.

```
c                                  If keyuns=1, this record will contain the
c                                  entire row.
c
c Load Vector

c  RHS   dp/cmp     1        neqn   Load vector terms.

c Imaginary part of Load Vector

c  IRHS    dp       1        neqn   Imaginary load vector terms.
c                                  This record EXISTS ONLY if its pointer in the header
c                                  is not zero. Record length will be GlbnVars for model
c                                  with across CE.

c Stiffness matrix diagonal vector

c  DIAGK   dp/cmp   1        neqn   diagonal vector data for stiffness matrix. Its length
c                                  will be GlbnVars for model with across CE.

c Stiffness matrix diagonal scaling vector

c  SCLK    dp/cmp   1        neqn   diagonal scaling vector for stiffness matrix. Record length
c                                  will be GlbnVars for across CE model.

c DOF marker array

c  MRK      i       1        neqn   marker array flagging various types of DOF
c                                  (1=U_EQN, 2=P_EQN, 3=E_EQN, 4=A_EQN).  Positive
c                                  values mean the DOF belongs to a user-defined node,
c                                  negative values mean the DOF belongs to an internal node.
c                                  Record length will be GlbnVars for models with across CE.

c PHYSICS marker array

c  PHYS     i       1        neqn   marker array flagging the various types of physics
c                                  (1=STRUCTURAL_EQN, 2=THERMAL_EQN, 3=ELECTRICAL_EQN,
c                                   4=MAGNETIC_EQN, 5=FLUID_EQN, 6=DIFFUSION_EQN)
c                                  Record length will be GlbnVars for models with across CE.

c DOF information

c  DOF      i       1      nNodes   Nodal extent vector. Number of DOFs at each node

c  ---      i       1   GlbnNodes   Global nodal extent vector giving numbers of DOFs at each
c                                  global nodes. This record EXISTS ONLY for the model with
c                                  across CE.

c  ---      i       1   GlbnNodes   A vector mapping global node number to local node number.
c                                  -1 in the vector means the node is not in this domain.
c                                  This record EXISTS ONLY for model with across CE.

c  ---      i       1        neqn   DOF vector. If negative, this DOF is constrained.

c  ---      i       1    GlbnVars   A vector of global DOF reference numbers.
c                                  this record EXISTS ONLY for the model with across CE.

c BC information

c  BC       i       1        neqn   DOFs with imposed values

c  ---   dp/cmp     1      varies   Imposed values

c Mass Matrix.
c    if lumpm = 0:
c      The next two records are repeated as a group neqn times.
c      It will be in global form the same way as stiffness matrix if model has across CE.

c  MAS      i       1      varies   Matrix row indices. The last item
c                                  corresponds to the diagonal. The
c                                  length of this record will vary (actual
c                                  length is returned from routine BINRD8)
```

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries and affiliates.

```
c  ---     dp       1      varies   Matrix terms

c    if lumpm = 1:
c  ---     dp       1        neqn   Matrix diagonals.
c                                   Record length will be GlbnVars for across CE model.

c Mass matrix diagonal vector

c  DIAGM   dp       1        neqn   diagonal vector data for mass matrix.
c                                   Record length will be GlbnVars for across CE model.

c Damping Matrix. The next two records are repeated as a group neqn times.
c For model with across CE, it will be in global form the same way as stiffness matrix.

c  DMP      i       1      varies   Matrix row indices. The last item
c                                   corresponds to the diagonal. The
c                                   length of this record will vary (actual
c                                   length is returned from routine BINRD8)

c  ---     dp       1      varies   Matrix terms

c Damping matrix diagonal vector

c  DIAGC   dp       1        neqn   diagonal vector data for damping matrix.
c                                   Record length will be GlbnVars for across CE model.

c K complex Matrix. The next two records are repeated as a group neqn times.
c For model with across CE, it will be in global form the same way as stiffness matrix.

c  KC      i       1       varies   Matrix row indices. The last item
c                                   corresponds to the diagonal. The
c                                   length of this record will vary (actual
c                                   length is returned from routine BINRD8)

c  ---     dp       1      varies   Matrix terms


c Nodal graph vectors

c  NGPH     i       1      nNodes   number of nonzeroes for each node.
c                                   Record length will be GlbnNodes for across CE model.

c   Repeat for each node

c           i       1      varies   Index vector. Node number in the vector is global when
c                                   model has across CE.

c G matrix if ncefull > 0.

c  CE       i       1     ncefull   List of slave DOFs of local CEs.
c                                   It EXISTS ONLY if ncefull>0. The slave DOF is local.

c  ---      i       1      lcAcCE   List of slave DOFs of local across CEs.
c                                   This record EXISTS ONLY if lcAcCE>0. The slave DOF is
c                                   local.

c  ---      i       1    GlbfAcCE   List of slave DOFs of all across CEs.
c                                   This record EXISTS ONLY if GlbfAcCE>0. And it is ONLY
c                                   in the full file of master domain.

c  ---      i       1      GlbfCE   List of slave DOFs of all CEs. This record EXISTS ONLY
c                                   if GlbAcCE>0.  And it is ONLY in the full file of the
c                                   master domain.

c  ---     dp       1     ncefull   g vector (constant terms) of local CEs. This record
c                                   EXISTS ONLY if ncefull>0.

c  ---     dp       1     ncefull   imaginary g vector (constant terms) of local CEs. This
c                                   vector only exists for FULL harmonic analyses (kan=3).
c                                   This record EXISTS ONLY if ncefull>0.

c  ---     dp       1     ncefull   g vector (constant terms) of local CEs for nonlinear
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries and affiliates.

```
c                                   analysis predictor logic.
c                                   This record EXISTS ONLY if ncefull>0 & predKey=1.

c  ---     dp       1    GlbfAcCE   g vector (constant terms) of across CEs. This record
c                                   EXISTS ONLY if GlbfAcCE>0 in the master full file.

c  ---     dp       1    GlbfAcCE   imaginary g vector (constant terms) of across CEs. This
c                                   vector only exists for FULL harmonic analyses (kan=3).
c                                   This record EXISTS ONLY if GlbfAcCE>0 in the master full file.

c  ---     dp       1    GlbfAcCE   g vector (constant terms) of across CEs for nonlinear
c                                   analysis predictor logic.  This record EXISTS ONLY
c                                   if GlbfAcCE>0 & predKey=1 in the master full file.

c  Following local CE data records EXIST ONLY in the full file of the domain with local CEs:
c  ---      i       1           4   Header for local CEs; 1=nRows, 2=nRows, 3=1, 4=0

c  ---      i       1       nRows   Vector of 1's

c  ---      i       1       nRows   Number of non-zero terms in each row for one local CE

c   Repeat for each row:

c  ---      i       1      varies   Column indices in local equation numbers

c  ---     dp       1      varies   Column values


c   Following across CE data records EXIST ONLY if GlbfAcCE>0 in the full file of master domain:
c  ---      i       1           4   Header for across CEs; 1=nRows, 2=nRows, 3=1, 4=0

c  ---      i       1       nRows   Vector of 1's

c  ---      i       1       nRows   Number of non-zero terms in each row for an across CE

c   Repeat for each row:

c  ---      i       1      varies   Column indices in global equation numbers

c  ---     dp       1      varies   Column values


c   Following CE data records EXIST ONLY if n>0 in the full file, where n == ncefull with
c   SMP and n == GlbfCE with DMP.  NOTE: for DMP these records only exist in the .full file
c   of master domain

c  ---      i       1           n   List of slave DOFs of all CEs. This record EXISTS ONLY
c                                   if n>0.  And it is ONLY in the full file of the master
c                                   domain.

c   NOTE: this matrix includes boundary d.o.f. which touch the CEs (for cnvfor.F)
c  ---      i       1           4   Header for across CEs; 1=nRows, 2=nRows, 3=1, 4=0

c  ---      i       1       nRows   Vector of 1's

c  ---      i       1       nRows   Number of non-zero terms in each row for an across CE

c   Repeat for each row:

c  ---      i       1      varies   Column indices in global equation numbers

c  ---     dp       1      varies   Column values



c Virtual constraint vector

c  VBC      i       1    nVirtBCs   marker array (bit 1 set -> constrained DOF for residual vector)
c                                                (bit 2 set -> constrained DOF for enforced motion)
c                                                (bit 3 set -> eliminated DOF for substructure master DOF)
c           i       1    nVirtBCs   virtual constraint DOFs
```

```
c  GVBC     i       1    GnVirtBCs  marker array (bit 1 set -> constrained DOF for residual vector)
c                                                (bit 2 set -> constrained DOF for enforced motion)
c                                                (bit 3 set -> eliminated DOF for substructure master DOF)

c                                   this record EXISTS ONLY for the model with across CE.

c           i       1    GnVirtBCs  virtual constraint DOFs
c                                   this record EXISTS ONLY for the model with across CE.


c  Meaning of K11, K12, and G matrices:
c     Given
c         [K]{x} = {F}
c     subject to the constraints
c           {x1} = [G]{x2} + {g}
c     where {x1} are the slave DOFs, {x2} the master DOFs

c     This results in
c         [K*]{x2} = {F*}
c     where
c         [K*] = [G]'[K11][G] + [G]'[K12] + [K21][G] + [K22]
c         {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) :
c         G' K11' g' = G' (K11,M11)*(g,gx)
c                    = G' [K11*g - M11*gx, M11*g + K11*gx]

c         K21' *g'   = (K21,M21)*(g,gx)
c                    = (K21*g- M21*gx, K21*gx + M21*g)
```

1.10. Description of the Substructure Displacement File

This section explains the contents of the substructure displacement file (`jobname.dsub`).

1.10.1. 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 13.

1.10.2. DSUB File Format

```
*comdeck,fddsub

c *** Copyright ANSYS.  All Rights Reserved.
c *** ansys, inc.

c     **********  description of substructure displacement file  ****

      character*8  DSUBNM
      parameter   (DSUBNM='dsub    ')

      LONGINT         dsubfpL, dsubfp
      integer         dsubbk, dsubut
      common /fddsub/ dsubfpL, dsubbk, dsubut
      equivalence  (dsubfp,dsubfpL)

c     ********************************************************
c     *                        CAUTION                       *
c     *   Please update  proc getDSUBInfo in SEManagement.eui   *
c     *   if the file format changes or GUI read of the DSUB    *
c     *   file will fail.                                      *
c     ********************************************************
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
c   ********** common variable descriptions ***********
co  dsubfpL      file position on file dsub
co  dsubbk       block number for file dsub
co  dsubut       file unit for file dsub

c  open:   slvstr
c  write:  setdis,ranbwv,eostrt,eofini,ranbas,slvstr,slvend,supsr1,supidx,supscl
c  close:  slvend
c  read:   matstr as .usub


c   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
c             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             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    type    nrec    lrec     contents

c  ---      i       1      100      standard ANSYS file header (see binhed for
c                                   details of header contents)

c   1       i       1       20      .DSUB FILE HEADER

c                                  fun13, fpeofS, fpeofL,    kcxp,  nmode,
c                                   knum,  kCXFM, senres,       0,      0,
c                                      0,      0,      0,       0,      0,
c                                      0,      0,      0,       0,      0,
c
c                           fun13   - file unit number
c                    fpeofS, fpeofL  - pointer to the eof
c                           kcxp    -
c                           nmode   -
c                           knum    - number of expanded modes
c                           senres  -

c      *** these records are repeated each iteration ***
c   2       i       1       50     fun13,    kan, lenbac,  numdof,   kcmplx,
c                                  itime,  itter, ncumit,  nitter,curdof(i),  (10)
c                                (curdof(i),i=1,numdof)
c                                (curdof(i),i=1,numdof)                      (20)
c                                (curdof(i),i=1,numdof)
c                                (curdof(i),i=1,numdof)                      (30)
c                                (curdof(i),i=1,numdof)
c                                (curdof(i),i=1,numdof)                      (40)
c                             curdof(i),      0,      0,       0,        0,
c                                      0,      0,      0,Glblenbac,  timint   (50)

c   3      dp       1       20    time/freq,(acel(i),i=1,3)

c      *** the following records are repeated for each superelement ***
c   4       i       1       20    iel,    nrow,   nvect,  ntrans, name1,
c                                 name2,  trok,   lrok,   name3,  name4,
c                                 name5,  name6,  name7,  name8,  kCXFM,
c                                 kdamp,      0,      0,      0,      0,
c
c                           iel    -   (iel=0 signals end of superelements
c                                    for this iteration)
c                           nrow   - number of dofs
c                           nvect  - number of load vectors
c                           ntrans - number of transformations
```

```
c                              name*  - name1/2/3/4/5/6/7/8 are the
c                                       inexc4 representation of the
c                                       32 character filename.
c                                       name3/4/5/6/7/8 will be "0"
c                                       for pre rev 5.2 files - cwa
c                              trok   - flag if transformations can be applied
c                              lrok   - flag if large deformation transformation
c                                       can be applied
c                              kCXFM  - key if CXFM transformations are present on file
c                              kdamp  - key if damping ratios:
c                                       rdamp(1),rdamp(2),dmprat,dmpst,spin
c                                       are present on file
c
c
c   5      dp       1      125    ntrans sets of transformations 15x5
c   6      dp       1      250    ntrans sets of transformations (CS) 50x5 - present if kCXFM = 1
c   7     LONG      1      nrow   (lL(i),i=1,nrow) - global dofs
c                                  (made LONGINT in version 14.0)
c   8      dp       1     nvect   (scalf(i),i=1,nvect)
c   9      dp       1        5    rdamp(1),rdamp(2),dmprat,dmpstr,spin     - present if kdamp = 1
c  10      dp       1      nrow   (disp(i),i=1,nrow)
c  11      dp       1      nrow   (vel(i),i=1,nrow)                        - present if senres >= 2
c  12      dp       1      nrow   (acel(i),i=1,nrow)                       - present if senres = 3
```


Release 18.2 - © ANSYS, Inc. All rights 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\V182\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/v182/ansys/customize/misc/<platform>/libbin.so` on Linux 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\V182\ANSYS\customize\user`. 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\V182\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\V182\ANSYS\custom\misc\<platform>\rdrwrt  ResRdDemo
```

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Appropriate files will then be copied from \Program Files\ANSYS Inc\V182\ANSYS\customize\user 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\V182\ANSYS\customize\user 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 Linux systems:

The Fortran source for these programs is located in /ansys_inc/v182/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/v182/ansys/customize/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/v182/ansys/customize/misc/rdrwrt.link  ResRdDemo
```

Appropriate files will then be copied from /ansys_inc/v182/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/v182/ansys/bin directory to run these programs, once linked. The procedure files are named ResRdDemo182, ResWrDemo182, rdsubs182, wrtsub182, and rdfull182.

Use the userprog command line option when writing your own customized program, naming the routine userprog.F. The resulting executable will be named userprog.e182. When userprog is used, no files are copied from /ansys_inc/v182/ansys/customize/misc to your working directory. The procedure file is named userprog182.

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.

2. 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.

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

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>1 on the command line), the **DUMP** command may not work as expected. In that case, 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 only
  out - output only
  inout - both an input and an output variable

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

```
*deck,binini
      subroutine binini (iott)
c *** primary function: initialize buffered binary i/o system
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 ***

c  input arguments:
c     iott      (int,sc,in)       - output unit number for error output

c  output arguments:   none
```

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

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:
c    variable (typ,siz,intent)    description
c    nunit    (int,sc,in)       - fortran unit number (used only for inqr='O')
c    fname    (chr,sc,in)       - name of file
c    lname_in (int,sc,in)       - length of file name (characters, max=50)
c    inqr_in  (chr,sc,in)       - character key for information requested
c                                  = 'E' - return whether file exists
c                                      sysiqr = 1 - file exists
c                                             = 0 - file does not exist
c                                             < 0 - error occured
c                                  = 'O' - return whether file is open
c                                      sysiqr = 1 - file is open
c                                             = 0 - file is closed
c                                             < 0 - error occured
c                                  = 'N' - return unit number of file
c                                      sysiqr > 0 - unit number for file
c                                             = 0 - file not assigned to a unit
c                                             < 0 - error occured

c  output arguments:
c    sysiqr   (int,func,out)    - the returned value of sysiqr is based on
c                                   setting of inqr
```

2.1.7. Function biniqr8 (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 ***

c  input arguments:
c     nblk      (int,sc,in)       - the block number for the inquiry
c                                   or zero (see below)
c     key       (int,sc,in)       - key for information requested
c                       nblk = 0 - return information about system/file
c                                key = 1 - return system block size
c                                    = 2 - return number of integers per dp
c                                    = 3 - return filename length
c                                      5 = return integers per LONG
c                       nblk > 0 - return information about this block
c                                key = 1 - return fortran unit number
c                                    = 2 - return number of pages in file
c                                    = 3 - return length of page (32 bit words)
c                                    = 4 - return open status
c                                          0 - file close
c                                          1 - file open
c                                    = 5 - return file format
c                                          0 - internal format
c                                          1 - external format
c                                    = 6 - return read/write status
c                                          0 - both read & write
c                                          1 - read
c                                          2 - write
c                                    = 7 - return current position on file
c                                    = 8 - return maximum length of file
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries and affiliates.

```
c                                          (in words)
c                                    = 9 - return starting word for this page
c                                          in buffer
c                                    =10 - return base location
c                                    =11 - return debug key
c                                    =12 - return absolute (non-base) key
c                                    =15 - return max record written
c                                    =16 - return swap and record header key
c                                    =17 - return precision key

c  output arguments:
c     biniqr    (int,func,out)    - the returned value of biniqr is based on
c                                     setting of nblk and key
```

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,
     x                 nchar,kext,Buffer4)

c *** primary function: initialize paging space for a blocked binary file.
c                       binset should be used to open a blocked file
c                       before binrd8 or binwrt8 are used.  binclo should
c                       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 ***

c  input arguments:
c     nblk      (int,sc,in)       - block number (1 to BIO_MAXFILES max)
c     nunit      (int,sc,in)      - fortran unit number for the file
c                                   (if 0, bit bucket)
c     ikeyrw    (int,sc,in)       - read/write flag
c                                   =  0 - both read & write
c                                   =  1 - read
c                                   =  2 - write
c                                   =  9 - read only
c       NOTE: 0 may write, but the file length may not be extended and
c               the file may or may not exist
c             1 reads only, but the file protection must set set to "rw"
c             2 may extend the file length and the file is a new file
c             9 reads only, but the file protection may be "r" only
c     istart    (int,sc,in)       - starting location in buffer array
c                                   usually 1 for nblk=1, paglen*npage+1
c                                   for nblk=2,etc.
c     paglen    (int,sc,in)       - page length in integer*4 words for external
c                                   files
c                                   paglen should always be a multiple of
c                                   512 words for efficiency
c     npage     (int,sc,in)       - number of pages (1 to BIO_MAXBLOCKS max)
c     pname     (chr,ar(*),in)    - name of the file
c     nchar     (int,sc,in)       - number of characters in the file name (not
c                                   used)
c     kext      (int,sc,in)       - no longer used, always external format
c     Buffer4   (i4, ar(*),inout) - work array for paging, should be
c                                   dimensioned to paglen*npage*nblk (max)

c  output arguments:
c     binset    (int,func,out)    - error status
c                                   = 0 - no error
c                                   <>0 - error occurred
c     Buffer4   (i4, ar(*),inout) - work array for paging
```

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

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
c --- This routine is intended to be used in standalone programs.
c --- This routine should not be linked into the ANSYS program.

c
c *** Notice - This file contains ANSYS Confidential information ***
c
c     typ=int,dp,log,chr,dcp   siz=sc,ar(n)   intent=in,out,inout
c
c  input arguments:
c     variable (typ,siz,intent)    description
c     title    (chr*80,ar(2),in) - main title and 1st subtitle
c     jobnam   (chr*8,sc,in)     - jobname
c     units    (int,sc,in)       - units
c                                    = 0 - user defined units
c                                    = 1 - SI
c                                    = 2 - CSG
c                                    = 3 - U.S. Customary, using feet
c                                    = 4 - U.S. Customary, using inches
c                                    = 5 - MKS
c                                    = 6 - MPA
c                                    = 7 - uMKS
c     code     (int,sc,in)       - code defining 3rd party vendor
c                                    (contact ANSYS, Inc. for code assignment)
c
c  output arguments:
c     none
c
```

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
c                          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 ***

c  input arguments:
c     nblk     (int,sc,in)       - block number of open binary file
c                                    (as defined with subroutine binset)
c     nunit    (int,sc,in)       - the unit number for this file
c     buffer   (int,ar(*),inout) - work array for paging, should be the
c                                    same array as used in binset

c  output arguments:
c     filpos   (int,sc,out)      - the position after the header
c     buffer   (int,ar(*),inout) - work array for paging

c   **********  ANSYS standard header data description (100 words)  **********
c  loc   no. words   contents
c   1       1        fortran unit number
c   2       2        file format
c                    = 0 - internal format
c                    = 1 - external format
c   3       1        time in compact form (ie 130619 is 13:06:19)
c   4       1        date in compact form (ie 19981023 is 10/23/1998)
c   5       1        units
c                    = 0 - user defined units
c                    = 1 - SI
c                    = 2 - CSG
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
c                   = 3 - U.S. Customary, using feet
c                   = 4 - U.S. Customary, using inches
c                   = 5 - MKS
c                   = 6 - MPA
c                   = 7 - uMKS
c   6       1       User_Linked
c  10       1       revision in text format ' 5.0' (inexc4)
c  11       1       date of revision release for this version
c  12       3       machine identifier - 3 4-character strings
c  15       2       jobname - 2 4-character strings
c  17       2       product name - 2 4-character strings
c  19       1       special version label - 1 4-character string
c  20       3       user name - 3 4-character strings
c  23       3       machine identifier - 3 4-character strings
c  26       1       system record size at file write
c  27       1       maximum file length
c  28       1       maximum record number
c  31       8       jobname - 8 4-character strings
c  41      20       main title - 20 4-character strings
c  61      20       first subtitle - 20 4-character strings
c  95       1       split point of file
c                    NOTE: Split files are not support by binlib!
c 97-98     2       LONGINT of file size at write
```

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 ***

c  input arguments:
c   nblk   (int,sc,in)     - block number.  see fd___(i.e. fdtri for tri
c
c   LongLocL(LONG,sc,inout)- location in integer*4 words of the startin
c                            position on the file.
c   leng   (int,sc,inout)  - number of words to read into ivect. (must be
c                            less or equal to dimension given to ivect in
c                            the calling routine).  if ivect is to be used
c                            as integers, use as is.  if ivect is to be
c                            used for double precision numbers, it must be
c                            increased by multiplying it by INTPDP.
c                            if negative, skip record and do not return
c                            data(results).
c   Buffer4 (i4,ar(*),inout) - work array for paging, should be the
c                              same array as used in binset

c  output arguments:
c   LongLocL(LONG,sc,inout)- location in integer*4 words of the current
c                            position on the file.  It is updated after
c                            each read operation
c   leng   (int,sc,inout)  - tells you how many items it actually read(in
c                            integer words).
c                            if zero, end of file(error case)
c   ivect  (int,ar(*),out) - results (can be either integer or double
c                            precision in the calling routine)
c   kbfint (int,sc,out)    - key for type(used only for AUX2 dump)
c                           = 0  double precision data
c                           > 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
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 ***

c  input arguments:
c   nblk   (int,sc,in)     - block number.  see fd___(i.e. fdtri for tri
c
c   LongLocL(LONG,sc,inout)- location in integer words of the starting
c                            position on the file.
c   leng   (int,sc,in)     - number of words to read from ivect. (must be
c                            less or equal to dimension given to ivect in
c                            the calling routine).  if ivect is to be used
c                            as integers, use as is.  if ivect is to be
c                            used for double precision numbers, it must be
c                            increased by multiplying it by INTPDP.
c   ivect  (int,ar(*),in)  - data to be written onto the file(can be either
c                            integer or double precision in the calling
c                            routine)
c   kbfint (int,sc,in)     - key for type(used only for AUX2 dump)
c                           = 0  double precision data
c                           > 0  integer data(usually the same as leng)
c   Buffer4 (int,ar(*),inout) - work array for paging, should be the
c                                  same array as used in binset on this
c                                  block

c  output arguments:
c   LongLocL(LONG,sc,inout)- location in integer words of the current
c                            position on the file.  It is updated after
c                            each write operation
c   ivect  (int,ar(*),out) - vector containing record to be written
c   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
c                   strings to plain 4-character strings (used to convert data
c                   from externally formatted files to data for interally
c                   formatted files)
c
c *** Notice - This file contains ANSYS Confidential information ***
c
c  input arguments:
c     ichext    (int,ar(n),in)    - externally formatted integer form of
c                                   4-character strings
c     n         (int,sc,in)       - number of strings to convert
c
c  output arguments:
c     chin     (char,ar(n),out)  - strings in character form
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
c

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
c                   integer versions of 4-character strings (used to convert
c                   data from internally formatted files to data for
c                   externally formatted files)
c
c *** Notice - This file contains ANSYS Confidential information ***
c
c  input arguments:
c     chin      (char,ar(n),in)   - strings in character form
c     n         (int,sc,in)       - number of strings to convert
c
c  output arguments:
c     ichext    (int,ar(n),out)   - externally formatted integer form of
c                                   4-character strings
c
c *** mpg inexc4 < sectype ansres anssav : ch4 -> int conversion
c
```

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
c                      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 ***

c  input arguments:
c     nblk      (int,sc,in)       - the block number to close
c                                   (as defined with subroutine binset)
c     pstat     (chr,sc,in)       - keep or delete flag
c                                   = 'K' - keep file
c                                   = 'D' - delete file
c     Buffer4   (int,ar(*),inout) - work array for paging, should be the
c                                   same array as used in binset

c  output arguments:
c     Buffer4    (int,ar(*),inout) - work array for paging
```

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

```
*deck,largeIntGet
      function largeIntGet (small,large)

c primary function:    Convert two short integers into a long integer

c object/library:  res

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

c  input arguments:
```

```
c     small    (int,sc,in)       - least significant part
c     large    (int,sc,in)       - most significant part

c  output arguments:
c     largeIntGet  (LONGINT,sc,out)  - 64 bit integer
```

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. 63) 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
c *** Notice - This file contains ANSYS Confidential information ***
c
c Copyright ANSYS.  All Rights Reserved.
c *** ansys, inc.
c
c     typ=int,dp,log,chr,dcp   siz=sc,ar(n)   intent=in,out,inout
c
c  input arguments:
c     variable  (typ,siz,intent)    description
c     pname     (chr,sc,in)       - name of binary file which is to
c                                   be dumped to the screen
c
c  output arguments:
c     none.
c
c  common variables:
c     iout      (int,sc,comm)     - output unit number
c     intpdp    (int,sc,comm)     - number of integers per double precision word
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

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

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

```
*deck,wrtsub
      program wrtsub
```

```
c primary function:    demonstrates use of binary access routines
c secondary function:  write an ANSYS substructure file

c *** Notice - This file contains ANSYS Confidential information ***
c
c *** Copyright ANSYS.  All Rights Reserved.
c *** ansys, inc.
c
c  ********************************************************************
c  *  Program to write ANSYS substructure file with usrsub.  To be     *
c  *  used as a base for 3rd party companies to create their routines  *
c  *  for writing the file.                                            *
c  ********************************************************************
```

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
c *** Copyright ANSYS.  All Rights Reserved.
c *** ansys, inc.
c
c  ********************************************************************
c  *  Reads a substructure file.  To be used as base for 3rd party     *
c  *  development of routines for reading ANSYS substructure files.    *
c  *  This demonstration program can handle up to:                     *
c  *   MAXNODE Nodes and MAXDOF DOFs                                   *
c  ********************************************************************
```

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 for most analysis types that use the sparse solver. 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
c *** Copyright ANSYS.  All Rights Reserved.
c *** ansys, inc.
c
c
c  ************************************************************
c  *  Reads a FULL file.  To be used as base for 3rd party      *
c  *  development of routines for reading ANSYS FULL files.     *
c  *  This demonstration program can handle up to:              *
c  *   MAXNODE nodes and MAXEQN equations                       *
c  ************************************************************
c
c
c  ************************************************************
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
c  *************************************************************
c  NOTICE - This program demonstrates how to read a .FULL file.

c      If the free-free stiffness and mass matrices are desired,
c        make sure there are no constraints on the model.
c  *************************************************************
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 (p. 75). 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\V182\ANSYS\customize\user` (on Windows systems) or `/ansys_inc/v182/ansys/customize/misc` (on Linux 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\V182\ANSYS\customize\user` (on Windows systems) or `/ansys_inc/v182/ansys/customize/misc` (on Linux systems).

2.3. Results File Access Routines

You can use the low-level routines described in Accessing ANSYS Binary Files (p. 63) 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\V182\ANSYS\customize\user` (on Windows systems) or `/ansys_inc/v182/ansys/customize/misc` (on Linux systems). See Access Routines to Results, Substructure, and Matrix Files (p. 63) 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 `ResRd`*`record`*`Begin`, where *`record`* is a descriptive name of the record, e.g., `ResRdNodeBegin`
- Read the data itself. These are named `ResRd`*`record`*, e.g., `ResRdNode`
- Deallocate space for the data. These are named `ResRd`*`record`*`End`, 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
   subroutine ResRdSectMatBegin
     subroutine ResRdSectBegin
       function ResRdSect
     subroutine ResRdSectEnd
     subroutine ResRdMatBegin
       function ResRdMat
     subroutine ResRdMatEnd
   subroutine ResRdSectMatEnd
   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 (p. 75) on the distribution medium for an example of the usage of these routines.

The memory allocation scheme is described in Memory Management Routines (p. 250) in Part II: Guide to 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,
      x                     Units, NumDOF, DOF, UserCode,
      x                     MaxNode, NumNode, MaxElem, NumElem,
      x                     MaxResultSet,NumResultSet)
 c primary function:    Open result file and return global information

 c object/library: ResRd

 c  input arguments:
 c     Nunit    (int,sc,in)       - Fortran Unit number for file (ANSYS uses 12)
 c     Lunit    (int,sc,in)       - Current print output unit (usually 6 <STDOUT>)
 c     Fname    (ch*(ncFname),sc,in) - The name (with extension) for the file
 c     ncFname  (int,sc,in)       - Number of characters in Fname

 c  output arguments:
 c     Title    (ch*80,ar(2),out) - Title and First subtitle
 c     JobName  (ch*32,sc,out)    - Jobname from file
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
c     Units    (int,sc,out)      - unit system
c                                  = 0 - user defined units
c                                  = 1 - SI
c                                  = 2 - CSG
c                                  = 3 - U.S. Customary, using feet
c                                  = 4 - U.S. Customary, using inches
c                                  = 5 - MKS
c                                  = 6 - MPA
c                                  = 7 - uMKS
c     NumDOF   (int,sc,out)      - Number of DOF per node
c     DOF      (int,ar(*),out)   - The DOFs per node
c     UserCode (int,sc,out)      - Code for this application
c     MaxNode  (int,sc,out)      - Maximum node number used
c     NumNode  (int,sc,out)      - Number of nodes attached to elements
c     MaxElem  (int,sc,out)      - Maximum element number used
c     NumElem  (int,sc,out)      - Number of elements used
c     MaxResultSet (int,sc,out)  - Maximum number of result sets (usually 1000)
c     NumResultSet (int,sc,out)  - Number of result sets on file
c     ResRdBegin (int,sc,out)    - 0, successful  other, error in file open
```

2.3.3. ResRdGeomBegin (Retrieving Global Geometry Information)

```
*deck,ResRdGeomBegin
      subroutine ResRdGeomBegin (MaxType, MaxReal, MaxCsys, nXYZ)
c primary function:    Read Geometry Header Record

c object/library: ResRd

c  input arguments:  none

c  output arguments:
c     MaxType  (int,sc,out)       - Maximum element type
c     MaxReal  (int,sc,out)       - Maximum real constant set number
c     MaxCsys  (int,sc,out)       - Maximum coordinate system number
c     nXYZ     (int,sc,out)       - number of nodes with coordinates
```

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

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

2.3.5. ResRdReal (Retrieving Real Constants)

```
*deck,ResRdReal
      function ResRdReal (iReal,Rcon)
c primary function:    Read real constant record

c object/library: ResRd

c  input arguments:
```

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries and affiliates.

```
c     iReal    (int,sc,in)             - Real set number

c  output arguments:  none
c     Rcon      (dp,ar(ResRdReal),out) - Real Constants
c     ResRdReal (int,sc,out)           - Number of real constants in set
```

2.3.6. ResRdCsys (Retrieving Coordinate Systems)

```
*deck,ResRdCsys
      function ResRdCsys (iCsys,Csys)
c primary function:    Read a coordinate system record

c object/library: ResRd

c  input arguments:
c     iCsys     (int,sc,in)           - Coordinate system number

c  output arguments:
c     Csys      (dp,ar(ResRdCsys),out)- Coordinate system description
c     ResRdCsys (int,sc,out)          - Number of values

c  output arguments:
```

2.3.7. ResRdNode (Retrieving Nodal Coordinates)

```
*deck,ResRdNode
      function ResRdNode (iNode,xyzang)
c primary function:    Get a node

c object/library: ResRd

c  input arguments:
c     iNode     (int,sc,in)     - node sequence number
c                                  (1 to nXYZnode)

c  output arguments:
c     xyzang    (dp,ar(6),out)  - x,y,z,thxy,thyz,thzx for node
c     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

c  input arguments:
c     iElem    (int,sc,in)       - The element number

c  output arguments:
c     ResRdElem(int,sc,out)      - Number of nodes
c     nodes    (int,ar(n),out)   - Element nodes
c     ElemData (int,ar(10),out)  - Element information
c                                     mat    - material reference number
c                                     type   - element type number
c                                     real   - real constant reference number
c                                     secnum - section number
c                                     esys   - element coordinate system
c                                     death  - death flag
c                                              = 0 - alive
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries and affiliates.

```
c                                          = 1 - dead
c                                  solidm - solid model reference
c                                  shape  - coded shape key
c                                  elnum  - element number
c                                  pexcl  - P-Method exclude key
```

2.3.9. ResRdSectMatBegin (Retrieving Global Section and Material Information)

```
*deck,ResRdSectMatBegin
      subroutine ResRdSectMatBegin (MaxSect, MaxMat)
c primary function:    Read maximum section and material number
c                      from the Geometry Header Record

c object/library: ResRd

c  input arguments:  none

c  output arguments:
c     MaxSect  (int,sc,out)       - Maximum section number
c     MaxMat   (int,sc,out)       - Maximum material number
```

2.3.10. ResRdSect (Retrieving Section Data)

```
*deck,ResRdSect
      function ResRdSect (iSect,SecData)
c primary function:    Read section record

c object/library: ResRd

c  input arguments:
c     iSect    (int,sc,in)             - Section set number

c  output arguments:
c     SecData   (dp,ar(ResRdSect),out) - Section data
c     ResRdSect (int,sc,out)           - Number of section data in set
```

2.3.11. ResRdMat (Retrieving Material Data)

```
*deck,ResRdMat
      function ResRdMat (iMat,iprop,MatData)
c primary function:    Read material record

c object/library: ResRd

c  input arguments:
c     iMat     (int,sc,in)             - Material set number
c     iprop    (int,sc,in)             - Property reference number
c                                        See mpinqr for details

c  output arguments:
c     MatData   (dp,ar(ResRdMat),out)  - Material data for type iprop
c     ResRdMat  (int,sc,out)           - Number of material data in set
```

See mpinqr Function (Getting Information About a Material Property) (p. 267) for details on the property reference number (iprop).

2.3.12. ResRdSolBegin (Retrieving Result Set Location)

```
*deck,ResRdSolBegin
      function ResRdSolBegin (key,lstep,substep,ncumit,kcmplx,time,
     x                        Title,DofLab)
c primary function:    Read the solution header records

c object/library: ResRd

c  input arguments:
c     key      (int,sc,in)       - 0, find by set number
c                                  1, find by lstep/substep
c                                  2, find by ncumit
c                                  3, find by time
c     lstep    (int,sc,in/out)   - Load step number
c                                   if key=0, this is the set number
c     substep  (int,sc,in/out)   - Substep of this load step
c     ncumit   (int,sc,in/out)   - Cumulative iteration number
c     kcmplx   (int,sc,in)       - 0, Real solution   1, Imaginary solution
c     time     (dp,sc,in/out)    - Current solution time

c  output arguments:
c     Title    (ch*80,ar(5),out)  - Title and 4 subtitles
c     DofLab   (ch*4,ar(nDOF),out)- Labels for DOFs
c     ResRdSolBegin (int,sc,out)  - 0, requested solution set found
c                                   1, not found
```

2.3.13. ResRdDisp (Retrieving Nodal Solution)

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

c object/library: ResRd

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

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

2.3.14. ResRdRfor (Retrieving Reaction Solution)

```
*deck,ResRdRfor
      function ResRdRfor (node,idof,value)
c primary function:    Retrieve a reaction force

c object/library: ResRd

c  input arguments:
c     node     (int,sc,in)       - External node number
c     idof     (int,sc,in)       - Internal dof number

c  output arguments:
c     value     (dp,sc,in)       - Value of reaction force
c     ResRdRfor (int,sc,out)     - Number of returned values (0 or 1)
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries and affiliates.

2.3.15. 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:
c     node     (int,sc,in)       - External node number
c     idof     (int,sc,in)       - Internal dof number

c  output arguments:
c     value    (dp,ar(4),in)     - Real,Imag, RealOld,ImagOld
c     ResRdFix (int,sc,out)      - Number of returned values (0 or 4)
```

2.3.16. 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:
c     node      (int,sc,in)       - External node number
c     idof      (int,sc,in)       - Internal dof number

c  output arguments:
c     value     (dp,ar(4),in)     - Real,Imag, RealOld,ImagOld
c     ResRdForc (int,sc,out)      - Number of returned values (0 or 4)
```

2.3.17. ResRdEstr (Retrieving Element Solutions)

```
*deck,ResRdEstr
      function ResRdEstr (iStr,Str)
c primary function:   Get an element's results

c object/library: ResRd


c  input arguments:
c     iStr    (int,sc,in)       - element record number (1-25)

c  output arguments:
c     ResRdEstr (int,sc,out)       - Number of element values
c     Str       (dp,ar(nStr),out)  - element values
```

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries and affiliates.

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries and affiliates.

Chapter 3: The CDWRITE (CDB) File Format

This chapter discusses how to write a coded database file, Jobname.CDB, that can be used to export a model from ANSYS into another application. The Jobname.cdb 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.

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

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 Number	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

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.

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

The **CDWRITE** command has an UNBLOCKED and a BLOCKED option. The UNBLOCKED option writes all data out in command format. The default BLOCKED option writes certain data items in a fixed format, including those that 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:

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
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 `Jobname.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.

The **CMBLOCK** command is not valid in a ***DO** loop.

3.2.4. CYCLIC Command

The **CYCLIC**,CDWR command defines the input and output of a cyclic symmetry analysis. The syntax is:

```
CYCLIC,CDWR,Value1,Value2,Value3,...
```

The following describes the values written to the `.CDB` file for cyclic options **CYCLIC**,CDWR:

Value1 **= 1**

Value2
Number of cyclic sectors

Value3
Number of solutions in cyclic space

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Value4
Harmonic index of this load

Value5
Cyclic coordinate system

Value6

< 0 or Static: only solve for given harmonic indices from **CYCOPT**,HIND
> 0: tolerance for the Fourier load

***Value1* = 2**

Value2
Cyclic edge type (0 = undefined; 1 = areas; 10 = lines; 100 = keypoints; 1000 = nodes)

Value3
0 or blank

Value4
Maximum possible harmonic index

Value5
Force load coordinate system (1 = global coordinate system; 0 = cyclic coordinate system)

Value6
Inertia load coordinate system (1 = global coordinate system; 0 = cyclic coordinate system)

***Value1* = 3 - 22**

Value2 - Value6
Cyclic edge constraint equation/coupling degree of freedom (DOF) (0 = all available DOF; otherwise bitmap) for pair IDs 1-5

(Repeat as necessary for other pair IDs (*Value1* = 4 - 22))

***Value1* = 23 - 30**
Cyclic harmonic index bit bins (each bin holds 32 harmonic indices by 5 containers corresponding to *Value2* - *Value6*)

Value2 - Value6
Cyclic harmonic index bits (0 = solve for harmonic index; nonzero values indicate skipped harmonic indices)

***Value1* = 31**

Value2
Max node number in base sector

Value3
Max element number in base sector

Value4
Number of defined nodes in base sector

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Value5
Number of defined elements in base sector

***Value1* = 32**

Value2
/CYCEXPAND number of sectors to expand (total)

Value3
Number of edge component pairs

Value4 - Value8
/CYCEXPAND number of sectors to expand (per window)

***Value1* = 33**
Not used

***Value1* = 34**
Cyclic **CSYS** coordinate system integer data (part 1)

Value2
Theta singularity key

Value3
Phi singularity key

Value4
Coordinate system type

***Value1* = 35**
Cyclic **CSYS** coordinate system integer data (part 2)

Value2
Coordinate system number

Value3
Not used (defaults to 0)

Value4
Not used (defaults to 0)

***Value1* = 36**

Value2
Number of user-defined cyclic edge pair components

Value3
Rotate cyclic edge nodes into cyclic coordinate system (0 = rotate edge nodes (default); 1 = do not rotate edge nodes)

Value4
NLGEOM flag (0 = no **NLGEOM** effects (default); 1 = include **NLGEOM** effects)

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Value5
Sector edge display key (-1 = suppresses display of edges between sectors even if the cyclic count varies between active windows; 0 = averages stresses or strains across sector boundaries. This value is the default (although the default reverts to 1 or ON if the cyclic count varies between active windows); 1 = no averaging of stresses or strains occurs and sector boundaries are shown on the plot)

***Value1* = 101**

Value2
Sector angle (degrees)

Value3
XYZ tolerance input for matching low/high nodes

Value4
Angle tolerance for matching low/high nodes (degrees)

Value5
Tolerance in the element coordinate system for unequal meshes

***Value1* = 102 - 104**
Cyclic **CSYS** coordinate system double precision data (part 1)

Value2* - *Value4
Coordinate system transformation matrix (total of 9 values)

***Value1* = 105**
Cyclic **CSYS** coordinate system double precision data (part 2)

Value2* - *Value4
Origin location (XYZ)

***Value1* = 106**
Cyclic **CSYS** coordinate system double precision data (part 3)

Value2
Used for elliptical, spheroidal, or toroidal systems. If CSYS = 1 or 2, *Value2* is the ratio of the ellipse Y-axis radius to X-axis radius (defaults to 1.0 (circle))

Value3
Used for spheroidal systems. If CSYS = 2, *Value3* is the ratio of ellipse Z-axis radius to X-axis radius (defaults to 1.0 (circle))

***Value1* = 107**
Cyclic **CSYS** coordinate system double precision data (part 4)

Value2
First rotation about local Z (positive X toward Y)

Value3
Second rotation about local X (positive Y toward Z)

Value4
Third rotation about local X (positive Z toward X)

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Value1 **= 121**

Value2
Root of component names defining low and high ranges

Value1 **= 122**

Value2
Cyclic low/high xref array parameter name (node)

Value1 **= 123**

Value2
Cyclic low/high xref array parameter name (line)

Value1 **= 124**

Value2
Cyclic low/high xref array parameter name (area)

Value1 **= 125**

Value2
The component name of the elements to expand (see **/CYCEXPAND**,,WHAT)

Value1 **= 201**

Value2
Total number of modes extracted during a cyclic modal solve. This value is only available after call to **CYCCALC**.

Value3
Mode superposition flag to limit results written to `.MODE` and `.RST` files

Value4
Excitation engine order

Value1 **= 202**

Value2
Type of mistuning (1 = stiffness; 2 = mass; 3 = both; -1 = use user macro CYCMSUPUSERSOLVE)

Value3
Cyclic mode superposition restart flag (1 = new frequency sweep; 2 = new mistuning parameters; -1 = form blade superelement and stop)

Value4
Cyclic mode superposition key to perform complex modal analysis of reduced system

Value5
Number of CMS modes for mistuned reduced order model (see **CYCFREQ**,BLADE)

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

***Value1* = 203**

Value2
Array name for aerodynamic coupling coefficients

***Value1* = 204**
Unused

***Value1* = 205**

Value2
The name of the nodal component containing the blade boundary nodes at the blade-to-disk interface (see **CYCFREQ**,BLADE). This is used for cyclic mode superposition analyses that include mistuning or aero coupling.

***Value1* = 206**

Value2
The name of the element component containing the blade superelements (see **CYCFREQ**,BLADE). This is used for cyclic mode superposition analyses that include mistuning or aero coupling.

***Value1* = 207**

Value2
The name of the array holding stiffness mistuning parameters

***Value1* = 208**
Unused

***Value1* = 209**
Rotational velocity from the base linear perturbation analysis.

Value2
X-component of rotational velocity

Value3
Y-component of rotational velocity

Value4
Z-component of rotational velocity

***Value1* = 210**

Value2
Beginning of frequency range for CMS modes (see **CYCFREQ**,BLADE). This is used for cyclic mode superposition analyses that include mistuning or aero coupling.

Value3
End of frequency range for CMS modes (see **CYCFREQ**,BLADE). This is used for cyclic mode superposition analyses that include mistuning or aero coupling.

***Value1* = 211**

Value2
Number of modes for a cyclic mode superposition damped modal solve

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Value3
Beginning of frequency range for cyclic mode superposition damped modal solve

Value4
End of frequency range for cyclic mode superposition damped modal solve

3.2.5. 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 - *Not used.*
- 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.

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

- 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.).

The **EBLOCK** command is not valid in a ***DO** loop.

3.2.6. EDCADAPT Command

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

```
EDCADAPT,R5.3,FREQ,TOL,OPT,MAXLVL,BTIME,DTIME,LCID,ADPSIZE,ADPASS,IREFLG,ADPENE,ADPTH,MAXEL
```

FREQ
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.

IREFLG
The uniform refinement level flag.

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

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

ADPTH
Absolute shell thickness level below which adaptivity should begin.

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

This command also appears in the *Command Reference*. The format shown here contains information specific to the **CDREAD**/**CDWRITE** file.

3.2.7. 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.

BTIME
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

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

BOXID2
Target volume as defined using EDBOX

This command also appears in the *Command Reference*. The format shown here contains information specific to the **CDREAD**/**CDWRITE** file.

3.2.8. 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.

This command also appears in the *Command Reference*. The format shown here contains information specific to the **CDREAD**/**CDWRITE** file.

3.2.9. 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.

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

DFFCTR
The dynamic relaxation factor.

DRTERM
The termination time for dynamic relaxation.

TSSFDR
The scale factor for each computed time step.

This command also appears in the *Command Reference*. The format shown here contains information specific to the **CDREAD**/**CDWRITE** file.

3.2.10. 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.

This command also appears in the *Command Reference*. The format shown here contains information specific to the **CDREAD**/**CDWRITE** file.

3.2.11. 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.

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

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).

LCID
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.

This command also appears in the *Command Reference*. The format shown here contains information specific to the **CDREAD**/**CDWRITE** file.

3.2.12. 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:

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
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.

3.2.13. 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.

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

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.

This command also appears in the *Command Reference*. The format shown here contains information specific to the **CDREAD**/**CDWRITE** file.

3.2.14. 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
```

Type
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.

I1,I2,I3,I4I5,I6,I7,I8
The integer values to be read:

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

- 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.

3.2.15. 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
```

Type
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.

3.2.16. M Command

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

```
M,R5.0,NODE,Dlab
```

NODE
The node number

Dlab
The DOF label

3.2.17. 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.

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

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*.

3.2.18. 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*

3.2.19. 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
```

Type
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.

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

3.2.20. 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,       849,       723
(3i9,6e21.13e3)
        1        0        0 8.7423930292124E-001 7.1843141243360E-001 8.2435547360131E-001
        3        0        0 9.2314873336026E-001 9.3459943382943E-001 4.8406643591666E-001
        4        0        0 1.1410427242574E+000 7.6883495387624E-001 2.5867801436812E-001
  .
  .
  .
      847        0        0 6.2146469267794E-001 8.0122597436764E-001 8.1352232529497E-001
      848        0        0 8.1179373384170E-001 6.6711479947438E-001 7.6547291135454E-001
      849        0        0 7.4952223718564E-001 7.6089019544242E-001 7.4112247735703E-001
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.).

The **NBLOCK** command is not valid in a ***DO** loop.

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

3.2.21. *PREAD Command

```
*PREAD,Par,NUMVALS
Format
END PREAD
```

Par
Alphanumeric name to identify this parameter.

NUMVALS
Number of values.

Format
Data descriptor defining the format of the subsequent lines (as needed). The format is always (4g20.13).

3.2.22. 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*.

3.2.23. 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

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

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.

The **RLBLOCK** command is not valid in a ***DO** loop.

3.2.24. SE Command

The **SE** command defines a superelement. The command format in `Jobname.CDB` is:

```
SE,R5.5,File,,,TOLER,TYPE,ELEM
```

File
The name (case-sensitive) of the file containing the original superelement matrix created by the generation pass (*Sename*.SUB).

TOLER
Tolerance for determining whether use-pass nodes are noncoincident with master nodes having the same node numbers. Default = 0.0001.

TYPE
Element type number.

ELEM
Element number.

This command command also appears in the *Command Reference*. The format shown here contains information specific to the **CDREAD**/**CDWRITE** file.

3.2.25. SECBLOCK Command

SECBLOCK for Beams

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**).

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

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
```

Cells Section:

```
      1   3   11   9    2   6    10   4    5    2
      7   9   23   21   8   16   22   14   15   1
      9   11  25   23   10  18   24   16   17   1
      11  13  27   25   12  20   26   18   19   1
```

Nodes Section:

```
...
0     0.0          0.0
0     0.025        0.0
0     0.05         0.0
0     5.0175       0.0
0     19.98        10.00
0     20.00        10.00
...
```

SECBLOCK for Shells

The **SECBLOCK** command also retrieves data for shell sections. The command format is:

```
SECBLOCK,N
TKn,MATn,THETAn,NUMPTn
```

N

Total number of layers. The second line (*TKn*, *MATn*, *THETAn*, *NUMPTn*) is repeated *N* times (once for each layer).

TKn

Layer thickness for layer number *n*

MATn

Material ID for layer number *n* (defaults to element material ID)

THETAn

Layer orientation angle for layer number *n*

NUMPTn

Number of integration points in layer number *n*

The **SECBLOCK** command is not valid in a ***DO** loop.

3.2.26. 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 4(1pg16.9) 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.

3.2.27. SFE Command

The **SFE** command defines a surface load. Values associated with this specification are on a new input line with a 4(1pg16.9) 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).

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

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/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.

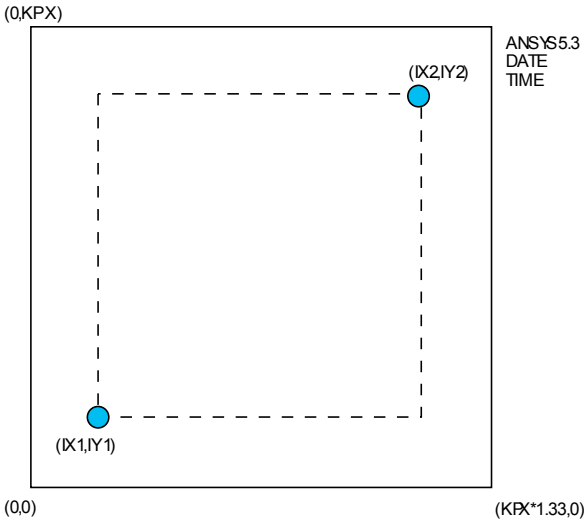
IX1,IY1 is the lower left corner of the z-buffer image.

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:

```
(BBAAA2A0AAAAAAPPPLPO>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).

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

`>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
Long	2 bytes, base 16 (letters A through P)	0 through 255
Byt3	3 bytes, base 16 (letters A through P)	0 through 65535
Xy	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 *Nchar* 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) kras - Defines whether the display uses raster mode or vector mode (1 = raster mode, 0 = vector mode)	Int, Int, Int, Int, Int

Graphics Directive	Command Character	Parameters	Parameter Types
		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	>	iclrt - 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	,	ltype - 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
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 Justification	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	-	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 string - Defines the text string itself	Xy, Long, String
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	Int, Int, Xy...Xy

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Graphics Directive	Command Character	Parameters	Parameter Types
		kedge - Defines whether the polygon edge is displayed (0, = do not display edge, 1 = display it) xy1,...xyn - Defines coordinates for the polygon	
No-Op	none	no parameters	none
End-Plot	)	no parameters	none
Pixmap	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 run-length encoded data which repeats until all pixels are read in, as defined by the window (X2-X1 + 1) * (Y2-Y1 + 1) n - Defines the number of pixels in a row inrm - Sets the normal for pixels iclr - Sets the color for the pixmap	Byt3, Xy, Xy, Long, Long, Long, ...

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

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Value	Color
10	Blue
11	Green-Cyan
12	Yellow
13	Dark Gray
14	Light Gray
15	White
16	
:	Reserved for future use
127	
128	Blue
:	:
	Cyan
	:
	Green
	:
	Indices 128 through 255 represent the color spectrum used to display the Low (Blue) to High (Red) contour values.
	:
	Yellow
	:
	Orange
:	:
255	Red

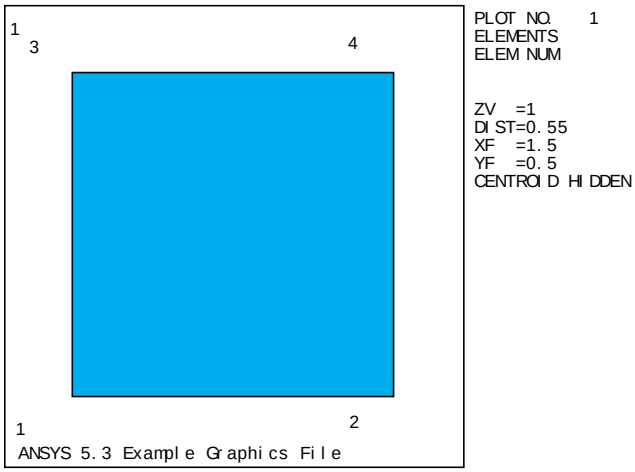
4.3. Decoding a Graphics File: an Example

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. 113) below
- The encoded graphics file that these commands produce
- The decoded graphics plot directives

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Figure 4.2: Example Display of a Graphics File



4.3.1. The Example Command Stream

To create the graphics display shown in Figure 4.2: Example Display of a Graphics File (p. 113), 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. 113) and the following graphics file:

```
(BBAAA2A0AAAAAAPPPLPO&#60AA&#62AP$MEKLKBAJANSYS 5.3$MEKLEFALNOV 16 1996$MEK
KOJAI15:57:07$MEKKIMAMPLOT NO.   1$MEKKDAAIELEMENTS$MEKJNEAIELEM NUM2
B0AAAAAALPOLPO&#60AB/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
&#60AA	Area-Color: Black
&#62AP	Text-Color: White

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

$MEKLKBAJANSYS 5.3	Text: 3146 2977 "ANSYS 5.3"
$MEKLEFALNOV 16 1996	Text: 3146 2885 "NOV 15 1996"
$MEKKOJAI15:57:07	Text: 3146 2793 "15:57:07"
$MEKKIMAMPLOT NO. 1	Text: 3146 2700 "PLOT NO. 1"
$MEKKDAAIELEMENTS	Text: 3146 2608 "ELEMENTS"
$MEKJNEAIELEM NUM	Text: 3146 2516 "ELEM NUM"
2B	Text-Size: Small
0AAAAAALPOLPO	Window: 0 0 3070 3070
&#60AB	Area-Color: Cyan
/PP	Normal: 255
+EBBBHBBHKOGBBHKOGKOGBBHKOG	Polygon: 279, 279, 2790, 279 2790, 2790 279, 2790
$FPPFPPAB1	Text: 1535 1535 "1"
$AILAILAB1	Text: 139 139 "1"
$LHCAILAB2	Text: 2930 139 "2"
$LHCLHCAB4	Text: 2930 2930 "4"
$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
-AAAAAA	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"

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

`$ABOABOCA ANSYS 5.3` `Example Graphics File`	Text: 30 30 "ANSYS 5.3 Example Graphics File"
`)`	End-Plot
	No-Op

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries and affiliates.

Part II: Guide to User-Programmable Features

Chapter 1: Understanding User Programmable Features (UPFs)

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

You can take advantage of user customization if you are licensed for the ANSYS Mechanical Enterprise family of products (ANSYS Mechanical Enterprise, ANSYS Mechanical Enterprise PrepPost, and ANSYS Mechanical Enterprise Solver).

Other ANSYS, Inc. products do not support customization of the Mechanical APDL program.

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

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

The following customization topics are available:

1.1. What Are User Programmable Features?
1.2. What You Should Know Before Using UPFs
1.3. Planning Your UPFs
1.4. Studying the Mechanical APDL User Routines
1.5. Programming in Languages Other than Fortran
1.6. Developing UPFs: a Suggested Strategy
1.7. Include Decks
1.8. Choosing a Linking Method
1.9. Compiling and Linking UPFs on Linux Systems
1.10. Compiling and Linking UPFs on Windows Systems
1.11. Activating UPFs
1.12. Running Your Custom Executable
1.13. Verifying Your Routines
1.14. Debugging Commands
1.15. Other Useful Commands
1.16. Generating Output
1.17. Reading Large Data Files More Rapidly

1.1. What Are User Programmable Features?

User programmable features are tools you can use to write your own routines. Using UPFs, you can tailor the Mechanical APDL program to your organization's needs. For instance, you may need to define a new material behavior, a special element, a contact interfacial model, or a modified failure criterion for composites.

UPFs provide the following capabilities:

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

- To read information into or retrieve information from the Mechanical APDL database, you can create subroutines and either link them into the program or use them in the external command feature (see Appendix A: Creating External Commands in Linux (p. 343) for more information about external commands). If you link these subroutines into Mechanical APDL, you are limited to 10 database access commands. Such commands, created through either method, operate at all levels of program operation, including the begin, preprocessor, general postprocessor, time-history postprocessor, and solution levels. For more information about accessing the database, see Accessing the Mechanical APDL Database (p. 257).

- Some UPF subroutines enable you 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. 227).

- Some UPF subroutines enable you to modify and monitor existing elements. For details, see Subroutines for Modifying and Monitoring Existing Elements (p. 180).

- Some UPF subroutines enable 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. 186).

- For analyses involving contact, another set of UPF subroutines enables you to define contact properties, friction models, and interaction behaviors. To see inputs and outputs for these routines, see Subroutines for Customizing Contact Interfacial Behavior.

- Several sets of UPFs enable you to define new elements and to adjust the nodal orientation matrix. See Creating a New Element (p. 141) for more information.

- You can call Mechanical APDL as a subroutine in a program you have written. To learn how, see Running Mechanical APDL as a Subroutine (p. 239).

1.2. What You Should Know Before Using UPFs

Before you do anything with linked UPFs, contact your on-site system support person to get the permissions needed to access the appropriate ANSYS, Inc. 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, Inc. distribution media. You can modify these routines to tailor the 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 Mechanical APDL 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 Mechanical APDL solution without errors. Even if you have in-depth knowledge of the Mechanical APDL 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.

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

- 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, Inc. installation manual. For more information on Fortran compilers, please refer to the installation guide specific to your operating system (ANSYS, Inc. Linux Installation Guide or ANSYS, Inc. Windows Installation Guide). 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 installation guide or your on-site ANSYS system support person

- Carefully consider whether you wish to use UPFs, especially if you are linking them into Mechanical APDL (rather than into a shared library for use as external commands). When you add your own routines to Mechanical APDL 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, Inc. 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 Mechanical APDL 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 Mechanical APDL more easily and safely with macros than with UPFs.

- When using shared memory, 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 Mechanical APDL program, see User-Programmable Features and Nonstandard Uses in the *Mechanical APDL Advanced Analysis Guide*.

1.3. 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 Mechanical APDL? Remember, a capability may not be obvious at first, especially to a novice user.

- Does your proposed subroutine fit into the Mechanical APDL 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.

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

1.4. Studying the Mechanical APDL User Routines

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

- If you are running Mechanical APDL under Linux, the source code for the UPF routines resides in directory `/ansys_inc/v182/ansys/customize/user/`.
- If you are running Mechanical APDL under Windows, the source code for the UPF routines resides in directory `Program Files\ANSYS Inc\V182\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.f` or `user.for`. Delete the routines you do not want and make appropriate changes to the others.

1.5. Programming in Languages Other than Fortran

If you wish to run Mechanical APDL with user customizations, 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 Mechanical APDL program. If you do use a language other than Fortran, such as the C or C++, your code may require a Fortran shell.

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

On many Linux 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 or C++ functions.

For more information on Fortran compilers please refer to the installation guide specific to your operating system (ANSYS, Inc. Linux Installation Guide or ANSYS, Inc. Windows Installation Guide).

1.6. Developing UPFs: a Suggested Strategy

When developing customizations for Mechanical APDL, 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 using the method described in Creating a New Element by Directly Accessing the Program Database (p. 159). You develop and test it by performing these steps:

1. Get the applicable element subroutines for `uel101` from the product distribution medium. Add a small change (such as a misspelling in an output heading), then compile and link the subroutines.

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

2. Using a production version of the program, run several analysis problems using various elements to form a base for comparison.

3. Run the same problem using USER101 on your custom version of the program.

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 Mechanical APDL element that most closely resembles your new custom element, and run some problems on a production version of Mechanical APDL 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 Mechanical APDL.

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 Mechanical APDL.

9. Test the changes with a series of increasingly complex problems for which you already know the answers.

1.7. 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 Mechanical APDL are on your product distribution medium. The include decks, also called *commons*, contain important but relatively small amounts of data. The program also handles large amounts of data using various access routines (GET and PUT), described elsewhere in this document.

Note

When you compile a user-programmable feature (UPF) in a shared library (ANSUSER-SHARED.BAT), you cannot access the common block variables. (All such variables will return a value of zero.)

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 include files and the definitions they contain:

Include File	Description
`acel-cm.inc`	Contains accelerations and angular velocities
`ansys-def.inc`	Defines general Mechanical APDL 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

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Include File	Description
elccmt.inc	Defines element characteristics (comments only)
elecom.inc	Contains element-specific information
elparm.inc	Defines pointers for the element data array
elucom.inc	Defines the element degree of freedom pointers
etycom.inc	Element type data
impcom.inc	Used by all routines and functions in the program
outpcm.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.
stepcm.inc	Contains load step information
usvrcm.inc	Defines storage of user-defined variables

1.8. Choosing a Linking Method

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

- Link your routines into shared libraries (as discussed starting in Appendix A: Creating External Commands in Linux (p. 343)), or
- Compile and link your custom routines into the Mechanical APDL program itself. This is discussed for Linux systems in Compiling and Linking UPFs on Linux Systems (p. 125) and for Windows systems in Compiling and Linking UPFs on Windows Systems (p. 127). You may need superuser or root privileges to run the procedure that does the linking.

For both Windows and Linux platforms, three methods of compiling and linking are available:

- **/UPF** command
- ANSUSERSHARED script (creates a shared library on Linux or a dynamic-link library on Windows)
- **ANS_ADMIN182** Utility

The **/UPF** command method is typically used by individuals wanting to occasionally use their customized code for certain runs. The advantages of this method are that it is very easy to use and the source code can be displayed in the output file.

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

The shared library (Linux) and dynamic link library (Windows) methods are typically used to run Mechanical APDL with frequently used user-libraries or third-party libraries (material libraries, and so on). This method is advantageous if customized code is frequently used or shared with other users.

The **ANS_ADMIN182** method is useful for someone wanting to create a permanently changed Mechanical APDL executable which will be used by many users, or used most of the time. Companies that validate their user-customized code might want to consider this option.

In some cases, you might want to combine two of the methods of compiling and linking. The program allows you to combine the **ANS_ADMIN182** method with either the **/UPF** command method or the ANSUSERSHARED method. Note that the **/UPF** command method cannot be combined with the ANSUSERSHARED method.

As an example of combining these methods, you might first create a custom executable with **ANS_ADMIN182** that contains user creep laws. Then, you might use the **ANS_USER_PATH** environment variable to include a user material (or third-party library) created with the ANSUSERSHARED method.

For detailed compiling and linking procedures, see Compiling and Linking UPFs on Linux Systems (p. 125) and Compiling and Linking UPFs on Windows Systems (p. 127).

1.9. Compiling and Linking UPFs on Linux Systems

There are three methods that you can use to link your custom routines into Mechanical APDL:

1.9.1. Using the /UPF Command
1.9.2. Creating a Shared Library
1.9.3. Using the ANS_ADMIN Utility

The source files for the user routines reside in the following subdirectory: `/ansys_inc/v182/ansys/customize/user/`

For all three methods, you can write your user routines in one language or a combination of languages (Fortran, C, and C++). Note that when using a combination of languages, you are responsible for the calling interfaces between languages (see Programming in Languages Other than Fortran (p. 122)).

The ANSYS, Inc. Linux Installation Guide lists the compilers that are required for specific platforms.

1.9.1. Using the /UPF Command

The **/UPF** command offers the simplest method for linking user programmable features into Mechanical APDL. The format of the command is:

```
/UPF, RoutineName
```

where *RoutineName* is the name of a user routine (`filename.ext`) that you want to link. The specified routine must reside in the current working directory.

To use this method start Mechanical APDL in batch mode and include one or more **/UPF** commands in the specified input listing. When the program reads the input and detects **/UPF**, Mechanical APDL will be relinked automatically.

You can include **/UPF** anywhere in your input file, and you can repeat **/UPF** as many times as needed to include multiple user routines in the relinked version. Any user routine can be linked using this method.

When you run a user-linked version of the program by this method, the output includes the following:

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
NOTE – This ANSYS version was linked by /UPF with n user supplied routine(s).
```

where *n* indicates the number of routines specified by **/UPF** commands. The routines that have been linked will be included in the output listing.

Example Using Mixed Languages

The following directory contains an example of using the **/UPF** command method to link user routines that are written in mixed languages (Fortran, C, C++):

`/ansys_inc/v182/ansys/custom/user/<platform>/Examples`

The README.txt file in this directory contains complete instructions on running this example. This is a simple, automated test that verifies whether compilers are correctly installed and configured.

1.9.2. Creating a Shared Library

You can also set up UPFs on some Linux systems through a shared library. All Fortran files (files ending with .F), C files (files ending with .c), and C++ files (files ending in .cpp) 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/v182/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.

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 Linux shell you are using:

```
setenv ANS_USER_PATH workingdirectory
```

or

```
export ANS_USER_PATH=workingdirectory
```

When you run a user-linked version of Mechanical APDL, the output echos 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, Inc. 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 executable(s) as distributed.

1.9.3. Using the ANS_ADMIN Utility

As mentioned previously, the source files for the user routines reside in subdirectory /ansys_inc/v182/ansys/customize/user/. If you modify any of these subroutines, you can select

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

the **Relink ANSYS** option from **ANS_ADMIN182** utility to link these changes. This method creates a custom Mechanical APDL executable.

The **Relink ANSYS** option compiles all Fortran files (files ending with `.F`), C files (files ending with `.c`), and C++ files (files ending in `.cpp`) in the current working directory. The procedure then loads all object files (files ending with `.o`) along with the default Mechanical APDL objects and libraries in `/ansys_inc/v182/ansys/customize/user/<platform>`. For Distributed ANSYS the location is: `/ansys_inc/v182/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.e` and will reside in the working directory.

When you run a user-linked version of Mechanical APDL, the output includes the following:

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

If you intend to run the linked version of Mechanical APDL in a distributed environment (for example, on a cluster), the executable (`ansyscust.e`) 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.

1.10. Compiling and Linking UPFs on Windows Systems

There are three methods that you can use to link your custom routines into the Mechanical APDL program:

Use the **/UPF** command (p. 128)
Create a dynamic-link library (p. 130)
Use the ANS_ADMIN Utility (p. 131)

The source files for the user routines reside in the following subdirectory: `Program Files\ANSYS Inc\V182\ansys\customize\user\`.

The user programmable features are loaded onto the system only if you perform a custom installation and choose to install the customization tools. If you intend to modify any of the user routines, make a duplicate copy of the `Program Files\ANSYS Inc\V182\ansys\customize\user\` directory to preserve the original files for later use, if necessary.

For all three methods, you can write your user routines in one language or a combination of languages (Fortran, C, and C++). Note that when using a combination of languages, you are responsible for the calling interfaces between languages (see Programming in Languages Other than Fortran (p. 122)).

The ANSYS, Inc. Windows Installation Guide lists the compilers that are required for Windows systems.

Note

You will need all the compilers specified in the Installation Guide specific to your operating system to use these user programmable features. ANSYS, Inc. does not provide assistance if customers are using unsupported compilers, or if the resulting objects are not compatible with the Mechanical APDL executable(s) as distributed.

Before linking UPFs, make sure that the INCLUDE, LIB, and PATH environment variables are set correctly for the required C/C++ and Intel Fortran compilers.

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Visual Studio 2012 Professional Update 4 is also required for linking user programmable features on Windows platforms. Before using any of the methods described below for linking UPFs, open the following Command Prompt window if you have Visual Studio 2012 Professional installed:

Start > Microsoft Visual Studio 2012 > Visual Studio Tools > Visual Studio 2012 x64 Native Tools Command Prompt

---

Note

For all three linking methods, you might have issues with write/modify access if you use the default working directory: `Program Files\ANSYS Inc\V182\ansys\custom\user\`. If the linking operation errors out or fails, try running the required applications as an administrator by right clicking and choosing "Run as administrator" while launching each application.

1.10.1. Using the /UPF Command

The **/UPF** command offers the simplest method for linking user programmable features into Mechanical APDL. The format of the command is:

```
/UPF, RoutineName
```

where *RoutineName* is the name of the user routine (`filename.ext`) that you want to link. The specified routine must reside in the current working directory.

Following are prerequisites for using the **/UPF** command method on a Windows system:

- A script named `findUPF.bat` is used to detect the **/UPF** command. You must include the path to this script in your system PATH variable. This script is typically located in `Program Files\Ansys Inc\V182\ansys\bin\<platform>` where <*platform*> is a directory that uniquely identifies the hardware platform version: "Winx64" for 64-bit Windows.
- Before starting Mechanical APDL, you must set the **ANS_USE_UPF** environment variable to TRUE. This causes the program to search for **/UPF** in the input file. This environment variable is required only on Windows systems and only when using the **/UPF** command method for linking UPFs. The **ANS_USE_UPF** environment variable should not be set when using other methods to link UPFs.

To use this method start Mechanical APDL in batch mode and include one or more **/UPF** commands in the specified input listing. When the program reads the input and detects **/UPF**, the appropriate DLL is created.

You can include **/UPF** anywhere in your input file, and you can repeat **/UPF** as many times as needed to include multiple user routines in the relinked version. The following user routines can be linked using this method:

UANBEG
UANFIN
UCNVRG
UELMATX
UITBEG
UITFIN
ULDBEG
ULDFIN
USER01 -USER10
USERCNPROP

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

USERCREEP
USERCV
USERCZM
USERELEM
USERFLD
USERFRIC
USERFX
USERHYPER
USERINTER
USERMAT
USERMATTH
USEROU
USERSWSTRAIN
USER_TBELASTIC
USERWEAR
USOLBEG
USOLFIN
USRCAL
USREFL
USRSHIFT
USRSURF116
USSBEG
USSFIN
UTIMEINC

To use the resulting DLL library, set the **ANS_USER_PATH** environment variable to the working directory where the UPF DLL library resides:

```
set ANS_USER_PATH=workingdirectory
```

If you are running in a distributed memory parallel environment (Distributed ANSYS), you must include the name of the master node when specifying the working directory:

```
set ANS_USER_PATH=\\masternode\workingdirectory
```

When you run a user-linked version of the program by this method, the output includes this message:

```
NOTE – This ANSYS version was linked by /UPF with n user supplied routine(s).
```

where *n* indicates the number of routines specified by **/UPF** commands. The routines that have been linked will be included in the output listing.

Example Using Mixed Languages

The following directory contains an example of using the **/UPF** command method to link user routines that are written in mixed languages (Fortran, C, C++):

`Program Files\ANSYS Inc\V182\ansys\custom\user\<platform>\Examples`

The `README.txt` file in this directory contains complete instructions on running this example. This is a simple, automated test that verifies whether compilers are correctly installed and configured.

1.10.1.1. Using the /UPF Command on a Windows HPC Server System

Running a distributed memory parallel (Distributed ANSYS) solution on a Windows HPC Server system is more involved than running across a simple Windows cluster. You can use the steps described here

to test the **/UPF** command on a Windows HPC Server system. Several files are included with the ANSYS, Inc. software installation for use in this example.

Before you begin, Visual Studio 2012 Professional and Intel Fortran 15.0 must be on the "master" compute node (which is usually not the head node).

The files needed for this example can be found in the following directory:

```
C:\Program Files\ANSYS Inc\v182\commonfiles\MPI\WindowsHPC\UPF
```

These include two user routines, a Mechanical APDL input file, and two files required by the HPC Job Manager:

```
usercreep.F
usermat.F
slupf.inp
RUNANSYS_UPF.xml
runansysupf.bat
```

Copy these files to:

```
C:\Temp\%USERNAME%
```

You will need to modify the value of the **ANS_USER_PATH** environment variable in RUNANSYS_UPF.xml to the appropriate location where the UPF DLL library resides.

Submit RUNANSYS_UPF.xml to the HPC Job Manager. When the job is complete, run the following commands from a Command Prompt window:

```
clusrun /exclude:%CCP_SCHEDULER% copy /y C:\Temp\%USERNAME%\Work\*.out \\%CCP_SCHEDULER%\Temp\%USERNAME%

findstr /I debug *.out
```

If the test worked correctly, this should display many lines of debug from all output files.

For more information on running Distributed ANSYS under Windows HPC Server, see Configuring Distributed ANSYS in the *Parallel Processing Guide*.

1.10.2. Creating a Dynamic-link (DLL) Library

You can also set up UPFs on Windows systems by using a DLL library. All Fortran files (files ending with .F), C files (files ending with .c), and C++ files (files ending in .cpp) that you want to include in your DLL library should reside in your working directory. To compile all *.F, *.c, and *.cpp routines, issue the following command:

```
\Program Files\Ansys Inc\v182\ansys\custom\user\<platform>\ANSUSERSHARED.bat
```

After you issue ANSUSERSHARED.bat, the output will display the options for building the DLL library. The first portion of the output is shown below:

```
This is the ANSYS 18.2 ANSUSERSHARED script.  It is used
to build a DLL of User Programmable Features for the program.

           NOTE:  The user subroutine source file(s) are expected to
                  reside in this directory.

 Enter one of the following choices to create your
 User Programmable Feature DLL:
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

The user routines that are supported by this method will appear in a list. (These are the same user routines as listed above for the **/UPF** command method (p. 128).) Enter the user routine name you wish to include. As an example, if you enter USERMAT the following output will display and prompt you to select another option:

```
You chose USERMAT
Microsoft (R) Incremental Linker Version 11.00.61030.0
Copyright (C) Microsoft Corporation.  All rights reserved.

-out:UserMatLib.dll
-def:UserMatLibex.def
-dll
-machine:AMD64
-map
-manifest
-manifestfile:UserMatLib.dll.intermediate.manifest
-defaultlib:ANSYS.lib
-defaultlib:bufferoverflowU.lib
*.obj
   Creating library UserMatLib.lib and object UserMatLib.exp

      ************************************************************************

      UserMatLib.dll has been successfully built.

      Set the environment variable ANS_USER_PATH to the directory where the
      UserMatLib.dll resides and run ansys182 to use your newly generated
      user shared library.

      ************************************************************************
```

After you have selected all desired user routines, press **Enter** to quit.

To use the resulting DLL library, set the **ANS_USER_PATH** environment variable to point to the working directory where the UPF DLL library resides. Use the following command:

```
set ANS_USER_PATH=workingdirectory
```

If you are running in a distributed memory parallel environment (Distributed ANSYS), you need to include the name of the master node when specifying the working directory:

```
set ANS_USER_PATH=\\masternode\workingdirectory
```

When you run a user-linked version of Mechanical APDL, the output echos 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 Mechanical APDL, return the **ANS_USER_PATH** environment variable to its original value.

Multiple UPF DLLs can be created via the ANSUSERSHARED`.bat` script but must exist in the same directory as designated by the **ANS_USER_PATH** environment variable.

1.10.3. Using the ANS_ADMIN Utility

The **ANS_ADMIN** procedure for compiling and linking UPFs on Windows Systems creates a custom executable. This executable can be used to run in a shared memory parallel (SMP) environment or a distributed memory parallel environment (Distributed ANSYS).

As mentioned previously, the source files for the user routines reside in subdirectory `Program Files\ANSYS Inc\V182\ansys\customize\user\`.

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

`Program Files\Ansys Inc\V182\ansys\custom\user\<platform>`

Where <*platform*> is a directory that uniquely identifies the hardware platform version: "Winx64" for 64-bit Windows.

Note

The user routines listed in Using the **/UPF** Command (p. 128) cannot be linked with the **ANS_ADMIN** utility or the ANSCUST.bat script. These user routines have reserved DLL names and must be built using either the **/UPF** command or the ANSUSERSHARED.bat script.

You can select the **Relink ANSYS** option from the **ANS_ADMIN182** utility to link these changes into Mechanical APDL. This procedure will ask which versions you intend to relink and then will compile all Fortran files (files ending with .F), C files (files ending with .c), and C++ files (files ending in .cpp) in the `Program Files\ANSYS Inc\V182\ansys\custom\user\<platform>` directory. The procedure then loads all object files (files ending with .obj), along with the default Mechanical APDL objects and libraries and creates custom executables. The executable file(s) created will be named ansys.exe and will reside in the folders described above.

Caution

When creating custom executables, they 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 Mechanical APDL by using ANSCUST instead of using the **ANS_ADMIN182** utility (as recommended above), error messages may occur. These messages may state that object files have been created, but the Mechanical APDL 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 Mechanical APDL with ANSCUST:

- Pick **Start > Microsoft Visual Studio 2012 > Visual Studio Tools > Visual Studio 2012 x64 Native Tools 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\V182\ansys\custom\user\`).
- Type ANSCUST in this command window. The process of creating the new user-customized Mechanical APDL version begins.

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

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

1. To execute the relinked version of the ANSYS program:

   - Click **Start>Programs>ANSYS 18.2>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:

     ```
     ansys182 -p <product variable> -g -custom <path>
     ```

   - Batch:

     ```
     ansys182 -b -p <product variable> -j jobname -i <input file> -o <output file> -custom <path>
     ```

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

Note

The `-custom` option must be the last option at the end of the command line.

Note

The commands above are for the SMP version of Mechanical APDL. See Activating Distributed ANSYS in the *Parallel Processing Guide* for instructions on running Distributed ANSYS.

Note

Output from a user-linked version contains the following statement:

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

1.11. Activating UPFs

The Mechanical APDL program activates many UPFs via a specific user action. This can be through a command option or a user selection. Below is a list of specific actions required for several types of UPF.

- To activate user elements created using the method described in Creating a New Element via the User-Defined Element API (p. 143), you need **USRELEM** and **USRDOF** 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 Program Database (p. 159), you must select it as one of the element types in a model using the **ET** command, then set the element attribute pointer using the **TYPE** command, and define elements using the solid modeling or direct generation method.

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

- To define a user material described in Subroutine UserMat (Creating Your Own Material Model) (p. 187), Subroutine UserCreep (Defining Creep Material Behavior) (p. 203), and Subroutine userswstrain (Defining Your Own Swelling Laws) (p. 210), you need to activate it with the corresponding **TB** commands.
- To customize contact interfacial behaviors as described in Subroutine USERFRIC (Writing Your Own Friction Laws) and Subroutine USERINTER (Writing Your Own Contact Interactions), you need to activate them with the corresponding **TB** commands.
- To program history-dependent contact properties described in Subroutine USERCNPROP (Defining Your Own Real Constants for Contact Elements), you need to activate the user routine with the **R**, **RMORE**, or **RMODIF** command. The real constant must be defined by the Mechanical APDL reserved table name _CNPROP and enclosed in % signs (i.e., %_CNPROP%).

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 Mechanical APDL logic is used by default. For example, when you apply a convection load, standard Mechanical APDL 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.

1.12. 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 Executable** field. Do not include the `-custom` argument.

When run from the command prompt, if no path is specified after the `-custom` argument, the `ansys182` script searches the current working directory for the custom Mechanical APDL executable (`ansyscust.e` by default on Linux or `ansys.exe` on Windows). If the custom Mechanical APDL executable resides in a separate directory (or has a name other than `ansyscust.e` on Linux), you can specify a different path and filename after the `-custom` argument.

Caution

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

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

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

```
ansys182 -custom /pathname/ansyscust.e
```

1.13. 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 Mechanical APDL works properly is *your* responsibility.

Make certain that your custom version of the 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 Mechanical APDL features. To do so, you can compare the results of a set of problems from the *Mechanical APDL Verification Manual* run on the standard version and on the customized version. Input for these problems is also available on your product 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 Mechanical APDL, always tell that person that you are using a user programmable feature. If you feel that an error exists in an unrelated feature of Mechanical APDL, demonstrate the suspected error in a non-customized, production version of the program before you report the error to an ANSYS, Inc. representative.

1.14. 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.

1.14.1. 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 Subroutines for Your Convenience (p. 327)) 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.

1.14.2. 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.

1.14.2.1. 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)

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

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)

1.14.2.2. 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)

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

1.14.2.3. 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)

H5

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

H9

-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)

1.15. Other Useful Commands

Two other Mechanical APDL 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. 239).

1.16. 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 `wrinqr`. See the discussion on the function `wrinqr` in Subroutines for Your Convenience (p. 327) for more details.

1.17. Reading Large Data Files More Rapidly

When files containing Mechanical APDL-related data are large, loading them into the program or writing them out to an external file can be a slow process. For example, consider a 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, Mechanical APDL requires much time to read it.

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

You can shorten the time the program requires 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 Mechanical APDL 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 The CDWRITE (CDB) File Format (p. 83) in the *Guide to Interfacing with ANSYS* for more information on the use of these commands.

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries and affiliates.

Chapter 2: UPF Subroutines and Functions

This guide describes the various subroutines, functions, and commands that allow you to customize the 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 `usercnprop` subroutine looks like this:

```
*deck,usercnprop                    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:

2.1. Creating a New Element
2.2. Supporting Subroutines for Element Creation
2.3. Subroutines for Modifying and Monitoring Existing Elements
2.4. Subroutines for Customizing Material Behavior
2.5. Subroutines for Customizing Contact Interfacial Behavior
2.6. Subroutines for Customizing Loads
2.7. Running Mechanical APDL as a Subroutine
2.8. Defining Your Own Commands
2.9. Supporting Subroutines
2.10. Access at the Beginning and End of Various Operations
2.11. Memory Management Subroutines
2.12. Parameter-Processing Subroutines
2.13. Other Useful Functions

2.1. Creating a New Element

Two tools are available for creating a user-defined element:

- The user-defined element API (p. 143)
- Direct access to the program database and files (p. 159)

ANSYS, Inc. 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	User-defined element API	Accessing program database and files directly
Description	Offers a simpler interface while preserving much of the underlying user-element capability. An	No special interface. With few exceptions, if a capability exists for an element, it will exist here. The logic

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Interface	User-defined element API	Accessing program database and files directly
	understanding of the database subroutines and the file structure is rarely necessary to use the interface.	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 included on your product distribution media	4-node quad and 20-node brick elements	`uel100` (a structural mass element) and `uel101` (a simple link element)
Typical linear material access subroutine	`getMatProp`	`propev`
New nonlinear material properties	Program in `UserMatTh`.	No special programming has been set up.
Existing nonlinear material properties	All 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 element 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 `tmp-get`, `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 *not* 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.

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

2.1.1. 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

2.1.2. 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.

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

Step	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 degrees of freedom.	Issue the **USRDOF** command. You can specify a maximum of 10 degrees of freedom per **USRDOF** command; to define additional degrees of freedom, issue the command again. Each node will have the same degrees of freedom. Although you can specify any valid degrees of freedom, the total number of degrees of freedom for your element cannot exceed 480, and the number of degrees of freedom for each node cannot exceed 32.
4.	Define real constants.	If needed.
5.	Create finite element models.	Use either of these methods: • *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

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

		create an element with a topology different from that of any standard element. • *Meshing commands* -- This method is available only if your element has the same topology as that of a standard 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).
8.	Perform postprocessing.	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.

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.
- The program may activate default solution settings automatically according to the USER300 element's degrees of freedom, but the default solution control settings may not be optimal for your element.
- The USER300 element does not support section (SEC*xxx*) commands. For composite beams and layered shells, you must input element data via real constants and code the `UserElem.F` subroutine accordingly.

2.1.2.1. Subroutine UserElem (Writing Your Own Elements)

The `UserElem` subroutine provides an interface to program 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 program database and files. With this API, you can create virtually any element type *without* having to access program database and files directly (p. 159). 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

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

elId	I	Element number	Output information	At FE model creation
matId	I	Material number	Output information Call material subroutines	At FE model creation
keyMtx	I	Formulation request	Specifying which matrices and load vectors to form	Program code
lumpm	I	Mass matrix format: 0 = Consistent mass matrix 1 = Lumped mass matrix	Specifying how to form the mass matrix	**LUMPM** command
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 degrees of freedom	Element coding -- The degrees of freedom 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	**USRELEM** and **USRDOF** commands

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

			the same way	
kEStress	I	Element stress state	Element coding Calling material subroutines if requested	**USRELEM** command
keyAnsMat	I	Element formulation key: 0 -- Write your own material formulation 1 -- Use standard material subroutines and call `ElemGetMat` subroutine	Specifying how to create material data	**USRELEM** command
keySym	I	Flag for symmetricity of element matrices	Element coding Program assembly logic	**USRELEM** command
nKeyOpt	I	Maximum number of element key options allowed (up to 2)	Element coding	Program code
KeyOpt	I	Element key options KEYOPT(1) = 0~99	Branching the user-element codes to different formulations. (This could be equivalent to 100 x 100 different types of elements.)	**ET** command
temper	I	Nodal temperatures at current time	Temperature dependence and thermal loads	**BF** and **BFE** commands (if **keyShape** is specified in the `UserElem` subroutine)
temperB	I	Nodal temperatures at the end of the last substep	Temperature dependence and thermal loads	Program code
tRef	I	Reference temperature	Temperature dependence and thermal loads	**TREF** command
kTherm	I	Key indicating whether a thermal load exists:	Element coding	---

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

		1 = Calculate the thermal load 0 = No thermal load calculation		
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	Program code
nReal	I	Number of real constants	Element coding	**USRELEM** command
RealConst	I	The list of real constants	Element coding Can pass material properties, section/layer information, element material control, and any element data	**R** command
nSaveVars	I	The number of variables saved in the `.esav` file for the element	Element coding	**USRELEM** command

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

			The size of **saveVars**	
saveVars	I/O	The data saved in the .esav file The program 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 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	Program code
TotValDofs	I	Total values of degrees of freedom (displacements for structural analysis) Values in global Cartesian coordinate system	Element coding	Program code
IncValDofs	I	Increment values of degrees of freedom occurring at the current substeps	Element coding	Program code

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

		Values in global Cartesian coordinate system		
ItrVal-Dofs	I	Iteration values of degrees of freedom occurring at the last iteration Values in global Cartesian coordinate system	Element coding	Program code
VelVal-Dofs	I	First time derivatives of degrees of freedom	Velocities	Program code
AccVal-Dofs	I	Second time derivatives of degrees of freedom	Accelerations	Program code
kfstps	I	Key indicating first time entering the element subroutine: 1 = First time 0 = Other than first time	Initializing data	Program 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	Program code
elPrint	I	Key indicating whether any element output should appear in the print file: 0 = No 1 = Yes	Element coding	Program code **OUTPR** command
iott	I	Output file number	The Fortran output file number. All information written in the specified file	Program code

			appears in the output file.	
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	Program code
The following variables are for debug, timing, and convergence-control purposes only. You can usually ignore them.				
ldstep	I	Current load step number	Output Debug	Program code
isubst	I	Current substep number	Output	Program code
ieqitr	I	Current iteration number	Output	Program code
timval	I	Current time	Output	Program code
keyEleErr	I/O	Formulation error key: 0 = No error (preset value) 1 = Error occurred in element formulation, possibly due to excessive deformation. (The program lessens deformation if possible by cutback.)	Element coding	Program code
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	Program code

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

			constraint for mixed u-P)	
End of special-purpose variable group				
eStiff	O	Small-deformation stiffness matrix In global Cartesian coordinate system	Solution	Requested when **keyMtx(1)** = 1
eMass	O	Mass matrix In global Cartesian coordinate system	Solution	Requested when **keyMtx(2)** = 1
eDamp	O	Damping matrix In global Cartesian coordinate system	Solution	Requested when **keyMtx(3)** = 1
eSStiff	O	Stress stiffness matrix In global Cartesian coordinate system	Solution	Requested when **keyMtx(4)** = 1
fExt	O	External load vector In global Cartesian coordinate system	Solution	Requested when **keyMtx(5)** = 1
fInt	O	Internal nodal force vector In global Cartesian coordinate system	Solution	Requested when **keyMtx(6)** = 1
elVol	O	Element volume	Output	`UserElem` subroutine
elMass	O	Element mass	Output	`UserElem` subroutine
elCG	O	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**	Program code
RsltBsc	O	Basic result data saved in results file These variables are accessible via the **PRESOL** and **PRNSOL** commands in the standard way and can also be plotted if you specify a *KeyShape* value via the	Postprocessing	`UserElem` subroutine

		USRELEM command.		
nRsltVar	I	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	O	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	O	Element energy vector: **elEnergy(1)** -- Strain energy **elEnergy(2)** -- Plastic energy **elEnergy(3)** -- Creep energy	Output	UserElem subroutine

```
*deck,UserElem                      USERSDISTRIB
c Copyright ANSYS.  All Rights Reserved.
      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,
     &                     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)
     &
c*************************************************************************
c
c *** Primary function: General User Element Subroutine
c *** Note:
c       This routine is completed with an example, see more details later.
c
c
c     PROGRAMMER SHOULD NOT CHANGE ANY PURE INPUT ARGUMENTS (marked by ....,in)!
c
c     elId      (int,sc,in)        element number
c     matId     (int,sc,in)        material number of this element
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
c     keyMtx    (int,ar(10),in)    matrix and load vector form requests
c                                      0 = not requested, 1 = requested
c                                      see below for more details
c     lumpm     (int,sc,in)        mass matrix format
c                                    = 0 no lumped mass matrix
c                                    = 1 lumped mass matrix
c     nDim      (int,sc,in)        number of dimensions of the problem
c                                        (defined on USRELEM command as NDIM)
c                                    = 2 2D
c                                    = 3 3D
c     nNodes    (int,sc,in)        number of nodes of the element
c                                        (defined on USRELEM command as NNODES)
c     Nodes     (int,ar(nNodes),in)node list of this element
c     nIntPnts  (int,sc,in)        maximum number of integration points
c                                        (defined on USRELEM command as NINTPNTS)
c     nUsrDof   (int,sc,in)        number of DOFs of this element (matrix and
c                                      load vector size)
c     kEStress  (int,sc,in)        kEStress
c                                        (defined on USRELEM command as KESTRESS)
c     keyAnsMat (int,sc,in)        key to indicate if ANSYS material
c                                     routine is going to be called
c                                     (defined on USRELEM command as KEYANSMAT)
c                                     = 0, No
c                                     = 1, Yes
c     keySym    (int,sc,in)        key to indicate if element matrices
c                                     is symmetric
c                                       (defined on USRELEM command as KEYSYM)
c                                     = 0, symmetric
c                                     = 1, unsymmetric
c     nKeyOpt   (int,sc,in)        number of element key options able to be
c                                     used in this routine
c     KeyOpt    (int,ar(nKeyOpt),in) values of element key option defined
c                                     by et or keyopt command for the
c                                     user elements, only the first
c                                     nKeyOpt values are passed in and can
c                                     be used to branch the routine for
c                                     different formulations
c     temper    (dp,ar(nNodes),in) nodal temperatures at current time
c     temperB   (dp,ar(nNodes),in) nodal temperatures at the beginning of this
c                                     incremental step (substep)
c     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,
c                                      thermal loading might be needed.
c                                      = 0, Temperatures at nodes are the same
c                                      as the reference temperature,
c                                      thermal loading is not needed.
c                                    output:  flag for thermal strains
c     nPress    (int,sc,in)        number of pressure values for this element
c     Press     (dp,ar(nPress),in) applied elemental face load (pressure)
c     kPress    (int,sc,in)        flag for pressure loading
c                                      = 1, pressure load is applied and
c                                      equivalent nodal forces should be
c                                      calculated
c                                      = 0, no pressure loading
c     nReal     (int,sc,in)        number of real constants
c                                        (defined on USRELEM command as NREAL)
c     RealConst (dp,ar(nReal),in)  user defined real constants
c     nSaveVars (int,sc,in)        number of saved variables
c                                        (defined on USRELEM command as NSAVEVARS)
c     saveVars  (dp,ar(nSaveVars),inout) user saved variables
c     xRef      (dp,ar(nDim,nNodes),in)
c                                    nodal coordinates in initial configuration
c     xCur      (dp,ar(nDim,nNodes),in)
c                                    nodal coordinates in current configuration
c     TotValDofs (dp,ar(nUsrDof),in) total values of DOFs (displacements)
c                                     from time = 0
c     IncValDofs (dp,ar(nUsrDof),in) incremental values of DOFs (displacements)
c                                     for the current step
c     ItrValDofs (dp,ar(nUsrDof),in) iterative values of DOFs (displacements)
c                                     for the current iteration
```

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


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries and affiliates.

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

2.1.2.2. Subroutine ElemGetMat (Calling the Standard Structural Material Library)

The `ElemGetMat` subroutine is the API to the standard materials. When you issue the **USRELEM** command after setting the command's *KEYANSMAT* argument, the subroutine accesses the standard structural material library. It allows you to focus on the kinematic portion of element formulation while the program 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	Material calculation	At FE model creation

		2 = 2-D element geometry 3 = 3-D element geometry Specifying the size of the nodal coordinates		
nTens	I	Number of stress/strain tensor components: 4 = 2-D and ordered as x, y, z, xy 6 = 3-D and ordered as x, y, z, xy, yz, xz	Specifying the data size	UserElem subroutine
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 integration point Values in global Cartesian coordinate system	Material calculation	UserElem subroutine
TemperIP	I	Integration point temperatures at the current time	Evaluating temperature-dependent material data	UserElem subroutine
TemperIPB	I	Integration point temperatures at the end of the last incremental step	Evaluating temperature-dependent material data	UserElem subroutine
IncStrain	I	Strain components [1] 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

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

kTherm	I/O	Thermal loading key: 0 = No thermal loading 1 = Has thermal loading	Thermal load calculation	UserElem subroutine
cMat	O	Material Jacobian [1]	Forming stiffness	Material subroutine
MatProp	O	Material data for element formulation	Forming mass matrix Handling transverse shear Output	Material subroutine
Stress	O	Cauchy stress [1]	Forming geometric stiffness Calculating internal forces	Material subroutine
Strain	O	Total strain components [1]	Output	Material subroutine
StressTh	O	Total thermal stress components [1]	Output Calculating thermal loading	Material subroutine
StrainTh	O	Total thermal strain components [1]	Output	Material subroutine
StrainPl	O	Total plastic strain components [1]	Output	---
StrainCr	O	Total creep strain components [1]	Output	---
StressBk	O	Back stress components [1]	Output	---
StrainSw	O	Swelling strain	*Not yet supported*	---
EnergyD	O	Energy density: 1 = Elastic energy density 2 = Plastic energy density 3 = Creep energy density	---	---
MatRotGlb	O	Rotation matrix	Rotation matrix from global Cartesian to rotated element coordinate system	Used only for solid elements when **nlgeom** = on

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

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,
     &                           defG0, defG, CMat, MatProp,
     &                           Stress, Strain, StressTh, StrainTh,
     &                           StrainPl, StrainCr, StressBk, StrainSw,
     &                           EnergyD, MatRotGlb)

c***********************************************************************
c
c *** Primary function: call ANSYS material routines to obtain material
c                       data for USER300 elements

c *** Notice - This file contains ANSYS Confidential information ***
c
c     input arguments
c     ===============
c     elId        (int,sc)           element number
c     matId       (int,sc)           material number of this element
c     nDim        (int,sc)           number of dimensions of the problem
c                                    = 2 2D
c                                    = 3 3D
c     nTens       (int,sc)           number of stress/strain components
c     nDirect     (int,sc)           number of stress/strain direct
c                                      components
c     intPnt      (int,sc)           current integration point number
c     xCurIP      (dp,ar(3))         coordinates of integration point
c     TemperIP    (dp,sc)            integration point  temperatures at
c                                      current time
c     TemperIPB   (dp,sc)            integration point  temperatures at
c                                      the end of last incremetal step
c     IncStrain   (dp,ar(nTens))     strain for the current substep step when
c                                       nlgeom = on
c                                    total strain when nlgeom = off
c     defG0       (dp,ar(3x3))       deformation gradient tensor at the end
c                                       of last incremental step
c
c     input output arguments         input desc     / output desc
c     ======================         ==========       ===========
c     defG        (dp, ar(3x3))      deformation gradient tensor at current
c                                      time, updated for thichness change in
c                                      plane stress when nlgeom=on
c     kTherm      (int,sc)           flag for thermal loading
c                                      input as:
c                                      = 0 if temp = tref
c                                      = 1 if temp .ne. tref
c                                      gets reset to 0
c                                                  if ALPX, ALPY, and ALPZ = 0
c
c     output arguments
c     ================
c     cMat        (nTens*nTens)      material Jacobian matrix
c     MatProp     (dp,ar(5))         commonly used materail properties
c                                    MatProp(1),Gxz: shear modulus
c                                    MatProp(2),Gyz: shear modulus
c                                    MatProp(3),Gxy: shear modulus
c                                    MatProp(4), density
c                                    MatProp(5), nuxy
c     Stress      (dp,ar(nTens))     total stress
c     Strain      (dp,ar(nTens))     total strain
c     StressTh    (dp,ar(nTens))     thermal stress
c     StrainTh    (dp,ar(nTens))     thermal strain
c     StrainPl    (dp,ar(nTens))     plastic strain
c     StrainCr    (dp,ar(nTens))     creep strain
c     StressBk    (dp,ar(nTens))     back stress for kinematic hardening
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
c     StrainSw    (dp,sc)           isotropic swelling strain
c                                       (swelling capability not available yet)
c     EnergyD      (dp,ar(3))       energy density
c                                   EnergyD(1) elastic energy density
c                                   EnergyD(2) plastic energy density
c                                   EnergyD(3) creep energy density
c     MatRotGlb   (dp,ar(3,3))      The rotation matrix from global
c                                    to  material coordinate system
```

2.1.3. Creating a New Element by Directly Accessing the Program 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 element library, and use them as "regular" elements. You can create up to six independent element types (names USER100 - USER105). For demonstration purposes, the subroutines `uel100` (for a structural mass element) and `uel101` (for a simple link element) are included on the product distribution media.

2.1.3.1. 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 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.

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

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

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

2.1.3.2.1. 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.

2.1.3.3. Subroutine uex100 (Overriding Element Characteristic Defaults)

```
*deck,uex100                       USERSDISTRIB
      subroutine uex100 (ielc,kerr)
c     *** subroutine to override element characteristic defaults ***
c     *** hence, this routine is needed only in rare cases.
c
c         *** Copyright ANSYS.  All Rights Reserved.
c         *** ansys, inc.
c *** Notice - This file contains ANSYS Confidential information ***
c
c     *** input and output are the same as for uec100, except that this
c     *** logic is called after the defaulting logic is finished.
c     *** this defaulting is done in ansys subroutine echdft(not a upf).
c     *** as indicated above, this routine is rarely needed, but if it is
c     *** desired to see the effect of echdft, you may print out the ielc array
c     *** 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
c  input arguments:
c     variable (typ,siz,intent)     description
c     ielc (int,ar(IELCSZ),inout) - element characteristics
c
c     kerr   (int,sc,inout)       - error flag up to this point.
c                                   (do not initialize to zero)
c
c  output arguments:
c     variable (typ,siz,intent)     description
c     ielc (int,ar(IELCSZ),inout) - element characteristics
c
c     kerr   (int,sc,inout)       - error flag (set to 1 if error)
c     *** standard defaults are taken.  the final results are given with
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
c     *** the debug accessed with /debug,,, ,,1
c
```

2.1.3.3.1. 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.

2.1.3.4. Subroutine uel100 (Calculating 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:
c        1. compute element matrices, load vectors, and results
c *** secondary functions:
c        2. maintain element solution data

c *** user programmable functions may not be used in parallel processing ***

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

c         *** Copyright ANSYS.  All Rights Reserved.
c         *** ansys, inc.

c  input arguments:
c       elem   (int,sc,in)         - element label (number)
c       ielc   (int,ar(IELCSZ),in) - array of element type characteristics
c                                     (IELCSZ = array size, defined in echprm)
c       elmdat (int,ar(EL_DIM),in) - array of element data
c       eomask (int,sc,in)         - bit pattern for element output
c                                    (see outpcm)
c       nodes  (int,ar(nnod),in)   - array of element node numbers
c                                   (nnod = number of nodes; 1 in this case)
c       locsvrL (int,sc,in)        - location of the saved variables
c                                              on file .esav for this element
c       kelreq (int,ar(10),in)     - matrix and load vector form requests
c                                   (indices for kelreq are given with output
c                                                          arguments below)
c       kelfil (int,ar(10),in)     - keys indicating incoming matrices and
c                                    load vectors (indices for kelfil are the
c                                    same as given for kelreq with output
c                                    arguments below)
c       nr     (int,sc,in)         - matrix and load vector size
c       xyz    (dp,ar(6,nnod),in)  - nodal coordinates (orig) and rotation angle
c       u      (dp,ar(nr,5),in)    - element nodal solution values

c  output arguments:
c       kelout (int,ar(10),out)    - keys indicating created matrices and
c                                         load vectors (indices for kelout
c                                            are the same as for kelreq below,
c                                            as well as kelin and kelout later)
c       zs     (dp,ar(nr,nr),inout)- stiffness/conductivity matrix (kelreq(1))
c       zass   (dp,ar(nr,nr),inout)- mass matrix                    (kelreq(2))
c       damp   (dp,ar(nr,nr),inout)- damping/specific heat matrix   (kelreq(3))
c       gstif  (dp,ar(nr,nr),inout)- stress stiffness matrix        (kelreq(4))
c       zsc    (dp,ar(nr),out)     - applied f vector               (kelreq(5))
c       zscnr  (dp,ar(nr),out)     - n-r restoring f vector         (kelreq(6))
c                                    or imaginary f vector          (kelreq(7))
c       elvol  (dp,sc,out)         - element volume
c       elmass (dp,sc,out)         - element mass
c       center (dp,ar(3),out)      - centroid location
c       elener (dp,ar(5),out)      - element energies
c       edindxL(LONG,ar(25),out)   - element result data file indexes
c       lcerstL(LONG,sc,inout)     - position on result file
```

```
c
```

2.1.3.4.1. Subroutines uel101 through uel105

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

2.1.3.5. 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)
c
c *** primary function:    produce printed output for user100
c
c         *** Copyright ANSYS.  All Rights Reserved.
c         *** ansys, inc.
c *** Notice - This file contains ANSYS Confidential information ***
c
c  **********  this subroutine is provided for user information  *********
c *** user programmable features may not be used in parallel processing ***
c
c  input arguments:
c     iott     (int,sc,in)         - output unit number
c     elem     (int,sc,in)         - element number
c     nodes    (int,ar(2),in)      - node numbers
c     mat      (int,sc,in)         - material number
c     kept     (int,sc,in)         - key to print temperatures
c     tem      (dp,ar(2),in)       - nodal temperatures
c     kemn     (inr,sc,in)         - key to print fluences
c     fluen    (dp,ar(2),in)       - neutron fluences
c     kems     (int,sc,in)         - key to print moment forces
c     force    (int,sc,in)         - member force fx
c     kens     (int,sc,in)         - key to print strains
c     sig      (dp,sc,in)          - axial stress
c     keel     (int,sc,in)         - key to print elastic strain
c     epel     (dp,sc,in)          - axial elastic strain
c     keth     (int,sc,in)         - key to print thermal,initial,swelling strai
c     eptho    (dp,sc,in)          - axial thermal strain
c     epswel   (dp,sc,in)          - swelling strain
c     epino    (dp,sc,in)          - initial axial strain
c     kenl     (int,sc,in)         - key set if any nonlinear materials present
c     sigepl   (dp,sc,in)          - stress in stress-strain curve
c     sigrat   (dp,sc,in)          - stress ratio
c     hpres    (dp,sc,in)          - hydrostatic pressure
c     epeq     (dp,sc,in)          - plastic equivalent strain
c     kepl     (int,sc,in)         - key to print plastic strain
c     eppl     (dp,sc,in)          - plastic strain
c     kecr     (int,sc,in)         - key to print creep strain
c     epcrp    (dp,sc,in)          - creep strain
c
c  output arguments:     none
c
```

2.1.3.5.1. Subroutines uep101 through uep105

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

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

2.1.3.6. Subroutine usertr (Adjusting the Nodal Orientation Matrix)

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

2.1.3.7. 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.
c
c         *** Copyright ANSYS.  All Rights Reserved.
c         *** ansys, inc.
c *** Notice - This file contains ANSYS Confidential information ***
c
c  input arguments:
c     variable (typ,siz,intent)    description
c     elem     (int,sc,in)       - element number
c
c  output arguments:
c     none
c
```

2.2. 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 Program Database (p. 159)).

2.2.1. Subroutine nminfo (Returning Element Reference Names)

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

2.2.2. Subroutine svgidx (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 ***

c  input arguments:
c     locsvr   (LONGINT,sc,in)   - pointer to location of index

c  output arguments:
c     svindx   (int,ar(20),out) - the 20 word index of svr variables
c                                 1,2-starting loc of this eles svr sets
c                                 3- length of eles svr sets
c                                 4-11-relative starting loc for each set
c                                    4-structural svrs
c                                    5-thermal/electric/fluid svrs
c                                    6-magnetic svrs
c                                    7-nonlinear svrs
c                                    8-plasticity svrs
c                                    9-creep svrs
c                                   10-coupled svrs
c                                   11-user svrs
c                                   12-initial strain svrs
c                                   13-section data after FiberSIM conversion
c                                                          (shell181 only)
c                                   14-20 spares
```

2.2.3. Subroutine svrget (Fetching Saved Variable Data for an Element)

```
*deck,svrget
      subroutine svrget (svindx,nset,nsvr,svr)
c *** primary function:    get svr data set for an element

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

c  input arguments:
c     svindx   (int,ar(20),in) - index for svr for this element (see svgidx)
c     nset     (int,sc,in)     - the set number in this index
c                                =  1 - structural svrs
c                                =  2 - thermal/electric/fluid svrs
c                                =  3 - magnetic svrs
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
c                                  =  4 - nonlinear svrs
c                                  =  5 - plasticity svrs
c                                  =  6 - creep svrs
c                                  =  7 - coupled svrs
c                                  =  8 - user svrs
c                                  =  9 - initial stress svrs
c                                            (2,42,82,45,92,95 only)
c                                  = 10 - section data after FiberSIM conversion
c                                                      (shell181 only)
c                                  = 11-17 - spares (note that the first three
c                                            items in svindx are not available)
c     nsvr     (int,sc,inout)    - number of dp words expected in this set

c  output arguments:
c     nsvr     (int,sc,inout)    - number of dp words in this set
c     svr      (dp,ar(nsvr),in)  - data in this set
```

2.2.4. Subroutine svrput (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 ***

c  input arguments:
c     svindx   (int,ar(20),inout)- the index for svr for this element
c                                                          (see svgidx)
c     nset     (int,sc,in)       - the set number in this index (same as svrget)
c                                  =  1 - structural svrs
c                                  =  2 - thermal/electric/fluid svrs
c                                  =  3 - magnetic svrs
c                                  =  4 - nonlinear svrs
c                                  =  5 - plasticity svrs
c                                  =  6 - creep svrs
c                                  =  7 - coupled svrs
c                                  =  8 - user svrs
c                                  =  9 - initial stress svrs
c                                            (2,42,82,45,92,95 only)
c                                  = 10 - section data after FiberSIM conversion
c                                                      (shell181 only)
c                                  = 11-17 - spares (note that the first three
c                                            items in svindx are not available)
c     leng     (int,sc,in)       - number of dp words in this set
c     svr      (dp,ar(leng),in)  - data in this set

c  output arguments:
c     svindx   (int,ar(10,2),inout)- updated index
```

2.2.5. Subroutine svpidx (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 ***

c  input arguments:
c     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  output arguments:
c     locsvr   (LONGINT,sc,inout)    - pointer to start of svr for next element
```

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

```
*deck,mreuse
      subroutine mreuse (kelrqq,kelfil,elem,ielc,kmasrt,knlmg,kconve,
     x kpheno,kprop,nprop,prop,propo,krvro,rvr,rvro,amodo,asymo, kelin)
c *** primary function:
c         determine which Matrices can be REUSEd and which must be recomputed
c          from iteration to iteration.
c          Note:  a few special elements have some supplementary logic
c          to adjust these results further.  No attempt as been made to
c          include all such logic in these routines.
c
c          Second note:  this logic is essentially the same as the old
c          sfrm logic.  Hopefully, further simplifications and enhancements
c          will be made in the future. (Especially in gap elements and in
c          multilayer elements)
c          the whole idea of kpheno, a holdover from the sfrm routines,
c          needs to be looked at and possibly eliminated.
c
c *** Notice - This file contains ANSYS Confidential information ***
c
c  input arguments:
c     kelrqq   (int,ar(10),in)     - request keys (needed for this analysis)
c     kelfil   (int,ar(10),in)     - keys indicating matrices on the file
c     elem     (int,sc,in)         - element number
c     ielc     (int,ar(IELCSZ),in) - array of element type characteristics
c     kmasrt   (int,sc,in)         - does the mass matrix have rotational DOF?
c                                       0 - no     1 - yes(with nlgeom, sfrm1n)
c     knlmg    (int,sc,in)         - nonlinear magnetic curve exists in this
c                                     element
c                                      0 - no     1 - yes
c     kconve   (int,sc,in)         - key indicating existence of convections
c                                     in this element
c                                      0,1 - no     2 or more - yes
c                                     must be input as 'i' if not used, as is
c                                     changed in this routine(for analyzer).
c                                     i = 0 must be used in calling routine
c                                     if kpheno = 1.
c     kpheno   (int,sc,in)         - key for type of phenomenon/level of check
c                                     0 - structural like old sfrm1n,1s,3n,3s,fl
c                                     1 - thermal  like old sfrm1c,1t,2t,3t
c                                     2 - electrical/magnetic like some of old
c                                                                      sfrmpo
c                                     3 - general    like old sfrmoo
c     kprop    (int,sc,in)         - key indicating which material properties
c                                     in the prop vector that need to be
c                                     checked (see below)
c     nprop    (int,sc,in)         - number of properties
c     prop     (dp,ar(nprop),in)   - current mat props
c     propo    (dp,ar(nprop),inout)- previous material properties
c     krvro    (int,sc,in)         -
c            = 0 - real constants are used by this element, and the old
c                  values(rvro) have been saved; or the element does not
c                  use real constants.  Any change of real constants
c                  causes all matrices to be reformed.
c            = 1 - real constants are used by this element and the old
c                  values(rvro) have been saved.  However, any change
c                  of real constants will cause the run to terminate,
c                  because the results would be too unpredictable.
c                  (e.g. gap elements)
c            = 2 - element is nonlinear, so do not bother to check
c            = 3 - real constants are used by this element, and the old
c                  values(rvro) have been saved.  However, no checking is
c                  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
c                  decided not to use this much storage.  therefore, no check
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
c                   can be made to determine if matrices should be reformed.
c                   (e.g. 100 layer elements)
c            = 5 - real constants are used by this element, but the old
c                   values(rvro) have not been saved because the real
c                   constants have no effect on matrix formulation.
c                   (e.g. acoustic elements)
c     rvr      (dp,ar(*),in)       - current real constants
c     rvro     (dp,ar(*),inout)    - previous real constants
c     amodo    (dp,sc,inout)       - previous value of mode
c     asymo    (dp,sc,inout)       - previous value of isym
c
c  output arguments:
c     propo    (dp,ar(nprop),inout)- current material properties
c     rvro     (dp,ar(*),inout)    - current real constants
c     amodo    (dp,sc,inout)       - current value of mode
c     asymo    (dp,sc,inout)       - current value of isym
c     kelin    (int,ar(10),out)    - keys indicating matrices to form
c
```

2.2.7. 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
c                          generation run
c *** secondary functions: none

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

c  input arguments:
c     iel      (int,sc,in)       - element number
c     key      (int,sc,in)       - type of load data
c                                   = 1 temperature
c                                   = 2 fluences
c                                   = 3 heat generation rates
c                                   = 4 current densities
c                                   = 9 end pressures (needed for beams/pipes)
c                                   =10 pressures
c                                   =11 film coefficients
c                                   =12 bulk temperatures
c                                   =13 extra displacement shapes
c                                   =14 thermal strains(eptho in el42)
c                                   =15 thermal flux (as in el55)
c                                   =16 initial strains(epino in el01)
c                                   =17 magnetic virtual displacements
c                                   =18 calculated source field(hsn in el96)
c                                   =20 element load vector
c                                   =30 copy - do not scale(tempev in el42)
c                                       first load step only
c     nd       (int,sc,in)       - number of data items

c  output arguments:
c     vect     (dp,ar(nd),out)   - array of load data
c     ka       (int,sc,out)      - load activation key
c                                    = 0 no load for this data
c                                    = 1 load is active
```

2.2.8. 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)
c *** primary function:    write element load vect to file for substructure
```

```
c                         generation run
c *** secondary functions: none

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

c  input arguments:
c     iel      (int,sc,in)       - element number
c     nvect    (int,sc,in)       - number of load vectors
c                                  (current load step number)
c     kkey     (int,sc,in)       - type of load vect
c                                  = 1 temperature
c                                  = 2 fluences
c                                  = 3 heat generation rates
c                                  = 4 current densities
c                                  = 9 end pressures
c                                  =10 pressures
c                                  =11 film coefficients
c                                  =12 bulk temperatures
c                                  =13 extra displacement shapes
c                                  =14 thermal strains(eptho in el42)
c                                  =15 thermal flux (as in el55)
c                                  =16 initial strains(epino in el01)
c                                  =17 magnetic virtual displacements
c                                  =18 calculated source field(hsn in el96)
c                                  =20 element load vector
c                                  =30 copy - do not scale(tempev in el42)
c     nd       (int,sc,in)       - number of vect items
c     vect     (dp,ar(nd),in)    - array of load data
c     ref      (dp,sc,in)        - reference value for zero load

c  output arguments: none
```

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

```
*deck,rvrget
      subroutine rvrget (iel,ireal,ielc,nrvr,rvr)
c *** primary function:  get the real constants for an element

c    typ=int,dp,log,chr,dcp   siz=sc,ar(n),func    intent=in,out,inout

c     variable (typ,siz,intent)     description
c        iel      (int,sc,in)       - element number
c        ireal    (int,sc,in)       - real constant set number
c        ielc     (int,ar(*),in)    - elment type characteristics

c     output arguments:
c        nrvr     (int,sc,out)      - number of real variables
c        rvr      (dp,ar(*),out)    - element real constants

c *** mpg magnetic element usage - iel ?
```

2.2.10. 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

c     propev is used to pass two or more material property numbers
c     thru the lp array to determine which temperature dependent
c     material properties are to be evaluated.
c     thus, the 3 prope1 calls:

c        call prope1 (elem,mat, 1,tem,e(1))
c        call prope1 (elem,mat,10,tem,alpha)
c        call prope1 (elem,mat,13,tem,dens)
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries and affiliates.

```
c     should be combined as:

c        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 ***

c  input arguments:
c     iel  (int,sc,in)       - element number
c     mtr  (int,sc,in)       - material number(input quantity mat, mat comma
c     lp   (int,ar(n),in)    - keys for which specific value is requested
c                                 each group must be in ascending
c                                 order  (ex,ey,ez, etc)
c                               if negative, a required property
c                               if zero, leave prop term unchanged
c        ---- MP command labels --------
c        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,
c        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,
c        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, BVIS=58, QRAT=59, REFT=60, CTEX=61, CTEY=62, CTEZ=63, THSX=64,
c        THSY=65, THSZ=66, DMPR=67, LSSM=68, BETD=69, ALPD=70, RH  =71, DXX =72,
c        DYY =73, DZZ =74, BETX=75, BETY=76, BETZ=77, CSAT=78, CREF=79, CVH =80
c                                              (see mpinit for uncommented code)
c      (see chapter 2 of the elements volume of the user's manual
c       for a detailed description))

c     tem      (dp,sc,in)        - temperature at which to evaluate material
c     n        (int,sc,in)       - number of properties to be evaluated.
c                                   (20 maximum)
c                                   If n = 1, use prope1 instead.

c  output arguments:
c     prop     (dp,ar(n),out)    - values of material property
```

2.2.11. Subroutine prope1 (Evaluating One Material Property)

```
*deck,prope1
      subroutine prope1 (iel,mtr,icon,tem,prop1)
c *** primary function:   to evaluate one material property
c                           (if multiple material properties are to
c                            be evaluated, use propev)
c *** secondary functions: to ensure that certain required props are present

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

c  input arguments:
c     iel      (int,sc,in)       - element number
c     mtr      (int,sc,in)       - material number
c     icon     (int,sc,in)       - key for which specific value is requested
c                                  (negative if property is required)
c        ---- MP command labels --------
c        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,
c        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,
c        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, BVIS=58, QRAT=59, REFT=60, CTEX=61, CTEY=62, CTEZ=63, THSX=64,
c        THSY=65, THSZ=66, DMPR=67, LSSM=68, BETD=69, ALPD=70, RH  =71, DXX =72,
c        DYY =73, DZZ =74, BETX=75, BETY=76, BETZ=77, CSAT=78, CREF=79, CVH =80
c                                              (see mpinit for uncommented code)
```

```
c     tem      (dp,sc,in)        - temperature at which to evaluate material

c  output arguments:
c     prop1    (dp,sc,out)       - value of material property
```

2.2.12. 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
c *** Notice - This file contains ANSYS Confidential information ***
c
c  input arguments:
c     elem     (int,sc,in)         - element number (for anserr)
c     matin    (int,sc,in)         - material reference number
c                                    if negative, no required properties
c     tem      (dp,sc,in)          - temperature for evaluation
c
c  output arguments:
c     prop     (dp,ar(5),out)      - material properties: ex,nuxy,gxy,alpx,dens
c
```

2.2.13. Subroutine tbuser (Retrieving User Table Data)

```
*deck,tbuser
      subroutine tbuser (mat,numitm,tbprop)
c *** primary function:    return the tb data for the user table

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

c  input arguments:
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
```

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

```
*deck,plast1
      subroutine plast1 (option,elem,intpt,mat,kstartL,tem,dtem,e,
     x                   ktform,dens,flu,dflu,epel,eppl,statev,usvr,
     x                   epeq,plwork,sigepl,sigrat,et)
c *** primary function:    to update the plastic history (for 1 component)
c                          used by:  LINK1, LINK8, BEAM23, BEAM24, and
c                                    SOLID65(reinforcing)
c *** secondary functions: to compute the material tangent matrix if requested

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

c  input arguments:
c     option   (int,sc,in)         - plasticity option
c     elem     (int,sc,in)         - element number (label)
c     intpt    (int,sc,in)         - element integration point number
c     mat      (int,sc,in)         - material reference number
c     kstartL  (intL,sc,in)        - virtual starting address of the data table
c     tem      (dp,sc,in)          - temperature at the end of this substep
c     dtem     (dp,sc,in)          - temperature increment over this substep
c     e        (dp,sc,in)          - elastic modulus
c     ktform   (int,sc,in)         - request key for tangent matrix formation
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries and affiliates.

```
c     dens     (dp,sc,in)          - material density
c     flu      (dp,sc,in)          - fluence at the end of this substep
c     dflu     (dp,sc,in)          - fluence increment over this substep
c     epel     (dp,sc,inout)       - modified total strain (trial strain)
c     eppl     (dp,sc,inout)       - plastic strain at previous substep
c     statev   (dp,ar(6),inout)    - state variables at previous substep
c     usvr     (dp,ar(*),inout)    - user-defined state variables (for userpl)
c     epeq     (dp,sc,inout)       - effective plastic strain at prev substep
c     plwork   (dp,sc,inout)       - accumulated plastic work at prev substep

c  output arguments:
c     epel     (dp,sc,inout)       - elastic strain
c     eppl     (dp,sc,inout)       - updated plastic strain
c     statev   (dp,ar(6),inout)    - updated state variables
c     usvr     (dp,ar(*),inout)    - updated user-defined state variables
c     epeq     (dp,sc,inout)       - updated effective plastic strain
c     plwork   (dp,sc,inout)       - updated accumulated plastic work
c     sigepl   (dp,sc,out)         - stress value on stress-strain curve
c     sigrat   (dp,sc,out)         - ratio of trial stress to yield stress
c     et       (dp,sc,out)         - tangent modulus

c  internal variables:
c     deppl    (dp,sc)             - equivalent plastic strain increment
```

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

```
*deck,plast3
      subroutine plast3 (option,elem,intpt,mat,kstartL,ncomp,tem,dtem,
     x prop,d,ktform,dens,flu,dflu,epel,eppl,statev,usvr,epeq,plwork,
     x sigepl,sigrat,dt,kplst,dtt,cmel)
c *** primary function:    to update the plastic history (for 4 or 6 components)
c       used by:  PLANE02, PLANE13, PIPE20, SHELL43, SHELL51, PIPE60,
c                 SOLID62, SOLID65, SHELL91, SHELL93, SHELL143, SOLID191
c       and by way of plast3creep : PLANE42, SOLID45, PLANE82, SOLID92, SOLID95

c *** secondary functions: to compute the material tangent matrix if requested

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

c  input arguments:
c     option   (int,sc,in)         - plasticity option
c     elem     (int,sc,in)         - element number (label)
c     intpt    (int,sc,in)         - element integration point number
c     mat      (int,sc,in)         - material reference number
c     kstartL  (intL,sc,in)        - virtual starting address of the data table
c     ncomp    (int,sc,in)         - number of stress/strain components (4 or 6)
c     tem      (dp,sc,in)          - temperature at the end of this substep
c     dtem     (dp,sc,in)          - temperature increment over this substep
c     prop     (dp,ar(9),in)       - material property array (ex,ey,ez,
c                                    gxy,gyz,gxz, uxy,uyz,uxz)
c     d        (dp,ar(ncomp,ncomp),in) - elastic stress-strain matrix
c     ktform   (int,sc,in)         - request key for tangent matrix formation
c     dens     (dp,sc,in)          - material density
c     flu      (dp,sc,in)          - fluence at the end of this substep
c     dflu     (dp,sc,in)          - fluence increment over this substep
c     epel     (dp,ar(ncomp),inout)- modified total strain (trial strain)
c     eppl     (dp,ar(ncomp),inout)- plastic strain at previous substep
c     statev   (dp,ar(ncomp,6),inout)- state variables at previous substep
c     usvr     (dp,ar(*),inout)    - user-defined state variables (for pluser)
c     epeq     (dp,sc,inout)       - effective plastic strain at prev substep
c     plwork   (dp,sc,inout)       - accumulated plastic work at prev substep
c     kplst    (int,sc,in)         - plane stress key (form dtt if kplst=1)

c  output arguments:
c     epel     (dp,ar(ncomp),inout)- elastic strain
c     eppl     (dp,ar(ncomp),inout)- updated plastic strain
c     statev   (dp,ar(ncomp,6),inout)- updated state variables
```

```
c     usvr     (dp,ar(*),inout)    - updated user-defined state variables
c     epeq     (dp,sc,inout)       - updated effective plastic strain
c     plwork   (dp,sc,inout)       - updated accumulated plastic work
c     sigepl   (dp,sc,out)         - stress value on stress-strain curve
c     sigrat   (dp,sc,out)         - ratio of trial stress to yield stress
c     dt       (dp,ar(ncomp,ncomp),out)- material modulus modified by dscpar
c     dtt      (dp,ar(ncomp,ncomp),out)- consistent tangent modulus
c                                        (formed only if kplst=1)

c  internal variables:
c     deppl    (dp,sc)             - equivalent plastic strain increment
```

2.2.16. 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
c                          used by:  LINK1, LINK8, BEAM23, BEAM24, and
c                                    SOLID65(reinforcing)

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

c  input arguments:
c     option    (int,sc,in)        - creep option
c     elem     (int,sc,in)         - element number (label)
c     intpt    (int,sc,in)         - element integration point number
c     mat      (int,sc,in)         - material reference number
c     kstartL  (intL,sc,in)        - virtual starting address of the data table
c     epel     (dp,sc,inout)       - elastic strain
c     e        (dp,sc,in)          - elastic modulus
c     epcrp    (dp,sc,inout)       - creep strain at previous substep
c     statev   (dp,ar(7),inout)    - state variables at previous substep
c     usvr     (dp,ar(*),inout)    - user-defined state variables (for usercr)
c     tem      (dp,sc,in)          - temperature at the end of this substep
c     dtem     (dp,sc,in)          - temperature increment over this substep
c     fluen    (dp,sc,in)          - fluence at the end of this substep
c     dflu     (dp,sc,in)          - fluence increment over this substep
c     epel     (dp,sc,inout)       - elastic strain adjusted for creep increment
c     sig      (dp,sc,inout)       - stress (not really used)

c  output arguments:
c     epcrp    (dp,sc,inout)       - updated creep strain
c     statev   (dp,ar(7),inout)    - updated state variables
c     usvr     (dp,ar(*),inout)    - updated user-defined state variables
c     sig      (dp,sc,inout)       - stress (recomputed if requested)
```

2.2.17. 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
c       used by:  PLANE02, PLANE13, PIPE20, PLANE42, SHELL43, SOLID45,
c                 SHELL51, PIPE60, SOLID62, SOLID65, PLANE82, SHELL91,
c                 SOLID92, SHELL93, SOLID95, SHELL143, SOLID191

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

c  input arguments:
c     option   (int,sc,in)         - creep option
c     elem     (int,sc,in)         - element number (label)
c     intpt    (int,sc,in)         - element integration point number
c     mat      (int,sc,in)         - material reference number
c     kstartL  (intL,sc,in)        - virtual starting address of the data table
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries and affiliates.

```
c     ncomp    (int,sc,in)         - number of stress/strain components (4 or 6)
c     epel     (dp,ar(ncomp),inout)- elastic strain
c     e        (dp,sc,in)          - elastic young'S MODULUS
c     posn     (dp,sc,in)          - poisson'S RATIO
c     d        (dp,ar(ncomp,ncomp),in) - elastic stress-strain matrix
c     epcrp    (dp,ar(ncomp),inout)- creep strain at previous substep
c     statev   (dp,ar(ncomp*5+2),inout)- state variables at previous substep
c     usvr     (dp,ar(*),inout)    - user-defined state variables (for usercr)
c     tem      (dp,sc,in)          - temperature at the end of this substep
c     dtem     (dp,sc,in)          - temperature increment over this substep
c     fluen    (dp,sc,in)          - fluence at the end of this substep
c     dflu     (dp,sc,in)          - fluence increment over this substep
c     kplst    (int,sc,in)         - plane stress/plane strain key
c     sig      (dp,ar(ncomp),inout)- stresses (not used in input)
c     hsig     (dp,ar(1),inout)    - hydrostatic stress (not used in input)

c  output arguments:
c     epel     (dp,ar(ncomp),inout)- elastic strain adjusted for creep increment
c     epcrp    (dp,ar(ncomp),inout)- updated creep strain
c     statev   (dp,ar(ncomp*5+2),inout)- updated state variables
c     usvr     (dp,ar(*),inout)    - updated user-defined state variables
c     sig      (dp,ar(ncomp),inout)- stresses (redefined if c13 > 0)
c     hsig     (dp,sc,inout)       - hydrostatic stress (redefined if c13 > 0)
```

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

```
*deck,swell1
      subroutine swell1 (option,elem,intpt,mat,kstartL,epswel,epel,e,
     x fluen,dfluen,tem,dtem,usvr)
c *** primary function:    to update the swelling history for 1-d elements
c                          used by:  LINK1, LINK8, BEAM23, and BEAM24

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

c  input arguments:
c     option   (int,sc,in)         - swelling option
c     elem     (int,sc,in)         - element number (label)
c     intpt    (int,sc,in)         - element integration point number
c     mat      (int,sc,in)         - material reference number
c     kstartL  (intL,sc,in)        - virtual starting address of the data table
c     epswel   (dp,sc,inout)       - swell strain at previous substep
c     epel     (dp,sc,inout)       - elastic strain
c     e        (dp,sc,in)          - elastic young'S MODULUS
c     fluen    (dp,sc,in)          - fluence at the end of this substep
c     dfluen   (dp,sc,in)          - fluence increment over this substep
c     tem      (dp,sc,in)          - temperature at the end of this substep
c     dtem     (dp,sc,in)          - temperature increment over this substep
c     usvr     (dp,ar(*),inout)    - user-defined state variables (for usersw)

c  output arguments:
c     epel     (dp,sc,inout)       - elastic strain adjusted for swelling inc
c     epswel   (dp,sc,inout)       - updated swelling strain
c     usvr     (dp,ar(*),inout)    - updated user-defined state variables
```

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

```
*deck,swell3
      subroutine swell3 (option,elem,intpt,mat,kstartL,ncomp,epswel,
     x                   epel,e,nuxy,fluen,dfluen,tem,dtem,usvr)
c *** primary function:    to update the swelling history for 3-d elements
c       used by:  PLANE02, PLANE13, PIPE20, PLANE42, SHELL43, SOLID45,
c                 SHELL51, PIPE60, SOLID62, PLANE82, SHELL91, SOLID92,
c                 SHELL93, SOLID95, SHELL143, SOLID191
```

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

c  input arguments:
c     option   (int,sc,in)         - swelling option
c     elem     (int,sc,in)         - element number (label)
c     intpt    (int,sc,in)         - element integration point number
c     mat      (int,sc,in)         - material reference number
c     kstartL  (intL,sc,in)        - virtual starting address of the data table
c     ncomp    (int,sc,in)         - number of stress/strain components (4 or 6)
c     epswel   (dp,sc,inout)       - swell strain at previous substep
c     epel     (dp,ar(ncomp),inout)- elastic strain
c     e        (dp,sc,in)          - elastic young'S MODULUS
c     nuxy     (dp,sc,in)          - poisson'S RATIO
c     fluen    (dp,sc,in)          - fluence at the end of this substep
c     dfluen   (dp,sc,in)          - fluence increment over this substep
c     tem      (dp,sc,in)          - temperature at the end of this substep
c     dtem     (dp,sc,in)          - temperature increment over this substep
c     usvr     (dp,ar(*),inout)    - user-defined state variables (for usersw)

c  output arguments:
c     epel     (dp,ar(ncomp),inout)- elastic strain adjusted for swelling inc
c     epswel   (dp,sc,inout)       - updated swelling strain
c     usvr     (dp,ar(*),inout)    - updated user-defined state variables
```

2.2.20. Function elLenPsvrBuf (Determining Additional ESAV Record for Plasticity)

```
*deck,elLenPsvrBuf
      function elLenPsvrBuf (mat, plOpt, ncomp)

c**********************************************************************

c     *** primary function:
c              determine additional esave record for plasticity

c     input arguments
c     ===============
c     mat      (int,sc,in)         - material ID
c     plOpt    (int,sc,in)         - plasticity option
c     ncomp    (int,sc,in)         - number of strain components (1,4, or 6)

c     output arguments
c     ================
c     elLenPsvrBuf  (int,sc,out)   - number of extra data items saved

c     local variables
c     ===============

c**********************************************************************
```

2.2.21. 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 ***

c     input arguments:
c        variable (typ,siz,intent)    description
c        mat      (int,sc,in)       - material number
c        iprop    (int,sc,in)       - property number (tbpnum in tblecm)
c                                        use 13 for tb,user
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
c                                       use 14 for tb,nl

c     output arguments:
c        variable (typ,siz,intent)    description
c        nlget   (int,sc,out)       - number of property values
c        prop    (dp,ar(nlget),out) - vector of the property values
c                                      (the first 15(tbhdsz) items are a header,
c                                      given below.  The terms are defined in
c                                      tblecm.inc)
c        --- terms of the descriptor record:
c        header(1) = tbtyp
c        header(2) = tbtems
c        header(3) = temloc
c        header(4) = dprtem
c        header(5) = tbrow
c        header(6) = tbcol
c        header(7) = rowkey
c        header(8) = nxtloc
c        header(9) = nxttem
c        header(10) = temptr
c        header(11) = tbpt
c        header(12) = tbsiz
c        header(13) = tbopt
c        header(14) = hypopt
c        header(15) = tbnpts
```

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

```
*deck,usereo
      subroutine usereo (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)
c
c *** primary function:  to call userou, which allows user to store
c                              data in nmisc record
c
c *** Notice - This file contains ANSYS Confidential information ***
c
c  input arguments:
c   variable (typ,siz,intent)       description
c    elem    (int,sc,in)           - element number
c    iout    (int,sc,in)           - output unit number
c    nbsvr   (int,sc,in)           - number of basic element variables
c    bsvr    (dp,ar(nbsvr),in)     - basic element variables
c    nnrsvr  (int,sc,in)           - number of nonlinear element variables
c    nrsvr   (dp,ar(nnrsvr),in)    - nonlinear element variables
c    npsvr   (int,sc,in)           - number of plasticity element variables
c    psvr    (dp,ar(npsvr),in)     - plasticity element variables
c    ncsvr   (int,sc,in)           - number of creep element variables
c    csvr    (dp,ar(ncsvr),in)     - creep element variables
c    nusvr   (int,sc,in)           - number of user-supplied element variables
c    usvr    (dp,ar(nusvr),in)     - user-supplied element variables
c    nnode   (int,sc,in)           - number of nodes
c    nodes   (int,ar(nnode),in)    - node numbers
c    xyz     (dp,ar(6,nnode),in)   - nodal coordinates and rotations (virgin)
c    vol     (dp,sc,in)            - element volume (or area if 2-d)
c    leng    (dp,sc,in)            - element length (beams,spars,etc)
c    time    (dp,sc,in)            - current time
c    timinc  (dp,sc,in)            - current sub step time increment
c    nutot   (int,sc,in)           - length of dof solution vector utot
c    utot    (dp,ar(nutot),in)     - solution vector
c    maxdat  (int,sc,in)           - size of user output array (3 x nnode)
c                                     actually, = ielc(nmnmup)
c                                     for contact element it is equale to nusvr
c                                     but it does not exceed 120
c
c  output arguments:
c   variable (typ,siz,intent)       description
```

```
c    numdat  (int,sc,out)          - number of user output items in array udbdat
c    udbdat  (dp,ar(maxdat),out)   - user output items to be placed at the end
c                                     of the nmisc record
c
```

2.2.23. 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:
c        ielem     (int,sc,in)        - element number
c        edtype    (int,sc,in)        - element data type (see elparm)
c        lcerstL   (LONG,sc,inout)    - pointer to results file position
c        edindxL   (LONG,ar(NUMELEDATASETS),inout)- index to results file data
c        nval      (int,sc,in)       - the total number of values
c                                         if edtype = EDEMS,
c                                       this should -always- be ielc(nmsmis),
c                                       unless there is a variable number, as
c                                       in the layered shell elements.
c        value     (dp,ar(nval),in)   - output values (real)
```

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

```
*deck,eldwrnL
      subroutine eldwrnL (elem,ielc,lcerstL,edindxL,nudb,udbdat,
     x                    nval,value,ndval)
c *** primary function:    output element nonsummable miscellaneous data
c                                 to result file
c *** Notice - This file contains ANSYS Confidential information ***

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

2.2.25. Subroutine trrot (Calculating 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 ***
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
c  input arguments:
c     tr       (dp,ar(3,3),in)     - transformation matrix

c  output arguments:
c     rot      (dp,ar(3),out)      - rotation vector
```

2.2.26. Subroutine rottr (Calculating 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 ***
c
c     ref(old): eqn. 20(transposed),rankin and brogan, jpvt,108(1986)165-174.
c     ref(new): eqn. (b.4), simo and vu-quoc, cmame, 58 (1986), 79-116
c                 (removes singularities at pi and 2*pi)
c
c  input arguments:
c     variable (typ,siz,intent)    description
c     rot      (dp,ar(4),in)     - rotation parameter in radians
c
c  output arguments:
c     variable (typ,siz,intent)    description
c     tr       (dp,ar(3,3),out)  - transformation matrix corresponding to rot
```

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

```
*deck,xyzup3
      subroutine xyzup3 (nnod,u,nr,xyz,nx,xyzup)
c *** primary function:  update a 3-d ele nodal coords for large deformation
c
c *** Notice - This file contains ANSYS Confidential information ***
c
c  input arguments:
c     nnod     (int,sc,in)          - number of nodes
c     u        (dp,ar(nr),in)       - displacement vector
c     nr       (int,sc,in)          - size of the u vector
c     xyz      (dp,ar(nx,nnod),in)  - coordinates to be updated
c     nx       (int,sc,in)          - row size of xy
c
c  output arguments:
c     xyzup     (dp,ar(3,nnod),out) - updated coordinates
c
```

2.2.28. 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
c *** Notice - This file contains ANSYS Confidential information ***
c
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
c     pseudovector uses the nomalization proposed by Rankin and Brogan (ref:
c     eqn. 15, JPVT, 108(1986)165-174).
c     CMAME = Computer Methods in Applied Mechanics and Engineering
c     JPVT = Journal of Presssure Vessel Technology (ASME)
c
c   variable descriptions:
c    input:
```

```
c     v2     - rotation increment
c     w1     - previous rotation pseudovector
c    output:
c     w1     - updated pseudovector
c
c        v1 = cos(v1/2) + 1/2*w1,   w1 = 2*sin(v1/2)*e1
c        v2 = cos(v2/2) + 1/2*w2,   w2 = 2*sin(v2/2)*e2
c        v21 = v2*v1 = cos(v21/2) + 1/2*w21 (quarternion multiplication)
c        w1 =: v21  (w1 is updated)
c
```

2.2.29. Subroutine tmpget (Defining Current Temperature Loads)

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

2.2.30. Subroutine prsget (Defining Current Pressure Loads)

```
*deck,prsget
      subroutine prsget (iel,ielc,nfac,ndat,begdat,dat,enddat,iexist)

c     primary function: define the current pressure loads

c     See also:  PrsRIGet

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

c     input arguments:
c        iel      (int,sc,in)      - element number
c        ielc  (int,ar(IELCSZ),in) - array of element type characteristics
c        nfac     (int,sc,in)      - number of pressure faces
c        ndat     (int,sc,in)      - number of pressure values
c        begdat   (dp,ar(ndat),in) - pressure at the beginning of load step

c     output arguments:
c        dat      (dp,ar(ndat),out) - pressures at this iteration
c        enddat   (dp,ar(ndat),out) - pressure at end of this load step
c        iexist   (int,sc,out)      - flag if pressure exist
c                                     = 0 - no pressure
c                                     = 1 - yes pressure
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

2.2.31. Subroutine cnvget (Defining Current Convection Loads)

```
*deck,cnvget
      subroutine cnvget (iel,ielc,nr,u,nfac,ndat,beghc,begtb,
     x   hc,tb,endhc,endtb,iexist)
c     primary function: define the current convection loads

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

c     input arguments:
c        iel      (int,sc,in)      - element number
c        ielc  (int,ar(IELCSZ),in) - array of element type characteristics
c        nr       (int,sc,in)      - dimension of u (temperature) vector
c        u        (dp,ar(nr),in)   - most current temperatures
c        nfac     (int,sc,in)      - number of convection faces
c        ndat     (int,sc,in)      - number of convection values
c        beghc    (dp,ar(ndat),in) - hcoef at the beginning of load step
c        begtb    (dp,ar(ndat),in) - tbulk at the beginning of load step

c     output arguments:
c        hc       (dp,ar(ndat),out) - hcoef at this substep
c        tb       (dp,ar(ndat),out) - tbulk at this substep
c        endhc    (dp,ar(ndat),in)  - hcoef at the end of this load step
c        endtb    (dp,ar(ndat),in)  - tbulk at the end of this load step
c        iexist   (int,sc,out)      - flag if convection exist
c                                     = 0 - no convection
c                                     = 1 - constant convection (with time)
c                                           does not require new element matrix
c                                     = 2 - changing convection (with time)
c                                           or deleted convection
c                                           requires new element matrix
```

2.2.32. Subroutine hgnget (Defining Current Heat Generation Loads)

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

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

2.2.33. Subroutine prinst (Calculating Principal Stress and Stress Intensity)

```
*deck,prinst
      subroutine prinst (s)
c    primary function:  computes principal stresses and stress intensity
c    secondary functions:  none
c *** Notice - This file contains ANSYS Confidential information ***

c  input arguments:
c     variable (typ,siz,intent)    description
c     s        (dp,ar(11),inout)   - stress vector
c                  s(1)=sx
c                  s(2)=sy
c                  s(3)=sz
c                  s(4)=sigxy
c                  s(5)=sigyz
c                  s(6)=sigzx

c
c  output arguments:
c     variable (typ,siz,intent)    description
c     s        (dp,ar(11),inout)   - stress vector
c                  s(7)=sig1
c                  s(8)=sig2
c                  s(9)=sig3
c                  s(10)=s.i.
c                  s(11)=sige

c ********* note: all changes to this routine must be made in
c                 post1  (paprst)
c
```

2.3. Subroutines for Modifying and Monitoring Existing Elements

Following are the user subroutines for modifying or monitoring existing elements:

2.3.1. Subroutine userou (Storing User-Provided Element Output)
2.3.2. Subroutine useran (Modifying Orientation of Material Properties)
2.3.3. Subroutine userrc (Performing User Operations on COMBIN37 Parameters)
2.3.4. Subroutine UElMatx (Accessing Element Matrices and Load Vectors)
2.3.5. Subroutine uthick (Getting User-Defined Initial Thickness)
2.3.6. Subroutine UsrFictive (Providing User-Defined Fictive Temperature Relationship)
2.3.7. Subroutine uflex (Calculating Flexibility Factors for PIPE288 and PIPE289)
2.3.8. Subroutine UsrShift (Calculating Pseudotime Time Increment)
2.3.9. Subroutine UTimeInc (Overriding the Program-Determined Time Step)
2.3.10. Subroutine UCnvrg (Overriding the Program-Determined Convergence)

2.3.1. Subroutine userou (Storing User-Provided 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)
c
c *** primary function:    store user supplied element output
c                           in nmisc record
c
c         in order to activate this user programmable feature,
c         the user must enter the usrcal command.
c
c
c         *** Copyright ANSYS.  All Rights Reserved.
c         *** ansys, inc.
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
c *** Notice - This file contains ANSYS Confidential information ***
c
c             this routine is called by almost every element
c             the data is stored on the nmisc record.
c             warning:  other data may be stored between the
c                       documented data and this data.
c             in order to see the actual information on the nmisc
c              record, insert the command:
c                  dblist,elp,elnum1,elnum2,elinc,11
c                       where elnum1 = the first element
c                             elnum2 = the last element
c                             elinc  = the element increment number
c              after a set command in post1.
c
c  input arguments:
c   variable (typ,siz,intent)       description
c    elem    (int,sc,in)           - element number
c    iout    (int,sc,in)           - output unit number
c    nbsvr   (int,sc,in)           - number of basic element variables
c    bsvr    (dp,ar(nbsvr),in)     - basic element variables
c    nnrsvr  (int,sc,in)           - number of nonlinear element variables
c    nrsvr   (dp,ar(nnrsvr),in)    - nonlinear element variables
c    npsvr   (int,sc,in)           - number of plasticity element variables
c    psvr    (dp,ar(npsvr),in)     - plasticity element variables
c    ncsvr   (int,sc,in)           - number of creep element variables
c    csvr    (dp,ar(ncsvr),in)     - creep element variables
c    nusvr   (int,sc,in)           - number of user-supplied element variables
c                                         (= nstv on the nsvr command)
c    usvr    (dp,ar(nusvr),in)     - user-supplied element variables
c    nnode   (int,sc,in)           - number of nodes
c    nodes   (int,ar(nnode),in)    - node numbers
c    xyz     (dp,ar(6,nnode),in)   - nodal coordinates and rotations (virgin)
c    vol     (dp,sc,in)            - element volume (or area if 2-d)
c    leng    (dp,sc,in)            - element length (beams,spars,etc)
c    time    (dp,sc,in)            - current time
c    timinc  (dp,sc,in)            - current sub step time increment
c    nutot   (int,sc,in)           - length of dof solution vector utot
c    utot    (dp,ar(nutot),in)     - solution vector
c    maxdat  (int,sc,in)           - size of user output array (3 x nnode)
c                                     for contact element it is equale to nusvr
c                                     but it dode not exceed 120
c
c  output arguments:
c   variable (typ,siz,intent)       description
c    numdat  (int,sc,out)          - number of user output items in array udbdat
c                                      (maximum size of numdat is ielc(NMNMUP)
c                                      which is usually three times the number
c                                      of nodes.
c                                      For contact elements CONTA171-178,it
c                                      should be equal or less than NSTV
c                                      on nsvr command). It cannot exceed 120.
c    udbdat  (dp,ar(maxdat),out)   - user output items to be placed at the end
c                                    of the nmisc record
c
```

2.3.2. Subroutine useran (Modifying Orientation of Material Properties)

```
*deck,useran                      USERSDISTRIB
      subroutine useran  (vn,vref,elem,thick,xyzctr,bsangl)
c     user written routine to modify orientation of material properties
c     and stresses ***************************
c       applicable to: shell43,63,91,93,99, solid46,64,191
c        accessed by keyopt
c
c         *** Copyright ANSYS.  All Rights Reserved.
c         *** ansys, inc.
c *** Notice - This file contains ANSYS Confidential information ***
c
c **** warning *** do not change any arguments other than bsangl.
```

```
c                 if you do, your results are probably wrong.
c
c  input(do not change)---
c     vn       = vector normal to element
c     vref     = unit vector orienting element, essentially edge i-j
c     elem     = element number
c     thick    = total thickness of element at this point (see note below)
c     xyzctr = location of element centroid or integration point
c
c  output---
c     bsangl = output from this subroutine.  it represents the angle(s)
c              between vref and the desired orientation.  it may have
c              the default orientation coming in to useran.
c                This will be combined with the angles derived from
c                the ESYS command.
c           use 1 angle for 2-d elements and shells
c           use 3 angles for 3-d solids
c
```

2.3.3. Subroutine userrc (Performing User Operations on COMBIN37 Parameters)

```
*deck,userrc                        USERSDISTRIB
      subroutine userrc (elem,ireal,type,nusvr,usvr,parm,parmld,
     x c1,c2,c3,c4,fcon)
c     primary function: user operation on parameter for combin37
c      accessed with keyopt(9) = 1
c
c         *** Copyright ANSYS.  All Rights Reserved.
c         *** ansys, inc.
c *** Notice - This file contains ANSYS Confidential information ***
c
c  input arguments:
c   variable (typ,siz,intent)       description
c    elem    (int,sc,in)           - element number
c    ireal    (int,sc,in)          - element real constant number
c    type    (int,sc,in)           - element type number
c    nusvr   (int,sc,in)           - number of user-supplied element variables
c                                     (input with the NSVR command)
c    usvr    (dp,ar(nusvr),inout)  - user-supplied element variables
c    parm    (dp,sc,in)            - current value of the paramater
c    parmld  (dp,sc,in)            - value of the parameter at previous time ste
c    c1      (dp,sc,in)            - real constant c1
c    c2      (dp,sc,in)            - real constant c2
c    c3      (dp,sc,in)            - real constant c3
c    c4      (dp,sc,in)            - real constant c4
c
c  output arguments:
c   variable (typ,siz,intent)       description
c    usvr    (dp,ar(nusvr),inout)  - user-supplied element variables
c                                     may be sent .rst file with usereo
c    fcon    (dp,sc,out)           - result of calculation
c
c     either c1 or c3 must be nonzero for this logic to be accessed,
c
```

2.3.4. Subroutine UElMatx (Accessing Element Matrices and Load Vectors)

```
*deck,UElMatx                       USERSDISTRIB
      subroutine UElMatx (elem,nr,ls,zs,zsc,uelm,ielc,nodes,
     x                    ElDofEachNode,elmdat,xyzang,lenu)

c primary function:    User routine to access element matrices and load vectors.
c                      Needs to have USRCAL,UELMATX to be accessed.
c                      Called after the call to the element routine and
c                       before the solver.
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
c                     May be used to monitor and/or modify the element matrices
c                                                    and load vectors.

c         *** Copyright ANSYS.  All Rights Reserved.
c         *** ansys, inc.

c     typ=int,dp,log,chr,dcp   siz=sc,ar(n)   intent=in,out,inout

c  input arguments:
c     variable (typ,siz,intent)        description
c     elem     (int,sc,in)          - User element number
c     nr       (int,sc,in)          - number of rows in element matrix
c     ls       (int,ar(nr),in)      - Dof Index vector for this element matrix
c     zs       (dp,ar(nr,nr,*),inout)- K,M,C,SS,KCPLX matrices for this element
c     zsc      (dp,ar(nr,2),inout)  - Element load vector and N-R correction vec
c     uelm     (dp,ar(nr,5),in)     - Nodal displacements for this element
c     ielc     (int,ar(*),in)       - Element type characteristics
c     nodes    (int,ar(*),in)       - Nodes for this element
c     ElDofEachNode (int,ar(nr),in) - list of dofs for each node in Global
c     elmdat   (int,ar(10),in)      - Element data for this element
c     xyzang   (dp,ar(6,*),in)      - X,Y,Z,THXY,THYZ,THZX for each element node
c     lenu     (int,sc,in)          - Length of global displacement vector

c  output arguments:
c     zs       (dp,ar(nr,nr,4),inout)- K,M,C,SS matrices for this element
c     zsc      (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
c      impact on the solution, possibly giving meaningless results.  The normal
c      usage of this routine is simply monitor what is happening.
```

2.3.5. Subroutine uthick (Getting User-Defined Initial Thickness)

```
*deck,uthick                       USERSDISTRIB
      SUBROUTINE uthick (elemId, elemType, matId, realId,
     $                   numDomIntPts, curCoords, thickness)
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c     *** primary function: get the user defined thickness
c
c         *** Copyright ANSYS.  All Rights Reserved.
c         *** ansys, inc.
c
c     input arguments
c     ===============
c     Variable        (type,sz,i/o)  description
c      elemId         (int,sc,i)     element number
c      elemType       (int,sc,i)     element TYPE (181 etc.)
c      matId          (int,sc,i)     material number
c      realId         (int,sc,i)     real constant set number
c      numDomIntPts   (int,sc,i)     number of integration points
c      curCoords      (dp,ar(3,numDomIntPts),i)
c                                    current coordinates
c
c     output arguments
c     ================
c      thickness      (dp,ar(3,numDomIntPts),o)
c                                    thickness at the integration points
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c --- parameters
c
```

2.3.6. 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
c                         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
c                            fictive temperature relationship
c                       this routine could also be used to modify the viscoelastic
c                       data during solution, i.e., to make the viscoelastic
c                       coefficients themselves time-dependent.
c
c *** notice- this routine contains ansys,inc. confidential information ***
c
c         *** Copyright ANSYS.  All Rights Reserved.
c         *** ansys, inc.
c
c  input arguments:
c     variable (type,sze,intent)    description
c     tref     (dp,sc,in)          - reference temperature
c     toffst   (dp,sc,in)          - temperature offset from absolute zero
c     tem      (dp,sc,in)          - temperature at the end of this substep
c     ftl      (dp,sc,in)          - previous fictive temperature
c     veinpt   (dp,ar(95),inout)   - table from tb,evisc
c
c  output arguments:
c     variable (type,sze,intent)    description
c     veinpt   (dp,ar(95),inout)   - table from tb,evisc
c     ftc      (dp,sc,in)          - fictive temperature
c
```

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


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
c  twten     (dp,sc,inout) - twist-tension factor
c
```

2.3.8. Subroutine UsrShift (Calculating Pseudotime Time Increment)

```
*deck,                      USERSDISTRIB
c Copyright ANSYS.  All Rights Reserved.
      subroutine UsrShift(dxi,dxihalf,timinc,
     &                    temp,dtemp,toffst,propsh,nTerms)
c*************************************************************************
c     calculate pseudotime time increment according
c     to a user specified shift function
c
c *** Notice - This file contains ANSYS Confidential information ***
c
c  input arguments:
c     timinc  (dp,sc,in)         - time increment
c     temp    (dp,sc,in)         - current temperature, t_n+1
c     dtemp   (dp,sc,in)         - temperature increment, t_n+1 - t_n
c     toffst  (dp,sc,in)         - temperature offset to absolute zero
c                                  (specified by TOFFST command)
c     propsh  (dp,ar,in)         - Constants for shift function
c                                  (User's input using TB,SHIFT,,,,USER)
c     nTerms  (int,ar,in)        - number of user defined constants
c                                  (specified in TB,SHIFT,,,nTerms,USER)
c  output arguments:
c    dxi      (dp,sc,out)        - pseudotime increment
c    dxihalf  (dp,sc,out)        - pseudotime increment over the upper half span
c*************************************************************************
```

2.3.9. Subroutine UTimeInc (Overriding the Program-Determined Time Step)

This subroutine allows you to create a user-defined time step to override the one determined by the program. Activate the subroutine via the **USRCAL**,UTIMEINC command.

```
*deck,UTimeInc                       USERSDISTRIB
      subroutine UTimeInc (deltmin,deltmax,delt)

c primary function:    User routine to override the program determined time step
c                      Needs to have USRCAL,UTIMEINC to be accessed.
c                      Called after the program determined the next time step
c                        increment (AUTOTS,ON only)

c         *** Copyright ANSYS.  All Rights Reserved.
c         *** ansys, inc.

c  input arguments:
c     deltmin  (int,dp,in)    - minimum time step size (user input)
c     deltmax  (int,dp,in)    - maximum time step size (user input)
c     delt     (int,dp,inout) - on input, the value determined by the program

c  output arguments:
c     delt     (int,dp,inout) - on output, the value you have determined
```

2.3.10. Subroutine UCnvrg (Overriding the Program-Determined Convergence)

This subroutine allows you to create user-defined convergence checking and to override the convergence determined by the program. Activate the subroutine via the **USRCAL**,UCNVRG command.

```
*deck,UCnvrg                        USERSDISTRIB
      subroutine UCnvrg (ConvergenceType,ConvergenceFlag)
```

```
c primary function:    User routine to perform custom convergence checking and
c                        override the program-determined convergence
c                      Needs to have USRCAL,UCNVRG to be accessed.
c                      Called after the program convergence checks.

c         *** Copyright ANSYS.  All Rights Reserved.
c         *** ansys, inc.

c  input arguments:
c     ConvergenceType  (int,sc,in)    - type of convergence to be checked
c                                         1, nonlinear element (called after
c                                            element matrix formation)
c                                         2, force convergence (called after
c                                            element matrix formation)
c                                         3, displacement convergence (called after
c                                            equation solution)
c     ConvergenceFlag  (int,sc,inout) - on input, the value the program determined
c                                       for this Type
c                                         0, not converged
c                                         1, converged

c  output arguments:
c     ConvergenceFlag  (int,sc,inout) - on output, the value the you have determined
c                                       for this Type
c                                         0, not converged
c                                         1, converged

c     Note: For overall convergence, all 3 Types must be converged. Not all
c           Types are evaluated (dependent on CNVTOL input and program defaults)
```

2.4. Subroutines for Customizing Material Behavior

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

2.4.1. Subroutine UserMat (Creating Your Own Material Model)
2.4.2. Subroutine UserMatTh (Creating Your Own Thermal Material Model)
2.4.3. Subroutine UserHyper (Writing Your Own Hyperelasticity Laws)
2.4.4. Subroutine UserCreep (Defining Creep Material Behavior)
2.4.5. Subroutine user_tbelastic (Defining Material Linear Elastic Properties)
2.4.6. Subroutine userfc (Defining Your Own Failure Criteria)
2.4.7. Subroutine userCZM (Defining Your Own Cohesive Zone Material)
2.4.8. Subroutine userswstrain (Defining Your Own Swelling Laws)
2.4.9. Subroutine userck (Checking User-Defined Material Data)
2.4.10. Supporting Function egen
2.4.11. Subroutine userfld (Update User-Defined Field Variables)
2.4.12. Subroutine userthstrain (Defining Your Own Thermal Strain)

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 the program 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.

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

2.4.1. 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, Inc. 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:

2.4.1.1. UserMat Overview
2.4.1.2. Stress, Strain, and Material Jacobian Matrix
2.4.1.3. The UserMat API
2.4.1.4. UserMat Variables
2.4.1.5. Table (TB) Commands for UserMat
2.4.1.6. Material Constitutive Integration with UserMat
2.4.1.7. UserMat Restrictions
2.4.1.8. Accessing Material and Element Data for UserMat
2.4.1.9. Utility Functions for UserMat

For a UserMat subroutine example, see Appendix C: User Material (UserMat) Subroutine Example (p. 355).

2.4.1.1. UserMat Overview

The UserMat subroutine defines the material stress-strain relationship of a material and applies to any 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. The program 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.

The two most important UserMat-calculated quantities are stress and the consistent tangent stiffness:

- The measure of the stress is Cauchy stress (true stress), σ.
- The consistent tangent stiffness is defined as $\partial \Delta \boldsymbol{\sigma} / \partial \Delta \boldsymbol{\varepsilon}$, where $\Delta \boldsymbol{\sigma}$ is the stress increment and $\Delta \boldsymbol{\varepsilon}$ is the strain increment. The measure of strain is logarithmic strain (true strain), $\boldsymbol{\varepsilon}$.

Generally, the material model can be formulated in either incremental (rate) or total forms.

- The incremental form is suitable for material models such as plasticity. The program uses a co-rotated framework for such material formulation. All variables are defined and updated in the co-rotated material coordinate system. The implementation of a constitutive model looks exactly the same for both small and large deformation (**NLGEOM**,ON).
- The total form is commonly used with hyperelastic material, and the material response is characterized by a potential relating to the invariants of the deformation gradient. The discrete equations are formulated with respect to the reference configuration. The deformation gradient is defined in the global coordinate system (with the exception of shell elements, where the deformation gradient is expressed on the co-rotated element coordinate system).

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

For most of the hyper-elastic models, material behavior is considered to be incompressible or nearly incompressible, resulting in a singular element matrix and possibly leading to element volumetric locking and solution failure. It is therefore necessary to take measures to avoid the singularity. The penalty method of carefully selecting the penalty parameter (which is also the compressibility parameter) is the simplest approach. Elements with mixed u-P formulation are the best way to address the singularity caused by material incompressibility. To use this option, set *TB0PT* = MXUP and element KEYOPT(6) = 1.

For a user-defined material with purely incompressible behavior, a mixed formulation in which the pressure is a degree of freedom is necessary. When a nearly incompressible hyperelastic material is formulated in `UserMat`, additional information (derivatives of volumetric potential with regard to determinant of deformation gradient J) is required. For more information, see UserMat Variables (p. 191).

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. 194).

Further Reading

ANSYS, Inc. recommends the following resource to help you understand how user-defined materials are implemented:

> 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.

2.4.1.2. 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 $\partial \Delta\sigma_{ij} / \partial \Delta\varepsilon_{ij}$. $\Delta\sigma_{ij}$ is the stress increment, and $\Delta\varepsilon_{ij}$ is the strain increment.

`UserMat` is based on the current configuration for nonlinear geometry analysis (**NLGEOM**,ON). The program uses a co-rotational approach to account for rigid body rotation. Because the program 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 strain and axisymmetric stress states
11, 22, 33, 12

2-D plane stress states
11, 22, 12

Beam element stress states
11, 13, 12

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

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

2.4.1.3. The UserMat API

Following is the interface for the `UserMat` subroutine:

```
*deck,usermat                        USERSDISTRIB
      subroutine usermat(
     &                   matId, elemId,kDomIntPt, kLayer, kSectPt,
     &                   ldstep,isubst,keycut,
     &                   nDirect,nShear,ncomp,nStatev,nProp,
     &                   Time,dTime,Temp,dTemp,
     &                   stress,ustatev,dsdePl,sedEl,sedPl,epseq,
     &                   Strain,dStrain, epsPl, prop, coords,
     &                   var0, defGrad_t, defGrad,
     &                   tsstif, epsZZ,
```

```
     &                  var1, var2, var3, var4, var5,
     &                  var6, var7, var8)
c************************************************************************
c     *** primary function ***
c
c           user defined material constitutive model
c
c      Attention:
c           User must define material constitutive law properly
c           according to the stress state such as 3D, plane strain
c           and axisymmetry, plane stress and 3D/1D beam.
c
c           a 3D material constitutive model can use for
c           plane strain and axisymmetry cases.
c
c           When using shell elements, a plane stress algorithm
c           must be use.
c
c                                          gal July, 1999
c
c       The following demonstrates a USERMAT subroutine for
c       a plasticity model, which is the same as TB, BISO,
c       for different stress states.
c       See "ANSYS user material subroutine USERMAT" for detailed
c       description of how to write a USERMAT routine.
c
c       This routine calls four routines,
c       usermat3d.F, usermatps.F usermatbm.F and usermat1d.F, w.r.t.
c       the corresponding stress states.
c       Each routine can be also a usermat routine for the specific
c       element.
c
c************************************************************************
c
c     input arguments
c     ===============
c      matId     (int,sc,i)              material #
c      elemId    (int,sc,i)              element #
c      kDomIntPt (int,sc,i)              "k"th domain integration point
c      kLayer    (int,sc,i)              "k"th layer
c      kSectPt   (int,sc,i)              "k"th Section point
c      ldstep    (int,sc,i)              load step number
c      isubst    (int,sc,i)              substep number
c      nDirect   (int,sc,in)             # of direct components
c      nShear    (int,sc,in)             # of shear components
c      ncomp     (int,sc,in)             nDirect + nShear
c      nstatev   (int,sc,l)              Number of state variables
c      nProp     (int,sc,l)              Number of material ocnstants
c
c      Temp      (dp,sc,in)              temperature at beginning of
c                                        time increment
c      dTemp     (dp,sc,in)              temperature increment
c      Time      (dp,sc,in)              time at beginning of increment (t)
c      dTime     (dp,sc,in)              current time increment (dt)
c
c      Strain   (dp,ar(ncomp),i)         Strain at beginning of time increment
c      dStrain  (dp,ar(ncomp),i)         Strain increment
c      prop     (dp,ar(nprop),i)         Material constants defined by TB,USER
c      coords   (dp,ar(3),i)             current coordinates
c      defGrad_t(dp,ar(3,3),i)           Deformation gradient at time t
c      defGrad  (dp,ar(3,3),i)           Deformation gradient at time t+dt
c
c     input output arguments
c     ======================
c      stress   (dp,ar(nTesn),io)        stress
c      ustatev   (dp,ar(nstatev),io)     user state variables
c      sedEl    (dp,sc,io)               elastic work
c      sedPl    (dp,sc,io)               plastic work
c      epseq    (dp,sc,io)               equivalent plastic strain
c      tsstif   (dp,ar(2),io)            transverse shear stiffness
c                                        tsstif(1) - Gxz
c                                        tsstif(2) - Gyz
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries and affiliates.

```
c                                       tsstif(1) is also used to calculate hourglass
c                                       stiffness, this value must be defined when low
c                                       order element, such as 181, 182, 185 with uniform
c                                       integration is used.
c      var?     (dp,sc,io)              not used, they are reserved arguments
c                                       for further development
c
c     output arguments
c     ================
c      keycut   (int,sc,io)             loading bisect/cut control
c                                       0 - no bisect/cut
c                                       1 - bisect/cut
c                                       (factor will be determined by ANSYS solution control)
c      dsdePl   (dp,ar(ncomp,ncomp),io) material jacobian matrix
c      epsZZ    (dp,sc,o)               strain epsZZ for plane stress,
c                                       define it when accounting for thickness change
c                                       in shell and plane stress states
c
c*************************************************************************
c
c      ncomp   6   for 3D  (nshear=3)
c      ncomp   4   for plane strain or axisymmetric (nShear = 1)
c      ncomp   3   for plane stress (nShear = 1)
c      ncomp   3   for 3d beam      (nShear = 2)
c      ncomp   1   for 1D (nShear = 0)
c
c      stresss and strains, plastic strain vectors
c          11, 22, 33, 12, 23, 13    for 3D
c          11, 22, 33, 12            for plane strain or axisymmetry
c          11, 22, 12                for plane stress
c          11, 13, 12                for 3d beam
c          11                        for 1D
c
c      material jacobian matrix
c        3D
c           dsdePl    |  1111   1122   1133   1112   1123   1113 |
c           dsdePl    |  2211   2222   2233   2212   2223   2213 |
c           dsdePl    |  3311   3322   3333   3312   3323   3313 |
c           dsdePl    |  1211   1222   1233   1212   1223   1213 |
c           dsdePl    |  2311   2322   2333   2312   2323   2313 |
c           dsdePl    |  1311   1322   1333   1312   1323   1313 |
c        plane strain or axisymmetric (11, 22, 33, 12)
c           dsdePl    |  1111   1122   1133   1112 |
c           dsdePl    |  2211   2222   2233   2212 |
c           dsdePl    |  3311   3322   3333   3312 |
c           dsdePl    |  1211   1222   1233   1212 |
c        plane stress (11, 22, 12)
c           dsdePl    |  1111   1122   1112 |
c           dsdePl    |  2211   2222   2212 |
c           dsdePl    |  1211   1222   1212 |
c        3d beam (11, 13, 12)
c           dsdePl    |  1111   1113   1112 |
c           dsdePl    |  1311   1313   1312 |
c           dsdePl    |  1211   1213   1212 |
c        1d
c           dsdePl    |  1111 |
c
c*************************************************************************
```

2.4.1.4. UserMat Variables

The `UserMat` subroutine uses the following Input (p. 191), Input/Output (p. 193), and Output (p. 193) variables. Do not change them in the subroutine code.

`UserMat` Input Arguments	
matId	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 containing the current temperature.
dTemp	Double-precision variable containing the current temperature increment.
Time	Double-precision variable containing the total time at the beginning of the time increment.
dTime	Double-precision variable containing the current time increment.
Strain	Double-precision array containing 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 approximately the logarithmic strains. When the mixed u-P formulation option (*TB0PT* = MXUP on the **TB**,USER command) is used for hyperelastic material, the strain array is the logarithmic strains at the current time. However, the strain array can be redefined within the `UserMat` subroutine. For nearly incompressible hyperelastic material, a mixed u-J formulation is used. The calculated J is passed from strain array as strain(ncomp+1).
dStrain	Double-precision array containing 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. When the mixed u-P formulation option (*TB0PT* = MXUP on the **TB**,USER command) is used for hyperelastic material, the dStrain array is zero.
prop	Double-precision array containing 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 containing the current coordinates of the material integration points. Array size is 3.
defGrad_t	Double-precision matrix containing deformation gradient at the beginning of the time increment. The matrix size is 3 x 3. The matrix components DefGrad_$t_{(i,j)}$ are equivalent to deformation gradient F_{ij} at the beginning of the time increment and are only available for continuum and shell elements with nonlinear deformation (**NLGEOM**,ON).

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

defGrad	Double-precision matrix containing 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 and are only available for continuum and shell elements with nonlinear deformation (**NLGEOM**,ON).

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`. When the mixed u-P formulation option (*TB0PT* = MXUP on the **TB**,USER command) is used for hyperelastic material, the stress is updated for deviatoric part of stress only. The calculated P is passed into usermat as stress(ncomp+1).
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.
epseq	Equivalent plastic strain.
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 the solution control. Set cutFactor to control the time stepping size.
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)	Transverse shear stiffness: Tsstif(1) - GXZ

	Tsstif(2) - GYZ
dsdePl(ncomp,ncomp)	Double-precision array containing the material Jacobian matrix $\partial \Delta\sigma_{ij} / \partial \Delta\varepsilon_{ij}$. 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 a change of the j-th strain component. By default, the program assumes that the element stiffness matrix is symmetric; therefore, you must provide a *symmetric* material Jacobian matrix *even if it is unsymmetric*. If your material requires an unsymmetric material Jacobian matrix, issue the **NROPT**,UNSYM command to define the unsymmetric stiffness matrix. When the mixed u-P formulation option (*TBOPT* = MXUP on the **TB**,USER command) is used for hyperelastic material, only the deviatoric material consistent tangent matrix is needed.
pVolDer(:)	Derivatives of volumetric potential with regard to the determinant of deformation gradient: pVolDer(1) = $\partial U / \partial J$ pVolDer(2) = $\partial^2 U / \partial J^2$ pVolDer(3) = $\partial^3 U / \partial J^3$ This argument is needed only when a hyperelastic material is defined and mixed u-P formulation is used. For nearly incompressible hyperelastic material, a mixed u-J formulation is used. The derivatives of volume potential is to Jc, where Jc is the calculated J and is passed from strain array as strain(ncomp+1). For purely incompressible hyperelastic material, set all three derivatives pVolDer(1:3) to zero.
cutFactor	Time-step size control factor.

2.4.1.5. 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*,*TBOPT*

where

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

MAT = User material ID number

NTEMPS = Number of temperature points.

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

TBOPT = NONLINEAR (default), LINEAR, or MXUP

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 2.1: 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, the program initializes state variables to zero at the beginning of an analysis. Use the **TBDATA** command to initialize your own values for state variables.

Example 2.2: 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.
```

2.4.1.6. 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. The program always uses the full Newton-Raphson scheme for the global solution to achieve a better convergence rate. The material Jacobian matrix (p. 188) (dsdePl(i,j) (p. 193)) must be consistent

with the material constitutive integration scheme for a better convergence rate of the overall Newton-Raphson scheme.

2.4.1.7. UserMat Restrictions

The following restrictions apply to the `UserMat` subroutine:

- The subroutine supports current-technology elements only and does not apply 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 **/GRAPHICS**,FULL command to do so.

- The `UserMat` interface for elements with mixed u-P formulation is supported only for current-technology continuum elements (such as PLANE182, PLANE183, SOLID185, SOLID186, SOLID187, SOLSH190, CPT212, CPT213, CPT215, CPT216 and CPT217), excluding plane stress state. KEYOPT(6) = 1 is required. For elements with a plane stress assumption, a mixed u-P formulation is not needed.

- The deformation gradient defGrad(3,3) and defGrad_t(3,3) are available only for continuum and shell elements with nonlinear deformation (**NLGEOM**,ON).

2.4.1.8. Accessing Material and Element Data for UserMat

Following is the interface for accessing the 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)

c        definition
c           inquire        - query argument (string, see variables)
c           elemId         - element number
c           kIntg          - gauss intg. number
c           nvect          - number of vector to be inquired
c           vect           - vector to be inquired
```

`get_ElmData` Input Arguments	
inquire	String variable containing a query argument (p. 196).
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.

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

TREF	Reference temperature.

2.4.1.9. Utility Functions for UserMat

The following functions are available for use with UserMat. ANSYS, Inc. 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 ***a*** by constant *c* and outputs the result as itself (that is, ***a*** = ***a*** * *c*). The value *n* represents the array dimension.
maxb(***a*** , ***b*** , ***c*** , *na*, *nb*, *nc*, *n1*, *n2*, *n3*)	Multiplies two double-precision matrices and outputs the result as ***c*** (that is, ***c*** = ***a*** * ***b***). 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 *columns* in matrix ***a*** *and* the number of *rows* 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.)

2.4.2. Subroutine UserMatTh (Creating Your Own Thermal Material Model)

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

The following UserMatTh topics are available:

2.4.2.1. UserMatTh Element Support
2.4.2.2. The UserMatTh API
2.4.2.3. UserMatTh Variables
2.4.2.4. Table (TB) Commands for UserMatTh
2.4.2.5. UserMatTh Restrictions
2.4.2.6. Utility Functions for UserMatTh

2.4.2.1. UserMatTh Element Support

The UserMatTh subroutine is available for current-technology elements SOLID278 and SOLID279. It is also available for FLUID116 with KEYOPT(1) = 1.

UserMatTh is called at every material integration point of the elements during the solution phase.

Input values and the number of state variables (if used) for the subroutine are specified via the **TB** command. For more information, see Table (**TB**) Commands for UserMatTh (p. 200).

2.4.2.2. The UserMatTh API

Following is the interface for the UserMatTh subroutine:

```
*deck,usermatth      USERDISTRIB  parallel
      subroutine usermatth(matId, elemId, kDomIntPt, kLayer, kSectPt,
     &                     ldstep, isubst, keycut, ncomp, nStatev,nProp,
     &                     Time, dTime, Temp, dTemp, tgrad,
     &                     ustatev, prop, coords,
     &                     dudt,dudg,flux,dfdt,dfdg,
     &                     cutFactor,hgen, dens, var1, var2, var3,
     &                     var4, var5, var6)
c*************************************************************************
c     *** primary function ***
c
c           user defined thermal material constitutive model
c
c     Attention:
c           User must define material constitutive behavior properly
c
c
c       The following demonstrates a USERMATTH subroutine for
c       a regular conductive heat transfer.
c       See "ANSYS user material subroutine USERMATTH" for detailed
c       description of how to write a USERMATTH routine.
c
c
c*************************************************************************
c Copyright ANSYS.  All Rights Reserved.
c
c     input arguments
c     ===============
c      matId     (int,sc,i)              material #
c      elemId    (int,sc,i)              element #
c      kDomIntPt (int,sc,i)              "k"th domain integration point
c      kLayer    (int,sc,i)              "k"th layer
c      kSectPt   (int,sc,i)              "k"th Section point
c      ldstep    (int,sc,i)              load step number
c      isubst    (int,sc,i)              substep number
c      ncomp     (int,sc,i)              # of components
c      nStatev   (int,sc,i)              Number of state variables
c      nProp     (int,sc,i)              Number of material constants
c
c      Temp      (dp,sc,in)              current temperature
c      dTemp     (dp,sc,in)              temperature increment
c      Time      (dp,sc,in)              time at beginning of increment (t)
c      dTime     (dp,sc,in)              current time increment (dt)
c
c      prop      (dp,ar(nprop),i)        Material constants defined by TB,USER
c      coords    (dp,ar(3),i)            current coordinates
c
c      tgrad     (dp,ar(ncomp),i)        Current values of the spatial gradients of temperature
c
c
c     input output arguments
c     ======================
c      ustatev   (dp,ar(nstatev),io)     user state variables
c      hgen      (dp,sr,io)              heat generation rate per unit mass
c      dens      (dp,sr,io)              density passed in as defined by mp command
c      var?      (dp,sc,io)              not used, they are reserved arguments
c                                        for further development
c
c     output arguments
c     ================
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
c      keycut    (int,sc,o)              loading bisect/cut control
c                                        0 - no bisect/cut
c                                        1 - bisect/cut
c                                        (factor will be determined by ANSYS solution control)
c
c      cutFactor (dp,sc,o)               time step size cut-back factor
c                                        define it if a smaller step size is wished
c                                        recommended value is 0~1
c
c      dudt      (dp,sr,o)               unit mass internal energy variation with respect to
c                                        temperature
c      dudg      (dp,ar(ncomp),o)        unit mass internal thermal energy variation  with
c                                        respect to the spatial gradients of temperature
c      flux      (dp,ar(ncomp),o)        heat flux vector
c      dfdt      (dp,ar(ncomp),o)        heat flux vector variation with respect to temperature
c      dfdg      (dp,ar(ncomp,ncomp),o)  heat flux vector variation with respect to the spatial
c                                        gradients of temperature
c
c***********************************************************************
c
c      ncomp   3   for 3D
c      ncomp   2   for 2D
c      ncomp   1   for 1D
c
c
c***********************************************************************
```

2.4.2.3. UserMatTh Variables

The `UserMatTh` subroutine uses the following Input (p. 199), Input/Output (p. 200), and Output (p. 200) variables. Do not change them in the subroutine code.

UserMatTh Input Arguments	
matId	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.
ncomp	Integer variable containing spatial dimension number: 1 for 1-D, 2 for 2-D, or 3 for 3-D.
nStatev	Number of state variables, specified via the *NPTS* value in the **TB**,STATE command.
nProp	Number of material constants, specified via the *NPTS* value in the **TB**,USER command.
Temp	Double-precision variable containing the current temperature.
dTemp	Double-precision variable containing the current temperature increment.
Time	Double-precision variable containing the total time at the beginning of the time increment.
dTime	Double-precision variable containing the current time increment.
prop	Double-precision array containing 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 containing the current coordinates of the material integration points. Array size is 3.

tgrad	Double-precision array of size ncomp containing temperature gradient at the beginning of the time increment.

UserMatTh Input/Output Arguments	
hgen	Double-precision variable with heat-generation information. It is passed in as the value that is predefined with the model. The value can be updated with user-defined heat generation.
ustatev	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.
dens	Double precision variable with density information. It is passed in as the value that is predefined with the model. The value can be updated with user-defined density.

UserMatTh 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 UserMatTh experiences convergence difficulty when solving constitutive equation integration. The bisect/cut factor is determined by the solution control.
cutFactor	Double-precision variable containing the user-defined time-step size-control factor.
dudt	Double-precision variable containing the unit mass internal energy variation with respect to temperature.
dudg	Double-precision array of size (ncomp,ncomp) containing the unit mass internal thermal energy variation with respect to the spatial gradients of temperature.
flux	Double-precision array of size ncomp containing the heat flux vector.
dfdt	Double-precision array of size ncomp containing the heat flux vector variation with respect to temperature.
dfdg	Double-precision array of size (ncomp,ncomp) containing the heat flux vector variation with respect to the spatial gradients of temperature.

2.4.2.4. Table (TB) Commands for UserMatTh

When creating your own material model, first define the material by specifying input values for the UserMatTh 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:

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

TB,USER,*MAT*,*NTEMPS*,*NPTS*,*TBOPT*

where

MAT = User material ID number

NTEMPS = Number of temperature points.

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

TBOPT = NONLINEAR (default) or LINEAR

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

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

Example 2.3: Defining the Material for UserMatTh

```
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 temperature
tbdata,1,19e5,0.3,1e3,100,     ! four material constants for
                               ! one temperature
tbtemp,2.0                     ! Second temperature
tbdata,1,21e5,0.3,2e3,100,     ! Four material constants for
                               ! two temperatures
```

TB,STATE Command

If you intend to use state variables with the `UserMatTh` 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, the program initializes state variables to zero at the beginning of an analysis. Use the **TBDATA** command to initialize your own values for state variables.

2.4.2.5. UserMatTh Restrictions

The following restrictions apply to the `UserMatTh` subroutine:

- The subroutine supports current-technology elements only and does not apply to legacy elements. For more information, see UserMatTh Element Support (p. 197).
- The state variables (defined via the **TB**,STATE command) are supported only by full graphics in the POST1 postprocessor.

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

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

- Element FLUID116 is supported only when KEYOPT(1) = 1. For this element, you can use `UserMatTh` to specify material properties KXX, C, and DENS only.

2.4.2.6. Utility Functions for UserMatTh

The following functions are available for use with `UserMatTh`. ANSYS, Inc. provides them for your convenience.

Utility Functions for `UserMatTh`	
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 ***a*** by constant *c* and outputs the result as itself (that is, ***a*** = ***a*** * *c*). The value *n* represents the array dimension.
maxb(***a*** , ***b*** , ***c*** , *na*, *nb*, *nc*, *n1*, *n2*, *n3*)	Multiplies two double-precision matrices and outputs the result as ***c*** (that is, ***c*** = ***a*** * ***b***). 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 *columns* in matrix ***a*** *and* the number of *rows* 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.)

2.4.3. 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 *TB0PT* = 100.

```
*deck,UserHyper                       USERSDISTRIB
c Copyright ANSYS.  All Rights Reserved.
      subroutine UserHyper(
     &                     prophy, incomp, nprophy, invar,
     &                     potential, pInvDer)
c*************************************************************************
c
c     *** Example of user hyperelastic routine
c
c           This example uses Arruda hyperelasticity model
c           which is the same ANSYS TB,BOYCE
c
c     input arguments
c     ===============
c      prophy       (dp,ar(*),i)     material property array
c      nprophy     (int,sc,i)        # of material constants
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
c      invar       dp,ar(3)          invariants
c
c     output arguments
c     ================
c      incomp      (log,sc,i)         fully incompressible or compressible
c      potential    dp,sc             value of potential
c      pInvDer       dp,ar(10)        der of potential wrt i1,i2,j
c                                      1 - der of potential wrt i1
c                                      2 - der of potential wrt i2
c                                      3 - der of potential wrt i1i1
c                                      4 - der of potential wrt i1i2
c                                      5 - der of potential wrt i2i2
c                                      6 - der of potential wrt i1j
c                                      7 - der of potential wrt i2j
c                                      8 - der of potential wrt j
c                                      9 - der of potential wrt jj
c
c**********************************************************************
c
c --- parameters
c
```

2.4.4. Subroutine UserCreep (Defining Creep Material Behavior)

Use the subroutine `UserCreep` to define creep material behavior. 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. The program always uses implicit time integration for this creep option. You can use plasticity options (BISO, BKIN, NLISO, PLASTIC) to define the plastic behavior of materials. Creep and plastic strain are calculated simultaneously when both creep and plasticity are defined for a material.

Using 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 the program. You can use and update internal state variables in the subroutine. The number of state variables must be defined (**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)
c**********************************************************************
c     *** primary function ***
c           Define creep laws when creep table options are
c           TB,CREEP with TBOPT=100.
c           Demonstrate how to implement usercreep subroutine
c
c            Creep equation is
c               dotcreq := k0 * seqv ^ n * creqv ^ m * exp (-b/T)
c
c               seqv  is equivalent effective stress (Von-Mises stress)
c               creqv is equivalent effective creep strain
c               T     is the temperature
c               k0, m, n, b are materials constants,
c
c           This model corresponds to  primary creep function  TBOPT = 1
c
```

```
c                                                       gal 10.01.1998
c
c*************************************************************************
c Copyright ANSYS.  All Rights Reserved.
c
c     input arguments
c     ===============
c      impflg   (in ,sc   ,i)            Explicit/implicit integration
c                                        flag (currently not used)
c      ldstep   (in ,sc   ,i)            Current load step
c      isubst   (in ,sc   ,i)            Current sub step
c      matId    (in ,sc   ,i)            number of material index
c      elemId   (in ,sc   ,i)            Element number
c      kDInPt   (in ,sc   ,i)            Material integration point
c      kLayer   (in ,sc   ,i)            Layer number
c      kSecPt   (in ,sc   ,i)            Section point
c      nstatv   (in ,sc   ,i)            Number of state variables
c      nprop    (in ,sc   ,i)            size of mat properties array
c
c      prop     (dp ,ar(*),i)            mat properties array
c                                        This array is passed all the creep
c                                        constants defined by command
c                                        TBDATA associated with TB,CREEP
c                                        (do not use prop(13), as it is used
c                                        elsewhere)
c                                        at temperature temp.
c      time                              Current time
c      dtime                             Current time increment
c      temp                              Current temperature
c      dtemp                             Current temperature increment
c      toffst   (dp, sc,   i)            temperature offset from absolute zero
c      seqv     (dp ,sc  , i)            equivalent effective stress
c      creqv    (dp ,sc  , i)            equivalent effective creep strain
c      pres     (dp ,sc  , i)            hydrostatic pressure stress, -(Sxx+Syy+Szz)/3
c                                        note that: constitutive consistency is not accounted for
c                                        if creep strains are function of pressure
c
c     input output arguments             input desc     / output desc
c     ======================             ==========       ===========
c      Ustatev  (dp,ar(*), i/o)          user defined iinternal state variables at
c                                        time 't' / 't+dt'.
c                                        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
c                                        time increment, if any of these internal
c                                        state variables are associated with the
c                                        creep behavior.
c
c     output arguments
c     ================
c      delcr    (dp ,sc  , o)            incremental creep strain
c      dcrda    (dp,ar(*), o)            output array
c                                        dcrda(1) - derivitive of incremental creep
c                                                   strain to effective stress
c                                        dcrda(2) - derivitive of incremental creep
c                                                   strain to creep strain
c
c     local variables
c     ===============
c      c1,c2,c3,c4 (dp, sc, l)           temporary variables as creep constants
c      con1        (dp ,sc, l)           temporary variable
c      t           (dp ,sc, l)           temporary variable
c
c*************************************************************************
c
c --- parameters
c
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries and affiliates.

2.4.5. 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:

2.4.5.1. Overview of the user_tbelastic Subroutine
2.4.5.2. Data Types Supported by user_tbelastic
2.4.5.3. Table (TB) Command for user_tbelastic
2.4.5.4. User Interface for user_tbelastic
2.4.5.5. The user_tbelastic API
2.4.5.6. Usage Example for user_tbelastic

2.4.5.1. 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.

You can use the subroutine with most current-technology elements and with most nonlinear material models.

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

2.4.5.2. 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.

2.4.5.3. Table (TB) Command for user_tbelastic

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

TB,ELASTIC,*mat*,,*npts*,USER

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.

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

2.4.5.4. 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* Output (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`	I / O	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`	O	The material elastic constants to be defined

2.4.5.5. The user_tbelastic API

Following is the interface to the `user_tbelastic` subroutine:

```
*deck,user_tbelastic                    USERSDISTRIB
      SUBROUTINE user_tbelastic(elemId, matId, coords, temp,
     &                          nprop,  prop)
c***********************************************************************
c     *** primary function ***
c           user interface for elastic material constants
c
c***********************************************************************
c Copyright ANSYS.  All Rights Reserved.
c
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

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

c     local variables
c     ===============
c
c**************************************************************************
c
c --- parameters
c
```

2.4.5.6. 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 **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 Input

```
/batch
/com
/com example for user elastic material property interface
/com
/com element 1 solid185 defined via standard MP command
/com element 2 solid185 defined using ansys elastic material interface
/com element 3 shell181 defined using ansys elastic material interface
/com
/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
```

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
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
```

2.4.6. Subroutine userfc (Defining Your Own Failure Criteria)

```
*deck,userfc                       USERSDISTRIB
      subroutine userfc (elem,matlay,iott,tem,elim,slim,
     x                   eps, sig, nfcOut, fc)
c     primary function: user subroutine for defining your own failure criterion
c *** secondary functions: none
c
c *** user programmable functions may not be used in parallel processing ***
c     this is currently only available with
c
c         *** Copyright ANSYS.  All Rights Reserved.
c         *** ansys, inc.
c
c *** Notice - This file contains ANSYS Confidential information ***
c
c  input arguments:
c     variable (typ,siz,intent)    description
c     elem     (int,sc,in)       - element number
c     elim     (dp,ar(9),in)     - failure strains at the current temperature
c                                    (see FC command input with Lab1 = EPEL)
c     slim     (dp,ar(12),in)    - failure stresses and coupling coefficients
c                                    at the current temperature
c                                    (see FC command input with Lab1 = S)
c     eps      (dp,ar(6),in)     - vector of strains
c     sig      (dp,ar(6),in)     - vector of stresses
c     tem      (dp,sc,in)        - temperature at this point in the model
c     matlay   (int,sc,in)       - material number
c     iott     (int,sc,in)       - unit number for writing
c
c  output arguments:
c     variable (typ,siz,intent)    description
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries and affiliates.

```
c     nfcOut   (int,sc, out)      - number of user fc computed
c     fc       (dp,ar(9),out)    - user failure criterion
c
```

2.4.7. Subroutine userCZM (Defining Your Own Cohesive Zone Material)

Define your own interfacial cohesive material law via the **TB**,CZM,,,,USER command.

Issue the **TBDATA** command to define the material constants. Data may be temperature-dependent and is interpolated at the current temperature of the material integration point and passed to the subroutine.

For more information, see User-Defined Cohesive Material (UserCZM) and Using State Variables with the UserCZM Subroutine in the *Material Reference*.

Following is the user cohesive material interface:

```
*deck,userCZM                       USERSDISTRIB
      subroutine userCZM (matId, elemId, kMatIntPt, ldstep,isubst,
     &                    keycut, ncomp,nProp, nstatev,
     &                    Time, dTime, Temp, dTemp,
     &                    coords, prop, Strain, dStrain,
     &                    stress, dsdePl, sedEl, sedPl, statev,
     &                    var1, var2, var3, var4, var5)
c
c**********************************************************************
c
c     *** primary function ***
c
c           user cohesive zone model example
c
c           Commands
c             TB,CZM,mat,NTEMP,NPTS,user
c                TBTEMP if mat. constants are temperature dependent
c                TBDATA define material constants
c
c**********************************************************************
c
c     input arguments
c     ===============
c      matId     (int,sc,in)             material #
c      elemId    (int,sc,in)             element #
c      kMatIntPt (int,sc,in)             material integration point #
c      ldstep    (int,sc,in)             load step number
c      isubst    (int,sc,in)             substep number
c      ncomp     (int,sc,in)             number of stress, strain components
c      nProp     (int,sc,in)             Number of material ocnstants
c      nstatev   (int,sc,in)             Number of state variables
c
c      Temp      (dp ,sc,in)             temperature at beginning of time increment
c      dTemp     (dp ,sc,in)             temperature increment
c      Time      (dp ,sc,in)             time at beginning of increment (t)
c      dTime     (dp ,sc,in)             time increment (dt)
c
c      prop     (dp,ar(nprop),i)         Material constants defined by TB command
c      Strain   (dp,ar(ncomp),i)         Interface separation at beginning of time increment
c      dStrain  (dp,ar(ncomp),i)         Interface separation increment
c      coords   (dp,ar(3),i)             current coordinates
c
c     output arguments
c     ====================
c      stress   (dp,ar(nTesn),io)        Traction stress
c      sedEl    (dp,sc,io)               elastic work
c      sedPl    (dp,sc,io)               plastic work
c      keycut   (int,sc,io)              loading bisect/cut control
c                                        0 - no bisect/cut
```

```
c                                       1 - bisect/cut
c                                       (factor will be determined by ANSYS solution control)
c      dsdePl   (dp,ar(ncomp,ncomp),io)   consistent tangent jacobian matrix
c
c     input output arguments
c     ======================
c      statev   (dp,ar(nstatev,io)        user defined solution state variables
c
c     misc.
c     ======================
c      var1, var2, var3, var4, var5       currently not used
c
c     local variables
c     ======================
c
c      debugflag (in,sc, l)                debugflag to print debug information
c
c
c**********************************************************************
c
```

2.4.8. Subroutine userswstrain (Defining Your Own Swelling Laws)

You can define your own swelling strain option via the **TB**,SWELL,,,,USER command.

Use the **TBDATA** command to define the material constants. Data may be temperature-dependent and is interpolated at the current temperature of the material integration point and passed to the subroutine.

For more information, see Swelling in the *Material Reference*.

```
*deck,userswstrain  USERDISTRIB
      subroutine userswstrain  (elemId,  kMatPoint
     &,                         matId,   nprop,   propv
     &,                         time,    dtime
     &,                         efvs,    defv
     &,                         sweqt,   dsweq
     &,                         swvi,    swvo)
c
c
c *** primary function:    compute user defined swelling strain
c
c *** Notice - This file contains ANSYS Confidential information ***
c Copyright ANSYS.  All Rights Reserved.
c
c  input arguments:
c     matId    (int,sc,in)       - material #
c     elemId   (int,sc,in)       - element  #
c     kMatPoint(int,sc,in)       - element integration point #
c     nprop    (int,sc,in)       - number of material constants
c     time     (int,sc,in)       - current time
c     dtime    (int,sc,in)       - current time increment
c     propv    (dp,ar(nprop),in) - array of material constants
c                                  (the data input via TBDATA command)
c     efvs     (dp,ar(*)    ,in) - field variables array
c                                  efvs(1) - current temperature
c                                  efvs(2) - current fluence
c     defv     (dp,ar(*)    ,in) - incremental field variables array
c                                  defv(1) - temperature increment
c                                  defv(2) - fluence increment
c     sweqt    (dp,sc       ,in) - equivalent swelling strain at time t
c
c  output arguments:
c     dsweq    (dp,sc       ,in) - incremental equivalent swelling strain
c  not used arguments:
c     swvi     (dp,sc       ,in) - not currently used
c     swvo     (dp,sc       ,in) - not currently used
c
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
c*****************************************************************
c
```

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

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

2.4.10. 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
c                      used in creep and plasticity calculations
c secondary functions: none

c    formulation of overall value is by:

c         ____________________________________________________________________
c        /1            2            2            2  1    2      2      2
c    \  / -*((ep  - ep ) + (ep  - ep ) + (ep  - ep ) + -*(ep   + ep   + ep ))
c     \/  2     1     2       2     3       3     1    2   4      5      6
c    ---------------------------------------------------------------------
c                                (1 + posn)

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

c     typ=int,dp,log,chr,dcp   siz=sc,ar(n)   intent=in,out,inout

c  input arguments:
c     variable (typ,siz,intent)    description
c     kcomp    (int,sc,in)       - number of components of strain
c     ep       (dp,ar(6),in)     - the strain components
c     posn     (dp,sc,in)        - poisson's ratio

c  output arguments:
c     egen     (dp,func,out)     - the combined strain value
c
```

2.4.11. Subroutine userfld (Update User-Defined Field Variables)

You can create your own field variables using via the **INISTATE** command and supported materials (**TB**). Node-based initial state helps to initialize the user-defined field variables that are then used by

the **TB** database to evaluate the material properties at an integration point. `userfld` allows you to examine the current state at the integration point and to modify the field variables as needed.

For more information, see Understanding Field Variables in the *Material Reference*.

```
*deck,userfld                       USERSDISTRIB
      subroutine userfld
     &          ( matId, elemId,
     &            ldstep, isubst, time, dtime,
     &            kDomIntPt, kLayer, kSectPt,
     &            nDirect, nShear, nComp, nStatev,
     &            coords,
     &            Temp,dTemp )
c**********************************************************************
c     *** primary function ***
c
c           Edit Field Variables During Solution
c
c      Attention:
c
c**********************************************************************
c Copyright ANSYS.  All Rights Reserved.
c
c     input arguments
c     ===============
c      matId     (int,sc,i)              material #
c      elemId    (int,sc,i)              element #
c      ldstep    (int,sc,i)              load step num
c      isubst    (int,sc,i)              substep num
c      time      (int,sc,d)              time
c      dtime     (int,sc,d)              time inc
c      kDomIntPt (int,sc,i)              "k"th domain integration point
c      kLayer    (int,sc,i)              "k"th layer
c      kSectPt   (int,sc,i)              "k"th Section point
c      nDirect   (int,sc,in)             # of direct components
c      nShear    (int,sc,in)             # of shear components
c      ncomp     (int,sc,in)             nDirect + nShear
c      nstatev   (int,sc,i)              Number of state variables
c      Temp      (dp,sc,in)              temperature at beginning of
c                                        time increment
c      dTemp     (dp,sc,in)              temperature increment
c      coords   (dp,ar(3),i)             current coordinates
c
c     input output arguments
c     ======================
c
c
c     output arguments
c     ================
c
c**********************************************************************
c                        List of Supported Field Variable Types
c**********************************************************************
c
c     FLD_USER_1_TYPE
c     FLD_USER_2_TYPE
c     FLD_USER_3_TYPE
c     FLD_USER_4_TYPE
c     FLD_USER_5_TYPE
c     FLD_USER_6_TYPE
c     FLD_USER_7_TYPE
c     FLD_USER_8_TYPE
c     FLD_USER_9_TYPE
c
c**********************************************************************
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

2.4.12. Subroutine userthstrain (Defining Your Own Thermal Strain)

You can define the thermal strain via the **TB**,CTE,,,,USER command and the `userthstrain` subroutine.

Issue the **TBDATA** command to define the material constants. Data can be field-dependent, and is interpolated at the current field values of the material integration point and passed to the subroutine.

For more information, see Thermal Expansion in the *Mechanical APDL Material Reference*.

```
*deck,userthstrain  USERDISTRIB
      subroutine userthstrain (nprop, propv,
     &                         ncomp, epth)
c
c *** primary function:    compute the thermal strain
c
c *** Notice - This file contains ANSYS Confidential information ***
c
c  input arguments:
c     nprop    (int,sc,in)       - number of material constants
c     propv    (dp,ar(ncomp),in) - array of material constants
c                                  e.g. coefficients of thermal
c                                       expansion in x,y,z order
c     ncomp    (int,sc,in)       - number of strain components
c                                  6  for 3-d elements   (x,y,z,xy,yz,xz)
c                                  4  for plane elements (x,y,z,xy)
c                                  3  for beam elements  (x,xy,xz)
c                                  1  for line elements  (x)
c
c  output arguments:
c     epth     (dp,ar(ncomp),out) - thermal strains
c
```

2.5. Subroutines for Customizing Contact Interfacial Behavior

This section describes user subroutines that you can use to customize contact interfacial behavior for contact elements (CONTA171 to CONTA178). These subroutines allow you to:

- Perform a user-defined operation on real constants (subroutine `usercnprop` (p. 213))
- Write your own friction laws (subroutine `userfric` (p. 217))
- Write your own contact interactions (subroutine `userinter` (p. 220))
- Write your own wear law (subroutine `userwear` (p. 226))

2.5.1. Subroutine usercnprop (Programming Your Own Contact Properties)

This subroutine applies to the CONTA17*x* contact elements.

```
*deck,usercnprop                      USERSDISTRIB
      subroutine usercnprop (ndim,coor,nkeyopt,keyopt,nrl,rlconst,
     x  nintIn,intIn,nrealIn,realIn,kupdhis,localr,nuval,nintp,usvr,
     x  ncomp,stress,strain0,strain,kstat,mu,kcnprop,cnprop,keyerr)
c
c *** primary function:   Allow users to define their own contact properties
c                         in real constant table
c                         This logic is accessed with real constant defined
c                         by table name: %_CNPROP%
c                         (e.g. rmod,cid,kcnprop,%_CNPROP%)
c
c *** Notice - This file contains ANSYS Confidential information ***
```

```
c
c
c         Copyright ANSYS.  All Rights Reserved.
c         *** ansys, inc.
c
c  input arguments:
c     variable (type,sze,intent)    description
c
c     elem     (int,sc,in)         - element number
c     intpt    (int,sc,in)         - element integration point number
c     ndim     (int,sc,in)         - number of dimensions of the problem
c                                    = 2 2D
c                                    = 3 3D
c     nintp    (int,sc,in)         - the total number of integration points of
c                                    an element to be used with this routine
c     nuval    (int)               - number of additional state variables per
c                                    integration point
c         note:  nuval x nintp = nstv(on nsvr command); cannot exceed 840!
c
c     intIn    (int,ar(*),in)      - integer variables passed in
c     intIn    (int,ar(*),in)      - integer variables passed in
c                                    intIn(1) = element number
c                                    intIn(2) = element integration point number
c                                    intIn(3) = material reference number
c                                    intIn(4) = element type ID number (absolute value)
c                                               > 0 for CONTA171-CONTA177
c                                               < 0 for CONTA178
c                                    intIn(5) = real constant ID number
c                                    intIn(6) = associated contact nodal number
c                                    intIn(7) = contact indicator
c                                               0: intersection is found
c                                               otherwise: no intersection
c                                    intIn(8) = target element number
c                                    intIn(9) = flag for forcing sliding
c                                               frictional case
c                                               0 : not forcing
c                                               1 : forcing (Slip direction is
c                                                   defined through CMROT command)
c                                    intIn(10) = 1 first pass through
c                                                  (1st iteration)
c                                                  (useful for initializing state
c                                                  variables to a non-zero value)
c                                              = 2 first pass through key of
c                                                  a restart
c                                              = 3 first pass through key of
c                                                  a rezoning
c                                    intIn(11) = current load step number
c                                    intIn(12) = current substep number
c                                    intIn(13) = current equilibrium iteration
c                                                number
c                                    intIn(14) = flag for using unsymmetric
c                                                matrices (nropt,unsym)
c                                                0 : symmetric
c                                                1 : unsymmetric
c                                    intIn(15) = Linear perturbation flag
c                                                0 : a general load step
c                                                1 : a linear perturbation step
c                                    intIn(16) = key to indicate output pass
c                                                0 : not a output pass
c                                                1 : output pass
c                                    intIn(17) = key to indicate if history-
c                                                dependent variables
c                                                (user defined) need to be
c                                                updated after the substep has
c                                                converged
c                                                1 : update (converged)
c                                                0 : do not update (not converged)
c                                    intIn(18) = key to indicate transient effects
c                                                1 : transient is active
c                                                0 : transient is not active
c                                    intIn(19) = large deformation key [nlgeom cmd]
c                                                1 : on
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries and affiliates.

```
c                                              0 : off
c                                  intIn(20) = analysis type (derived from
c                                              antype cmd)
c                                              0 : a static analysis
c                                              1 : a buckling analysis
c                                              2 : a modal analysis
c                                              3 : a harmonic analysis
c                                              4 : a transient analysis
c                                              7 : a substructure analysis
c                                              8 : a spectrum analysis
c                                  intIn(21) = key for displacement & force
c                                              convergence
c                                              1 : converged
c                                              0 : not converged
c     realIn   (dp,ar(*),in)       - real variables passed in
c                                    realIn(1) = contact element length
c                                    realIn(2) = contact element depth
c                                    realIn(3) = area associated with the contact
c                                                detection point
c                                    realIn(4) = pinball radius
c                                    realIn(5) = un-scaled normal penalty stiffness
c                                    realIn(6) = time (or frequency for a harmonic
c                                                analysis) at the beginning of this
c                                                load step
c                                    realIn(7) = time (or frequency for a harmonic
c                                                analysis) at the end of this load step
c                                    realIn(8) = current time value (or frequency value
c                                                for a harmonic analysis)
c                                    realIn(9) = time increment (or frequency increment
c                                                for a harmonic analysis) over this
c                                                substep
c                                    realIn(10)= temperature offset from absolute
c                                                 zero
c                                    realIn(11)= geometric penetration/gap
c                                                (current substep)
c                                                > 0 : gap
c                                                < 0 : penetration
c                                    realIn(12)= time increment scaling factor to
c                                                be used for structural transient
c                                                dynamics
c     nkeyopt  (int,sc,in)         - number of key options
c     keyopt   (int,ar(nkeyopt),in)- array containing key options
c                                    keyopt(1) : Select degree of freedom
c                                    keyopt(2) : Contact algorithm
c                                    ... so on (see ANSYS documentation)
c     nrl      (int,sc,in)         - number of real constants
c     rlconst  (dp,ar(nrl),in)     - array containing real constants
c                                    Elements CONTA171 to CONTA177
c                                    rlconst(1) : R1
c                                    rlconst(2) : R2
c                                    rlconst(3) : FKN
c                                    rlconst(4) : FTOLN
c                                    ... so on (see ANSYS documentation)
c                                    Element CONTA178
c                                    rlconst(1) : FKN
c                                    rlconst(2) : GAP
c                                    ... so on (see ANSYS documentation)
c
c     kcnprop  (int,sc,in)         - the position of constant in the real set
c
c                                    (see ANSYS contact element manual)
c     ncomp    (int,sc,in)         - number of stress/force component
c                                    = 9 for CONTA171-CONTA177
c                                    = 7 for CONTA178

c     stress   (dp,ar(ncomp),in)   - stress components at the beginning of
c                                    the current iteaation/substep.
c                                    stress(1) = frictional stress in direction 1
c                                    stress(2) = frictional stress in direction 2
c                                                (3D only)
c                                    stress(3) = contact normal pressure
c                                    > 0 : compression
```

```
c                                    < 0 : tension
c                                    the above contact traction must be defined in
c                                    a local coordinate system (see localr)
c                                    stress(4) = heat flux (per area)
c                                                flowing into contact
c                                    stress(5) = heat flux (per area)
c                                                flowing into target
c                                    < 0 heat flowing into a surface
c                                    > 0 heat flowing out of a surface
c                                    stress(6) = electrical current density
c                                                (or pore fluid flux density)
c                                                (per area) flowing into contact
c                                    stress(7) = electrical current density
c                                                (or pore fluid flux density)
c                                                (per area) flowing into target
c                                    > 0 current flowing out of a surface
c                                    < 0 current flowing into a surface
c                                    stress(8) = diffusion flux density
c                                                (per area) flowing into contact
c                                    stress(9) = diffusion flux density
c                                                (per area) flowing into target
c                                    > 0 flux flowing out of a surface
c                                    < 0 flux flowing into a surface
c     strain0  (dp,ar(ncomp),in) - strain components in the end of the previous
c                                    substep
c                                    (see strain for each component definition)
c     strain  (dp,ar(ncomp),in)  - current strain components
c                                    strain(1) = slip increment in direction 1
c                                    strain(2) = slip increment in direction 2
c                                              (3D only)
c                                    strain(3) = contact normal gap/penetration
c                                    < 0 : gap
c                                    > 0 : penetration
c                                    strain(4) = temperature at the contact point
c                                                (from TEMP DOF or temperature load)
c                                    strain(5) = temperature at the target point
c                                                (only from TEMP DOF)
c                                    strain(6) = voltage (or pore pressure)
c                                                at the contact point
c                                    strain(7) = voltage (or pore pressure)
c                                                at the target point
c                                    strain(8) = concentrationat the contact point
c                                    strain(9) = concentrationat the target point
c     kstat    (int,sc,in)         - contact status at the end of the previous
c                                    substep
c                                    3 : stick
c                                    2 : sliding
c                                    1 : open contact (near)
c                                    0 : open contact (far)
c     mu       (dp,sc,in)          - The frictional coef at the end of previous
c                                    substep
c     coor     (dp,ar(6),in)       - Coordinates of the contact detection point
c                                    coor(1) current x
c                                    coor(2) current y
c                                    coor(3) current z
c                                    coor(4) initial x
c                                    coor(5) initial y
c                                    coor(6) initial z
c     localr   (dp,ar(3,3),in)     - the direction cosines of the local surface
c                                    coordinate system at contact detection
c                                    localr(1,1),localr(1,2),localr(1,3) in slip
c                                                                        direction 1
c                                    localr(2,1),localr(2,2),localr(2,3) in slip
c                                                                        direction 2
c                                    localr(3,1),localr(3,2),localr(3,3) in normal
c                                                                        direction
c
c     usvr     (dp,ar(nuval,nintp),inout)- additional state variables from
c                                    previous equilibrium iteration (saved
c                                    if the nsvr command is used)
c     kupdhis  (int,sc,in)         - key to indicate if history-dependent
c                                    variables (user defined) need to be
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries and affiliates.

```
c                                  updated after the substep has converged
c                                  1 : update (converged)
c                                  0 : do not update (not converged)
c
c  output arguments:
c     variable (type,sze,intent)    description

c     cnprop   (dp,ar(5),out)      - user defined real constant value and
c                                    derivatives w.r.t. kcnprop position
c                                    cnprop(1) = user defined real constant value
c                                    (e.g. kcnprop = 3 for normal contact
c                                      stiffness FKN.
c                                      positive as scaling factor;
c                                      negative value as the absolute value)
c                                    cnprop(2) = derivative of the real constant
c                                                w.r.t. geometric penetration/gap
c                                    cnprop(3) = derivative of the real constant
c                                                w.r.t. contact normal pressure
c                                    cnprop(4) = derivative of the real constant
c                                                w.r.t. temperature at contact
c                                    cnprop(5) = derivative of the real constant
c                                                w.r.t. temperature at target
c     usvr     (dp,ar(nuval,nintp),inout)- updated additional state variables
c                                    They are passed in as the values at the
c                                    beginning of this substep.
c                                    They are updated to be the values at the
c                                    end of this substep
c                                    Use NSVR command to size usvr array and
c                                    set nuval to same value as number of
c                                    variables on NSVR commands
c                                    Use userou.F to save these values
c                                    on NMISC record for output purposes.
c                                    The number of user defined output items on
c                                    NMISC should be equal or less than NSTV
c                                    on nsvr command). It cannot exceed 120.
c
c      keyerr   (int,sc,inout)      - key to indicate if there is any element
c                                    formulation error.
c                                    The error could be caused by too
c                                    large incremental step, illegal model.
c                                    = 0 no error (present value before calling)
c                                    = 1 some error happens. ANSYS will
c                                    decide to stop the analysis or cutback
c                                    the substep (bi-section) based on other
c                                    user input and information at higher
c                                    level.
c
```

2.5.2. Subroutine userfric (Writing Your Own Friction Laws)

This subroutine applies to the CONTA17*x* contact elements.

```
*deck,userfric                       USERSDISTRIB
      subroutine userfric (elem,mat,intpt,nkeyopt,keyopt,nrl,rlconst,
     x ncomp,npropu,uprop,kfirst,kfsteq,kn,kt,elen,kstat,timval,
     x timinc,tcont,ttarg,toffst,dslip,slip,pres,tau,dt,usvr,
     x fdiss,elener,kupdhis,mu,dtdp)
c
c *** primary function:   Allow users to write their own friction laws.
c                         This logic is accessed with tb,fric with tbopt=user.
c                         The below demonstration logic is the same as using
c                         tb,fric for isotropic Coulomb friction.
c                         Other friction laws may require more general
c                         definition of friction forces.
c *** secondary function: demonstrate the use of user-written friction laws
c                         in this routine:
c                         a. update history variables
c                         b. compute consistent tangent matrix
c
```

```
c *** Notice - This file contains ANSYS Confidential information ***
c
c
c         Copyright ANSYS.  All Rights Reserved.
c         *** ansys, inc.
c
c  input arguments:
c     variable (type,sze,intent)    description
c
c     elem     (int,sc,in)         - element number (label)
c     mat      (int,sc,in)         - material reference number
c     intpt    (int,sc,in)         - element integration point number
c     nkeyopt  (int,sc,in)         - number of key options
c     keyopt   (int,ar(nkeyopt),in)- array containing key options
c                                    keyopt(1) : Select degree of freedom
c                                    keyopt(2) : Contact algorithm
c                                    ... so on (see ANSYS documentation)
c     nrl      (int,sc,in)         - number of real constants
c     rlconst  (dp,ar(nrl),in)     - array containing real constants
c                                    Elements CONTA171 to CONTA177
c                                    rlconst(1) : R1
c                                    rlconst(2) : R2
c                                    rlconst(3) : FKN
c                                    rlconst(4) : FTOLN
c                                    ... so on (see ANSYS documentation)
c                                    Element CONTA178
c                                    rlconst(1) : FKN
c                                    rlconst(2) : GAP
c                                    ... so on (see ANSYS documentation)
c     ncomp    (int,sc,in)         - no. of friction stress components (1 or 2)
c     npropu   (int,sc,in)         - no. of user-defined friction properties
c     uprop    (dp,ar(npropu),in)  - user-defined material properties
c     kfirst   (int,sc,in)         - 1 if first time through, 0 otherwise
c                                    (useful for initializing state variables
c                                    to a non-zero value)
c     kfsteq   (int,sc,in)         - 1 if first equilibrium iteration of a
c                                    substep, 0 otherwise
c     kn       (dp,sc,in)          - normal penalty stiffness
c     kt       (dp,sc,in)          - tangential penalty stiffness
c                                    (an initial guess is provided but
c                                     the user must pick a suitable
c                                     value that allows minimal tangential
c                                     slip during sticking without
c                                     adversely affecting the convergence;
c                                     a possible choice could be kt=mu*kn).
c                                     For Lagrange multiplier method (keyopt(2)=4
c                                     use a small kt (several orders of magnitude
c                                     smaller than mu*pres).
c     elen     (dp,sc,in)          - length of contact element
c     kstat    (int,sc,inout)      - contact status
c                                    3 : stick
c                                    2 : sliding
c                                    1 : open contact (near)
c                                    0 : open contact (far)
c     timval   (dp,sc,in)          - current time value
c     timinc   (dp,sc,in)          - time increment over this substep
c     tcont    (dp,sc,in)          - contact surface temperature
c                                    (from temperature DOF or temperature load)
c     ttarg    (dp,sc,in)          - target surface temperature
c                                    (only from temperature DOF)
c     toffst   (dp,sc,in)          - temperature offset from absolute zero
c     dslip    (dp,ar(ncomp),in)   - slip increment (current substep)
c     slip     (dp,ar(ncomp),inout)- accumulated slip (previous substep)
c     pres     (dp,sc,in)          - normal pressure/force (current substep)
c                                    > 0 : compression
c                                    < 0 : tension
c     tau      (dp,ar(ncomp),inout)- frictional stress (previous substep)
c                                    Lagrange multiplier contribution is added
c                                    if keyopt(2)=4
c     usvr     (dp,ar(nuval,nintp),inout)- additional state variables from
c                                    previous equilibrium iteration (saved
c                                    if the nsvr command is used)
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries and affiliates.

```
c     kupdhis  (int,sc,in)         - key to indicate if history-dependent
c                                    variables (user defined) need to be
c                                    updated after the substep has converged
c                                    1 : update (converged)
c                                    0 : do not update (not converged)
c
c  output arguments:
c     variable (type,sze,intent)    description
c
c     kstat    (int,sc,inout)      - updated contact status
c     mu       (dp,sc  inout)      - updated friction coefficient
c     slip     (dp,ar(ncomp),inout)- updated accumulated slip
c     tau      (dp,ar(ncomp),inout)- updated frictional stress
c     dt       (dp,ar(5,5),out)    - material tangent modulus
c                                    rows and columns of dt matrix are
c                                    associated to:
c                                    row 1 : frictional stress in direction 1
c                                    row 2 : frictional stress in direction 2
c                                    row 3 : normal pressure
c                                    row 4 : blank
c                                    row 5 : blank
c                                    col 1 : sliding in direction 1
c                                    col 2 : sliding in direction 2
c                                    col 3 : normal gap
c                                    col 4 : temperature at contact
c                                    col 5 : temperature at targte
c                                    relevant components to be filled in are:
c                                    dt(1,1): d(tau1)/d(slip1)
c                                    dt(1,2): d(tau1)/d(slip2)
c                                    dt(1,3): d(tau1)/d(normal gap)
c                                    dt(2,1): d(tau2)/d(slip1)
c                                    dt(2,2): d(tau2)/d(slip2)
c                                    dt(2,3): d(tau2)/d(normal gap)
c                                    dt(3,3): d(pres)/d(normal gap)
c                                    dt(3,3) set to kn internally
c                                    dt(1,4) : d(tau1)/d(tcont)
c                                    dt(1,5) : d(tau1)/d(ttarg)
c                                    dt(2,4) : d(tau2)/d(tcont)
c                                    dt(2,5) : d(tau2)/d(ttarg)
c     dtdp     (dp,ar(ncomp),out)  - partial derivative of the frictional
c                                    stress in direction 1/2 w.r.t. normal
c                                    pressure used in Lagrange multiplier
c                                    method (keyopt(2)=3,4).
c     usvr     (dp,ar(nuval,nintp),inout)- updated additional state variables
c                                    For example, mu value and absolute
c                                    accumulated slip could be output as follows:
c                                    usvr(1,intpt) : mu
c                                    usvr(2,intpt) : abs. acc. slip in dir1
c                                    usvr(3,intpt) : abs. acc. slip in dir2
c                                    Use NSVR command to size usvr array and
c                                    set nuval to same value as number of
c                                    variables on NSVR commands
c                                    Use userou.F to save these values
c                                    on NMISC record for output purposes.
c                                    The number of user defined output items on
c                                    NMISC should be equal or less than NSTV
c                                    on nsvr command). It cannot exceed 120.
c
c     fdiss    (dp,sc,out)         - incremental frictional dissipation
c                                    per unit area
c     elener   (dp,sc,out)         - incremental elastic stored energy
c                                    per unit area
c
c  fortran parameters (to be defined by the user):
c     variable (type)              description
c     nuval    (int)               - number of additional state variables per
c                                    integration point
c     nintp    (int)               - maximum number of integration points of
c                                    an element to be used with this routine
c                                    (14 is the maximum)
c         note:  nuval x nintp = nstv(on nsvr command); cannot exceed 840!
c
```

```
c  internal variables:
c     variable (type,sze)      description
c     dtfac    (dp,sc)         - temporary variable
c     taulim   (dp,sc)         - limit frictional stress
c     taueq    (dp,sc)         - equivalent frictional stress
c     dir1     (dp,sc)         - slip increment direction 1
c     dir2     (dp,sc)         - slip increment direction 2
c     dslipeq  (dp,sc)         - equivalent slip increment
c     oldt1    (dp,sc)         - frictional stress 1 from prev substep
c     oldt2    (dp,sc)         - frictional stress 2 from prev substep
c     err      (dp,ar(2))      - data array for diagnostic message
c
```

2.5.3. Subroutine userinter (Writing Your Own Contact Interactions)

This subroutine applies to the CONTA17*x* contact elements.

```
*deck,userinter                       USERSDISTRIB
      subroutine userinter (ndim,coor,nkeyopt,keyopt,nrl,rlconst,
     x  npropu,uprop,nintIn,intIn,nrealIn,realIn,kupdhis,localr,
     x  nuval,nintp,usvr,ncomp,stress,strain0,strain,
     x  kstat,mu,dt,dtdp,kdamp,damp,fdiss,elener,keyerr,keycnv)
c
c *** primary function:   Allow users to write their own interaction behavior.
c                         This logic is accessed with tb,inter with tbopt=user.
c *** secondary function: demonstrate the use of user-written interface laws
c                         in this routine:
c                         a. update history variables
c                         b. compute consistent tangent matrix
c
c *** Notice - This file contains ANSYS Confidential information ***
c
c
c         *** Copyright ANSYS.  All Rights Reserved.
c         *** ansys, inc.
c
c  input arguments:
c     variable (type,sze,intent)    description
c
c     elem     (int,sc,in)         - element number
c     intpt    (int,sc,in)         - element integration point number
c     ndim     (int,sc,in)         - number of dimensions of the problem
c                                    = 2 2D
c                                    = 3 3D
c     nintp    (int,sc,in)         - the total number of integration points of
c                                    an element to be used with this routine
c     nuval    (int)               - number of additional state variables per
c                                    integration point
c         note:  nuval x nintp = nstv(on nsvr command); cannot exceed 840!
c
c     intIn    (int,ar(*),in)      - integer variables passed in
c                                    intIn(1) = element number
c                                    intIn(2) = element integration point number
c                                    intIn(3) = material reference number
c                                    intIn(4) = element type ID number (absolute value)
c                                               > 0 for CONTA171-CONTA177
c                                               < 0 for CONTA178
c                                    intIn(5) = real constant ID number
c                                    intIn(6) = associated contact nodal number
c                                    intIn(7) = contact indicator
c                                               0: intersection is found
c                                               otherwise: no intersection
c                                    intIn(8) = target element number
c                                    intIn(9) = flag for forcing sliding
c                                               frictional case
c                                               0: - not forcing
c                                               1: - forcing (Slip direction is
c                                                    defined through CMROT command)
c                                    intIn(10) = 1 first pass through
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries and affiliates.

```
c                                                (1st iteration)
c                                                (useful for initializing state
c                                                 variables to a non-zero value)
c                                             = 2 first pass through key of
c                                                 a restart
c                                             = 3 first pass through key of
c                                                 a rezoning
c                                       intIn(11) = current load step number
c                                       intIn(12) = current substep number
c                                       intIn(13) = current equilibrium iteration number
c                                       intIn(14) = flag for using unsymmetric matrices
c                                                   (nropt,unsym)
c                                                   0: - symmetric
c                                                   1: - unsymmetric
c                                       intIn(15) = Linear perturbation flag
c                                                   0: - a general load step
c                                                   1: - a linear perturbation step
c                                       intIn(16) = key to indicate output pass
c                                                   0: not a output pass
c                                                   1: output pass
c                                       intIn(17) = key for displacement & force
c                                                   convergence
c                                                   1: converged
c                                                   0: not converged
c                                       intIn(18) = key to indicate transient effects
c                                                   1 : transient is active
c                                                   0 : transient is not active
c                                       intIn(19) = large deformation key [nlgeom cmd]
c                                                   1 : on
c                                                   0 : off
c                                       intIn(20) = analysis type (derived from antype)
c                                                   0 : a static analysis
c                                                   1 : a buckling analysis
c                                                   2 : a modal analysis
c                                                   3 : a harmonic analysis
c                                                   4 : a transient analysis
c                                                   7 : a substructure analysis
c                                                   8 : a spectrum analysis
c     realIn   (dp,ar(*),in)       - real variables passed in
c                                       realIn(1) = contact element length
c                                       realIn(2) = contact element depth
c                                       realIn(3) = area associated with the contact
c                                                   detection point
c                                       realIn(4) = pinball radius
c                                       realIn(5) = unscaled normal penalty stiffness
c                                       realIn(6) = time (or frequency for a harmonic
c                                                   analysis) at the beginning of this
c                                                   load step
c                                       realIn(7) = time (or frequency for a harmonic
c                                                   analysis) at the end of this load step
c                                       realIn(8) = current time value (or frequency value
c                                                   for a harmonic analysis)
c                                       realIn(9) = time increment (or frequency increment
c                                                   for a harmonic analysis) over this
c                                                   substep
c                                       realIn(10) = temperature offset from absolute
c                                                    zero
c                                       realIn(11) = geometric penetration/gap
c                                                    (current substep)
c                                                    > 0 : gap
c                                                    < 0 : penetration
c                                       realIn(12) = time increment scaling factor to
c                                                    be used for structural transient
c                                                    dynamics
c                                       realIn(13) = convection coefficient (SFE command)
c                                       realIn(14) = bulk temp (SFE command)
c     nkeyopt  (int,sc,in)         - number of key options
c     keyopt   (int,ar(nkeyopt),in)- array containing key options
c                                       keyopt(1) : Select degree of freedom
c                                       keyopt(2) : Contact algorithm
c                                       ... so on (see ANSYS documentation)
c     nrl      (int,sc,in)         - number of real constants
```

```
c     rlconst  (dp,ar(nrl),in)     - array containing real constants
c                                    Elements CONTA171 to CONTA177
c                                    rlconst(1) : R1
c                                    rlconst(2) : R2
c                                    rlconst(3) : FKN
c                                    rlconst(4) : FTOLN
c                                    ... so on (see ANSYS documentation)
c                                    Element CONTA178
c                                    rlconst(1) : FKN
c                                    rlconst(2) : GAP
c                                    ... so on (see ANSYS documentation)
c     ncomp    (int,sc,in)         - number of stress/force component
c                                    = 9 for CONTA171-CONTA177
c                                    = 7 for CONTA178
c     stress   (dp,ar(ncomp),inout)- stress components (current substep)
c                                    It is passed in as the stress at the beginning
c                                    of the current substep. It is updated to be
c                                    the stress at the end of this current substep
c                                    stress(1) = frictional stress in direction 1
c                                    stress(2) = frictional stress in direction 2
c                                                (3D only)
c                                    stress(3) = contact normal pressure
c                                    > 0 : compression
c                                    < 0 : tension
c                                    the above contact traction must be defined in
c                                    a local coordinate system (see localr)
c                                    Lagrange multiplier contribution is added
c                                    if keyopt(2)=3,4
c                                    stress(4) = heat flux (per area)
c                                                flowing into contact
c                                    stress(5) = heat flux (per area)
c                                                flowing into target
c                                    < 0 heat flowing into a surface
c                                    > 0 heat flowing out of a surface
c                                    stress(6) = electrical current density
c                                                (or pore fluid flux density)
c                                                (per area) flowing into contact
c                                    stress(7) = electrical current density
c                                                (or pore fluid flux density)
c                                                (per area) flowing into target
c                                    > 0 current flowing out of a surface
c                                    < 0 current flowing into a surface
c                                    stress(8) = diffusion flux density
c                                                (per area) flowing into contact
c                                    stress(9) = diffusion flux density
c                                                (per area) flowing into target
c                                    > 0 flux flowing out of a surface
c                                    < 0 flux flowing into a surface
c     strain0  (dp,ar(ncomp),in)   - strain components in the end of the previous
c                                    substep
c                                    (see strain for each component definition)
c     strain   (dp,ar(ncomp),in)   - current strain components
c                                    strain(1) = slip increment in direction 1
c                                    strain(2) = slip increment in direction 2
c                                                (3D only)
c                                    strain(3) = contact normal gap/penetration
c                                    < 0 : gap
c                                    > 0 : penetration
c                                    strain(4) = temperature at the contact point
c                                               (from TEMP DOF or temperature load)
c                                    strain(5) = temperature at the target point
c                                               (only from TEMP DOF)
c                                    strain(6) = voltage (or pore pressure)
c                                                at the contact point
c                                    strain(7) = voltage (or pore pressure)
c                                                at the target point
c                                    strain(8) = concentrationat the contact point
c                                    strain(9) = concentrationat the target point
c
c     kstat    (int,sc,inout)      - contact status (current substep)
c                                    It is passed in as the status at the
c                                    beginning of the current substep.
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries and affiliates.

```
c                                   It is updated to be the status at the
c                                   end of the current substep
c                                   3 : stick
c                                   2 : sliding
c                                   1 : open contact (near)
c                                   0 : open contact (far)
c     coor     (dp,ar(6),in)       - Coordinates of the contact detection point
c                                   coor(1) current x
c                                   coor(2) current y
c                                   coor(3) current z
c                                   coor(4) initial x
c                                   coor(5) initial y
c                                   coor(6) initial z
c     localr   (dp,ar(3,3),in)     - the direction cosines of the local surface
c                                   coordinate system at contact detection
c                                   localr(1,1),localr(1,2),localr(1,3) in
c                                                          slip direction 1
c                                   localr(2,1),localr(2,2),localr(2,3) in
c                                                          slip direction 2
c                                   localr(3,1),localr(3,2),localr(3,3) in
c                                                           normal direction
c
c     npropu   (int,sc,in)         - number of user-defined interaction properties
c     uprop    (dp,ar(npropu),in)  - user-defined material properties
c
c     usvr     (dp,ar(nuval,nintp),inout)- additional state variables from
c                                    previous equilibrium iteration (saved
c                                    if the nsvr command is used)
c     kupdhis  (int,sc,in)         - key to indicate if history-dependent
c                                   variables (user defined) need to be
c                                   updated after the substep has converged
c                                   1 : update (converged)
c                                   0 : do not update (not converged)
c
c  output arguments:
c     variable (type,sze,intent)   description
c
c     kstat    (int,sc,inout)      - updated contact status
c     stress   (dp,ar(ncomp),inout)- updated stress components
c
c     dt   (dp,ar(ncomp,ncomp),out)- interface stiffness matrix:
c                                   dt(i,j) defines the partial derivative of
c                                   the ith stress component at the current
c                                   substep w.r.t. the jth component of the
c                                   relative strain increment array.
c                                   If symmetric solver option used, ANSYS will
c                                   symmetrize the matrix bu averaging the
c                                   off-diagonal terms.
c                                   rows and columns of dt matrix are
c                                   associated to:
c                                   row 1 : frictional stress in direction 1
c                                   row 2 : frictional stress in direction 2
c                                   row 3 : normal pressure
c                                      > 0 : compression
c                                      < 0 : tension
c                                   row 4 : heat flux out the contact surface
c                                      < 0 heat flowing into contact
c                                      > 0 heat flowing out of target
c                                   row 5 : heat flux out the target surface
c                                      < 0 heat flowing into target
c                                      > 0 heat flowing out of target
c                                   row 6 : electrical current density
c                                           (or pore prssure)
c                                           flowing out the contact surface
c                                      > 0 current flowing out of contact
c                                      < 0 current flowing into contact
c                                   row 7 : electrical current density
c                                           (or pore prssure)
c                                      > 0 current flowing out of target
c                                      < 0 current flowing into target
c                                   row 8 : diffusion flux density
c                                           flowing out the contact surface
```

```
c                                       > 0 flux flowing out of contact
c                                       < 0 flux flowing into contact
c                                     row 9 : diffusion flux density
c                                       > 0 flux flowing out of target
c                                       < 0 flux flowing into target
c                                     col 1 : sliding in direction 1
c                                     col 2 : sliding in direction 2
c                                     col 3 : normal gap
c                                       < 0 : gap
c                                       > 0 : penetration
c                                     col 4 : temperature at the contact surface
c                                     col 5 : temperature at the target surface
c                                     col 6 : voltage at the contact surface
c                                     col 7 : voltage at the target surface
c                                     col 8 : concentration at the contact surface
c                                     col 9 : concentration at the target surface
c                                     relevant components to be filled in are:
c                                     dt(1,1): d(tau1)/d(slip1)
c                                     dt(1,2): d(tau1)/d(slip2)
c                                     dt(1,3): d(tau1)/d(normal gap)
c                                     dt(1,4): d(tau1)/d(tempC)
c                                     dt(1,5): d(tau1)/d(tempT)
c                                     dt(1,6): d(tau1)/d(voltC)
c                                     dt(1,7): d(tau1)/d(voltT)
c                                     dt(1,8): d(tau1)/d(concC)
c                                     dt(1,9): d(tau1)/d(concT)
c                                     dt(2,1): d(tau2)/d(slip1)
c                                     dt(2,2): d(tau2)/d(slip2)
c                                     dt(2,3): d(tau2)/d(normal gap)
c                                     ...
c                                     dt(3,1): d(pres)/d(slip 1)
c                                     dt(3,2): d(pres)/d(slip 2)
c                                     dt(3,3): d(pres)/d(normal gap)
c                                     ...
c                                     dt(4,1): d(fluxC)/d(slip 1)
c                                     dt(4,2): d(fluxC)/d(slip 2)
c                                     dt(4,3): d(fluxC)/d(normal gap)
c                                     dt(4,4): d(fluxC)/d(tempC)
c                                     dt(4,5): d(fluxC)/d(tempT)
c                                     dt(4,6): d(fluxC)/d(voltC)
c                                     dt(4,7): d(fluxC)/d(voltT)
c                                     dt(4,8): d(fluxC)/d(concC)
c                                     dt(4,9): d(fluxC)/d(concT)
c                                     ...
c                                     dt(5,4): d(fluxT)/d(tempC)
c                                     dt(5,5): d(fluxT)/d(tempT)
c                                     dt(5,6): d(fluxT)/d(voltC)
c                                     dt(5,7): d(fluxT)/d(voltT)
c                                     dt(5,8): d(fluxT)/d(concC)
c                                     dt(5,9): d(fluxT)/d(concT)
c                                     ...
c                                     dt(6,4): d(eleC)/d(tempC)
c                                     dt(6,5): d(eleC)/d(tempT)
c                                     dt(6,6): d(eleC)/d(voltC)
c                                     dt(6,7): d(eleC)/d(voltT)
c                                     dt(6,8): d(eleC)/d(concC)
c                                     dt(6,9): d(eleC)/d(concT)
c                                     ...
c                                     dt(7,4): d(eleT)/d(tempC)
c                                     dt(7,5): d(eleT)/d(tempT)
c                                     dt(7,6): d(eleT)/d(voltC)
c                                     dt(7,7): d(eleT)/d(voltT)
c                                     dt(7,8): d(eleT)/d(concC)
c                                     dt(7,9): d(eleT)/d(concT)
c                                     ...
c                                     dt(8,4): d(diffC)/d(tempC)
c                                     dt(8,5): d(diffC)/d(tempT)
c                                     dt(8,6): d(diffC)/d(voltC)
c                                     dt(8,7): d(diffC)/d(voltT)
c                                     dt(8,8): d(diffC)/d(concC)
c                                     dt(8,9): d(diffC)/d(concT)
c                                     ...
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries and affiliates.

```
c                                   dt(9,4): d(diffT)/d(tempC)
c                                   dt(9,5): d(diffT)/d(tempT)
c                                   dt(9,6): d(diffT)/d(voltC)
c                                   dt(9,7): d(diffT)/d(voltT)
c                                   dt(9,8): d(diffT)/d(concC)
c                                   dt(9,9): d(diffT)/d(concT)
c     dtdp     (dp,ar(ncomp),out)  - partial derivative of the frictional stress
c                                   in direction 1,2 w.r.t. normal pressure
c                                   used in Lagrange multiplier method
c                                   (keyopt(2)=3,4).
c     damp     (dp,ar(3,3),out)    - interface damping matrix (structure only)
c                                   it can be used only in Linear perturbation
c                                   modal analysis or transient analysis or
c                                   harmonic analysis in frequence domain.
c                                   damp(i,j) defines the partial derivative of
c                                   the ith stress component at the current
c                                   substep w.r.t. the jth component of the
c                                   strain increment rate array.
c                                   rows and columns of dt matrix are
c                                   associated to:
c                                   row 1 : frictional stress in direction 1
c                                   row 2 : frictional stress in direction 2
c                                   row 3 : normal pressure
c                                   col 1 : sliding rate in direction 1
c                                   col 2 : sliding rate in direction 2
c                                   col 3 : normal gap rate
c     kdamp    (int,sr,out)        - damping matrix index
c                                   0 : no damping matrix
c                                   1 : taking damping matrix into account
c     usvr     (dp,ar(nuval,nintp),inout)- updated additional state variables
c                                   For example, mu value and absolute/relative
c                                   accumulated slip could be output as follows:
c                                   usvr(1,intpt) : mu
c                                   usvr(2,intpt) : abs. acc. slip in dir1
c                                   usvr(3,intpt) : abs. acc. slip in dir2
c                                   usvr(4,intpt) : acc. slip in dir1
c                                   usvr(5,intpt) : acc. slip in dir2
c                                   They are passed in as the values at the
c                                   beginning of this substep. They are updated
c                                   to be the values at the end of this substep.
c                                   Use NSVR command to size usvr array and
c                                   set nuval to same value as number of
c                                   variables on NSVR commands
c                                   Use userou.F to save these values
c                                   on NMISC record for output purposes.
c                                   The number of user defined output items on
c                                   NMISC should be equal or less than NSTV
c                                   on nsvr command). It cannot exceed 120.
c
c     mu       (dp,sc,inout)       - The current frictional coefficient
c     fdiss    (dp,sc,out)         - incremental frictional dissipation
c                                   per unit area
c     elener   (dp,sc,out)         - incremental elastic stored energy
c                                   per unit area
c
c     keyerr (int,sc,out)          - key to indicate if there is any element
c                                   formulation error, like
c                                   contact status changes abruptly,
c                                   too much penetration.
c                                   The error could be caused by too
c                                   large incremental step, illegal model.
c                                   = 0 no error (preset value before calling)
c                                   = 1 some error happens. ANSYS will
c                                   decide to stop the analysis or cutback
c                                   the substep (bi-section) based on other
c                                   user input and information at higher
c                                   level.
c     keycnv (int,sc,inout)        - key to flag if this element satisfies
c                                   the user defined element convergence
c                                   criterion.
c                                   = 1, yes, the criterion is satisfied
c                                     or don't have any criterion at all
```

```
c                                     it is preset value before calling
c                                   = 0, no, the element doesn't satisfy
c                                     element convergence criterion. If
c                                     this is the case, the iteration will
c                                     not converge even when both force
c                                     and displacement converge
c
c  internal variables:
c     variable (type,sze)      description
c
```

2.5.4. Subroutine userwear (Writing Your Own Wear Law)

This subroutine applies to contact elements CONTA171 - CONTA175.

```
*deck,userwear                        USERSDISTRIB
      subroutine userwear(WearInc,WearDir,TotWearOld,strain,
     x     stress,temperature,dtime,YieldStress,nTbprop,Tbprop,
     x     coor,kstat,elem,intpt,ndim,localr)
c *** Primary Function:   Calculates the Wear Increment and (optionally) wear direction
c *** Notice - This file contains ANSYS Confidential information ***
c
c
c         *** Copyright ANSYS.  All Rights Reserved.
c         *** ansys, inc.
c input arguments:
c     variable (type,sze,intent)    description
c
c     ndim     (int,sc,in)         - number of dimensions of the problem
c                                    = 2 2D
c                                    = 3 3D
c     TotWearOld(dp,ar(ndim),in)   - Total Wear at the contact point at the previous substep
c     strain   (dp,ar(3),in)   - current strain components in contact surface coordinate system
c                                    strain(1) = slip increment in direction 1
c                                    strain(2) = slip increment in direction 2
c                                                (3D only)
c                                    strain(3) = contact normal gap/penetration
c                                    < 0 : gap
c                                    > 0 : penetration
c     stress   (dp,ar(3),in)   - stress components  in contact surface coordinate system
c                                    stress(1) = frictional stress in direction 1
c                                    stress(2) = frictional stress in direction 2
c                                                (3D only)
c                                    stress(3) = contact normal pressure
c                                    > 0 : compression
c                                    < 0 : tension
c     temperature  (dp,sc,in)      - temperature
c     dtime (dp,sc,in)             - time increment
c     YieldStress (dp,sc,in)       - Yield stress of underlying element (defined only for Plastic material-see doc
c     nTbprop (int,sc,in)          - Number of TBdata for Tb,Wear per field
c     Tbprop (dp,ar(nTbprop,in)    - TB data for the the Tb,Wear option at the given temperature
c     coor     (dp,ar(6),in)       - Coordinates of the contact detection point
c                                     coor(1) current x
c                                     coor(2) current y
c                                     coor(3) current z
c                                     coor(4) initial x
c                                     coor(5) initial y
c                                     coor(6) initial z
c     kstat    (int,sc,in)          - contact status (current substep)
c                                     3 : stick
c                                     2 : sliding
c                                     1 : open contact (near)
c                                     0 : open contact (far)
c     elem     (int,sc,in)         - element number
c     intpt    (int,sc,in)         - element integration point number
c     localr   (dp,ar(3,3),in)     - the direction cosines of the local surface
c                                    coordinate system at contact detection
c                                    localr(1,1),localr(1,2),localr(1,3) in
c                                                          slip direction 1
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries and affiliates.

```
c                                   localr(2,1),localr(2,2),localr(2,3) in
c                                                          slip direction 2
c                                   localr(3,1),localr(3,2),localr(3,3) in
c                                                           normal direction
c Output Arguments:
c     variable (type,sze,intent)   description
c
c     WearInc  (dp,sc)             - Increment in the Wear (magnitude)- User must define
c     WearDir  (dp,ar(ndim),inout) - Direction cosines in global coordinate system
c                                    in which wear increment will be applied- Optional
c                                    default coming in -Contact normal direction
```

2.6. Subroutines for Customizing Loads

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

2.6.1. Subroutine usrefl (Changing Scalar Fields to User-Defined Values)
2.6.2. Subroutine userpr (Changing Element Pressure Information)
2.6.3. Subroutine usercv (Changing Element Face Convection Surface Information)
2.6.4. Subroutine userfx (Changing Element Face Heat Flux Surface Information)
2.6.5. Subroutine userch (Changing Element Face Charge Density Surface Information)
2.6.6. Subroutine userfd (Calculating the Complex Load Vector for Frequency Domain Logic)
2.6.7. Function userpe (Calculating Rotation Caused by Internal Pressure)
2.6.8. Subroutine usrsurf116 (Modifying SURF151 and SURF152 Film Coefficients and Bulk Temperatures)
2.6.9. Subroutine User116Cond (Calculating the Conductance Coefficient for FLUID116)
2.6.10. Subroutine User116Hf (Calculating the Film Coefficient for FLUID116)
2.6.11. Subroutine userPartVelAcc (Calculating Particle Velocities and Accelerations of Ocean Waves)
2.6.12. Subroutine userPanelHydFor (Calculating Panel Loads Caused by Ocean Loading)

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

```
*deck,usrefl                       USERSDISTRIB
      subroutine usrefl (key,iel,ielc,nnod,nodes,time,defalt,nd,dat)
c *** primary function:  change the scalar fields (temperatures, fluences,
c                  heat generation, etc.) to what user desires.
c *** secondary functions: none
c
c         in order to activate this user programmable feature,
c         the user must enter the usrcal command.
c
c         this routine is called at each substep of each load step
c         for which element or nodal temperatures(etc) are used.
c         it is called for each equilibrium iteration.
c         the call to get the standard ansys input element or nodal values
c         is made just before entering this routine.
c
c         *** Copyright ANSYS.  All Rights Reserved.
c         *** ansys, inc.
c
c *** Notice - This file contains ANSYS Confidential information ***
c
c     typ=int,dp,log,chr,dcp   siz=sc,ar(n)   intent=in,out,inout
c
c  input arguments:
c     variable (typ,siz,intent)    description
c     key      (int,sc,in)       - type of data desired
c                                    = 1 temperatures
c                                    = 2 fluences
c                                    = 3 heat generation rates
c                                    = 4 moisture contents
c                                    = 5 magnetic virtual displacements
c     iel      (int,sc,in)       - element number
c     ielc   (int,ar(IELCSZ),in) - array of element type characteristics
```

```
c     nnod     (int,sc,in)       - number of nodes
c     nodes    (int,ar(nnod),in) - list of nodes
c     time     (dp,sc,in)        - time of current substep
c     defalt   (dp,sc,in)        - default value (e.g. tunif)
c     nd       (int,sc,in)       - size of dat array
c     dat      (dp,ar(nd),inout) - array of data as normally computed by element
c                                  as selected by key
c
c  output arguments:
c     variable (typ,siz,intent)    description
c     dat      (dp,ar(nd),inout) - array of data passed back to element
c                                    this data represents values at the end
c                                    of the load step
c
c        the input argument dat may be used in one of three ways:
c            1.  it may be simply passed thru
c            2.  it may be used as a flag(e.g. if dat(1) = -3.0, use
c                                 a certain set of logic)
c            3.  it may be completely ignored and instead defined with new logic
c
```

2.6.2. Subroutine userpr (Changing Element Pressure Information)

```
*deck,userpr                       USERSDISTRIB
      subroutine userpr (ielc,elem,time,ndat,dat)
c *** primary function:    change element pressure information.

c         *** Copyright ANSYS.  All Rights Reserved.
c         *** ansys, inc.

c         in order to activate this user programmable feature,
c         the user must enter the 'usrcal,userpr' command.

c         this routine is called at each substep of each load step for which
c         pressures are used. it is called for each equilibrium iteration.
c         it is called once per element.
c         the call to get the standard ansys input pressures is made just before
c         entering this routine.

c     input arguments:
c        variable (typ,siz,intent)    description
c        ielc   (int,ar(IELCSZ),in) - array of element type characteristics
c        elem     (int,sc,in)       - element number for operation.
c        time     (dp,sc,in)        - time of current substep
c        ndat     (int,sc,in)       - number of pressure items for this element
c        dat      (dp,ar(ndat,2),inout) - the element pressure vector
c                                      (has input values for each corner
c                                      of each face)


c     output arguments:
c        variable (typ,siz,intent)    description
c        dat      (dp,ar(ndat,2),inout) - the element pressure vector
c                                      (defines input values for each corner
c                                      of each face)
c                                      dat(1:ndat,1) - real pressures
c                                      dat(1:ndat,2) - complex pressures
c                                                      (surface elements only)


c        the input array  dat  may be used in one of three ways:
c          1.  it may be simply passed thru
c          2.  it may be used as a flag(e.g. if dat(1) = -3.0, use
c                 a certain set of logic)
c          3.  it may be completely ignored and instead defined with new logic
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries and affiliates.

2.6.3. Subroutine usercv (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
c         *** Copyright ANSYS.  All Rights Reserved.
c         *** ansys, inc.
c
c         in order to activate this user programmable feature,
c         the user must enter the 'usrcal,usercv' command.
c
c        the input arguments hc and tb may be used in one of three ways:
c            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).
c            3.  they may be completely ignored.
c                                 and instead redefined with new logic

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         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         is made just before entering this routine, so this information is
c         available to be modified, if desired.
c
c         velocity-dependent film coefficients can be computed by inputting the
c         velocity as the input film coefficient or bulk temperature or
c         by inputting the velocity as a function of location in space.  this
c         routine could then compute the effective film coefficient.


c
c     input arguments:
c        variable (typ,siz,intent)    description
c        elem     (int,sc,in)       - element number for operation.
c        ielc   (int,ar(IELCSZ),in) - array of element type characteristics
c        time     (dp,sc,in)        - time of current substep
c        nr       (int,sc,in)       - number of nodal temperatures
c                                         of the element
c        u        (dp,ar(nr),in)    - vector of most recent values of the
c                                      temperatures
c        ndat     (int,sc,in)       - number of data points per element
c                                      for example, for solid70, ndat = 24 = 6*4
c                                      where 6 = faces per element
c                                            4 = corners per face
c        hc     (dp,ar(ndat),inout) - film coefficients
c                                      (has input values for each corner
c                                      of each face)
c        tb     (dp,ar(ndat),inout) - bulk temperature
c                                      (has input values for each corner
c                                      of each face)
c
c     output arguments:
c        variable (typ,siz,intent)    description
c        hc     (dp,ar(ndat),inout) - film coefficients
c                                      (defines input values for each corner
c                                      of each face)
c        tb     (dp,ar(ndat),inout) - bulk temperature
c                                      (defines input values for each corner
c                                      of each face)
c
```

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

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

2.6.5. 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
c
c         in order to activate this user programmable feature,
c         the user must enter the usrcal command.
c
c         this routine is called during each substep of each load step.
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         evaluation stage.
c         the call to get the standard ansys input charge densities of surfaces
c         is made just before entering this routine, so this information is
c         available to be modified, if desired.
c
c         *** Copyright ANSYS.  All Rights Reserved.
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
c         *** ansys, inc.
c
c     input arguments:
c        variable (typ,siz,intent)    description
c        ielc   (int,ar(IELCSZ),in) - array of element type characteristics
c        ielem    (int,sc,in)       - element number for operation.
c        time     (dp,sc,in)        - time of current substep
c        nr       (int,sc,in)       - number of nodal temperatures
c                                        of the element
c        u        (dp,ar(nr),in)    - vector of most recent values of the
c                                      temperatures
c        ndat     (int,sc,in)       - number of data points per element
c        dat    (dp,ar(ndat),inout) - fluxes
c
c     output arguments:
c        variable (typ,siz,intent)    description
c        dat    (dp,ar(ndat),inout) - fluxes
c
c        the input argument dat may be used in one of three ways:
c            1.  they may be simply passed thru.
c            2.  they may be used as a flag(e.g. if dat(2) = -3.0, use
c                                a certain set of logic).
c            3.  they may be completely ignored.
c                                 and instead redefined with new logic
c
```

2.6.6. Subroutine userfd (Calculating the Complex Load Vector for Frequency Domain Logic)

```
*deck,userfd                        USERSDISTRIB
      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
c                       for pipe59
c *** secondary functions: none
c     -- accessed with keyopt(12) = 2
c
c         *** Copyright ANSYS.  All Rights Reserved.
c         *** ansys, inc.
c *** Notice - This file contains ANSYS Confidential information ***
c
c  input arguments:
c     nr       (int,sc,in)       - matrix size
c     kcbrm    (int,sc,in)       - key for reduced matrices/cable option
c     kpfor    (int,sc,in)       - keyopt for hydrodynamic printout
c     ktrsur   (int,sc,in)       - keyopt for surface treatment(unfinished)
c     isur     (int,sc,in)       - surface flag
c     cb       (dp,sc,in)        - buoyancy coefficient (real constant)
c     do       (dp,sc,in)        - outside diameter of pipe
c     doext    (dp,sc,in)        - outside diameter of insulation
c     aread    (dp,sc,in)        - area of displaced water
c     alenv    (dp,sc,in)        - length of element
c     denswat  (dp,sc,in)        - water density
c     faclen   (dp,sc,in)        - wetted fraction of pipe
c     conac    (dp,sc,in)        - added mass per unit length
c     fluidt   (dp,sc,in)        - fluid temperature
c     visc     (dp,sc,in)        - viscosity
c     watbas   (dp,ar(*),in      - water basic table
c     watcur   (dp,ar(*),in      - water current table
c     watwav   (dp,ar(*),in      - water wave table
c     xyzup    (dp,ar(3,2),in)   - updated coordinates
c     tr       (dp,ar(3,3),in)   - local to global transformation matrix
c     accel    (dp,ar(3),in)     - acceleration vector
c     puvel    (int,sc,in)       - index for velocities in u matrix
c     u        (dp,ar(nr,5),in   - displacements and velocities
c     zass     (dp,ar(nr,nr),in) - mass matrix
c     forl     (dp,ar(12),inout) - force vector in element coordinates
```

```
c     zsc      (dp,ar(nr),inout) - real load vector for frequency domain
c     zsc2     (dp,ar(nr),inout) - complex load vector for frequency domain
c
c  output arguments:
c     forl     (dp,ar(12),inout) - force vector in element coordinates
c     zsc      (dp,ar(nr),inout) - real load vector for frequency domain
c     zsc2     (dp,ar(nr),inout) - complex load vector for frequency domain
c     pdyn     (dp,ar(2),out)    - dynamic pressure
c     holdwv   (dp,ar(60),out)   - wave information held for printout
c
```

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

```
*deck,userpe                       USERSDISTRIB
      function userpe (prs,rvrp,angle,ex,nuxy)

c primary function:    calculate the rotation caused by internal pressure
c                      on an elbow element
c                      This function is only called by el18(pipe18)
c                                          if keyopt(5) = 1

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

c         *** Copyright ANSYS.  All Rights Reserved.
c         *** ansys, inc.

c     typ=int,dp,log,chr,dcp   siz=sc,ar(n)   intent=in,out,inout

c  input arguments:
c     variable (typ,siz,intent)    description
c     prs      (dp,ar(5),in)     - pressure vector
c     rvrp     (dp,ar(11),in)    - real constants(see elements manual)
c     angle    (dp,sc,in)        - subtended angle
c     ex       (dp,sc,in)        - Young's modulus
c     nuxy     (dp,sc,in)        - Poisson's ratio

c  output arguments:
c     variable (typ,siz,intent)    description
c     userpe   (dp,sc,out)       -  rotation caused by internal pressure on the
c                                      elbow element
```

2.6.8. Subroutine usrsurf116 (Modifying SURF151 and SURF152 Film Coefficients and Bulk Temperatures)

```
*deck,usrsurf116                       USERSDISTRIB
      subroutine usrsurf116 (elem,ielc,center,jdim,kaxis,time,nr,u,
     x                omeg,ndat,temvel,hc,tb,temfluid,mdot,key)
c *** primary function: change element convection surface info
c     for surf151 and/or surf152 based on information from fluid116.
c     It is called by el151 and el152.
c
c         in order to activate this user programmable feature,
c         the user must have used fluid116 with keyopt(2) = 1.
c         Further, surf151 and/or surf152 must have keyopt(5) = 1 or 2
c         (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
c         nonzero number.  There is no usrcal control over this routine.
c
c         *** Copyright ANSYS.  All Rights Reserved.
c         *** ansys, inc.
c *** Notice - This file contains ANSYS Confidential information ***
c
c     input arguments:
c        variable (typ,siz,intent)    description
c        elem   (int,sc,in)         - element number for operation.
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
c        ielc   (int,ar(IELCSZ),in) - array of element type characteristics
c        center (dp,ar(3),in)       - coordinates of center of surface element
c        jdim   (int,sc,in)         - dimensionality key
c                                        1 = 2d
c                                        2 = axisymmetric
c                                        3 = 3d
c        kaxis  (int,sc,in)         - axis of rotation (keyopt(3) for el152)
c                                       (see getv116 for definition)
c        time   (dp,sc,in)          - time of current substep
c        nr     (int,sc,in)         - number of nodal temperatures
c                                         of the element
c        u      (dp,ar(nr),in)      - vector of most recent values of the
c                                      temperatures
c        omeg   (dp,sc,in)          - spin real constant (may be from table)
c        ndat   (int,sc,in)         - number of data points per element
c        hc     (dp,ar(ndat),inout) - film coefficients
c                                      (has input values for each corner
c                                      of element)
c        tb     (dp,ar(ndat),in) - bulk temperature
c                                      (has input values for each corner
c                                      of element)
c        temfluid (dp,sc,in)        - temp of fluid at surf151/152 centroid
c                                   - when using kyop5 = 1 or 2
c        mdot (dp,sc,in)            - mass flow rate of fluid when using
c                                   - kyop5 = 2 ( 0 otherwise )
c
c     output arguments:
c        variable (typ,siz,intent)    description
c        temvel (dp,sc,out)         - user defined bulk temperature in excess of
c                                      fluid node temperature
c        hc     (dp,ar(ndat),inout) - film coefficients
c                                      (defines input values for each corner
c                                      of element)
c        key    (int,ar(2),out)     - key if to use this logic
c                                      key(1) = 0 = no new film coefficient
c                                      key(1) = 1 = define new film coefficient
c                                      key(2) is not used currently
c
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         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         is made just before entering this routine, so this information is
c         available to be modified, if desired.
c
c         This routine may be thought of as a specialized version of usercv.
c         Indeed, el151 and el152 also call usercv.  Either (or both, rarely)
c         could be used.
c
c         velocity-dependent film coefficients and bulk temperatures can
c         be computed by using the velocities and other information from
c         fluid116.
c         Details of this procedure are:
c            -- SURF151 or SURF152 are 'pasted' onto the actual solid model.
c            -- flow rate is input to or is computed by FLUID116,
c                 with KEYOPT(2) = 1
c            -- flow rate may be a function of time
c            -- the user defines nodes on the FLUID116 network to be the same
c                 nodes as the 'extra' nodes of SURF151 or SURF152.  If more
c                 than one FLUID116 element is attached to one of these nodes,
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.
```

2.6.9. Subroutine User116Cond (Calculating 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 ***
c
c         *** Copyright ANSYS.  All Rights Reserved.
c         *** ansys, inc.

c  input arguments:
c     elem     (int,sc,in)       - element number
c     prop     (dp,ar(4),in)     - material property vector
c                                    order is:  dens,visc,kxx,c
c     rvr      (dp,ar(24),in)    - real constant vector
c     aleng    (dp,sc,in)        - element length
c     re       (dp,sc,in)        - reynold's number
c     fric     (dp,sc,in)        - friction factor
c     uptot    (dp,ar(2),in      - nodal pressure values from previous iteration
c     uttot    (dp,ar(4),in      - nodal temperature values from prev iteration
c     bco      (dp,sc,inout)     - the conductance coefficient from TB,fcon

c  output arguments:
c     bco      (dp,sc,inout)     - the desired conductance coefficient
```

2.6.10. Subroutine User116Hf (Calculating 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 ***
c
c         *** Copyright ANSYS.  All Rights Reserved.
c         *** ansys, inc.

c  input arguments:
c     elem     (int,sc,in)       - element number
c     prop     (dp,ar(4),inout)  - material property vector
c                                    order is:  dens,visc,kxx,c
c     rvr      (dp,ar(18),in)    - real constant vector
c     aleng    (dp,sc,in)        - element length
c     re       (dp,sc,in)        - reynold's number
c     uptot    (dp,ar(2),in      - nodal pressure values from previous iteration
c     uttot    (dp,ar(4),in      - nodal temperature values from prevs iteration
c     hf       (dp,sc,inout)     - the film coefficient from TB,hflm

c  output arguments:
c     hf       (dp,sc,inout)     - the desired film coefficient
```

2.6.11. Subroutine userPartVelAcc (Calculating Particle Velocities and Accelerations of Ocean Waves)

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 line elements (such as LINK180, BEAM188, BEAM189, PIPE288, and PIPE289). The `userPartVelAcc` subroutine works with the following subroutines:

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

- userPartVelAccSetup (p. 236), which initializes the data for use by userPartVelAcc, and
- userWavHt (p. 236), which calculates the wave height for a user-defined wave.

For your convenience, two I/O service subroutines are called by the userPartVelAcc subroutine: wvhybl (p. 237) and wvargu (p. 237).

```
*deck,userPartVelAcc                        USERSDISTRIB
      subroutine userPartVelAcc (elemId,domInt,xyzg,doIns,depth,denswat,
     x                           ncm, pCur,watcur,
     x                           nw,  pWav,watwav, timval,
     x                           argu,eta,vxyz,axyz,ar,pdynam)
c        ---- accessed only if kwav .ge. 101 -----
c   ***** primary function:  compute particle velocities and accelerations
c                           due to waves and current
c   ***** secondary function:  compute dynamic pressures
c
c         *** Copyright ANSYS.  All Rights Reserved.
c         *** ansys, inc.
c *** Notice - This file contains ANSYS Confidential information ***
c
c  input arguments:
c     elemId   (int,sc,in)       - element id
c     domInt   (int,sc,in)       - integration point number
c     xyzg     (dp,ar(3),in)     - coordinates of point of interest
c     doIns    (dp,sc,in)        - outside diameter with insulation
c     depth    (dp,sc,in)        - water depth
c     denswat  (dp,sc,in)        - water density
c     ncm      (int,sc,in)       - number of current measurements
c     pCur     (int,sc,in)       - pointer in current table (= 30 at 12.0)
c                                     i.e. first item is at watcur(pCur+1)
c     watcur   (dp,ar(*),in)     - water current table
c                                          ic = current reading number
c                                   watcur( 6) = ncm = number of current measurements
c                                   watcur(pCur + (ic-1)*6 + 1) = Z Coor
c                                   watcur(pCur + (ic-1)*6 + 2) = Velocity
c                                   watcur(pCur + (ic-1)*6 + 3) = Angle
c                                   watcur(pCur + (ic-1)*6 + 4) = Temperature
c                                   watcur(pCur + (ic-1)*6 + 5) = Spare
c                                   watcur(pCur + (ic-1)*6 + 6) = Spare
c     nw       (dp,sc,in)        - number of wave components
c     pWav     (int,sc,in)       - pointer to wave table (= 30 at 12.0)
c     watwav   (dp,ar(*),in)     - water wave table
c                                   watwav( 6) = nw = number of wave components
c                                   watwav(11) = KWAVE  (kwav)
c                                   watwav(12) = THETA
c                                   watwav(13) = WAVLOC (kpeak)
c                                   watwav(14) = KCRC
c                                   watwav(15) = KMF
c                                   watwav(16) = PRKEY
c                                          iw = wave number
c                                   watwav(pWav + (iw-1)*6 + 1) = Wave Height
c                                   watwav(pWav + (iw-1)*6 + 2) = Period
c                                   watwav(pWav + (iw-1)*6 + 3) = Phase Shift
c                                   watwav(pWav + (iw-1)*6 + 4) = Wave Length
c                                   watwav(pWav + (iw-1)*6 + 5) = Spare
c                                   watwav(pWav + (iw-1)*6 + 6) = Spare
c     timval   (dp,sc,in)        - current time value
c
c  output arguments:
c       While the below 7 arguments are output, they can also
c       be used as input, based on other ANSYS input.
c     argu     (dp,sc,out)       - position in wave (radians) (passed out only for output)
c     eta      (dp,sc,out)       - total wave height
c     vxyz     (dp,ar(3),out)    - particle velocities
c     axyz     (dp,ar(3),out)    - particle accelerations
c     ar       (dp,sc,out)       - radial particle acceleration
c     pdynam   (dp,sc,out)       - dynamic pressure head
c
c  local variable
```

```
c     phead    (dp,sc,out)       - pressure head
c
```

2.6.11.1. Subroutine userPartVelAccSetup (Initializing Data for Use by the userPartVelAcc Subroutine)

This subroutine initializes the data for the `userPartVelAcc` (p. 234) subroutine.

```
*deck,userPartVelAccSetup                      USERSDISTRIB
      subroutine userPartVelAccSetup ( kch,ptr_Ocean,
     x                                 nsize,nsizec,nsizew,
     x                                 dWork,dWorkC,dWorkW,
     x                                 rkd,wvmax )
c        ---- accessed only if kwav .ge. 101 -----
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 Copyright ANSYS.  All Rights Reserved.
c
c
c  input arguments:
c     kch      (int,sc,in)       - key for checking or defaulting (not used by PIPE288)
c     ptr_Ocean (int,sc,in)      - storage offset
c     nsize    (int,sc,in)       - size of ocean basic data
c     nsizec   (int,sc,in)       - size of ocean current data
c     nsizew   (int,sc,in)       - size of ocean wave data
c     dWork    (dp,ar(*),inout)  - raw ocean basic data (dWork = watbas)
c                                    watbas( 6) = nReN = number of Reynold's numbers
c                                    watbas(11) = DEPTH
c                                    watbas(12) = MATOC
c                                    watbas(13) = KFLOOD
c                                    watbas(14) = Ci
c                                    watbas(15) = Cb
c                                          pBas = 30 (at Rev 12.0) (to be added to argument list)
c                                            ir = Reynold's number number
c                                    watbas(pBas + (ir-1)*9 + 1) = RE
c                                    watbas(pBas + (ir-1)*9 + 2) = CDy
c                                    watbas(pBas + (ir-1)*9 + 3) = CDz
c                                    watbas(pBas + (ir-1)*9 + 4) = CT
c                                    watbas(pBas + (ir-1)*9 + 5) = CMy
c                                    watbas(pBas + (ir-1)*9 + 6) = CMz
c     dWorkC   (dp,ar(*),inout)  - raw ocean current data (dWorkC = watcur)
c     dWorkW   (dp,ar(*),inout)  - raw ocean wave    data (dworkW = watwav)
c                                     - see userPartVelAcc.F for details for watcur and watwav
c
c  output arguments:
c     dWork    (dp,ar(*),inout)  - adjusted ocean basic data
c     dWorkC   (dp,ar(*),inout)  - adjusted ocean current data
c     dWorkW   (dp,ar(*),inout)  - adjusted ocean wave data
c     rkd      (dp,sc,out)       - value of k*d
c     wvmax    (dp,sc,out)       - total wave height
```

2.6.11.2. Subroutine userWavHt

The `userWavHt` subroutine calculates the wave height of a user-defined wave for the `userPartVelAcc` (p. 234) subroutine.

```
*deck,userWavHt                        USERSDISTRIB
      subroutine userWavHt (xyzg,doext,depth,nw,pWav,watwav,timval,
     &                      eta,etadot)
c        ---- accessed only if kwave .ge. 101 -----
c     *** primary function: calculate wave height for user wave
c     ***                        over point at xyzg of the element
c     *** secondary functions:  none
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
c
c *** Notice - This file contains ANSYS Confidential information ***
c Copyright ANSYS.  All Rights Reserved.
c
c  input arguments:
c     xyzg     (dp,ar(3),in)       - updated coordinates of point of interest in
c     doext    (dp,sc,in)          - outside diameter with insulation
c                                    if timval<0.0, argu = doext
c     depth    (dp,sc,in)          - water depth
c     nw       (int,sc,in)         - number of waves
c     pWav     (int,sc,in)         - pointer to wave table
c     watwav   (dp,ar(*),in)       - water wave table
c     timval   (dp,sc,in)          - current time value
c                                    if timval < 0.0
c                                       pass directly in doext position
c                                       (used for stream function only)
c                                    else compute value in wvargu
c
c  output arguments:
c     eta      (dp,sc,out)         - wave height
c     etadot   (dp,sc,out)         - time derivative of wave height
c
```

2.6.11.3. 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` (p. 234) subroutine.

```
*deck,wvhybl
      function wvhybl (kclass,x,y)
c *** primary function: to compute the ratio of two hyperbolic functions,
c                        specialized to the needs of wave loading.
c                        The options are as given with kclass below.
c                        Further, only positive values of x and y are used
c
c *** secondary functions: none
c
c *** Notice - This file contains ANSYS Confidential information ***
c
c  input arguments:
c     variable (typ,siz,intent)    description
c     kclass   (int,sc,in)       - 0 - cosh(x)/cosh(y)
c                                  - 1 - sinh(x)/cosh(y)
c                                  - 2 - cosh(x)/sinh(y)
c                                  - 3 - sinh(x)/sinh(y)
c     x        (dp,sc,in)        - argument of numerator
c     y        (dp,sc,in)        - argument of denominator
c
c  output arguments:
c     variable (typ,siz,intent)    description
c     wvhybl   (dp,sc,out)       - resulting fraction
c
```

2.6.11.4. Subroutine wvargu

The `wvargu` subroutine computes the appropriate position with regard to the wave. It is a utility subroutine called by the `userPartVelAcc` (p. 234) subroutine.

```
*deck,wvargu
      function wvargu (kpeak,kmf,wavdat,timval,r,doext)
c     *** primary function: to find appropriate position wrt wave
c     *** secondary functions:  none
c
c *** Notice - This file contains ANSYS Confidential information ***
c
```

```
c  input arguments:
c     kpeak    (int,sc,in)       - keyopt for when peak effect occurs
c     kmf      (int,sc,in)       - key for maccamy-fuchs adjustment
c     wavdat   (dp,ar(6),in)     - wave data (from water wave table)
c                                       wavdat(1) = wave height(not used)
c                                       wavdat(2) = period
c                                       wavdat(3) = phase shift
c                                       wavdat(4) = wave length
c     timval   (dp,sc,in)        - current time value
c     r        (dp,sc,in)        - radial location of point of interest
c     doext    (dp,sc,in)        - effective outside diameter of pipe
c
c  output arguments:
c     wvargu   (dp,sc,out)       - wave position(as determined by the argument)
c                                  output in radians
c
```

2.6.12. Subroutine userPanelHydFor (Calculating Panel Loads Caused by Ocean Loading)

The `userPanelHydFor` subroutine applies loads and other effects onto SURF154 surface elements. This capability is accessed via KEYOPT(8) of SURF154, together with data read in via the `userOceanRead (p. 238)` subroutine.

```
*deck,userPanelHydFor                      USERSDISTRIB
      subroutine userPanelHydFor (kPOcean, elemId, intPnt,
     x depth, denswat,
     x ncm, pCur, watcur,
     x nw , pWav, watwav,
     x xyzupp, vn,
     x presoc,admsoc)

c        ---- accessed only if kwave on the OCDATA command .ge. 101 -----
c primary function:    Get pressure loading on panel
c secondary functions: Get hydrodynamic mass on panel
c       load is applied on SURF154 with keyopt(8)

c *** Notice - This file contains ANSYS Confidential information ***
c Copyright ANSYS.  All Rights Reserved.

c  parameter definition include files:
```

2.6.12.1. Subroutine userOceanRead

The `userOceanRead` subroutine reads in ocean data to be used by the `userPanelHydFor (p. 238)` subroutine.

```
*deck,userOceanRead                      USERSDISTRIB

      subroutine userOceanRead (iott,kpr,fUnitNo,iOption,
     x                          pdWaveData,lenWavDat)

c        ---- accessed only if kwave on the OCDATA command .ge. 101 -----
c Primary Function: read in ocean file for later use
c Secondary Functions:
c
c -----------------
c Notice:
c ------
C    This routine contains ANSYS, Inc. confidential information
c Copyright ANSYS.  All Rights Reserved.
c -----------------
c  input arguments:
c     iott         (int,sc,in)     output unit number, based on then /OUT command
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
c     kpr          (log,sc,in)     print flag, based on the /NOPR command
c     fUnitNo      (int,sc,in)     file unit number, based on the command
c                                  OCREAD,file,ext,dir
c     iOption      (int,sc,in)     integer from the command line, based on
c                                  OCREAD,file,ext,dir,iOption
c     pdWaveData   (ptr,sc,out)    pointer to wave data array
c     lenWavDat    (int,sc,out)    length of wave data
c                                  0 = an error, no wave data is stored
```

2.7. Running Mechanical APDL as a Subroutine

To call the Mechanical APDL program, use the following:

```
program 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 cannot use the database access subroutines; however, other user-programmable features can use the database access subroutines freely.

There may be times when Mechanical APDL exits abnormally. Check the `file.err` file to see if Mechanical APDL wrote an exit code to the file before ending. These error codes may help you to understand what caused the abnormal program exit:

Table 2.1: 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
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		

2.8. Defining Your Own Commands

ANSYS, Inc. provides a set of user subroutines, named `user01` through `user10`, which you can use to define custom 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 program.

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

3. Issue the 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 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 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.

2.8.1. Function user01

```
*deck,user01                      USERDISTRIB
      function  user01()
c *** primary function:    user routine number  01

c         *** Copyright ANSYS.  All Rights Reserved.
c         *** ansys, inc.
c *** Notice - This file contains ANSYS Confidential information ***

c  /*****************************************************************\
c  | this is a user routine that may be used by users to include their |
c  | special coding.  accesss to this routine is by the command usr1.  |
c  | usr1 may be changed by the user using the command /ucmd.  the     |
c  | user may then use this routine to call his/her special routines.  |
c  | ansys routines to access information in the ansys database may be |
c  | found in the "ansys programmer's manual", available from ansys,inc|
c  | see user02 for a simple example usage.                            |
c  | routines user03 to user10 are also available.                     |
c  \*****************************************************************/

c  input arguments:  none

c  output arguments:
c     user01   (int,sc,out)      - result code (should be zero)
c                                   (which is ignored for now)

c    ************************************************************
c    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
c    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
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
c    ************************************************************
c
#include "impcom.inc"
#include "ansysdef.inc"

      external  wrinqr
      integer   wrinqr

      integer  user01, iott

      iott = wrinqr(2)

c          *****  USER'S CODE IS INSERTED HERE *****
      write (iott,2000)
 2000 format (//' *****  CALL TO ANSYS,INC DUMMY USER01  *****'//)

c          *****  do not return this result code in a real user routine
      user01 = -654321
c         *****  instead return a zero   *****
c     user01 = 0

      return
      end
```

2.8.2. Function user02 (Demonstrates Offsetting Selected Nodes)

```
*deck,user02                      USERDISTRIB
      function  user02()
c *** primary function:    user routine number  02
c      --- This demonstration offsets selected nodes with the command:
c              usr2,dx,dy,dz

c         *** Copyright ANSYS.  All Rights Reserved.
c         *** ansys, inc.
c *** Notice - This file contains ANSYS Confidential information ***

c  /*****************************************************************\
c  | see user01 for additional information on user routines          |
c  \*****************************************************************/

c  input arguments:  none

c  output arguments:
c     user02   (int,sc,out)      - result code (should be zero)
c                                  (which is ignored for now)

c    ************************************************************
c    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
c    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
c    ************************************************************
c
#include "impcom.inc"
#include "ansysdef.inc"

      external  TrackBegin, TrackEnd
      external  wrinqr,ndinqr,ndgxyz,ndpxyz,erhandler, dpinfun
      integer   wrinqr,ndinqr,ndgxyz
      double precision  dpinfun

      integer  user02, iott, maxnp, i ,ksel
      double precision  xyz(3), offset(3)


c     *****  start timing check  *****
      call TrackBegin ('user02')
```

```
      maxnp = ndinqr(0,DB_MAXDEFINED)

c          *****  get the desired offsets from the command line  *****
      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

c          *****  write to output file  *****
      iott = wrinqr(WR_OUTPUT)
      write (iott,2000)
 2000 format (/'  NODE OFFSET COMPLETE  '/)

c         *****  write to GUI window  *****
      call erhandler ('user02',3000,
     x             2,'NODE OFFSET COMPLETE',0.0d0,' ')

c          *****  required return value  *****
      user02 = 0

c          *****  end timing check  *****
      call TrackEnd ('user02')
      return
      end
```

2.8.3. Function user03 (Demonstrates Using Memory)

```
*deck,user03                       USERDISTRIB
      function  user03()
c *** primary function:    user routine number 03. Gives example of
c                          ANSYS Memory usage

c         *** Copyright ANSYS.  All Rights Reserved.
c         *** ansys, inc.
c *** Notice - This file contains ANSYS Confidential information ***

c  /*****************************************************************\
c  | see user01 for additional information on user routines          |
c  \*****************************************************************/

c  input arguments:  none

c  output arguments:
c     user03   (int,sc,out)      - result code (should be zero)
c                                  (which is ignored for now)

c    ***********************************************************
c    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
c    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
c    ***********************************************************

#include "impcom.inc"
#include "ansysdef.inc"
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
      external  TrackBegin, TrackEnd
      external  wrinqr, ndinqr, ndgxyz, ndnext, fAnsMemAlloc,
     x          fAnsMemFree,erhandler, parreturn, parstatus
      integer   wrinqr, ndinqr, ndgxyz, ndnext
      PTRFTN    fAnsMemAlloc

      integer   user03, iott, i, ksel, numnp, node, istat
      double precision  xyz(3), xmean, ymean, zmean, stdxyz(3),
     x          sodx, sody, sodz

c  pointers:
      pointer (pdXnodeL,Xnode)
      pointer (pdYnodeL,Ynode)
      pointer (pdZnodeL,Znode)
      double precision  Xnode(*), Ynode(*), Znode(*)


c       *****  call to start timing  *****
      call TrackBegin ('user03')

c     Get nodal xyz locations and calculate standard deviation of
c     x coordinates, y coordinates, & z coordinates


c     get number of currently selected nodes
      numnp = ndinqr(0,DB_NUMSELECTED)

      istat = 1
      if (numnp .le. 0) go to 999

c     allocate memory for x, y, & z coordinates of nodes
      pdXnodeL = fAnsMemAlloc(numnp,MEM_DOUBLE,'XCoords ')
      pdYnodeL = fAnsMemAlloc(numnp,MEM_DOUBLE,'YCoords ')
      pdZnodeL = fAnsMemAlloc(numnp,MEM_DOUBLE,'ZCoords ')

c     loop through all selected nodes
      i = 1
      node = 0
      xmean = 0.0d0
      ymean = 0.0d0
      zmean = 0.0d0

 10   node = ndnext(node)

      if (node .gt. 0) then

c         get xyz coordinates
          ksel = ndgxyz(node,xyz(1))

c         store this node's xyz coordinates
          Xnode(i) = xyz(1)
          Ynode(i) = xyz(2)
          Znode(i) = xyz(3)

c         while we're looping, accumulate sums to calculate means
          xmean = xmean + xyz(1)
          ymean = ymean + xyz(2)
          zmean = zmean + xyz(3)

c         increment index
          i = i + 1

c         loop back up for next selected node
          goto 10

      endif

c     node = 0, at the end of node list

c     calculate mean of xyz coordinates
      xmean = xmean / numnp
      ymean = ymean / numnp
```

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
      zmean = zmean / numnp

c     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))

c     *****  write to output file  *****
      iott = wrinqr(WR_OUTPUT)
      write (iott,2000) xmean,ymean,zmean,
     x                  stdxyz(1),stdxyz(2),stdxyz(3)
 2000 format (/' MEAN FOR X COORDINATES:',G12.5/
     x         ' MEAN FOR Y COORDINATES:',G12.5/
     x         ' MEAN FOR Z COORDINATES:',G12.5/
     x         ' STD  FOR X COORDINATES:',G12.5/
     x         ' STD  FOR Y COORDINATES:',G12.5/
     x         ' STD  FOR Z COORDINATES:',G12.5)

c     *****  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),' ')

c      *****  set _STATUS to 0 for success  *****
      istat = 0

c     release dynamically allocated memory
      call fAnsMemFree (pdZnodeL)
      call fAnsMemFree (pdYnodeL)
      call fAnsMemFree (pdXnodeL)

c     *****  required return value  *****
 999  user03 = 0

c     *****  set _RETURN  to number of nodes processed  *****
      call parreturn (dble(numnp))

c     *****  set _STATUS for success (0) or no nodes (1)  *****
      call parstatus (istat)

c     *****  call to end timing  *****
      call TrackEnd ('user03')
      return
      end
```

2.8.4. Function user04

```
*deck,user04                      USERDISTRIB
      function  user04()
c *** primary function:   user routine number  04; demonstrates gettting a
c                         list of nodes attached to a keypoint, line, or area

c         *** Copyright ANSYS.  All Rights Reserved.
c         *** ansys, inc.
c *** Notice - This file contains ANSYS Confidential information ***

c  /*****************************************************************\
c  | see user01 for additional information on user routines          |
c  \*****************************************************************/
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
c  input arguments:  none

c  output arguments:
c     user04   (int,sc,out)      - result code (should be zero)
c                                   (which is ignored for now)

c    ************************************************************
c    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
c    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
c    ************************************************************
c
#include "impcom.inc"
#include "ansysdef.inc"


      external  ndkpnt

      external  wrinqr, ndline, ndarea, intinfun
      integer   wrinqr, ndline, ndarea, intinfun
      external     ch4infun
      character*4  ch4infun

      integer  user04,   iott, listk(20),listl(20),lista(20), listin(1),
     x           i, num,ktype, nkpnts, nlines, nareas
      character*4  type, lab2


      iott = wrinqr (WR_OUTPUT)

c     --- setup with:  /UCMD,GNSME,4
c        !gnsme,group,num,type
c        ! group = kp, ln, or ar
c        ! num   = entity number of kp, ln, or ar
c        ! type = interior, or all
c

      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)
      listin(1) = num

      if (lab2 .ne. 'KP  ' ) then
         type = ch4infun(4)
         if (type .eq. 'INTE') then
            write (iott,2030)
 2030       format (' interior nodes only ')
            ktype = 0
         elseif (type .eq. 'ALL ') then
            write (iott,2040)
 2040       format (' all (interior and edge/end) nodes ')
            ktype = 1
         else
            write (iott,2050)
 2050       format ('Only INTE or ALL are acceptable in last field',
     x       ' on user-written gnsme command')
         endif
      endif

      if (lab2 .eq. 'KP  ' ) then
         nkpnts = 0
         call ndkpnt (1,listin(1),nkpnts,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)
 2125    format (' list of nodes on line'/(3x,i4))

      elseif (lab2 .eq. 'AR  ' ) then
         nareas = ndarea (num,ktype,lista(1))
         write (iott,2130)  nareas
 2130    format (' number of nodes on area = ',i4)
         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 ',
     x    'gnsme command')
      endif

      user04 = 0

      return
      end
```

2.8.5. Functions user05 through user10

The source code for user subroutines `user05`, `user06`, `user07`, `user08`, `user09`, and `user10` is identical to function `user01` shown above.

2.9. Supporting Subroutines

The following routines may be used for general applications.

2.9.1. 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.
c                       warning: This routine is called after the elements
c                       are formed, but before solution.  Therefore,
c                       F is from the current iteration, but
c                       u is from the previous iteration.  At convergence,
c                       this difference will have little effect.
c                       The computations are done immediately after the
c                       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                        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 ***
c     Prolog is not CONFIDENTIAL INFORMATION

c  input arguments:
c     variable (typ,siz,intent)    description
c     Node     (int,sc,in)       - Node Number (User)
c     Label    (ch*4,sc,in)      - DOF Label (Upper Case)
c                                  'UX  ','UY  ','TEMP','VOLT','ROTY', etc

c  output arguments:
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
c     GetRForce (int,func,out)   - status/pointer
c                                  = 0 - data not valid
c                                  > 0 - Rfsum pointer to data for fast access
c                                        see comments below
c     Value    (dp,sc,out)       - Solution value for Node,Label
c                                   All results are in the nodal coordinate
c                                   system

c  example usage:

c       external  GetRForce
c       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...
c       Value = Rfsum(ptr)
```

2.9.2. 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 ***
c     Prolog is not CONFIDENTIAL INFORMATION

c     typ=int,dp,log,chr,dcp   siz=sc,ar(n)   intent=in,out,inout

c  input arguments:
c     variable (typ,siz,intent)    description
c     Node     (int,sc,in)       - Node Number (User)
c     Label    (ch*4,sc,in)      - DOF Label (Upper Case)
c                                  'UX  ','UY  ','TEMP','VOLT','ROTY', etc

c  output arguments:
c     variable (typ,siz,intent)    description
c     GetStackDisp (int,sc,out)  - status/pointer
c                                  = 0 - data not valid
c                                  > 0 - UDisp pointer to data for fast access
c                                        see comments below
c     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)
```

2.9.3. 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 ***
c               (prolog is not confidential)

c  input arguments:
c     Label    (ch*4,sc,in)          - Result Type
c     Comp     (ch*4,sc,in)          - Result Component (8 char for ESTR)
c     LabAvg   (ch*4,sc,in)          - 'AVG ' or 'NOAV' ('AVG ' default)

c  output arguments:
c     TypeData (int,sc,out)          - Code for data type
c     nVal     (int,sc,out)          - Number of values per point
c                                      If 0, no data
c     iLoc      (int,sc,out)          - Location of Comp in values
```

2.9.4. Subroutine ElResultGet (Getting Results Values at Selected Points)

```
*deck,ElResultGet
      subroutine ElResultGet (nPoints,ebest,elcord,TypeData,iLoc,
     x                        nVal,result)
c *** primary function:    (post1) Get results at selected points

c *** Notice - This file contains ANSYS Confidential information ***
c               (prolog is not confidential)

c  input arguments:
c     nPoints  (int,sc,in)           - Number of evaluation points
c                    *** from ElInterp ***
c     ebest    (int,ar(nPoints),in)  - Element(s) containing points
c     elcord   (dp,ar(3,nPoints),in) - Element coordinates
c                    *** from ElResultStrt ***
c     TypeData (int,sc,in)           - Data type code
c     iLoc      (int,sc,in)           - Start of selected data
c     nVal     (int,sc,in)           - Number of results per point

c  output arguments:
c     Result   (dp,ar(nvar,nPoints),out)  - Array of results
```

2.9.5. Subroutine ElInterp (Finding Element Coordinates)

```
*deck,ElInterp
      subroutine ElInterp (piFEML,nPoints,xyzPoints,tolInsidein,
     x                     tolOutsidein,MoveTol,ebest,elcord)

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 ***
c     (Prolog is not CONFIDENTIAL INFORMATION)

c input arguments:
c   piFEML   (ptr,sc,in)           - If non 0, pointer of a FEM Object
c   nPoints  (int,sc,in)           - Number of points to find (do in one group)
c   xyzPoints(dp,ar(3,nPoints),in)- XYZ coordinates of each point
c   tolInsidein(dp,sc,in)          - Tolerance for point inside element
c                                      (0.0d0 defaults to 1.0d-4)
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries and affiliates.

```
c   tolOutsidein(dp,sc,in)        - Maximum distance outside to be associated
c                                     with an element (0.0d0 defaults to 0.25)
c   MoveTol   (dp,sc,in)          - Node move tolerance (0.0d0, no move)

c output arguments:
c   ebest    (int,ar(nPoints),out)  - Best element number for each point
c   elcord   (dp,ar(3,nPoints),out) - Element coordinates of the point
```

2.10. Access at the Beginning and End of Various Operations

You can access the logic just before a run begins or just after a run ends, and at many other intermediate points, by using the 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 [1]	At 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 a run

1. The UAnBeg subroutine that allows user access at the start of a run does not require activation by the **USRCAL** command; it is automatically activated when the program is started.

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 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 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 the program, as described in Understanding User Programmable Features (UPFs) (p. 119).

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 a 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,
```

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
/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
array(1) = 1
array(2) = 2
array(3) = 3
array(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
*msg,note
THIS IS A TEST MESSAGE
*vwrite,array(1)
(/ b = ,f10.4)
```

2.11. Memory Management Subroutines

ANSYS, Inc. provides UPF subroutines that you can use for memory management.

2.11.1. Using the Memory-Management Subroutines

The program 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 the program 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:

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
piArray = fAnsMemAlloc (size,...)
```

To use the 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:

   - *x*leng 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 Memory) (p. 242), which appears earlier in this chapter.

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

2.11.2. 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 ***

c  input arguments:
c     iLen (int,sc,in)           - length of the block (in data elements)
c     c16Label (chr*16,sc,in)    - 16 character name for the Block
c     key (int,sc,in)            - type of data for this block (see ansysdef)

c  output arguments:
c     fAnsMemAlloc (PTRFTN,sc,out)  - Pointer to this data block -- needs to be
c                                     tied to a local variable in the calling
c                                     routine
```

2.11.3. Subroutine fAnsMemFree (Deallocating Space)

```
*deck,fAnsMemFree
      subroutine fAnsMemFree (memPtr)
c primary function:   Free a Data Block, given a pointer

c keywords:  subroutine to free a mem block

c object/library:  mem

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

c  input arguments:
c     ptr   (PTRFTN,sc,inout)  - pointer for this block

c  output arguments:
c     ptr   (PTRFTN,sc,inout)  - pointer will be set to zero
```

2.12. Parameter-Processing Subroutines

The product distribution medium contains three subroutines that you can use for parameter processing: `pardim`, `parevl`, and `pardef`.

2.12.1. Subroutine pardim (Creating a Dimensioned Parameter)

```
*deck,pardim
      subroutine pardim (cName,labl4,nDim,nxyz,cLabels)
c *** primary function:    create a dimensioned parameter

c       *dim,parm32,type,d1,d2,d3,cName1,cName2,cName3
c       *dim,parm32,type,d1,cName1
c       *dim,parm32,type,d1,d2,d3,d4,d5,cName1,cName2,cName3,cName4,cName5

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

c  input arguments:
c     cName    (chr*32,sc,in)    - the name of the parameter to create
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
c     labl4    (chr*4,sc,in)     - 'TABL' or 'ARRA' or 'CHAR' or 'STRI'
c     nDim     (int,sc,in)       - Dimension of array
c     nxyz     (int,ar(nDim),in) - the dimensions of the array
c     cLabels  (chr*32,ar(nDim),in) - Names for the directions in table

c  output arguments:  none
```

2.12.2. 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  input arguments:
c     ParName  (chr*(PARMSIZE),sc,in) - the name of the parameter
c                                          (must be upper case, left justified)
c     nDim     (int,sc,in)              - the number of subscripts (0,scaler)
c     subc     (dp,ar(*),in)            - values for the subscripts (if any)
c     lvl      (int,sc,in)                 - 0,1  no error output  2, report error
c                                           -1, set kerr flag with no anserr call

c  output arguments:
c     dpValue  (dp,sc,out)           - the value of the parameter (may be a
c                                        packed character*8
c     chValue  (chr*(STRING_MAX_LENG),sc,out) - character output
c     kerr     (int,sc,out)              - error flag  (0,ok  -1,output is packed
c                                         0=ok,  1=error,  2=error but TINY is used
c                                         -2, output is string in chValue
```

2.12.3. Subroutine pardef (Adding a Parameter)

```
*deck,pardef
      subroutine pardef (cNameIn,ctype,nval,subc,valuein,kerr,string)
c *** primary function:    add a parameter to parameter list
c
c *** Notice - This file contains ANSYS Confidential information ***
c
c  input arguments:
c     cNameIn (chr*(PARMSIZE),sc,in) - name of parameter
c                              cNameIn is a character variable that
c                              contains the name of the parameter that
c                              is to be defined. (Length = PARMSIZE characters)
c
c     ctype  (int,sc,in)   - 0, dp   1,character  2,string
c                              ctype is an integer key which describes
c                              the type of data that the parameter data
c                              holds.  This would also indicate the
c                              contents of "value" (arg 5).
c                              0=double precision data
c                              1=character data packed in value
c                              2=character data in string
c
c     nval   (int,sc,in)   - number of subscripts
c                              nval is the number of subscripts that the
c                              "cNameIn" (arg 1) contains.
c                              1=single dimensioned variable (ex. x(10))
c                              2=double dimensioned variable (ex. y(10,3))
c                              3=triple dimensioned variable (ex. z(10,3,2))
c                              -1=delete this parameter from the internal
c                              tables.
c
c     subc (dp,ar(*),in)   - values of subscripts
c                              subc is a double precision vector that
c                              contains the subscripts of "cNameIn" (arg 1).
c                              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
c                               to set "x(5,1,1)=123.0", then "nval" (arg 3)
c                               should be set to 3, and "subc" should be set
c                               to 5.0, 1.0, 1.0, and "value" (arg 5) should
c                               be 123.0.  Another example is if "y" was
c                               dimensioned to as "y(20,20)" and you were
c                               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
c                               proper dimensions were set by a "*dim" command.
c
c                               Please note that although the values of "subc"
c                               should be double precision, subroutine "pardef"
c                               uses the intrinsic "nint" (nearest integer)
c                               function to get and use the integer equivalent.
c
c                               You should also note the "nval" (arg 3) and
c                             "subc" (arg 4) must fall within the range that was
c                               set with a "*dim" or "*set" command or an error
c                               will occur.
c
c     valuein(dp,sc,in)     - the value for this parameter
c                               (should be a packed character*8 if
c                               ctype=1. To pack a char into a dp
c                               variable use "call chtodp(ch8,dp)".
c                               To unpack a dp variable into a char
c                               use "call dptoch(dp,ch8)" )
c                               Value is the data value that is to be stored for
c                               "cNameIn" (arg 1). If "ctype=1" (arg 2) then this
c                               value would be a "packed character" data from the
c                               "chtodp" Ansys function.
c
c  output arguments:
c     kerr    (int,sc,out)  - error flag (0=ok, 1=error)
c                               kerr is an integer error flag that is
c                               returned to the calling subroutine.  Any
c                               non zero number would indicate an error
c                               was detected in subroutine "pardef"
c
c *** mpg pardef < parstore pardim ntableget rdsset<rdmac<rdcmd: define param
c
```

2.13. Other Useful Functions

The program has several miscellaneous functions that you may find useful for working with UPFs:

- The `erhandler` subroutine displays output messages (notes, warnings, and errors).
- The `RunCommand` function lets you issue a 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.
- The **/TRACK** command enables you to do program tracing and timing.

For further descriptions of `erhandler` and **/TRACK**, see Subroutines for Your Convenience (p. 327). For details about the `GetStackDisp` function, see Function GetStackDisp (Getting Current Displacement Values) (p. 247).

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

2.13.1. Using Function RunCommand

This function enables you to issue a 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

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

c  input arguments:
c     nChar    (int,sc,in)           - Length of the command string (8 min)
c     command  (ch*(nChar),sc,in)    - A character string containing a
c                                              valid ANSYS command

c  output arguments:
c     RunCommand   (int,sc,out)      - An internally defined value, ignore
```

2.13.2. 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 the program ask permission before writing the file
STATUS, to restore the file as it existed after executing the last command issued before the **/UNDO** command.

2.13.3. Using the /HOLD command

Issue the **/HOLD** command to synchronize tasks. The 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 program reads a command from the file and then deletes the file.

TimeInterval
The length of time, in seconds, that the program waits before trying to read the message file again.

Timeout
The maximum length of time, in seconds, that the program can wait between attempts to read the file.

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries and affiliates.

Chapter 3: Accessing the Mechanical APDL Database

This chapter describes how you can retrieve information in the Mechanical APDL database (or store information in the database) by linking subroutines you create into the Mechanical APDL program.

You can use the database access routines with any of the user-programmable features. For example, you can create your own Mechanical APDL 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 Mechanical APDL database. The function and subroutine descriptions are grouped into the following sections.

3.1. Routines for Selecting and Retrieving Nodes and Elements
3.2. Node Information Routines
3.3. Element Attribute Routines
3.4. Coupling and Constraint Routines
3.5. Nodal Loading Routines
3.6. Element Loading Routines
3.7. Results Information Routines

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

3.1. Routines for Selecting and Retrieving Nodes and Elements

3.1.1. 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 ***

c     input arguments:
c        next     (int,sc,in)       - the last node number used
c                                     = 0 - use for initial value

c     output arguments:
c        ndnext   (int,func,out)    - the next selected node number
c                                     = 0 - no more nodes
```

3.1.2. 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
c *** Notice - This file contains ANSYS Confidential information ***
c
c     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
c        next     (int,sc,in)       - the next node number used
c                                     = 0 - use for initial value
c
c     output arguments:
c        ndprev   (int,func,out)    - the previous selected node number
c                                     = 0 - no more nodes
c
```

3.1.3. 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 ***
c
c     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
c        next     (int,sc,in)       - the last node number used
c                                     = 0 - use for initial value
c
c     output arguments:
c        ndnxdf   (int,func,out)    - the next defined node number
c                                     = 0 - no more nodes
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

3.1.4. 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 ***

c     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        ndmi     (int,sc,in)       - node number
c                                       = 0 - all nodes
c                                       < 0 - do not delete CPs and CEQNs
c                                             (merge/offset/compress)
c        ksel     (int,sc,in)       - type of operation to be performed.
c                                       ksel = 0 - delete node.
c                                            = 1 - select node.
c                                            =-1 - unselect node.
c                                            = 2 - invert select status of node.

c     output arguments:
c        none.
```

3.1.5. 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 ***

c     input arguments:
c        next     (int,sc,in)       - the last element number used
c                                       = 0 - use for initial value

c     output arguments:
c        elnext   (int,func,out)    - the next selected element
c                                       = 0 - no more elements
```

3.1.6. 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
c
c *** Notice - This file contains ANSYS Confidential information ***
c
c     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
c        prev     (int,sc,in)       - the last element used
c                                       = 0 - use for initial value
c
c     output arguments:
c        elprev   (int,func,out)    - the previous selected element
c                                       = 0 - no more elements
c
```

3.1.7. 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
c *** Notice - This file contains ANSYS Confidential information ***
c
c     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
c        next     (int,sc,in)       - the last element used
c                                     = 0 - use for initial value
c
c     output arguments:
c        elnxdf   (int,func,out)    - the next defined element
c                                     = 0 - no more elements
```

3.1.8. 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 ***
```

3.2. Node Information Routines

3.2.1. ndinqr Function (Getting Information About a Node)

The primary function of `ndinqr` 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.

```
*deck,ndinqr
      function ndinqr (node,key)
c *** primary function:    get information about a node.
c *** secondary functions: set current node pointer to this node.

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

c     input arguments:
c        node     (int,sc,in)       - node number
c                                      Should be 0 for key=11, DB_NUMDEFINED,
c                                      DB_NUMSELECTED, DB_MAXDEFINED, and
c                                      DB_MAXRECLENG
c        key      (int,sc,in)       - key as to information needed about
c                                      the node.
c                 = DB_SELECTED    - return select status:
c                     ndinqr  = 0 - node is undefined.
c                             =-1 - node is unselected.
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
c                            = 1 - node is selected.
c                 = DB_NUMDEFINED  - return number of defined nodes
c                 = DB_NUMSELECTED - return number of selected nodes
c                 = DB_MAXDEFINED  - return highest node number defined
c                 = DB_MAXRECLENG  - return maximum record length (dp words)
c                 =  2, return length (dp words)
c                 =  3,
c                 =  4, pointer to first data word
c                 = 11, return void percent (integer)
c                 = 17, pointer to start of index
c                 = -1,
c                 = -2, superelement flag
c                 = -3, master dof bit pattern
c                 = -4, active dof bit pattern
c                 = -5, solid model attachment
c                 = -6, pack nodal line parametric value
c                 = -7, constraint bit pattern
c                 = -8, force bit pattern
c                 = -9, body force bit pattern
c                 = -10, internal node flag
c                 = -11, orientation node flag =1 is =0 isnot
c                 = -11, contact node flag <0
c                 = -12, constraint bit pattern (for DSYM)
c                 = -13, if dof constraint written to file.k (for LSDYNA only)
c                 = -14, nodal coordinate system number (set by NROTATE)
c                 =-101, pointer to node data record
c                 =-102, pointer to angle record
c                 =-103,
c                 =-104, pointer to attached couplings
c                 =-105, pointer to attacted constraint equations
c                 =-106, pointer to nodal stresses
c                 =-107, pointer to specified disp'S
c                 =-108, pointer to specified forces
c                 =-109, pointer to x/y/z record
c                 =-110,
c                 =-111,
c                 =-112, pointer to nodal temperatures
c                 =-113, pointer to nodal heat generations
c                 =-114,
c                 =-115, pointer to calculated displacements
c                 =-116,

c     output arguments:
c        ndinqr   (int,func,out)   - the returned value of ndinqr is based on
c                                         setting of key.
```

3.2.2. getnod Function (Getting a Nodal Point)

```
*deck,getnod
      subroutine getnod (node,v,kerr,kcrot)
c *** primary function:    get a nodal point

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

c  input arguments:
c     node     (int,sc,in)       - node number
c     kerr     (int,sc,inout)    - message flag
c                                    = 0 - print no message if node is unselected
c                                           or undefined
c                                    = 1 - print message if node is undefined
c                                    = 2 - print message if node is undefined
c                                           or unselected
c     kcrot    (int,sc,in)       - output coordinates in this coordinate system.
c                                      if kcrot is negative, output theta and
c                                      phi coordinates in radians

c  output arguments:
c     v        (dp,ar(6),out)    - Coordinates (first 3 values) and rotation
```

```
c                                    angles (last 3 values)
c     kerr     (int,sc,inout)    - select status
c                                  = 0 - node is selected
c                                  = 1 - node is not defined
c                                  =-1 - node is unselected
```

3.2.3. 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 ***

c     input arguments:
c        node     (int,sc,in)       - node number
c        vctn     (dp,ar(6),in)     - array of 3 nodal coordinates and
c                                              3 nodal rotation angles.
c        kcrot    (int,sc,in)       - local coordinate system in which the nodal
c                                     coordinates and angles are defined

c     output arguments:  none.
```

3.2.4. 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 ***

c     input arguments:
c        node     (int,sc,in)       - node number for operation.

c     output arguments:
c        ndgall   (int,sc,out)      - status of node.
c                                       0=node is undefined.
c                                      -1=node is unselected.
c                                       1=node is selected.
c        xyz      (dp,ar(6),out)    - vector containing x,y,z,rotx,roty,rotz
```

3.2.5. ndspgt Subroutine (Getting the Nodal Solution for a Node of an Element)

```
*deck,ndspgt
      subroutine ndspgt (node,dofs,ndof,nrot,xyzang,nuvect,unode)
c *** primary function:  get the nodal solution for a node of an element

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

c     input arguments:
c        node     (int,sc,in)    - The node number
c        dofs     (int,ar(DOFBITLENG),in) - The dofs to retrieve for the node.
c                                    dof = degree of freedom
c                                    The dofs array should be zeroed out,
c                                    except for the needed parts.
c                                    dofs is a bit pattern with true bits
c                                    representing the GLOBAL Dof set desired.
c                                    That is, dofs(1) is used for UX to SP06,
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
c                                  and dofs(2) is used for TBOT to TTOP.
c                                  See ECHPRM for details.  For example,
c                                        dofs(1) = UX + TEMP
c                                        dofs(2) = TE3
c                                  TTOP is a special case.  If you want
c                                  TTOP alone, use:
c                                        dofs(2) = ibset(0,TTOP)
c                                  If TBOT and TTOP are desired, you must use:
c                                        dofs(2) = TBOT
c                                        dofs(2) = ibset(dofs(2),TTOP)
c        ndof     (int,sc,in)    - The number of node dofs (1, 2 or 3).
c        nrot     (int,sc,in)    - Key to rotate dofs from nodal to global
c                                   coordinate systems.
c                                   if 0, none. if 2, 2-d. if 3, 3-d
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).
c        xyzang   (dp,ar(6),in)  - The xyz virgin node coordinates
c                                    (including angles).  Not used if
c                                    nrot = 0 or ndof < 2.
c        nuvect   (int,sc,in)    - Number of vectors to retrieve.  Can vary
c                                   between 1 and 5.  Normally 1 is what is
c                                   wanted.  Other vectors include previous
c                                   values and/or velocities.  See elucom for
c                                   all possibilites.  Contents are analysis
c                                   type dependent.

c     output arguments:
c        unode    (dp,ar(ndof,nuvect),out) - Element nodal solution vectors in
c                                             the global coordinate system.
```

3.3. Element Attribute Routines

3.3.1. 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 ***

c     input arguments:
c        ielem    (int,sc,in)       - element number
c                                     should be zero for key=11, DB_NUMDEFINED,
c                                       DB_NUMSELECTED, or DB_MAXDEFINED
c        key      (int,sc,in)       - information flag.
c                 = DB_SELECTED    - return select status:               (1)
c                      elmiqr = 0 - element is undefined.
c                               =-1 - element is unselected.
c                               = 1 - element is selected.
c                 = DB_NUMDEFINED  - return number of defined elements    (12)
c                 = DB_NUMSELECTED - return number of selected elements   (13)
c                 = DB_MAXDEFINED  - return maximum element number used   (14)
c                 = DB_MAXRECLENG  - return maximum record length         (15)
c                                    (int words)
c                 = 2 - return length (int words)
c                 = 3 - return layer number
c                       (for cross reference files return number of entities)
c                 = 4 - return address of first data word
c                 = 5 - return length (in record type units)
c                 = 6 - return compressed record number.
c                 = 11 - return void percent (integer)
c                 = 16 - return location of next record
c                        (this increments the next record count)
c                 = 17 - pointer to start of index
c                 = 18 - return type of file.
```

```
c                     elmiqr = 0 - integer
c                              = 1 - double precision
c                              = 2 - real
c                              = 3 - complex
c                              = 4 - character*8
c                              = 7 - index
c                 = 19 - return virtual type of file.
c                     elmiqr = 0 - fixed length (4.4 form)
c                              = 1 - indexed variable length (layer data)
c                              = 2 - xref data tables
c                              = 3 - bitmap data (for 32 data item packed records)
c                              = 4 - data tables (three dimensional arrays)
c                 = -1 - material number           ( = -EL_MAT)
c                   -2 - element type              ( = -EL_TYPE)
c                   -3 - real constant number      ( = -EL_REAL)
c                   -4 - element section ID number ( = -EL_SECT)
c                   -5 - coordinate system number  ( = -EL_CSYS)
c                         (see elmcmx for rest)
c                 =-101 - pointer to element integers etc.
c                            (see elmcmx with elmilg and 1 instead of -101)
c

c     output arguments:
c        elmiqr  (int,sc,out)  - the returned value of elmiqr is based on
c                                      setting of key.
c
c *** mpg elmiqr < el117,edgrde,edgrecc,edgmul: elem inquire
c
```

3.3.2. elmget Function (Getting an Element's Attributes and Nodes)

```
*deck,elmget
      function elmget (ielem,elmdat,nodes)
c *** primary function:    get element attributes and nodes.

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

c *** mpg magnetic usage to be checked

c arguments:
      integer,intent(in)  :: ielem    ! element number
      integer             :: elmget   ! status of element.
                                      ! = 0 - element undefined
                                      ! < 0 - number of nodes on unselected
                                      !       element
                                      !> 0 - number of nodes on selected element
      integer,intent(out) :: elmdat(*)! element attributes.
                          !   elmdat(EL_MAT)  - material number
                          !         (EL_TYPE)  - element type
                          !         (EL_REAL)  - real constant number
                          !         (EL_SECT)  - section number
                          !         (EL_CSYS)  - coordinate system number
                          !         (EL_DEAD)  - death flag (bit 0)
                          !                 if clear - alive
                          !                 if set   - dead
                          !         (EL_SOLID) - solid model reference
                          !         (EL_SHAPE) - 100*shape + specific shape
                          !         (EL_SHAPE) - 100*shape + specific shape
                          !         (EL_OBJOPTIONS)  - reserved
                          !         (EL_PEXCLUDE) - p element include flag
                          !                 (bit 0)
                          !                 if clear - include
                          ! (element may need to have its p-level increased)
                          !                 if set   - exclude
                          ! (element does not need to have its p-level increased)
                          ! EL_PEXCLUDE is also used for the LSDYNA part number
      integer,intent(out) :: nodes(*) ! node numbers for element.
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

3.3.3. 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
c              subroutine will NULL the element centroid previously defined.

c     input arguments:
c        ielem    (int,sc,in)       - element number
c        elmdat   (int,ar(EL_DIM),in) - element attributes.
c                           elmdat(EL_MAT)  - material number
c                                  (EL_TYPE)  - element type
c                                  (EL_REAL)  - real constant number
c                                  (EL_SECT)  - section number
c                                  (EL_CSYS)  - coordinate system number
c                                  (EL_DEAD)  - death flag (bit 0)
c                                         if clear - alive
c                                         if set   - dead
c                                  (EL_SOLID) - solid model reference
c                                  (EL_SHAPE) - 100*shape + specific shape
c                                  (EL_OBJOPTIONS)  - reserved
c                                  (EL_PEXCLUDE) - p element include flag
c                                         (bit 0)
c                                         if clear - include
c                                         if set   - exclude
c                                         For LSDYNA, it means part ID
c                                         in regular ANSYS, it is never part ID
c        nnod     (int,sc,in)       - number of nodes for this element.
c        nodes    (int,ar(*),in)    - node numbers for this element.

c     output arguments:  none.
```

3.3.4. 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 ***

c     input arguments:
c        itype    (int,sc,in)       - element type number
c                                     Should be 0 for key=11, DB_NUMDEFINED,
c                                     DB_NUMSELECTED, DB_MAXDEFINED, and
c                                     DB_MAXRECLENG
c        key      (int,sc,in)       - information flag.
c                 = DB_SELECTED    - return select status:
c                      etyiqr = 0 - element type is undefined.
c                             =-1 - element type is unselected.
c                             = 1 - element type is selected.
c                 = DB_NUMDEFINED  - return number of defined element types
c                 = DB_NUMSELECTED - return number of selected element types
c                 = DB_MAXDEFINED  - return highest element type number defined
c                 = DB_MAXRECLENG  - return maximum record length (int words)
c                = -n, return element characteristic n from etycom for element
c                       type itype.
c                      n is correlated to the parameter names in echprm.
c                      see elccmt for definitions of element characteristics.
c                      note- this will not overwrite the current setting of
c                       etycom.
```

```
c     output arguments:
c        etyiqr   (int,func,out)    - the returned value of etyiqr is based on
c                                      setting of key.
```

3.3.5. 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 ***

c     input arguments:
c        itype    (int,sc,in)       - element type number

c     output arguments:
c        etyget   (int,func,out)    - status of element type.
c                                     = 0 - element type is undefined.
c                                     < 0 - number of data items on unselected
c                                           element type.
c                                     > 0 - number of data items on selected
c                                           element type.
c        ielx     (int,ar(*),out)   - element type data. see elccmt for
c                                     description of data.

c *** mpg etyget<el117,edgcntf1,edgcntsz,edgrde,edgrecc,edgmul:get elem type
```

3.3.6. 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 ***

c     input arguments:
c        itype    (int,sc,in)       - element type number for operation.
c        n        (int,sc,in)       - length of data vector to store.
c        ielx     (int,ar(*),in)    - element type data.  see elccmt for
c                                     description.

c     output arguments:  none

c 2007 nov 5
c *** mpg etyput<etymod etydef dirasmdft dasupd dasdft:update elem type active
```

3.3.7. echrtr Subroutine (Getting Information About Element Characteristics)

```
*deck
      subroutine echrtr (iott,elcdn,ielc,kerr)
c   primary function: collect all element characteristics based on
c                                                    ityp,jtyp, and keyopts

c *** Notice - This file contains ANSYS Confidential information ***
c
c     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
c     iott     (int,sc,in)            - printout file
c     ielc     (int,ar(IELCSZ),inout) - input element characteristics
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries and affiliates.

```
c                                     in positions 1 to 20.
c                                     (itype, jstif, keyopts, etc.)
c
c  output arguments:
c     elcdn    (chr,sc,out)           - element descriptive name as character
c                                       string
c     ielc     (int,ar(IELCSZ),inout) - input element characteristics
c                                       in positions 21 to 150.
c                                       (kdim, ishap, idegen, etc.)
c                                       see elccmt for a full list
c     kerr     (int,sc,out)           - error flag
c                                       = 0 - no errors
c                                       = 1 - errors

c
```

3.3.8. 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
c                          element type.
c *** secondary functions: none.
c
c *** Notice - This file contains ANSYS Confidential information ***
c
c     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
c        itypi    (int,sc,in)       - element type number
c                                       = 0 - all element types
c        ksel     (int,sc,in)       - type of operation to be performed.
c                                       = 0 - delete element type.
c                                       = 1 - select element type.
c                                       =-1 - unselect element type.
c                                       = 2 - invert element type.
c
c     output arguments:
c        none.
c
```

3.3.9. mpinqr Function (Getting Information About a Material Property)

```
*deck,mpinqr
      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:
c        mat      (int,sc,in)       - material number
c                                       should be 0 for key=11,
c                                       DB_NUMDEFINED(12),
c                                       DB_MAXDEFINED(14), and
c                                       DB_MAXRECLENG(15)

c        iprop    (int,sc,in)       - property reference number:
c         if iprop = 0, test for existence of any material property with this
c                   material number (with key = DB_SELECTED(1))
c
c        ---- MP command labels --------
c        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
c        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
c        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, BVIS=58, QRAT=59, REFT=60, CTEX=61, CTEY=62, CTEZ=63, THSX=64,
c        THSY=65, THSZ=66, DMPR=67, LSSM=68, BETD=69, ALPD=70, RH  =71, DXX =72,
c        DYY =73, DZZ =74, BETX=75, BETY=76, BETZ=77, CSAT=78, CREF=79, CVH =80
c
c               (see mpinit for uncommented code and for TB command information)

c        key      (int,sc,in)       - key as to the information needed
c                                     about material property.
c             = DB_SELECTED(1)- return select status:
c                        mpinqr = 0 - material prop is undefined.
c                               = 1 - material prop is selected.
c             = DB_NUMDEFINED(12) - number of defined material properties
c             = DB_MAXDEFINED(14) - highest material property number defined
c             = DB_MAXRECLENG(15) - maximum record length (dp words)
c             =  2 - return length (dp words)
c             =  3 - return number of temp. values
c             = 11 - return void percent (integer)

c     output arguments:
c        mpinqr   (int,func,out)    - returned value of mpinqr is based on
c                                       setting of key.
```

3.3.10. 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 ***

c     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        mat      (int,sc,in)       - material number
c        iprop    (int,sc,in)       - property reference number:
c        ---- MP command labels --------
c        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
c        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
c        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, BVIS=58, QRAT=59, REFT=60, CTEX=61, CTEY=62, CTEZ=63, THSX=64,
c        THSY=65, THSZ=66, DMPR=67, LSSM=68, BETD=69, ALPD=70, RH =71,  DXX =72,
c        DYY =73, DZZ =74, BETX=75, BETY=76, BETZ=77, CSAT=78, CREF=79, CVH =80
c
c                  (see mpinit for uncommented code and TB command information)

c     output arguments:
c        mpget   (int,func,out)     - number of temperature values
c        temp    (dp,ar(mpget),out) - vector of the temperature values
c        prop    (dp,ar(mpget),out) - vector of the property values
```

3.3.11. mpput Subroutine (Storing a Material Property Table)

```
*deck,mpput
      subroutine mpput (mat,iprop,ntab,temp,prop)
c *** primary function:    store material property tables.
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

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

c     input arguments:
c        mat      (int,sc,in)       - material number.
c        iprop    (int,sc,in)       - property reference number:
c        ---- MP command labels --------
c        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
c        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
c        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, BVIS=58, QRAT=59, REFT=60, CTEX=61, CTEY=62, CTEZ=63, THSX=64,
c        THSY=65, THSZ=66, DMPR=67, LSSM=68, BETD=69, ALPD=70, RH  =71, DXX =72,
c        DYY =73, DZZ =74, BETX=75, BETY=76, BETZ=77, CSAT=78, CREF=79, CVH =80
c
c                  (see mpinit for uncommented code and TB command information)

c        ntab     (int,sc,in)       - number of entries in the table
c                                        (1 to 100)
c        tem     (dp,ar(ntab),in)  - temperature vector (ascending)
c        prp     (dp,ar(ntab),in)  - property vector

c     output arguments:
c        none.
```

3.3.12. 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.
c        iprop    (int,sc,in)       - property reference number:
c                                        (0 = all properties)
c        ---- MP command labels --------
c        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
c        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
c        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, BVIS=58, QRAT=59, REFT=60, CTEX=61, CTEY=62, CTEZ=63, THSX=64,
c        THSY=65, THSZ=66, DMPR=67, LSSM=68, BETD=69, ALPD=70, RH  =71, DXX =72,
c        DYY =73, DZZ =74, BETX=75, BETY=76, BETZ=77, CSAT=78, CREF=79, CVH =80
c
c                (see mpinit for uncommented code and for TB command information)

c     output arguments:  none.
```

3.3.13. rlinqr Function (Getting Information About a Real Constant Set)

```
*deck,rlinqr
      function rlinqr (nreal,key)
c *** primary function:    get information about a real constant set
c *** secondary functions: none

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

c     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        nreal   (int,sc,in)       - real constant table number
c                                      should be 0 for key=11, DB_NUMDEFINED,
c                                      DB_NUMSELECTED, DB_MAXDEFINED, and
c                                      DB_MAXRECLENG
c        key      (int,sc,in)       - information flag.
c             = 5              - return number of values stored for nreal
c             = DB_SELECTED    - return select status
c                      rlinqr = 0 - real constant table is undefined.
c                             =-1 - real constant table is unselected.
c                             = 1 - real constant table is selected
c             = 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
c             = DB_MAXRECLENG  - return maximum record length (dp words)

c  output arguments:
c        rlinqr   (int,func,out)    - the returned value of rlinqr is based on
c                                       setting of key.
c *** mpg magnetic interface usage
c
```

3.3.14. rlget 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:
c        nreal    (int,sc,in)       - real constant table number

c     output arguments:
c        rlget    (int,func,out)    - number of real constant data obtained
c        rtable   (dp,ar(*),out)    - real constant data obtained
```

3.3.15. 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
c *** Notice - This file contains ANSYS Confidential information ***
c
c     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
c     nreai    (int,sc,in)       - real constant table
c                                    = 0 - all real constant tables
c     ksel     (int,sc,in)       - type of operation to be performed.
c                                    = 0 - delete real constant table.
c                                    = 1 - select real constant table.
c                                    =-1 - unselect real constant table.
c                                    = 2 - invert real constant table.
c
c  output arguments:
c     none
c
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries and affiliates.

3.3.16. csyiqr Function (Getting Information About a Coordinate System)

```
*deck,csyiqr
      function csyiqr (ncsy,key)
c *** primary function:    get information about a coordinate system

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

c  input arguments:
c     ncsy     (int,sc,in)       - coordinate system reference number
c                                    should be zero for key= DB_NUMDEFINED
c                                    or DB_MAXDEFINED
c     key      (int,sc,in)       - information flag.
c              = DB_SELECTED    - return status:
c                            csyiqr = 0 - coordinate system is not defined
c                                    -1 - coordinate system is not selected
c                                     1 - coordinate system is selected
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
c                                    setting of key.
```

3.3.17. 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 ***

c  NOTE:  As a time-saving device, this routine will not fetch the coordinate
c         system data from the database (an expensive operation)
c         if ncsy = csyinx(4), as this would indicate that the data is current.
c         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.

c     typ=int,dp,log,chr,dcp   siz=sc,ar(n),func    intent=in,out,inout

c  input arguments:
c     variable (typ,siz,intent)    description                  csycom name
c     ncsy     (int,sc,in)       - coordinate system number
c     csyinx(4) (int,sc,inout)   - coordinate system number     csyact

c  output arguments:
c     csydpx   (dp,ar(18),out)
c                           csydpx(1-9)   - transformation matrix
c                                 (10-12) - origin (XC, YC, ZC)
c                                 (13-14) - coordinate system parameters  cparm
c                                                                         cparm2
c                                 (15)    - spare
c                                 (16-18) - defining angles
c     csyinx   (int,ar(6),out)
c                           csyinx(1-2)   - theta, phi singularity keys
c                                 (3)     - coordinate system type        icdsys
c         (csyinx(4) is inout)    (4)     - coordinate system number      csyact
c                                 (5)     - spare
c                                 (6)     - spare
c     csyget   (int,func,out)      - status of coordinate system
c                                    = 0 - coordinate system exists
c                                    = 1 - coordinate system doesn't exist
```

3.3.18. 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 ***

c  input arguments:
c     ncsy     (int,sc,in)       - coordinate system number
c     csydpx   (dp,ar(18),out)
c                           csydpx(1-9)   - transformation matrix
c                                 (10-12) - origin (XC, YC, ZC)
c                                 (13-14) - coordinate system parameters     cparm
c                                                                           cparm2
c                                 (15)    - spare
c                                 (16-18) - defining angles
c     csyinx   (int,ar(6),out)
c                           csyinx(1-2) - theta, phi singularity keys
c                                 (3)   - coordinate system type            icdsys
c                                 (4)   - coordinate system number          csyact
c                                 (5)   - spare
c                                 (6)   - spare


c  output arguments: none
```

3.3.19. csydel Subroutine (Deleting a Coordinate System)

```
*deck,csydel
      subroutine csydel (ncsy)
c *** primary function:    delete a coordinate system
c *** secondary functions: none
c
c     typ=int,dp,log,chr,dcp   siz=sc,ar(n),func    intent=in,out,inout
c
c *** Notice - This file contains ANSYS Confidential information ***
c
c  input arguments:
c     variable (typ,siz,intent)    description
c     ncsy     (int,sc,in)       - coordinate system number
c
c  output arguments:
c     none
c
```

3.3.20. userac Subroutine (Demonstrates Use of Element Attribute Routines)

See Subroutine userac (Accessing Element Information) (p. 163) 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, Inc. distribution media.

3.4. Coupling and Constraint Routines

3.4.1. cpinqr Function (Getting Information About a Coupled Set)

```
*deck,cpinqr
      function cpinqr (nce,key)
c *** primary function:    get information about a coupled set
c *** secondary functions: none
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
c
c *** Notice - This file contains ANSYS Confidential information ***
c
c     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
c     nce       (int,sc,in)     - coupled set number
c     key       (int,sc,in)     - inquiry key:
c                                  should be zero for key=11, DB_NUMDEFINED,
c                                   DB_NUMSELECTED, DB_MAXDEFINED, and
c                                   DB_MAXRECLENG
c                = DB_SELECTED    - return select status
c                            cpinqr = 1 - coupled set is selected
c                                   = 0 - coupled set in undefined
c                                   =-1 - coupled set in unseleted
c                = DB_NUMDEFINED  - return number of defined coupled sets
c                = DB_NUMSELECTED - return number of selected coupled sets
c                = DB_MAXDEFINED  - return the number of the highest numbered
c                                    coupled set
c                = DB_MAXRECLENG  - return length of largest coupled set record
c                                    (max record length)
c                =  2             - return length (data units)
c                =  3             - return layer number
c                =  4             - return address of first data word
c                = 11             - return void percent (integer)
c                = 16             - return location of next record
c                = -1             - return master node for this eqn (this is
c                                   currently only used by solution DB object)
c
c  output arguments:
c     cpinqr   (int,func,out)    - the returned value of cpinqr is based on
c                                       setting of key
c
```

3.4.2. 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 ***

c  input arguments:
c     ncp      (int,sc,in)                  - coupled set number
c
c  output arguments:
c     cpget         (int,func,out)        - number of nodes in list
c     ieqn          (int,ar(cpget+2),out) - coupled set info:
c                             ieqn(1:cpget) - list of coupled nodes
c                             ieqn(cpget+1) - set degree of freedom
c                             ieqn(cpget+2) - number of nodes in list
c                                             (copy of return value)
```

3.4.3. 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 ***

c  input arguments:
c     ncp       (int,sc,in)        - coupled set number
c     n         (int,sc,in)        - number of nodes in coupled set
c     ieqn      (int,ar(n+2),in)   - info for storage
```

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
c                          ieqn(1:n) - list of coupled nodes
c                          ieqn(n+1) - degree of freedom label for set
c    (ieqn(n+2) is inout)  ieqn(n+2) - number of nodes in coupled set
c                                       (copy of n)

c  output arguments:
c     ieqn(n+2) (int,sc,inout)     - number of nodes in coupled set
c                                    (another copy of n)
```

3.4.4. 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
c *** Notice - This file contains ANSYS Confidential information ***
c
c     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
c     ncpi      (int,sc,in)       - coupled set number
c     ksel      (int,sc,in)       - select/delete flag
c                                   = 0 - delete coupled set
c                                   = 1 - select coupled set
c  output arguments:
c     none
c
```

3.4.5. ceinqr Function (Getting Information About a Constraint Equation Set)

```
*deck,ceinqr
      function ceinqr (nce,key)
c *** primary function:    get information about a constraint equation set

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

c     input arguments:
c     nce         (int,sc,in)       - constraint equation number
c     key         (int,sc,in)       - inquiry key:
c                                     should be zero for key=11, DB_NUMDEFINED,
c                                      DB_NUMSELECTED, DB_MAXDEFINED, and
c                                      DB_MAXRECLENG
c           = DB_SELECTED    - return select status
c                                   ceinqr = 1 - equation is selected
c                                          = 0 - equation is undefined
c                                          =-1 - equation is unselected
c           = DB_NUMDEFINED  - return number of defined contraint equations
c           = DB_NUMSELECTED - return number of selected contraint equations
c           = DB_MAXDEFINED  - return number of highest numbered constraint
c                              equation defined
c           = DB_MAXRECLENG  - return length of longest contraint equation set
c                              (max record length)
c           =  2             - return length (data units)
c           =  3             - return layer number
c           =  4             - address of first data word
c           = 11             - return void percent (integer)
c           = 16             - return location of next record
c           = CE_NONLINEAR   - return 1 if CE is nonlinear

c     output arguments:
c        ceinqr   (int,func,out)    - the returned value of ceinqr is based on
c                                     setting of key
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries and affiliates.

3.4.6. ceget Function (Getting a 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  arguments:
      integer,intent(in)   ::  nce         ! constraint equation set number
      integer,intent(out)  ::  ieqn(2,*)   ! ieqn(1:2,1:n) - node, dof
                                           ! ieqn(2,n+1) - number of dof in set
                                           !  negative means internal CE
      double precision,intent(out) :: deqn(*)  ! deqn(1:n) - coefficients of
                                               !  each dof in set
                                               ! deqn(n+1) - constant term
```

3.4.7. 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 ***

c  arguments:
      integer,intent(in)  ::  ncei  ! constraint equation number
      integer,intent(in)  ::  ksel  ! select/delete flag
                                    !   = 0 - delete equation

      external  fAnsMemAlloc, fAnsMemFree
      external  vmsel,  ceinqr, cexsub, vmgetx, inquireAnsCe,
     x          activateAnsCe, erhandler, checkForDupTerms
      integer   vmgetx, ceinqr, inquireAnsCe, activateAnsCe
      PTRFTN    fAnsMemAlloc
```

3.4.8. cesel Subroutine (Deleting or Selecting a Constraint Equation)

```
*deck,cesel
      subroutine cesel (ncei,ksel)
c *** primary function:    select or delete a constraint equation

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

c  arguments:
      integer,intent(in)  ::  ncei  ! constraint equation number
      integer,intent(in)  ::  ksel  ! select/delete flag
                                    !   = 0 - delete equation
```

3.5. Nodal Loading Routines

3.5.1. disiqr Function (Getting 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.
c        key     (int,sc,in)       - key as to the information needed
c                          = 1              - return constraint mask
c                          = DB_MAXDEFINED,
c                            DB_NUMDEFINED  - return number of nodal constraints
```

```
c                                          NOTE: both DB_MAXDEFINED and
c                                          DB_NUMDEFINED produce the same
c                                          functionality

c     output arguments:
c        disiqr   (int,func,out)   - the returned value of disiqr is based on
c                                    setting of key.
```

3.5.2. 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 ***
c     input arguments:
c     variable (typ,siz,intent)     description
c        inode    (int,sc,in)       - node number (negative value for no
c                                                   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         ,        ,        , WARP=16
c    CONC=17, HDSP=18, PRES=19, TEMP=20, VOLT=21, MAG =22, ENKE=23, ENDS=24
c    EMF =25, CURR=26  SP01=27, SP02=28, SP03=29, SP04=30, SP05=31, SP06=32
c                                 (missing entries are spares)

c     output arguments:
c        disget   (int,func,out)    - status of constraint.
c                                     = 0 - no constraint on this node
c                                           for this DOF
c                                     = 4 - this node has a constraint
c                                           defined for this DOF
c                                     = -4 - this node has a pseudo-support
c                                           defined for this DOF
c        value    (dp,ar(4),out)    - constraint values
c                         value(1-2) - (real,imag) values of present settings
c                         value(3-4) - (real,imag) values of previous settings
```

3.5.3. 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 ***

c     input arguments:
c        node     (int,sc,in)       - node number
c        idf      (int,sc,in)       - reference number of 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         CONC=17           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.
```

3.5.4. disdel Subroutine (Deleting a Constraint at a Node)

```
*deck,disdel
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries and affiliates.

```
      subroutine disdel (node,idf)
c *** primary function:    delete a constraint at a node

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

c     input arguments:
c        node     (int,sc,in)       - node number.
c        idf      (int,sc,in)       - reference number of 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         CONC=17,          PRES=19, TEMP=20, VOLT=21, MAG =22, ENKE=23, ENDS=24
c         EMF =25, CURR=26           (missing entries are spares)

c     output arguments:  none.
```

3.5.5. foriqr Function (Getting Information About Nodal Loads)

```
*deck,foriqr
      function foriqr (node,key)
c *** primary function: get information about nodal loads.

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

c     input arguments:
c        node     (int,sc,in)       - number of node being inquired about.
c                                     should be 0 for key=DB_MAXDEFINED or
c                                     DB_NUMDEFINED
c        key      (dp,sc,in)        - key as to information needed
c                             = 1              - return force mask for node
c                             = DB_MAXDEFINED,
c                               DB_NUMDEFINED  - return number of nodal loadings
c                                                in model
c                                    NOTE: both DB_MAXDEFINED and DB_NUMDEFINED
c                                    produce the same functionality

c     output arguments:
c        foriqr   (int,func,out)    - the returned value of foriqr is based on
c                                     setting of key.
```

3.5.6. 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 ***

c     input arguments:
c        inode    (int,sc,in)       - node number (negative value for no
c                                                   partabeval)
c        idf      (int,sc,in)       - reference number for the DOF: (1-32)
c                                     (see echprm.inc)

c     output arguments:
c        forget   (int,func,out)  - status of constraint.
c                                   = 0 - no loading on this node for this DOF
c                                   = 4 - this node has a loading for this DOF
c        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
```

3.5.7. forput Subroutine (Storing a Nodal Load at a Node)

```
*deck,forput
      subroutine forput (node,idf,value)
c *** primary function:    store a nodal load at a node

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

c     input arguments:
c        node     (int,sc,in)       - node number
c        idf      (int,sc,in)       - reference number for the DOF: (1-32)
c         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
c         RATE=17           FLOW=19, HEAT=20, AMPS=21, FLUX=22, NPKE=23, NPDS=24
c         CURT=25,                   (missing entries are spares)

c        value    (dp,ar(2),in)     - (real,imag) values of force

c     output arguments:  none.
```

3.5.8. 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 ***

c     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        node     (int,sc,in)       - node number
c        idf      (int,sc,in)       - reference number for the DOF: (1-32)
c         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
c         RATE=17,          FLOW=19, HEAT=20, AMPS=21, FLUX=22, NPKE=23, NPDS=24
c         CURT=25,                   (missing entries are spares)

c     output arguments:
c        none.
```

3.5.9. ntpiqr Function (Getting Information About a Nodal Temperature)

```
*deck,ntpiqr
      function ntpiqr (node,key)
c *** primary function:    get information about a nodal temperature

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

c     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        node     (int,sc,in)       - node number
c                                     should be zero for key=2
c        key      (int,sc,in)       - key for operation
c                                     = 1 - return temperature status
c                                      ntpiqr = 0 - node has no temperature
c                                                   constraint defined
c                                             = 1 - node has a temperature
c                                                   constraint defined
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
c                                     = 2 - return total number of nodal
c                                           temperatures defined in model

c     output arguments:
c        ndinqr   (int,func,out)    - the returned value of ndinqr is based on
c                                     setting of key.
```

3.5.10. 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 ***

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

c     output arguments:
c        ntpget   (int,func,out)    - heat generation status of node.
c                                     = 0 - nodal heat generation undefined
c                                     = 1 - nodal heat generation is defined
c        tmp       (dp,ar(2),out)    - the nodal heat generation (new,old).
```

3.5.11. ntpput Subroutine (Storing a Nodal Temperature)

```
*deck,ntpput
      subroutine ntpput (node,temp)
c *** primary function:    store nodal temperature.

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

c     input arguments:
c        node    (int,sc,in)       - node number
c        temp    (dp ,sc,in)       - nodal temperature

c     output arguments:  none.
```

3.5.12. 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
c     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
c        node    (int,sc,in)       - node number
c
c     output arguments:
c        none.
```

3.5.13. nhgiqr Function (Getting Information About Nodal Heat Generations)

```
*deck,nhgiqr
```

```
      function nhgiqr (node,key)
c *** primary function:    get information about nodal heat generations

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

c     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        node     (int,sc,in)       - node number
c                                      should be 0 for key=2
c        key      (int,sc,in)       - key for operation
c                          = 1 - return whether node has a heat generation rate
c                                 defined
c                              nhgiqr = 0 - no heat generation defined for node
c                                     = 1 - heat generation is defined for node
c                          = 2 - return total number of nodal heat generation
c                                 rates defined in model

c     output arguments:
c        nhgiqr   (int,func,out)    - the returned value of nhgiqr is based on
c                                        setting of key.
```

3.5.14. nhgget Function (Getting a Nodal Heat Generation)

```
*deck,nhgget
      function nhgget (node,hg)
c *** primary function:    get specified nodal heat generation (in raw form)

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

c     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        node     (int,sc,in)       - node number

c     output arguments:
c        nhgget   (int,func,out)    - heat generation status of node.
c                                     = 0 - nodal heat generation undefined
c                                     = 1 - nodal heat generation is defined
c        hg       (dp,ar(2),out)    - the nodal heat generation (new,old).
```

3.5.15. nhgput Subroutine (Storing Nodal Heat Generation)

```
*deck,nhgput
      subroutine nhgput (node,hg)
c *** primary function:    store nodal heat generation.

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

c     input arguments:
c        node    (int,sc,in)       - node number
c        hg      (dp ,sc,in)       - nodal heat generation

c     output arguments:  none.
```

3.5.16. nhgdel Subroutine (Deleting a Nodal Heat Generation)

```
*deck,nhgdel
      subroutine nhgdel (node)
c *** primary function:    delete nodal heat generations.
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
c *** Notice - This file contains ANSYS Confidential information ***
c
c     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
c        node    (int,sc,in)       - node number

c     output arguments:
c        none.
```

3.5.17. nfuiqr Function (Getting Information About Nodal Fluences)

```
*deck,nfuiqr
      function nfuiqr (node,key)
c *** primary function:    get information about nodal fluences

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

c     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        node     (int,sc,in)       - node number
c                                     should be zero for key=2
c        key      (int,sc,in)       - key for operation
c                         = 1 - return status:
c                           nfuiqr = 0 - node does not have a fluence constraint
c                                  = 1 - node has a fluence constraint
c                         = 2 - return total number of nodal fluences defined on
c                                model


c     output arguments:
c        nfuiqr   (int,func,out)    - the returned value of nfuiqr is based on
c                                      setting of key.
```

3.5.18. 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 ***

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

c     output arguments:
c        nfuget   (int,func,out)    - fluence status of node.
c                                     = 0 - node has no fluence constraint
c                                     = 1 - node has a fluence constaint
c        fluen    (dp ,ar(2),out)   - the nodal fluences (new,old).
```

3.5.19. 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 ***
```

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
c     input arguments:
c        node    (int,sc,in)       - node number
c        fluen   (dp ,sc,in)       - nodal fluence

c     output arguments:  none.
```

3.5.20. nfudel Subroutine (Deleting a Nodal Fluence)

```
*deck,nfudel
      subroutine nfudel (node)
c *** primary function:    delete node fluences.
c
c *** Notice - This file contains ANSYS Confidential information ***
c
c     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
c        node    (int,sc,in)       - node number
c
c     output arguments:
c        none.
c
```

3.5.21. 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 ***

c     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        node     (int,sc,in)       - node number
c                                     should be zero for key=2
c        key      (int,sc,in)       - key for operation
c                       = 1 - return nodal current status:
c                          ndciqr = 0 - no current density defined for this node
c                                 = 1 - node has a current density defined
c                       = 2 - total number of nodal current densities defined
c                             on model


c     output arguments:
c        ndciqr   (int,func,out)    - the returned value of ndciqr is based on
c                                     setting of key.
```

3.5.22. ndcget Function (Getting a Nodal Current Density)

```
*deck,ndcget
      function ndcget (node,currd)
c *** primary function:    get specified nodal current density.

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

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


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
c     output arguments:
c        ndcget   (int,func,out)    - current density status of node.
c                                     = 0 - node has no current density defined
c                                     = 1 - node has a current density defined
c        currd    (dp,ar(4,2),out)  - the node current density (new,old).
```

3.5.23. 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:
c        node    (int,sc,in)       - node number
c        currd   (dp ,ar(4),in)    - nodal current densities

c     output arguments:  none.
```

3.5.24. 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 ***
c
c     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
c        node    (int,sc,in)       - node number
c
c     output arguments:
c        none.
```

3.5.25. nvdiqr Function (Getting Information About Nodal Magnetic Virtual Displacements)

```
*deck,nvdiqr
      function nvdiqr (node,key)
c *** primary function:    get information about nodal mag virtual disps

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

c     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        node     (int,sc,in)       - node number
c                                      should be zero for key=2
c        key      (int,sc,in)       - key for operation
c                     = 1 - return magnetic virtual displacement status
c                         nvdiqr = 0 - no mag. virt. disps defined for this node
c                                = 1 - node has mag. virt. disps defined
c                     = 2 - return total number of nodal magnetic virtual
c                            displacements defined on model

c     output arguments
```

```
c        nvdiqr   (int,func,out)    - the returned value of nvdiqr is based on
c                                      setting of key.
```

3.5.26. 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 ***

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

c     output arguments:
c        nvdget   (int,func,out)    - virtual disp status of node.
c                                       = 0 - node has no magnetic virtual
c                                             displacement
c                                       = 1 - node has a magnetic virtual
c                                             displacement
c        virtd    (dp ,sc,out)      - the nodal virtual displacement value
```

3.5.27. nvdput Subroutine (Storing a Nodal Virtual Displacement)

```
*deck,nvdput
      subroutine nvdput (node,virtd)
c *** primary function:    store nodal virtual displacement

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

c     input arguments:
c        node    (int,sc,in)       - node number
c        virtd   (dp ,sc,in)       - nodal virtual displacement

c     output arguments:  none.
```

3.5.28. 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 ***
c
c     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
c        node    (int,sc,in)       - node number
c
c     output arguments:
c        none.
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries and affiliates.

3.6. Element Loading Routines

3.6.1. 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 ***

c     input arguments:
c        ielem    (int,sc,in)       - element number
c                                      should be zero for key=DB_NUMDEFINED or
c                                      DB_MAXRECLENG
c        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
c                                      meaning (see below)
c        key      (int,sc,in)       - key as to the information needed
c                 = 1              - return pressure mask for element
c                 = 5              - return number of pressures for this
c                                      element face
c                 = DB_NUMDEFINED,
c                 = DB_MAXDEFINED  - return value is based on setting of iface
c                                    NOTE: both DB_NUMDEFINED and
c                                    DB_MAXDEFINED produce the same
c                                    functionality
c                              iface = 0 - return number of surface loads defined
c                                    = 1-6 - return number of pressure loads
c                                      defined for this element.
c                                      NOTE:  only 1-6 is valid, but this
c                                      routine simply checks that iface is in
c                                      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
c                                    length)

c     output arguments:
c        epriqr   (int,func,out)    - the returned value of epriqr is based on
c                                      setting of key.
```

3.6.2. 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 ***

c     input arguments:
c        elem     (int,sc,in)       - element number (negative value for
c                                       no partabeval)
c        iface    (int,sc,in)       - face number (1-68)

c     output arguments:
c        eprget   (int,func,out)    - status of element.
c                                     =-1 - element has no pressures
c                                     = 0 - this element face has no pressures
c                                     > 0 - number of values defined
c        value    (dp ,ar(*),out)   - the element pressures (real,imag) at each
c                                      face
```

3.6.3. 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 ***

c     input arguments:
c        ielem    (int,sc,in)       - element number for operation.
c        iface    (int,sc,in)       - face number (1-68)
c        nval     (int,sc,in)       - number of values to put
c        value    (dp ,ar(nval),in) - the element pressures (real,imag) at each
c                                      face

c     output arguments:  none.
```

3.6.4. 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 ***

c     input arguments:
c        ielem     (int,sc,in)       - element number
c        iface     (int,sc,in)       - face number
c                                       = 0 - delete all pressures on this
c                                             element
c                                       = 1-6 - delete pressure on this face

c     output arguments:  none.
```

3.6.5. 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 ***

c     input arguments:
c        ielem    (int,sc,in)       - element number for inquire
c                                       should be zero for key=DB_NUMDEFINED or
c                                       DB_MAXRECLENG
c        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
c                                        meaning (see below)
c        key      (int,sc,in)       - key as to the information needed
c                 = 1              - return convection mask for element
c                 = 5              - return number of convections for this
c                                      element face
c                 = DB_NUMDEFINED,
c                 = DB_MAXDEFINED  - return value is based on setting of iface
c                                      NOTE: both DB_NUMDEFINED and
c                                      DB_MAXDEFINED produce the same
c                                      functionality
c                              iface = 0 - return number of surface loads
c                                          defined (rec length)
c                                    = 1-6 - return number of convection loads
c                                       defined for this element.
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries and affiliates.

```
c                                       NOTE:  only 1-6 is valid, but this
c                                       routine simply checks that iface is in
c                                       the range.  The actual value of iface
c                                       does not matter in this case.
c                 = DB_MAXRECLENG  - return the maximum number of convections
c                                      on any element (max rec length)

c     output arguments:
c        ecviqr   (int,func,out)    - the returned value of ecviqr is based on
c                                       setting of key.
```

3.6.6. 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 ***

c     input arguments:
c        elem     (int,sc,in)       - element number
c        iface    (int,sc,in)       - face number (1-6)

c     output arguments:
c        ecvget   (int,func,out)    - status of element.
c                                      =-1 - element has no convections/heat
c                                             fluxes
c                                      = 0 - this element face has no
c                                             convections/heat fluxes
c                                      > 0 - number of values defined
c        value    (dp ,ar(*),out)   - the element convections

c                                     NOTE: Two values at each node of an
c                                      element face: if loading is a convection,
c                                      the first first value is the film
c                                      coefficient and the second value is the
c                                      bulk temperature. If loading is a heat
c                                      flux, the first value is the heat flux,
c                                      and the second value is a large number
c                                      (2**100)
```

3.6.7. 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 ***

c     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        nval     (int,sc,in)       - number of values to put
c        value    (dp ,ar(nval),in) - the element convections.
c                                      NOTE: Two values at each node of an
c                                       element face: if loading is a convection,
c                                       the first first value is the film
c                                       coefficient and the second value is the
c                                       bulk temperature. If loading is a heat
c                                       flux, the first value is the heat flux,
c                                       and the second value is a large number
c                                       (2**100)
c     output arguments:
c        none.
```

3.6.8. 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 ***

c     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
c                                       = 0 - delete all convections on this
c                                             element
c                                       = 1-6 - delete convections on this face

c     output arguments:
c        none.
```

3.6.9. etpiqr Function (Getting Information About Element Temperatures)

```
*deck,etpiqr
      function etpiqr (ielem,key)
c *** primary function: get information about element temperatures.

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

c     input arguments:
c        ielem    (int,sc,in)       - element number
c                                        Should be 0 for key=11, DB_NUMDEFINED,
c                                        DB_MAXDEFINED, and DB_MAXRECLENG
c        key      (int,sc,in)       - information flag.
c                 = DB_SELECTED    - return status:
c                      etpiqr = 0 - element has no temperatures
c                             = 1 - element has temperatures defined
c                 = DB_NUMDEFINED  - return number of temperatures defined for
c                                    this element (rec length)
c                 = DB_MAXDEFINED  - return number of temperatures defined in
c                                    model
c                 = DB_MAXRECLENG  - return maximum number of temperatures
c                                    defined for any element (max rec length)
c                 = 2 - return length (dp words)
c                 = 3 - return layer number (for cross reference files return
c                           number of entities)
c                 = 4 - return address of first data word
c                 = 5 - return length (dp words)
c                 = 6 - return compressed record number.
c                 = 11 - return void percent (integer)
c                 = 16 - return location of next record (this increments the
c                           next record count)
c                 = 18 - return type of file.
c                          etpiqr = 0 - integer
c                                 = 1 - double precision
c                                 = 2 - real
c                                 = 3 - complex
c                                 = 4 - character*8
c                                 = 7 - index
c                 = 19 - return virtual type of file.
c                          etpiqr = 0 - fixed length (4.4 form)
c                                 = 1 - indexed variable length
c                                         (layer data)
c                                 = 2 - xref data tables
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries and affiliates.

```
c                                = 3 - bitmap data (for 32 data item packed
c                                       records)
c                                = 4 - data tables (three dimensional arrays)


c     output arguments:
c        etpiqr   (int,func,out)    - the returned value of etpiqr is based on
c                                     setting of key.
```

3.6.10. 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:
c        ielem    (int,sc,in)       - element number

c     output arguments:
c        etpget   (int,func,out)    - status of element.
c                                     = 0 - this element has no element
c                                           temperatures
c                                     > 0 - number of element temperatures
c                                           retrieved
c        tem     (dp,ar(n,2),out)  - the element temperatures (new,old).

c    NOTE THAT TEM MUST DOUBLE THE NUMBER OF DESIRED
c    TEMPERATURES IN THE CALLING ROUTINE!

c                                       NOTE: If a value is not defined (i.e.,
c                                             defaults to TUNIF), value will be a
c                                             very small number (2**-100)
```

3.6.11. 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 ***

c     input arguments:
c        ielem    (int,sc,in)       - element number
c        n        (int,sc,in)       - number of element temperature values
c        temp     (dp ,ar(n),in)    - element temperatures.

c     output arguments:  none.
c                                       NOTE: If a value is not defined (i.e.,
c                                             defaults to TUNIF), a very small
c                                             number should be used (2**-100)
```

3.6.12. 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:
```

```
c        ielem    (int,sc,in)       - element number

c     output arguments:  none.
```

3.6.13. 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:
c        ielem    (int,sc,in)       - element number
c                                       should be 0 for key=11, DB_NUMDEFINED,
c                                       DB_MAXDEFINED, and DB_MAXRECLENG
c        key      (int,sc,in)       - information flag.
c                 = DB_SELECTED    - return status:
c                      ehgiqr = 0 - heat generation is undefined
c                             = 1 - heat generation is defined
c                 = DB_NUMDEFINED  - return number of defined heat generations
c                                    in model
c                 = DB_MAXRECLENG  - return maximum number of heat generations
c                                    on any element (max rec length)
c                 = 2 - return length (dp words)
c                 = 3 - return layer number (for cross reference files return
c                                    number of entities)
c                 = 4 - return address of first data word
c                 = 5 - return length (record type units)
c                 = 6 - return compressed record number.
c                 = 11 - return void percent (integer)
c                 = 16 - return location of next record (this increments the
c                                    next record count)
c                 = 18 - return type of file.
c                      ehgiqr = 0 - integer
c                             = 1 - double precision
c                             = 2 - real
c                             = 3 - complex
c                             = 4 - character*8
c                             = 7 - index
c                 = 19 - return virtual type of file.
c                      ehgiqr = 0 - fixed length (4.4 form)
c                             = 1 - indexed variable length
c                                          (layer data)
c                             = 2 - xref data tables
c                             = 3 - bitmap data (for 32 data
c                                          item packed records)
c                             = 4 - data tables (three
c                                          dimensional arrays)


c     output arguments:
c        ehgiqr   (int,func,out)    - the returned value of ehgiqr is based on
c                                     setting of key.
```

3.6.14. ehgget Function (Getting an Element Heat Generation)

```
*deck,ehgget
      function ehgget (ielem,qgen)
c *** primary function:    get element heat generations (in raw form)

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

c     input arguments:
c        ielem    (int,sc,in)       - element number
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
c     output arguments:
c        ehgget   (int,func,out)    - status of element.
c                                     = 0 - heat generations undefined for this
c                                           element
c                                     > 0 - number of heat generations defined
c        qgen     (dp ,ar(*),out)   - the element heat generations.
c                                     NOTE: If a value is not defined, it will
c                                           be a very small number (2**-100)
```

3.6.15. 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 ***

c     input arguments:
c        ielem    (int,sc,in)       - element number
c        n        (int,sc,in)       - number of element heat generation values
c        qgen     (dp ,ar(n),in)    - element heat generations

c     output arguments:  none
c                                     NOTE: If a value is not defined, a very
c                                        small number should be used (2**-100)
```

3.6.16. 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 ***

c     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     output arguments:
c        none
```

3.6.17. 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:
c        ielem    (int,sc,in)       - element number or zero (see below)
c        key      (int,sc,in)       - key as to the information needed
c                 = 1 or DB_MAXRECLENG - return element fluences info
c                         for ielem > 0 - return number of fluences for this
c                                         element (record length)
c                                   = 0 - return maximum number of fluences
c                                         defined for any element
c                                         (max rec length)
```

```
c                = DB_NUMDEFINED,
c                = DB_MAXDEFINED  - return number of defined fluences
c                                    in model
c                                   NOTE: both DB_NUMDEFINED and DB_MAXDEFINED
c                                         produce the same functionality

c     output arguments:
c        efuiqr   (int,func,out)    - the returned value of efuiqr is based on
c                                       setting of key
```

3.6.18. 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 ***

c     input arguments:
c        ielem    (int,sc,in)       - element number

c     output arguments:
c        efuget   (int,func,out)    - status of element.
c                                      = 0 - element has no fluences defined
c                                      > 0 - number of element fluences defined
c        value    (dp,ar(*),out)    -  element fluences.

c                                     NOTE: If a value is not defined, it will
c                                           be a very small number (2**-100)
```

3.6.19. 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 ***

c     input arguments:
c        ielem    (int,sc,in)        - element number
c        n        (int,sc,in)        - the number of values to store
c        value    (dp,ar(n),in)      - element fluences.

c     output arguments:  none

c                                      NOTE: If a value is not defined, a very
c                                         small number should be used (2**-100)
```

3.6.20. 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
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries and affiliates.

3.6.21. edciqr Function (Getting Information About Element Current Densities)

```
*deck,edciqr
      function edciqr (ielem,key)
c *** primary function:    get information about element current densities

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

c     input arguments:
c        ielem    (int,sc,in)       - element number or zero (see below)
c        key      (int,sc,in)       - key as to the information needed
c                 = 1 or DB_MAXRECLENG - return element densities info
c                         for ielem > 0 - number of current densities for this
c                                         element (rec length)
c                                   = 0 - maximum number of current densities
c                                         defined for any element
c                                         (max rec length)
c                 = DB_NUMDEFINED,
c                 = DB_MAXDEFINED  - return total number of current densities
c                                      defined in model
c                                   NOTE: both DB_NUMDEFINED and DB_MAXDEFINED
c                                         produce the same functionality

c     output arguments:
c        edciqr   (int,func,out)    - the returned value of edciqr is based on
c                                        setting of key
```

3.6.22. 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 ***

c     typ=int,dp,log,chr,dcp   siz=sc,ar(n),func    intent=in,out,inout

c        ielem    (int,sc,in)       - element number
c     output arguments:

c                                     = 0 - element has no current densities
c                                           defined
c                                     > 0 - number of element current
c                                           densities defined
c        value    (dp,ar(*),out)    -  element current densities

c                                     NOTE: If a value is not defined, it will
c                                           be a very small number (2**-100)
```

3.6.23. edcput Subroutine (Storing an Element Current Density)

```
*deck,edcput
      subroutine edcput (ielem,n,value)
c *** primary function:    store element current densities

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

c     input arguments:
c        ielem    (int,sc,in)        - element number
c        n        (int,sc,in)        - the number of current densities to store
c        value    (dp,ar(n),in)      - element current densities
```

```
c     output arguments:
c        none
c                                      NOTE: If a value is not defined, a very
c                                         small number should be used (2**-100)
```

3.6.24. 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
```

3.6.25. evdiqr Function (Getting Information About Element Virtual Displacements)

```
*deck,evdiqr
      function evdiqr (ielem,key)
c *** primary function:    get information about element virt disps

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

c     input arguments:
c        ielem    (int,sc,in)       - element number or zero (see below)
c        key      (int,sc,in)       - key as to the information needed
c                 = 1 or DB_MAXRECLENG - return element virt disps info
c                         for ielem > 0 - number of virt disps defined for this
c                                         element (rec length)
c                                   = 0 - maximum number of virt disps defined
c                                         for any element (max rec length)
c                 = DB_NUMDEFINED,
c                 = DB_MAXDEFINED  - return total number of virt disps defined
c                                    in model
c                                  NOTE: both DB_NUMDEFINED and DB_MAXDEFINED
c                                         produce the same functionality

c     output arguments:
c        evdiqr   (int,func,out)    - the returned value of evdiqr is based on
c                                     setting of key
```

3.6.26. 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:
c        evdget   (int,func,out)    - status of element.
c                                     = 0 - no virt disps defined for this
c                                           element
c                                     > 0 - number of  element virtual
c                                           displacements
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
c        value    (dp,ar(*),out)    -  element virtual displacements
c
c                                     NOTE: If a value is not defined, it will
c                                           be a very small number (2**-100)
```

3.6.27. 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 ***

c     input arguments:
c        ielem    (int,sc,in)        - element number
c        n        (int,sc,in)        - the total number of values
c        value    (dp,ar(n),in)      - element virtual displacements

c     output arguments:  none
c                                     NOTE: If a value is not defined, a very
c                                        small number should be used (2**-100)
```

3.6.28. eimiqr Function (Getting Information About Element Impedances)

```
*deck,eimiqr
      function eimiqr (ielem,iface,key)
c *** primary function: get information about element impedences

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

c     input arguments:
c        ielem    (int,sc,in)       - element number for inquire.
c                                        should be zero for key=DB_NUMDEFINED,
c                                        DB_MAXDEFINED or DB_MAXRECLENG
c        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
c                                        meaning (see below)
c        key      (int,sc,in)       - key as to the information needed
c                 =  1             - return impedence mask for element
c                 =  5             - return number of impedences for this
c                                       element face
c                 = DB_NUMDEFINED,
c                 = DB_MAXDEFINED  - return value is based on setting of iface
c                                      NOTE: both DB_NUMDEFINED and
c                                      DB_MAXDEFINED produce the same
c                                      functionality
c                              iface = 0 - return number of surface loads defined
c                                          in model
c                                    = 1-6 - return number of pressure loads
c                                        defined for this element. (rec length)
c                                        NOTE:  only 1-6 is valid, but this
c                                        routine simply checks that iface is in
c                                        the range.  The actual value of iface
c                                        does not matter in this case.
c                 = DB_MAXRECLENG  - return the maximum number of element
c                                      impedences defined for any element
c                                      (max rec length)

c     output arguments:
c        eimiqr   (int,func,out)    - the returned value of eimiqr is based on
c                                       setting of key.
```

3.6.29. eimget Function (Getting an Element Face Impedance)

```
*deck,eimget
       function eimget (elem,iface,value)
c *** primary function:    get an element face impedance (in raw form)

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

c     input arguments:
c        elem     (int,sc,in)       - element number
c        iface    (int,sc,in)       - face number (1-6)

c     output arguments:
c        eimget   (int,func,out)    - status of element.
c                                     =-1 - element has no impedance
c                                     = 0 - this element face has no impedance
c                                     > 0 - number of values defined
c        value    (dp ,ar(*),out)   - the element impedances
```

3.6.30. 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 ***

c     input arguments:
c        ielem    (int,sc,in)       - element number
c        iface    (int,sc,in)       - face number (1-6)
c        nval     (int,sc,in)       - number of values to put
c        value    (dp ,ar(nval),in) - the element impedances (real,imag)

c     output arguments:  none
```

3.6.31. 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 ***

c     input arguments:
c        ielem     (int,sc,in)       - element number
c        iface     (int,sc,in)       - face number
c                                       = 0 - delete all impedances on this
c                                             element
c                                       = 1-6 - delete impedance on this face

c     output arguments:  none
```

3.6.32. esfiqr Function (Getting Information About Element Surface Stress Data)

```
*deck,esfiqr
      function esfiqr (ielem,key)
c *** primary function:    get information about element surface stress data
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

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

c     input arguments:
c        ielem     (int,sc,in)      - element number (or zero, see below)
c        key     (int,sc,in)        - key as to the information needed
c                 = 1 - return info about surface stress
c                         ielem > 0 - return number of surface stresses on this
c                                     element (rec length)
c                                = 0 - return maximum number of surface stresses
c                                     on any element (max rec length)
c                 = DB_NUMDEFINED - return the number of surface stresses
c                                   defined in model

c     output arguments:
c        esfiqr   (int,func,out)    - the returned value of esfiqr is based on
c                                     setting of key
```

3.6.33. 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 ***

c     input arguments:
c        ielem    (int,sc,in)       - element number

c     output arguments:
c        esfget   (int,func,out)    - status of element.
c                                     = 0 - element undefined
c                                     > 0 - number of values returned
c        value    (dp,ar(*),out)    - element surface stress data.
```

3.6.34. 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:
c        ielem     (int,sc,in)       - element number
c        nval      (int,sc,in)       - the total number of values
c                                      (19 * number of stress faces)
c                                      There is a max of 2 stress faces
c        value    (dp,ar(nval),in)   - the values

c     output arguments:  none
```

3.6.35. esfdel Subroutine (Deleting an Element's Surface Stress Data)

```
*deck,esfdel
      subroutine esfdel (ielem)
c *** primary function:    delete element surface stress data

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

```
c     input arguments:
c        ielem    (int,sc,in)       - element number
c                                      = 0 - delete for all defined elements

c     output arguments:  none.
```

3.6.36. 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 ***

c     input arguments:
c        ielem     (int,sc,in)       - element number
c        iface     (int,sc,in)       - face number
c                                      = 0 - all flagged surfaces
c                                      = 1-6 - this flagged surface

c     output arguments:  none.
```

3.6.37. 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 ***

c     input arguments:
c        ielem    (int,sc,in)       - element number
c        iface    (int,sc,in)       - face number (1-6)

c     output arguments:
c        efsget   (int,func,out)    - status of element.
c                                      =-1 - no values for this element
c                                      = 0 - zero flagged surfaces defined
c                                      > 0 - number of values defined
c        value    (dp ,ar(*),out)   - the element flagged surfaces
```

3.6.38. 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.
c                                        should be zero for key=DB_NUMDEFINED,
c                                        DB_MAXDEFINED or DB_MAXRECLENG
c        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
c                                        meaning (see below)
c        key      (int,sc,in)       - key as to the information needed
c                 =  1             - return flagged surfaces mask for element
c                 =  5             - return number of flagged surfaces for this
c                                      element face
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
c                 = DB_NUMDEFINED,
c                 = DB_MAXDEFINED  - return value is based on setting of iface
c                                    NOTE: both DB_NUMDEFINED and
c                                    DB_MAXDEFINED produce the same
c                                    functionality
c                             iface = 0 - return total number of pressures,
c                                         convections, etc defined in model
c                                   = 1-6 - return number of flagged surfaces
c                                       defined for this element. (rec length)
c                                       NOTE:  only 1-6 is valid, but this
c                                       routine simply checks that iface is in
c                                       the range.  The actual value of iface
c                                       does not matter in this case.
c                 = DB_MAXRECLENG  - return maximum number of flagged surfaces
c                                       for any element (max rec length)

c     output arguments:
c        efsiqr   (int,func,out)    - the returned value of efsiqr is based on
c                                      setting of key.
```

3.6.39. 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 ***

c     input arguments:
c        ielem    (int,sc,in)       - element number
c        iface    (int,sc,in)       - face number (1-6)
c        nval     (int,sc,in)       - number of values to put
c        value    (dp ,ar(nval),in) - the element flagged surface values

c     output arguments:  none.
```

3.7. Results Information Routines

3.7.1. dspiqr Function (Getting Information About Nodal Results)

```
*deck,dspiqr
      function dspiqr (node,key)
c *** primary function: get information about nodal results

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

c     input arguments:
c        node    (int,sc,in)       - node number
c                                    > 0 - return result mask at this node
c                                    = 0 - return number of calculated
c                                          displacements in model
c        key     (int,sc,in)       - key as to the information needed
c                                    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
```

3.7.2. 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 ***

c     input arguments:
c        node     (int,sc,in)       - node number
c        ndf      (int,sc,in)       - number of results requested
c        idf      (int,ary(ndf),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    CONC=17           PRES=19, TEMP=20, VOLT=21, MAG =22, ENKE=23, ENDS=24
c    EMF =25, CURR=26  SP01=27, SP02=28, SP03=29, SP04=30, SP05=31, SP06=32
c                                (missing entries are spares)

c     output arguments:
c        value    (dp,ar(ndf),out)  - result values
```

3.7.3. 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 ***

c     input arguments:
c        node     (int,sc,in)       - node number
c        ndf      (int,sc,in)       - number of results to be stored
c        idf      (int,ary(ndf),in) - reference number for the DOF: (1-32)
c        value    (dp,ar(ndf),in)   -  displacement values

c     output arguments:  none
```

3.7.4. 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 ***

c     input arguments:
c        node     (int,sc,in)       - node number. (0 to delete DOF at all
c                                     nodes)
c        ndf      (int,sc,in)       - number of DOFs to delete (0 to delete
c                                     all DOFs)
c        idf      (int,ar(*),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
c         EMF =25, CURR=26           (missing entries are spares)

c     output arguments:  none
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

3.7.5. emsiqr Function (Getting Information About an Element's Miscellaneous Summable Data)

```
*deck,emsiqr
      function emsiqr (ielem,key)
c *** primary function:    get information about element misc summable data

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

c     input arguments:
c        ielem     (int,sc,in)      - element number (or zero, see below)
c        key     (int,sc,in)        - key as to the information needed
c                     =  1 - return info about misc summed data records
c                                    ielem > 0 - return number of misc summed
c                                                data items for this element
c                                                (record length)
c                                         = 0 - return maximum number of misc
c                                                summed data items on any
c                                                element (max record length)
c                     = DB_NUMDEFINED - return total number of misc summed data
c                                       items defined in model

c     output arguments:
c        emsiqr   (int,func,out)    - the returned value of emsiqr is based on
c                                     setting of key
```

3.7.6. 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 ***

c     input arguments:
c        ielem    (int,sc,in)       - element number

c     output arguments:
c        emsget   (int,func,out)    - status of element.
c                                     = 0 - element is undefined
c                                     > 0 - number of data items returned
c        value    (dp,ar(*),out)    - element misc summed data.

c                                    NOTE: the contents of this record is element
c                                        dependent.  See SMISC on ETABLE command
```

3.7.7. 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
c        nval      (int,sc,in)       - number of values to be stored
c        value    (dp,ar(nval),in)   - the misc summed data values

c     output arguements:  none
```

```
c                                  NOTE: the contents of this record is element
c                                      dependent.  See SMISC on ETABLE command
```

3.7.8. emsdel Subroutine (Deleting an Element's Miscellaneous Summable Data)

```
*deck,emsdel
      subroutine emsdel (ielem)
c *** primary function:    delete element misc summable data

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

c     input arguments:
c        ielem    (int,sc,in)       - element number
c                                     = 0 - delete data for all defined elements

c     output arguments:  none
```

3.7.9. 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 ***

c     input arguments:
c        ielem     (int,sc,in)      - element number (or zero, see below)
c        key     (int,sc,in)        - key as to the information needed
c                     =  1 - return info about element nodal forces
c                                    ielem > 0 - return number of element nodal
c                                                forces for this element
c                                                (record length)
c                                         = 0 - return maximum number of element
c                                                nodal forces on any element
c                                                (max record length)
c                     = DB_NUMDEFINED - return total number of element nodal
c                                       forces defined in model

c     output arguments:
c        enfiqr   (int,func,out)    - the returned value of enfiqr is based on
c                                     setting of key
```

3.7.10. 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:
c        ielem    (int,sc,in)       - element number

c     output arguments:
c        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
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

3.7.11. 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 ***

c     input arguments:
c        ielem     (int,sc,in)       - element number
c        nval      (int,sc,in)       - the total number of values
c                                       NOTE: There may be a maximum of 3 sets of
c                                     nodal forces in the record: static
c                                     forces, inertia forces, and damping forces
c        value    (dp,ar(nval),in)   - nodal force results

c     output arguments:  none
```

3.7.12. enfdel Subroutine (Deleting an Element's Nodal Forces)

```
*deck,enfdel
      subroutine enfdel (ielem)
c *** primary function:    delete element nodal forces data

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

c     input arguments:
c        ielem    (int,sc,in)       - element number
c                                       = 0 - delete for all defined elements

c     output arguments:  none
```

3.7.13. ensiqr Function (Getting Information About an Element's Nodal Stresses)

```
*deck,ensiqr
      function ensiqr (ielem,key)
c *** primary function: get information about element nodal stresses

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

c     input arguments:
c        ielem     (int,sc,in)      - element number (or zero, see below)
c        key     (int,sc,in)        - key as to the information needed
c                     =  1 - return info about element nodal stresses
c                                    ielem > 0 - return number of element nodal
c                                                stresses for this element
c                                                (record length)
c                                         = 0 - return maximum number of element
c                                                nodal stresses on any element
c                                                (max record length)
c                     = DB_NUMDEFINED - return total number of element
c                                       nodal stresses defined in model

c     output arguments:
c        ensiqr   (int,func,out)    - the returned value of ensiqr is based on
c                                       setting of key
```

3.7.14. 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 ***

c     input arguments:
c        ielem    (int,sc,in)       - element number

c     output arguments:
c        ensget   (int,func,out)    - status of element.
c                                     = 0 - element undefined
c                                     > 0 - number of nodal stresses
c                                           returned
c        value    (dp,ar(*),out)    - element nodal stresses

c                                NOTE: Stresses at each corner node in the order
c                                         X, Y, Z, XY, YZ, XZ, S1, S2, S3, SI, SE
c                                      For solid elements, stresses at each
c                                         corner node
c                                      For shell elements, stresses at each
c                                         corner node (first top durface, then
c                                         bottom)
c                                      For layered elements (w/KEYOPT(8)=0),
c                                         stresses for "first" layer at each
c                                         corner node (first at the bottom
c                                         surface of the bottom layer, then the
c                                         top surface of the top layer).
c                                         Stresses for "second" layer at each
c                                         corner node (first the bottom surface,
c                                         then the top surface for the layer with
c                                         the largest failure criteria).
c                                         The second layer is not present if
c                                         failure criteria were not used or are
c                                         not appropriate
c                                      For layered elements (w/KEYOPT(8)=1),
c                                         stresses 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 LS
c                                         item of ETABLE command.
```

3.7.15. 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 ***

c     input arguments:
c        ielem     (int,sc,in)       - element number
c        nval      (int,sc,in)       - the total number of values
c                                       (11*nnod*nface)
c        value    (dp,ar(nval),in)   - the stress values

c     output arguments:  none

c                                NOTE: Stresses at each corner node in the order
c                                         X, Y, Z, XY, YZ, XZ, S1, S2, S3, SI, SE
c                                      For solid elements, stresses at each
c                                         corner node
c                                      For shell elements, stresses at each
c                                         corner node (first top surface, then
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
c                                       bottom)
c                                    For layered elements (w/KEYOPT(8)=0),
c                                       stresses for "first" layer at each
c                                       corner node (first at the bottom
c                                       surface of the bottom layer, then the
c                                       top surface of the top layer).
c                                       Stresses for "second" layer at each
c                                       corner node (first the bottom surface,
c                                       then the top surface for the layer with
c                                       the largest failure criteria).
c                                       The second layer is not present if
c                                       failure criteria were not used or are
c                                       not appropriate
c                                    For layered elements (w/KEYOPT(8)=1),
c                                       stresses 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 LS
c                                       item of ETABLE command.
```

3.7.16. ensdel Subroutine (Deleting an Element's Nodal Stresses)

```
*deck,ensdel
      subroutine ensdel (ielem)
c *** primary function:    delete element nodal stresses

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

c     input arguments:
c        ielem    (int,sc,in)       - element number
c                                     = 0 - delete for all defined elements

c     output arguments:  none.
```

3.7.17. engiqr Function (Getting Information About an Element's Energies)

```
*deck,engiqr
      function engiqr (ielem,key)
c *** primary function: get information about element energies

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

c     input arguments:
c        ielem     (int,sc,in)      - element number (or zero, see below)
c        key     (int,sc,in)        - key as to the information needed
c                 =  1 - return info about element energies
c                             ielem > 0 - return number of element energies on
c                                         this element (rec length)
c                                   = 0 - return maximum number of element
c                                         energies on any element
c                                         (max rec length)
c                 = DB_NUMDEFINED - return the number of element energies
c                                    defined in model

c     output arguments:
c        engiqr   (int,func,out)    - the returned value of engiqr is based on
c                                        setting of key
```

3.7.18. 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 ***

c     input arguments:
c        ielem    (int,sc,in)       - element number

c     output arguments:
c        engget   (int,func,out)    - status of element.
c                                     =  0 - element undefined
c                                     = 11 - energies returned
c        value    (dp,ar(6),out)
c                                value(1) = volume of element
c                                     (2) = strain energy
c                                     (3) = dissipation energy
c                                     (4) = kinetic energy
c                                     (5) = plastic energy
c                                     (6) = creep energy
c                                     (7) = stabilization energy
c                                     (8) = strain energy density
c                                     (9) = thermal energy
c                                 (10-11) = spares
```

3.7.19. 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 ***

c     input arguments:
c        ielem     (int,sc,in)       - element number
c        nval      (int,sc,in)       - the total number of values to be stored
c                                        Must be 11!
c        value    (dp,ar(6),in)      - volume and energies
c                                value(1) = volume of element
c                                     (2) = strain energy
c                                     (3) = dissipation energy
c                                     (4) = kinetic energy
c                                     (5) = plastic energy
c                                     (6) = creep energy
c                                     (7) = stabilization energy
c                                     (8) = spare
c                                     (9) = thermal energy
c                                    (10) = viscous regularization energy for CZM
c                                    (11) = spares

c     output arguments:  none
```

3.7.20. 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:
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
c        ielem    (int,sc,in)       - element number
c                                     = 0 - delete for all defined elements

c     output arguments:  none.
```

3.7.21. 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 ***

c     input arguments:
c        ielem     (int,sc,in)      - element number (or zero, see below)
c        key     (int,sc,in)        - key as to the information needed
c                 =  1 - return info about nodal gradients
c                         for ielem > 0 - return number of nodal gradients on
c                                         this element (record length)
c                                     = 0 - return maximum number of nodal
c                                         gradients on any element
c                                         (maximum record length)
c                 = DB_NUMDEFINED - return the number of nodal gradients defined
c                                     in model

c     output arguments:
c        egriqr   (int,func,out)    - the returned value of egriqr is based on
c                                       setting of key
```

3.7.22. 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 ***

c     input arguments:
c        ielem    (int,sc,in)       - element number

c     output arguments:
c        egrget   (int,func,out)    - status of element.
c                                     = 0 - element undefined
c                                     > 0 - number of nodal gradients
c                                             returned
c        value    (dp,ar(*),out)    - element nodal gradients

c                                     Note: If a coupled field, a set of
c                                       gradients are stored in the following
c                                       order (as available): fluid, thermal,
c                                       electric, magnetic

c *** mpg egrget < pagend,magget<hsnget2: get elem gradient, H,
```

3.7.23. egrput Subroutine (Storing an Element's Nodal Gradients)

```
*deck,egrput
      subroutine egrput (ielem,nval,value)
c *** primary function:    store nodal gradients at an element.
```

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

c     input arguments:
c        ielem     (int,sc,in)       - element number
c        nval      (int,sc,in)       - the total number of values
c                                       (ndir*nnod*nscalr)
c        value    (dp,ar(nval),in)   - the gradient values

c                                     Note: If a coupled field, a set of
c                                       gradients are stored in the following
c                                       order (as appropriate): fluid, thermal,
c                                       electric, magnetic
c     output arguments:  none
```

3.7.24. egrdel Subroutine (Deleting an Element's Nodal Gradients)

```
*deck,egrdel
      subroutine egrdel (ielem)
c *** primary function:    delete element nodal gradients

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

c     input arguments:
c        ielem    (int,sc,in)       - element number
c                                       = 0 - delete for all defined elements

c     output arguments:  none.
```

3.7.25. eeliqr Function (Getting Information About an Element's Nodal Elastic Strains)

```
*deck,eeliqr
      function eeliqr (ielem,key)
c *** primary function: get information about element nodal elastic strains

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

c     input arguments:
c        ielem     (int,sc,in)      - element number (or zero, see below)
c        key     (int,sc,in)        - key as to the information needed
c                 =  1 - return info about elastic strains
c                             ielem > 0 - return number of nodal elasic strains
c                                         on this element (rec length)
c                                   = 0 - return maximum number of nodal elastic
c                                         strains on any element
c                                         (max rec length)
c                = DB_NUMDEFINED - return the number of nodal elastic strains
c                           defined in model

c     output arguments:
c        eeliqr   (int,func,out)    - the returned value of eeliqr is based on
c                                       setting of key
```

3.7.26. eelget Function (Getting an Element's Nodal Elastic Strains)

```
*deck,eelget
      function eelget (ielem,value)
c *** primary function:    get element nodal elastic strains.

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


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
c     input arguments:
c        ielem    (int,sc,in)       - element number

c     output arguments:
c        eelget   (int,func,out)    - status of element.
c                                     = 0 - element undefined
c                                     > 0 - number of nodal elastic strains
c                                           returned
c        value    (dp,ar(*),out)    - element nodal elastic strains


c                                NOTE: Strains at each corner node in the order
c                                         X, Y, Z, XY, YZ, XZ, EQV
c                                      For solid elements, strains at each
c                                         corner node
c                                      For shell elements, strains at each
c                                         corner node (first top durface, then
c                                         bottom)
c                                      For layered elements (w/KEYOPT(8)=0),
c                                         strains for "first" layer at each
c                                         corner node (first at the bottom
c                                         surface of the bottom layer, then the
c                                         top surface of the top layer).
c                                         Strains for "second" layer at each
c                                         corner node (first the bottom surface,
c                                         then the top surface for the layer with
c                                         the largest failure criteria).
c                                         The second layer is not present if
c                                         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
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 LEPEL
c                                         item of ETABLE command.
```

3.7.27. 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 ***

c     input arguments:
c        ielem     (int,sc,in)       - element number
c        nval      (int,sc,in)       - the total number of values
c                                       (7*nnod*nface)
c        value    (dp,ar(nval),in)   - nval strain values

c     output arguments:  none

c                                NOTE: Strains at each corner node in the order
c                                         X, Y, Z, XY, YZ, XZ, EQV
c                                      For solid elements, strains at each
c                                         corner node
c                                      For shell elements, strains at each
c                                         corner node (first top durface, then
c                                         bottom)
c                                      For layered elements (w/KEYOPT(8)=0),
c                                         strains for "first" layer at each
c                                         corner node (first at the bottom
c                                         surface of the bottom layer, then the
c                                         top surface of the top layer).
```

```
c                                       Strains for "second" layer at each
c                                       corner node (first the bottom surface,
c                                       then the top surface for the layer with
c                                       the largest failure criteria).
c                                       The second layer is not present if
c                                       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
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 LEPEL
c                                       item of ETABLE command.
```

3.7.28. eeldel Subroutine (Deleting an Element's Nodal Elastic Strains)

```
*deck,eeldel
      subroutine eeldel (ielem)
c *** primary function:    delete element elastic strains

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

c     input arguments:
c        ielem    (int,sc,in)       - element number
c                                     = 0 - delete for all defined elements

c     output arguments:  none.
```

3.7.29. epliqr Function (Getting Information About an Element's Nodal Plastic Strains)

```
*deck,epliqr
      function epliqr (ielem,key)
c *** primary function: get information about element nodal plastic strains

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

c     input arguments:
c        ielem     (int,sc,in)      - element number (or zero, see below)
c        key     (int,sc,in)        - key as to the information needed
c                 =  1 - return info about plastic strains
c                             ielem > 0 - return number of nodal plastic strains
c                                         on this element
c                                         (record length)
c                                    = 0 - return maximum number of nodal plastic
c                                         strains on any element
c                                         (max record length)
c                 = DB_NUMDEFINED - return the number of nodal plastic strains
c                                    defined in model

c     output arguments:
c        epliqr   (int,func,out)    - the returned value of epliqr is based on
c                                        setting of key
```

3.7.30. eplget Function (Getting an Element's Nodal Plastic Strains)

```
*deck,eplget
      function eplget (ielem,value)
c *** primary function:    get element nodal plastic strains.
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

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

c     input arguments:
c        ielem    (int,sc,in)       - element number

c     output arguments:
c        eplget   (int,func,out)    - status of element.
c                                     = 0 - element undefined
c                                     > 0 - number of nodal plastic strains
c                                           returned
c        value    (dp,ar(*),out)    - element nodal plastic strains

c                                 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
c                                       For shell elements, strains at each
c                                          corner node (first top durface, then
c                                          bottom)
c                                       For layered elements (w/KEYOPT(8)=0),
c                                          strains for "first" layer at each
c                                          corner node (first at the bottom
c                                          surface of the bottom layer, then the
c                                          top surface of the top layer).
c                                          Strains for "second" layer at each
c                                          corner node (first the bottom surface,
c                                          then the top surface for the layer with
c                                          the largest failure criteria).
c                                          The second layer is not present if
c                                          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
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 LEPPL
c                                          item of ETABLE command.
```

3.7.31. 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 ***

c     input arguments:
c        ielem     (int,sc,in)       - element number
c        nval      (int,sc,in)       - the total number of values
c                                       (6*nnod*nface)
c        value    (dp,ar(nval),in)   - the strain values

c     output arguments:  none
c                                 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
c                                       For shell elements, strains at each
c                                          corner node (first top durface, then
c                                          bottom)
c                                       For layered elements (w/KEYOPT(8)=0),
c                                          strains for "first" layer at each
c                                          corner node (first at the bottom
c                                          surface of the bottom layer, then the
c                                          top surface of the top layer).
c                                          Strains for "second" layer at each
c                                          corner node (first the bottom surface,
```

```
c                                       then the top surface for the layer with
c                                       the largest failure criteria).
c                                       The second layer is not present if
c                                       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
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 LEPPL
c                                       item of ETABLE command.
```

3.7.32. epldel Subroutine (Deleting an Element's Nodal Plastic Strains)

```
*deck,epldel
      subroutine epldel (ielem)
c *** primary function:    delete element plastic strains

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

c     input arguments:
c        ielem    (int,sc,in)       - element number
c                                     = 0 - delete for all defined elements

c     output arguments:  none.
```

3.7.33. 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 ***

c     input arguments:
c        ielem     (int,sc,in)      - element number (or zero, see below)
c        key     (int,sc,in)        - key as to the information needed
c                 =  1 - return info about creep strains
c                              ielem > 0 - return number of nodal creep strains
c                                          on this element
c                                          (record length)
c                                    = 0 - return maximum number of nodal creep
c                                          strains on any element
c                                          (max record length)
c                 = DB_NUMDEFINED - return the number of nodal creep strains
c                                    defined in model

c     output arguments:
c        ecriqr   (int,func,out)    - the returned value of ecriqr is based on
c                                        setting of key
```

3.7.34. 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 ***
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
c     input arguments:
c        ielem    (int,sc,in)       - element number

c     output arguments:
c        ecrget   (int,func,out)    - status of element.
c                                     = 0 - element undefined
c                                     > 0 - number of nodal creep strains
c                                           returned
c        value    (dp,ar(*),out)    - element nodal creep strains

c                                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
c                                      For shell elements, strains at each
c                                         corner node (first top durface, then
c                                         bottom)
c                                      For layered elements (w/KEYOPT(8)=0),
c                                         strains for "first" layer at each
c                                         corner node (first at the bottom
c                                         surface of the bottom layer, then the
c                                         top surface of the top layer).
c                                         Strains for "second" layer at each
c                                         corner node (first the bottom surface,
c                                         then the top surface for the layer with
c                                         the largest failure criteria).
c                                         The second layer is not present if
c                                         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
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 LEPCR
c                                         item of ETABLE command.
```

3.7.35. 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 ***

c     input arguments:
c        ielem     (int,sc,in)       - element number
c        nval      (int,sc,in)       - the total number of values
c                                       (6*nnod*nface)
c        value    (dp,ar(nval),in)   - the strain values

c     output arguments:  none
c                                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
c                                      For shell elements, strains at each
c                                         corner node (first top durface, then
c                                         bottom)
c                                      For layered elements (w/KEYOPT(8)=0),
c                                         strains for "first" layer at each
c                                         corner node (first at the bottom
c                                         surface of the bottom layer, then the
c                                         top surface of the top layer).
c                                         Strains for "second" layer at each
c                                         corner node (first the bottom surface,
c                                         then the top surface for the layer with
c                                         the largest failure criteria).
```

```
c                                       The second layer is not present if
c                                       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
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 LEPCR
c                                       item of ETABLE command.
```

3.7.36. ecrdel Subroutine (Deleting an Element's Nodal Creep Strains)

```
*deck,ecrdel
      subroutine ecrdel (ielem)
c *** primary function:    delete element creep strains

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

c     input arguments:
c        ielem    (int,sc,in)       - element number
c                                     = 0 - delete for all defined elements

c     output arguments:  none.
```

3.7.37. ethiqr Function (Getting Information About an Element's Nodal Thermal Strains)

```
*deck,ethiqr
      function ethiqr (ielem,key)
c *** primary function: get information about element nodal thermal strains

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

c     input arguments:
c        ielem     (int,sc,in)      - element number (or zero, see below)
c        key     (int,sc,in)        - key as to the information needed
c                 =  1 - return info about thermal strains
c                              ielem > 0 - return number of nodal thermal strains
c                                          on this element
c                                          (record length)
c                                    = 0 - return maximum number of nodal thermal
c                                          strains on any element
c                                          (max record length)
c                 = DB_NUMDEFINED - return the number of nodal thermal strains
c                                    defined in model

c     output arguments:
c        ethiqr   (int,sc,out)      - the returned value of ethiqr is based on
c                                        setting of key
```

3.7.38. ethget Function (Getting an Element's Nodal Thermal Strains)

```
*deck,ethget
      function ethget (ielem,value)
c *** primary function:    get element nodal thermal strains.
c                             also the volumetric swelling strain
c *** Notice - This file contains ANSYS Confidential information ***

c     input arguments:
c        ielem    (int,sc,in)       - element number
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
c     output arguments:
c        ethget   (int,func,out)    - status of element.
c                                     = 0 - element undefined
c                                     > 0 - number of nodal thermal strains
c                                           returned
c        value    (dp,ar(*),out)    - element nodal thermal strains

c                                NOTE: Strains at each corner node in the order
c                                      X, Y, Z, XY, YZ, XZ, epswel
c                                   For solid elements, strains at each
c                                      corner node
c                                   For shell elements, strains at each
c                                      corner node (first top durface, then
c                                      bottom)
c                                   For layered elements (w/KEYOPT(8)=0),
c                                      strains for "first" layer at each
c                                      corner node (first at the bottom
c                                      surface of the bottom layer, then the
c                                      top surface of the top layer).
c                                      Strains for "second" layer at each
c                                      corner node (first the bottom surface,
c                                      then the top surface for the layer with
c                                      the largest failure criteria).
c                                      The second layer is not present if
c                                      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
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.
```

3.7.39. ethput Subroutine (Storing an Element's Nodal Thermal Strains)

```
*deck,ethput
      subroutine ethput (ielem,nval,value)
c *** primary function:    store nodal thermal strains at an element.
c                           also the volumetric swelling strain

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

c     input arguments:
c        ielem     (int,sc,in)       - element number
c        nval      (int,sc,in)       - the total number of values
c                                      (7*nnod*nface)
c        value    (dp,ar(nval),in)   - the strain values

c     output arguments:  none
c                                NOTE: Strains at each corner node in the order
c                                      X, Y, Z, XY, YZ, XZ, epswel
c                                   For solid elements, strains at each
c                                      corner node
c                                   For shell elements, strains at each
c                                      corner node (first top durface, then
c                                      bottom)
c                                   For layered elements (w/KEYOPT(8)=0),
c                                      strains for "first" layer at each
c                                      corner node (first at the bottom
c                                      surface of the bottom layer, then the
c                                      top surface of the top layer).
c                                      Strains for "second" layer at each
c                                      corner node (first the bottom surface,
c                                      then the top surface for the layer with
c                                      the largest failure criteria).
c                                      The second layer is not present if
```

```
c                                        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
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.
```

3.7.40. ethdel Subroutine (Deleting an Element's Thermal, Initial, and Swelling Strains)

```
*deck,ethdel
      subroutine ethdel (ielem)
c *** primary function:    delete element thermal, initial, and
c                          swelling strains

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

c     input arguments:
c        ielem    (int,sc,in)       - element number
c                                     = 0 - delete for all defined elements

c     output arguments:  none.
```

3.7.41. euliqr Function (Getting Information About an Element's Euler Angles)

```
*deck,euliqr
      function euliqr (ielem,key)
c *** primary function: get information about element euler angles

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

c     input arguments:
c        ielem     (int,sc,in)      - element number (or zero, see below)
c        key       (int,sc,in)      - key as to the information needed
c                  =  1 - return info about element euler angles
c                              ielem > 0 - return number of euler angles on this
c                                          element
c                                          (record length)
c                                    = 0 - return maximum number of euler angles
c                                          on any element
c                                          (max record length)
c                  = DB_NUMDEFINED - return the number of element euler angles
c                                    defined in model

c     output arguments:
c        euliqr   (int,func,out)    - the returned value of euliqr is based on
c                                        setting of key
```

3.7.42. 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:
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
c        ielem    (int,sc,in)       - element number

c     output arguments:
c        eulget   (int,func,out)    - status of element.
c                                     = 0 - element undefined
c                                     > 0 - number of euler angle values
c                                           returned
c        value    (dp,ar(*),out)    - element euler angles

c                                  NOTE: For lower-ordered elements, rotations
c                                          at centroid
c                                        For higher-ordered elements, rotations
c                                          at each corner node
c                                        For layered shells, rotations at each
c                                          corner node, plus layer rotation angle
c                                          for each layer (real constant THETA)
c                                        For layered solids, rotation angles at
c                                          centroid, plus layer rotation angle
c                                          for each layer (real constant THETA)
c                                        For surface element, no euler angles
c                                          are saved
```

3.7.43. 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 ***

c     input arguments:
c        ielem     (int,sc,in)       - element number
c        nval      (int,sc,in)       - the total number of values
c                                       (3 * number of display nodes)
c        value    (dp,ar(nval),in)   - the euler angle values

c     output arguments:  none
c                                  NOTE: For lower-ordered elements, rotations
c                                          at centroid
c                                        For higher-ordered elements, rotations
c                                          at each corner node
c                                        For layered shells, rotations at each
c                                          corner node, plus layer rotation angle
c                                          for each layer (real constant THETA)
c                                        For layered solids, rotation angles at
c                                          centroid, plus layer rotation angle
c                                          for each layer (real constant THETA)
```

3.7.44. euldel Subroutine (Deleting an Element's Euler Angles)

```
*deck,euldel
      subroutine euldel (ielem)
c *** primary function:    delete element euler angles

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

c     input arguments:
c        ielem    (int,sc,in)       - element number
c                                       = 0 - delete for all defined elements

c     output arguments:  none.
```

3.7.45. efxiqr Function (Getting Information About Element Fluxes)

```
*deck,efxiqr
      function efxiqr (ielem,key)
c *** primary function:    get information about element fluxes

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

c     input arguments:
c        ielem     (int,sc,in)      - element number (or zero, see below)
c        key       (int,sc,in)      - key as to the information needed
c                 =  1 - return info about element fluxes
c                             ielem > 0 - return number of fluxes on this
c                                         element
c                                         (record length)
c                                    = 0 - return maximum number of fluxes
c                                         on any element
c                                         (max record length)
c                 = DB_NUMDEFINED - return the number of element fluxes defined
c                                    in model

c     output arguments:
c        efxiqr   (int,func,out)    - the returned value of efxiqr is based on
c                                     setting of key
```

3.7.46. 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 ***

c     input arguments:
c        ielem    (int,sc,in)       - element number

c     output arguments:
c        efxget   (int,func,out)    - status of element.
c                                     = 0 - element undefined
c                                     > 0 - number of nodal fluxes returned
c        value    (dp,ar(*),out)    - element nodal fluxes

c                                    Note: If a coupled field, a set of fluxes is
c                                          stored in the following order (as
c                                          available):  fluid, thermal,
c                                          electric, magnetic

c *** mpg efxget<pagend<paberrwb,edgzzx,panavg,papres,paterr: get ele nd flx, B
```

3.7.47. 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 ***

c     input arguments:
c        ielem     (int,sc,in)       - element number
c        nval      (int,sc,in)       - the total number of values
c                                      (ndir*nnod*nscalr)
c        value    (dp,ar(nval),in)   - the flux values

c     output arguments:  none
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
c                                  Note: If a coupled field, a set of fluxes is
c                                        stored in the following order (as
c                                        available):  fluid, thermal,
c                                        electric, magnetic
```

3.7.48. efxdel Subroutine (Deleting Element Fluxes)

```
*deck,efxdel
      subroutine efxdel (ielem)
c *** primary function:    delete element nodal fluxes

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

c     input arguments:
c        ielem    (int,sc,in)       - element number
c                                     = 0 - delete for all defined elements

c     output arguments:  none.
```

3.7.49. 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 ***

c     input arguments:
c        ielem     (int,sc,in)      - element number (or zero, see below)
c        key       (int,sc,in)      - key as to the information needed
c                 =  1 - return info about element local forces
c                              ielem > 0 - return number of local forces on this
c                                          element
c                                          (record length)
c                                    = 0 - return maximum number of local forces
c                                          on any element
c                                          (max record length)
c                 = DB_NUMDEFINED - return the number of element local forces
c                                    defined in model

c     output arguments:
c        elfiqr   (int,func,out)    - the returned value of elfiqr is based on
c                                        setting of key
```

3.7.50. 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 ***

c     input arguments:
c        ielem    (int,sc,in)       - element number

c     output arguments:
c        elfget   (int,func,out)    - status of element.
c                                     = 0 - element has no local nodal forces
c                                     > 0 - number of nodal forces returned
c        value    (dp,ar(*),out)    - element local nodal forces.
```

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
c *** mpg elfget<pagend<paberrwb,edgzzx,panavg,papres,paterr: get ele nd frc, F
```

3.7.51. 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 ***

c     input arguments:
c     variable (typ,siz,intent)     description
c        ielem     (int,sc,in)       - element number
c        nval      (int,sc,in)       - the total number of values
c                                       NOTE: There may be a maximum of 3 sets of
c                                      nodal forces in the record: static
c                                      forces, inertia forces, and damping forces
c        value    (dp,ar(nval),in)   - element local nodal forces

c     output arguments:  none
```

3.7.52. elfdel Subroutine (Deleting Element Local Forces)

```
*deck,elfdel
      subroutine elfdel (ielem)
c *** primary function:    delete element local forces

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

c     input arguments:
c        ielem    (int,sc,in)       - element number
c                                       = 0 - delete for all defined elements

c     output arguments:  none.
```

3.7.53. emniqr Function (Getting Information About Element Miscellaneous Non-summable Data)

```
*deck,emniqr
      function emniqr (ielem,key)
c *** primary function:    get information about element misc non-summable
c                          data

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

c     input arguments:
c        ielem     (int,sc,in)      - element number (or zero, see below)
c        key       (int,sc,in)      - key as to the information needed
c                 =  1 - return info about element misc non-summed data
c                             ielem > 0 - return number of data items on this
c                                         element
c                                         (record length)
c                                    = 0 - return maximum number of data items
c                                         on any element
c                                         (max record length)
c                 = DB_NUMDEFINED - return the number of element misc non-summed
c                                    data items defined in model

c     output arguments:
c        emniqr   (int,func,out)    - the returned value of emniqr is based on
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
c                                       setting of key
```

3.7.54. 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 ***

c     input arguments:
c        ielem    (int,sc,in)       - element number

c     output arguments:
c        emnget   (int,func,out)    - status of element.
c                                     = 0 - no non-summed misc data at this
c                                           element
c                                     > 0 - number of data items returned
c        value    (dp,ar(*),out)    - element misc non-summed data.

c                                    NOTE: the contents of this record is element
c                                        dependent.  See NMISC on ETABLE command
```

3.7.55. 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 ***

c     input arguments:
c        ielem     (int,sc,in)       - element number
c        nval      (int,sc,in)       - the total number of values
c        value    (dp,ar(nval),in)   - the misc. non-summed data items

c     output arguments:  none
c                                    NOTE: the contents of this record is element
c                                        dependent.  See NMISC on ETABLE command
```

3.7.56. emndel Subroutine (Deleting an Element's Miscellaneous Non-summable Data)

```
*deck,emndel
      subroutine emndel (ielem)
c *** primary function:    delete element misc non-summable data

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

c     input arguments:
c        ielem    (int,sc,in)       - element number
c                                     = 0 - delete for all defined elements

c     output arguments:  none.
```

3.7.57. ecdiqr 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 ***

c     input arguments:
c        ielem     (int,sc,in)      - element number (or zero, see below)
c        key       (int,sc,in)      - key as to the information needed
c                 =  1 - return info about element current densities
c                             ielem > 0 - return number of current densities on
c                                         this element
c                                         (record length)
c                                    = 0 - return maximum number of current
c                                         densities on any element
c                                         (max record length)
c                 = DB_NUMDEFINED - return the number of element current
c                                    densities defined in model

c     output arguments:
c        ecdiqr   (int,func,out)    - the returned value of ecdiqr is based on
c                                        setting of key
```

3.7.58. 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 ***

c     input arguments:
c        ielem    (int,sc,in)       - element number

c     output arguments:
c        ecdget   (int,func,out)    - status of element.
c                                     = 0 - element has no current densities
c                                     > 0 - number of calculated element
c                                           current densities
c        value    (dp,ar(*),out)    - calculated element current densities.

c                                    NOTE: current densities are in the order
c                                          X, Y, Z
```

3.7.59. 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 ***

c     input arguments:
c        ielem    (int,sc,in)        - element number
c        nval     (int,sc,in)        - the total number of values
c        value    (dp,ar(nval),in)   - calculated element current densities.

c     output arguments:   none
c                                    NOTE: current densities are in the order
c                                          X, Y, Z
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries and affiliates.

3.7.60. ecddel Subroutine (Deleting Element Current Densities)

```
*deck,ecddel
      subroutine ecddel (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                                     = 0 - delete for all defined elements

c     output arguments:  none.
```

3.7.61. enliqr Function (Getting Information About Element Nonlinear Tables)

```
*deck,enliqr
      function enliqr (ielem,key)
c *** primary function: get information about element nonlinear tables

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

c     input arguments:
c        ielem     (int,sc,in)      - element number (or zero, see below)
c        key       (int,sc,in)      - key as to the information needed
c                 =  1 - return info about element nonlinear tables
c                             ielem > 0 - return number of nonlinear tables for
c                                         this element
c                                         (record length)
c                                   = 0 - return maximum number of nonlinear
c                                         tables for any element
c                                         (max record length)
c                 = DB_NUMDEFINED - return the number of element nonlinear
c                                    tables defined in model

c     output arguments:
c        enliqr   (int,func,out)    - the returned value of enliqr is based on
c                                        setting of key
```

3.7.62. 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:
c        ielem    (int,sc,in)       - element number

c     output arguments:
c        enlget   (int,func,out)    - status of element.
c                                     = 0 - nonlinear tables undefined
c                                     > 0 - number of nonlinear tables defined
c        value    (dp ,ar(n),out)   - the element nonlinear tables.

c                                   NOTE: Nonlinear data at each node are in the
c                                         order SEPL, SRAT, HPRES, EPEQ, PSV,
c                                         PLWK, and 4 spares
c                                        For beam elements, the contents and
c                                         number of information is element
```

```
c                                       dependent.  See NLIN on ETABLE
c                                       command
```

3.7.63. enlput Subroutine (Storing an Element's Nonlinear Tables)

```
*deck,enlput
      subroutine enlput (ielem,n,temp)
c *** primary function:    store element nonlinear tables

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

c     input arguments:
c        ielem    (int,sc,in)       - element number
c        n        (int,sc,in)       - number of element nonlinear table values
c        temp     (dp ,ar(6),in)    - element nonlinear table,etc.

c     output arguments:  none.
c                                  NOTE: Nonlinear data at each node are in the
c                                        order SEPL, SRAT, HPRES, EPEQ, PSV,
c                                        PLWK, and 4 spares
c                                       For beam elements, the contents and
c                                        number of information is element
c                                        dependent.  See NLIN on ETABLE
c                                        command
```

3.7.64. enldel Subroutine (Deleting Element Nonlinear Tables)

```
*deck,enldel
      subroutine enldel (ielem)
c *** primary function:    delete element nonlinear tables

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

c     input arguments:
c        ielem    (int,sc,in)       - element number
c                                     = 0 - delete for all defined elements

c     output arguments:  none.
```

3.7.65. 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 ***

c     input arguments:
c        ielem     (int,sc,in)      - element number (or zero, see below)
c        key       (int,sc,in)      - key as to the information needed
c                 =  1 - return info about calculated element heat gens
c                        for ielem > 0 - return number of heat gens for
c                                        this element
c                                        (record length)
c                                     = 0 - return maximum number of heat gens
c                                        for any element
c                                        (max record length)
c                 = DB_NUMDEFINED - return the number of calculated element heat
c                                    generations defined in model
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries and affiliates.

```
c     output arguments:
c        ehciqr   (int,func,out)    - the returned value of ehciqr is based on
c                                        setting of key
```

3.7.66. 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 ***

c     input arguments:
c        ielem    (int,sc,in)       - element number

c     output arguments:
c        ehcget   (int,func,out)    - status of element.
c                                     = 0 - element undefined
c                                     > 0 - number of calculated element
c                                            heat generations
c        value    (dp,ar(*),out)    - calculated element heat generations.
```

3.7.67. 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 ***

c     input arguments:
c        ielem    (int,sc,in)        - element number
c        nval     (int,sc,in)        - the total number of values
c        value    (dp,ar(nval),in)   - calculated element heat generations.

c     output arguments:  none
```

3.7.68. ehcdel Subroutine (Deleting Element Calculated Heat Generations)

```
*deck,ehcdel
      subroutine ehcdel (ielem)
c *** primary function:    delete calculated element heat generations

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

c     input arguments:
c        ielem    (int,sc,in)       - element number
c                                     = 0 - delete for all defined elements

c     output arguments:  none.
```

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries and affiliates.

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries and affiliates.

Chapter 4: Subroutines for Your Convenience

This chapter describes 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:

4.1. Input and Output Abbreviations
4.2. General Subroutines
4.3. Vector Functions
4.4. Matrix Subroutines

4.1. 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 argument
  out - output argument
  inout - both an input and an output argument

4.2. General Subroutines

4.2.1. dptoch Subroutine (Retrieve Eight Characters From a Double Precision Variable)

```
*deck,dptoch
      subroutine dptoch (dp8,ch8)
c *** primary function:    retreive 8 characters from a dp variable

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

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
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
```

4.2.2. wrinqr Function (Obtain Information About Output)

```
*deck,wrinqr
      function wrinqr (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).
c ---           note that the whole common cannot be "saved/resumed".  cwa

c     typ=int,dp,log,chr,dcp   siz=sc,ar(n),func    intent=in,out,inout

c  input arguments:
c     variable (typ,siz,intent)    description                        wrcom name
c     key      (int,sc,in)
c                 = WR_PRINT       - print flag (kprint)                 prtkey
c                      wrinqr = 0 - no output
c                             = 1 - print
c                 = WR_OUTPUT      - current output unit number(iott) outfil
c                 = WR_MASTEROUT   - master output file               frstot
c                 = WR_COLINTER    - interactive columns per page     intcol
c                 = WR_COLBATCH    - batch columns per page           batcol
c                 = WR_LINEINTER   - interactive lines per page       intlin
c                 = WR_LINEBATCH   - batch lines per page             batlin
c                 = WR_COMMASEP    - 1 for comma separated output     CommaSep
c                 = WR_CHARITEM    - characters per output item       chrper
c                 = WR_CHARDECIMAL - characters past decimal          chrdec
c                 = WR_CHARINTEGER - characters in leading integer    chrint
c                 = WR_CHARTYPE    -                                  chrtyp
c                       wrinqr = 1 - using E format in output
c                              = 2 - using F format in output
c                              = 3 - using G format in output
c                 = WR_SUPTITLE    - tlabel supress key               keyhed
c                 = WR_SUPSUBTITLE - subtitle supress key            keytit
c                 = WR_SUPLSITER   - ls,iter id supress key          keyid
c                 = WR_NOTELINE    - note line supress key           keynot
c                 = WR_SUPCOLHEADER - column header supress key      keylab
c                 = WR_SUPCOLMAX   - column maximum supress key      keysum
c                 = WR_LISTOPT     - ListOpt from /output command    ListOpt

c  output arguments:
c     wrinqr   (int,func,out)      - the value corresponding to key
```

4.2.3. erinqr Subroutine (Obtaining Information from the Errors Common)

```
*deck,erinqr
      function erinqr (key)
c *** primary function:    obtain information from errors common
c
c *** Notice - This file contains ANSYS Confidential information ***
c
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
c  input arguments:
c     key      (int,sc,in)       - item to be returned
c                                  1=keyerr, 2=errfil,    3=numnot, 4=numwrn,
c                                  5=numerr, 6=numfat,    7=maxmsg, 8=lvlerr,
c                                  9=mxpcmd, 10=nercmd,  11=nertim,12=nomore,
c                                  13=eropen,14=ikserr,  15=kystat,16=mxr4r5,
c                                  17=mshkey,            19=opterr,20=flowrn,
c                                            22=noreport,23=pdserr,24=mxpcmdw
c                                  25=kystop,26=icloads, 27=ifkey
c
c ---- below definitions copied from errcom 7/92 for user information
c
c                            *** key number= ........................
c                     (see ansysdef for parameter definitions)          |
c                                                                       \/
c
co keyerr - master error flag                                    (ER_ERRORFLAG)
co errfil - errors file unit number                              (ER_ERRORFILE)
co numnot - total number of notes displayed                      (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)
co           -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)
co          (as per rsg/pft 5/1/92 - only for "info" calls
co           -1=message is timed before removal
co            0=message needs pick or keyboard before removal
co            1=message stays up untill replaced by another message
co nomore   display any more messages                            (ER_NOMOREMSG)
co           0=display messages
co           1=display discontinue message and stop displaying
co eropen - 0=errors file is closed                              (ER_FILEOPEN)
co          1=errors file is opened
co ikserr - 0=if interactive do not set keyerr                   (ER_INTERERROR)
c         - 1=if interactive set keyerr (used by mesher and tessalation)
co kystat - flag to bypass keyopt tests in the elcxx routines    (ER_KEYOPTTEST)
c           associated with status/panel info  inquiries.
c            0=do not bypass keyopt tests
c            1=perform all keyopt tests
c           also flag to bypass setting of _STATUS upon resume
co mxr4r5 - mixed rev4-rev5 input logic (*do,*if,*go,*if-go)     (ER_MIXEDREV)
c            (used in chkmix called from rdmac)
c            1=rev5 found (*do,*fi-then-*endif)
c            2=rev4 found (*go,:xxx,*if,....,:xxx)
c            3=warning printed. do not issue any more.
co mshkey - cpu intensive meshing etc. this will cause           (ER_MESHING)
c           "nertim (11)" to be set to -1 for "notes", 1 for "warnings",
c           and 0 for "errors". checking of this key is done in "anserr".
c            0=not meshing or cpu intensive
c            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)
c           1=an error has happened in main ansys during opt looping
co flowrn - flag used by "floqa" as to list floqa.ans            (20)
c           0=list "floqa.ans"
c           1="floqa.ans" has been listed. do not list again.
co noreport- used in GUI for turning off errors due to strsub calls (22)
c           0=process errors as usual
c           1=do NOT report errors
co pdserr - 0=no error in main ansys during pds looping          (ER_PDSLOOPING)
c           1=an error has happened in main ansys during pds looping
co mxpcmdw- number of messages written to file.err for any one   (24)
co          command
c           0=write all errors to file.err
c           1=only write displayed errors to file.err
co icloads - key to forbid the iclist command from listing solution    (26)
c            data instead of the input data.
```

```
c           0=iclist is OK
c           1=do not permit iclist
co ifkey   - key on whether or not to abort during /input on error     (27)
c           0=do not abort
c           1=abort
c
co espare - spare integer variables
c
c  --- end of information from errcom
c
c  output arguments:
c     erinqr   (int,sc,out)      - value corresponding to key
c
c *** mpg erinqr < el117,el115,el126,el109,el53,el96,el97,edg?: get error stat
c
```

4.2.4. TrackBegin Subroutine (Beginning Tracking for a Subroutine Call)

```
*deck,TrackBegin
      subroutine TrackBegin (sub32)

c *****function: mark beginning of track ansys call

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

c  input arguments:
c     sub32   (char*(*),sc,in)    - name of subroutine being entered and left
c                                     (32 characters max)

c  output arguments:  none
```

4.2.5. TrackEnd Subroutine (Ending Tracking for a Subroutine Call)

```
*deck,TrackEnd
      subroutine TrackEnd (sub32)

c *****function: mark end of track ansys call

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

c  input arguments:
c     sub32     (char*(*),sc,in)  - name of subroutine being left
c                                   (32 characters max)

c  output arguments:  none
```

4.2.6. erhandler Subroutine (Displaying Program 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 ***

c  input arguments:
c
c               In order to make life for vismg easier,
c               do NOT use variables for the first four arguments
c
c     filein   (ch*40,sc,in)     - Filename used for character portion of
c                                  message ID (this is the file name of the
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
c                                 file which contains the source for this
c                                 routine)
c
c                                 if 'ErrorMessageProbe', then error was
c                                   generated on another processor (distributed
c                                   ANSYS). In that case, dperr contains the
c                                   message already made ASCII and expanded
c
c     msgid    (int,sc,in)       - Numeric portion of the message ID
c                                   1 - 9999, unique for each erhandler
c                                   call in the FILE.  Recommend using
c                                   a sequence, similar to format conventions,
c                                   i.e., 5000, 5010, 5020
c                                   if filein='ErrorMessageProbe', this is the
c                                     CPU # that originally generated the error
c     msglvl   (int,sc,in)       - level of error (same as lngerr)
c                                   0=no label (used for u/i pop-ups)
c                                  -1=no label (used for u/i pop-ups) timed
c                                     as a note message
c                                   1=note, 2=warning, 3=error, 4=fatal
c                                                      -3=error w/tech supp note
c                                                      -4=fatal w/tech supp note
c                                      (see lngerr.F for text of tech supp note)
c     lngstrng (ch*(*),sc,in)    - error message to display. use keywords
c                                   of %i %g %c %/ for formating (same as
c                                   lngerr)
c     dperr    (dp,ar(*),in)     - vector of data to display. contains both
c                                   integer and double precision data.
c                                   (same as lngerr)
c                                     if filein='ErrorMessageProbe', dperr
c                                     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
c                                   characters
```

4.2.7. 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 ****
c                       (if double interpolation is needed, see intrpt)
c
c *** Notice - This file contains ANSYS Confidential information ***
c
c     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
c     klog      (int,sc,in)       - interpolation type
c                                     = 0 - use linear interpolation
c                                     = 1 - use log-log interpolation
c                                       -- note:  there is no option yet for
c                                                 lin-log or log-lin
c     kppx      (int,sc,in)       - X value end of table signal
c                                     = 0 - a repeated x-value will signal the end
c                                           of the table
c                                     = 1 - a repeated x-value will not signal the
c                                           end of the table
c                                           (only known use = c evaluation)
c     kstpz     (int,sc,in)       - Y value end of table signal
c                                     = 0 - a yval of zero will not signal the end
c                                           of the table (e.g. stress fitting)
c                                     = 1 - a yval of zero will signal the end of
c                                           the table (in general, material
c                                           properties (exception: alpx))
c
c                         NOTE: the end of the table will be signaled thru
```

```
c                               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
c                               the previous one (ax(i) .lt. ax(i-1)).
c                               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
c                               beginning of the table are done as if they were
c                               done at the beginning of the table.
c
c     xval      (dp,sc,in)          - value of x with which to go into the table
c     ax        (dp,ar(*),in)       - table of x values, in ascending order
c     ay        (dp,ar(*),in)       - table of y values
c     nmax      (int,sc,in)         - maximum table size allowed
c
c  output arguments:
c     yval      (dp,sc,out)         - value of y which comes back from the table
c     kyoff     (int,sc,out)        - xval status flag
c                                     = 0 - xval in x range
c                                     = 1 - xval less than minimum x
c                                     = 2 - xval greater than maximum x
c
```

4.2.8. 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
c                       with or without a 3rd node
c
c *** Notice - This file contains ANSYS Confidential information ***
c
c  input arguments:
c     nnod      (int,sc,in)        - number of nodes (2 or 3)
c     xyz       (dp,ar(nx,*),in)   - coordinates (x,y,z down)
c     nx        (int,sc,in)        - row dimension of xyz array
c
c  output arguments:
c     tr        (dp,ar(3,3),in)    - transformation matrix
c
```

4.2.9. systop Subroutine (Stopping a 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
c *** Notice - This file contains ANSYS Confidential information ***
c
c  input arguments:
c     icode     (int,sc,in)        - stop error code (0<icode<127)
c                                      0 - normal exit
c                                      1 - stack overflow error
c                                      2 - stack level overflow
c                                      3 - stack pop below zero
c                                      4 - names do not match in stkpxp
c                                      5 - command line argument error
c                                      6 - unused (was: accounting file error)
c                                      7 - licensing failure
c                                      8 - indicated error or end-of-run
c                                     11 - error in user routine
c                                     12 - macro stop command
c                                     13 - job already running
c                                     14 - untrapped xox error
c                                     15 - anserr fatal error
c                                     16 - possible full disk
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
c                                  17 - possible corrupted or missing file
c                                  18 - Error in VM routines (corrupt db?)
c                                  21 - unauthorized code section entered
c                                  25 - unable to open x11 server
c                                  30 - quit signal
c                                  31 - failure to get signal in max time
c                                          (syhold)
c                                 >32 - system dependent error
c                                  35 - fatal error on another process
c                                       (distributed ANSYS)
c
c  output arguments:  none
```

4.3. Vector Functions

4.3.1. 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:
c     v1       (dp,ar(n),in)     - vector v1
c     v2       (dp,ar(n),in)     - vector v2
c     n        (int,sc,in)       - length of vectors v1 and v2

c  output arguments:
c     vdot     (dp,sc,out)       - dot product of v1 and v2

c
```

4.3.2. 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
c *** Notice - This file contains ANSYS Confidential information ***
c
c     ---- inc1 and inc2 must be positive!
c
```

4.3.3. vsum Function (Summing Vector Components)

```
*deck,vsum
      function vsum (va,n)
c *** primary function: sum the components of a vector

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

c  input arguments:
c     va       (dp,ar(n),in)     - vector va
c     n        (int,sc,in)       - length of vector va

c  output arguments:
c     vsum     (dp,sc,out)       - vector sum
```

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

4.3.4. vmax Function (Retrieving the Maximum Vector Value at a Given Location)

```
*deck,vmax
      function vmax (v,n,locmax)
c *** primary function: get the biggest value and location in v
c *** Notice - This file contains ANSYS Confidential information ***
```

4.3.5. 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 *********
```

4.3.6. izero Function (Setting an Integer Vector to Zero)

```
*deck,IZERO
      subroutine izero (ivect,n)
c **********  set an integer vector to zero  **********
```

4.3.7. imove Function (Assigning Equal Values to Two Integer Vectors)

```
*deck,imove
      subroutine imove (i1,i2,n)
c **********  move a vector from one to another  **********
```

4.3.8. vzero Subroutine (Initializing a Vector to Zero)

```
*deck,vzero
      subroutine vzero (v,n)
c *** primary function:    initialize a vector to zero
c *** Notice - This file contains ANSYS Confidential information ***
```

4.3.9. vmove Subroutine (Moving One Vector into Another)

```
*deck,vmove
      subroutine vmove (v1,v2,n)
c *** primary function: copy v1 vector into another vector
c *** Notice - This file contains ANSYS Confidential information ***
```

4.3.10. 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 ***
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

4.3.11. vinit Subroutine (Assigning a Scalar Constant to a Vector)

```
*deck,vinit
      subroutine vinit (v,n,const)
c *** primary function: initialize a vector to a constant
c *** Notice - This file contains ANSYS Confidential information ***
```

4.3.12. 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 *** Notice - This file contains ANSYS Confidential information ***
```

4.3.13. 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 ***
```

4.3.14. vapb1 Subroutine (Combining Two Vectors in One)

```
*deck,vapb1
      subroutine vapb1 (a,b,n)
c *** primary function: add vector b to vector a and store in vector a
```

4.3.15. vapcb1 Subroutine (Multiplying a Vector to a Constant)

```
*deck,vapcb1
      subroutine vapcb1 (a,b,n,const)
c *** primary function: multiply vector b to constant, add to vector a,
c                       and store in vector a
```

4.3.16. 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 ***
```

4.3.17. 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 ***
```

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

4.3.18. vmult Subroutine (Multiplying a Vector by a Constant)

```
*deck,vmult
      subroutine vmult (v1,v2,n,const)
c *** primary function: multiply a vector by a constant
c *** Notice - This file contains ANSYS Confidential information ***
```

4.3.19. 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 ***
```

4.3.20. 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 ***

c     typ=int,dp,log,chr,dcp   siz=sc,ar(n)   intent=in,out,inout

c  input arguments:
c     a        (dp,ar(3),in)     - first vector to be cross-multiplied
c     b        (dp,ar(3),in)     - second vector to be cross-multiplied

c  output arguments:
c     c        (dp,ar(3),out)    - resulting vector
c
c
```

4.3.21. vnorme Subroutine (Normalizing a Three-Component Vector)

```
*deck,vnorme
      subroutine vnorme (iel,v)
c primary function:  normalize a vector to unit length
c  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 ***

c  input arguments:
c     iel      (int,sc,inout)    - element number
c     v        (dp,ar(3),inout)  - vector to be normalized

c  output arguments:
c     iel      (int,sc,inout)    - if 0, vector has zero length
c     v        (dp,ar(3),inout)  - normalized vector
```

4.3.22. vnorm Subroutine (Normalizing a Vector to Unit Length)

```
*deck,vnorm
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
      subroutine vnorm (v,n)
c *** primary function: normalize a vector to unit length

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

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
```

4.3.23. 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 ***

c     input arguments:
c        node     (int,sc,in)       - node number for operation.

c     output arguments:
c        ndgxyz   (int,sc,out)      - status of node.
c                                       0=node is undefined.
c                                      -1=node is unselected.
c                                       1=node is selected.
c        xyz      (dp,ar(3),out)    - vector containing x,y,z
```

4.3.24. ndpxyz Subroutine (Storing X,Y,Z for a Node)

```
*deck,ndpxyz
      subroutine ndpxyz (node,xyz)
c *** primary function:    store x,y,z vector for a node.

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

c     input arguments:
c        node     (int,sc,in)       - node number for operation.
c        xyz      (dp,ar(3),in)     - vector containing x,y,z
c                                       (vector should be in global system)

c     output arguments:  none
```

4.4. Matrix Subroutines

4.4.1. maxv Subroutine (Multiplying a Vector by a Matrix)

```
*deck,maxv
      subroutine maxv (a,v,w, nr,nc)
c *** primary function: multiply a matrix by a vector

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

c  input arguments:
c     a        (dp,ar(nr,*),in)  - matrix a
c     v        (dp,ar(*),in)     - vector v
c     nr       (int,sc,in)       - number of rows in matrix a
```

```
c     nc       (int,sc,in)       - number of columns to multiply in matrix a

c  output arguments:
c     w        (dp,ar(*),out)    - product vector w
c
c *** mpg w = A v : A(nr,nc) : matrix vector product
c
```

4.4.2. 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:
c     a        (dp,ar(nr,nc),in) - matrix a
c     v        (dp,ar(nc),inout) - vector v
c     nr       (int,sc,in)       - number of rows in matrix a
c                                  *** nr limited to 60 ***
c     nc       (int,sc,in)       - number of columns to multiply in matrix a

c  output arguments:
c     v        (dp,ar(nr),inout) - product, stored in vector v
c
c *** mpg v = A v : A(nr,nc) : matrix vector product, max 60 rows
c
```

4.4.3. 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 ***

c  input arguments:
c     a        (dp,ar(nr,*),in)   - matrix a (first dimension must = nr)
c     v        (dp,ar(nv),in)     - vector v (nv must be greater or equal
c                                                 to nr)
c     nr       (int,sc,in)        - first dimension and number of active
c                                      rows of the untransposed matrix a
c                                      (also the number of active rows
c                                       of vector v)
c     nc       (int,sc,in)        - number of columns of the untransposed
c                                      matrix a
c                                      (also the number of computed items
c                                      in the product vector w)
c                                      if negative, accumulate

c  output arguments:
c     w        (dp,ar(na,*),out)  - product vector w
c
c
c *** mpg A(nr,nc) : matrix transpose vector product
c         w =     A+ v : if nr > 0
c         w = w + A+ v : if nr < 0
c
```

4.4.4. matxv1 Subroutine (Multiplying a Vector by a Full Transposed Matrix)

```
*deck,matxv1
      subroutine matxv1 (a,v, nr,nc)
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
c *** primary function: multiply vector by full transposed matrix

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

c  input arguments:
c     a        (dp,ar(nr,*),in)   - matrix a
c     v        (dp,ar(nr),inout)  - vector v
c     nr       (int,sc,in)        - number of rows in matrix (un-transposed)
c     nc       (int,sc,in)        - number of columns in matrix (un-transposed)
c                                   *** nc limited to 60 ***

c  output arguments:
c     v        (dp,ar(nc),inout)  - product, stored in vector v

c
c *** mpg A(nr,nc) : matrix transpose vector product
c         v = A+ v : max 60 nc
```

4.4.5. 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)     t means transpose

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

c  input arguments:
c     a        (dp,ar(na,*),in)   - matrix a
c     b        (dp,ar(nb,*),in)   - matrix b
c     na       (int,sc,in)        - number of rows in matrix a
c     nb       (int,sc,in)        - number of rows in matrix b
c     nc       (int,sc,in)        - number of rows in matrix c
c     n1       (int,sc,in)        - number of rows in matrix c to fill
c     n2       (int,sc,in)        - number of columns in matrix c to fill
c     n3       (int,sc,in)        - number of rows in matrix a and
c                                   number of rows of matrix b
c                                   to work with (the two need
c                                   to be the same for the inner product)
c                                   if n3 is negative, accumulate results in c

c  output arguments:
c     c        (dp,ar(nc,*),out)  - product matrix c

c *** mpg C =     A+ B     if n3 > 0
c         C = C + A+ B     if n3 < 0
c         A(na,*) B(nb,*) C(nc,*)  C:minor n1 * n2  n3: dot length
c
```

4.4.6. 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 ***

c     typ=int,dp,log,chr,dcp   siz=sc,ar(n)   intent=in,out,inout

c  input arguments:
c     a        (dp,ar(*),in)        - vector to be multiplied by itself to
c                                     generate an nxn square matrix
c                                      (a by a-transposed)
c     c        (dp,ar(nc,*),inout) - matrix to be accumulated onto
c     nc       (int,sc,in)         - number of rows in the c matrix
```

```
c     n        (int,sc,in)         - size of square matrix
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
c
```

4.4.7. 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
c *** Notice - This file contains ANSYS Confidential information ***
c  input arguments:
c     a        (dp,ar(na,*),in)    - matrix a
c     b        (dp,ar(nb,*),in)    - matrix b (must be square,
c                                           and maximum dimension is (15,15)
c     c        (dp,ar(nc,*),inout)- matrix c (see output)
c     na       (int,sc,in)         - number of rows in matrix a
c     nb       (int,sc,in)         - number of rows in matrix b
c     nc       (int,sc,in)         - number of rows in matrix c
c     n1       (int,sc,in)         - number of rows in matrix a and
c                                    number of rows of matrix b
c                                    to work with (the two need
c                                    to be the same for the inner product)
c     n2       (int,sc,in)         - number of columns in matrix c to fill
c     con      (dp,sc,in)          - multiplier on product added to sum

c  output arguments:
c     c        (dp,ar(nc,*),inout)- c = c + con*at*b*a
c     work     (dp,ar(n2,*),out)   - at*b (this byproduct is occasionally useful)

c *** mpg C = C + con A+ B A  A(na,*) B(nb,*) C(nc,*)  C:minor n1 * n2
c     see matbabd for block diagonal
c
```

4.4.8. 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 ***

c     typ=int,dp,log,chr,dcp   siz=sc,ar(n)   intent=in,out,inout

c  input arguments:
c     a        (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     n        (int,sc,in)         - size of matrix to be processed

c  output arguments:
c     a        (dp,ar(nd,*),inout) - matrix that has its lower triangular part
c                                     copied to its upper triangular part
c
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

4.4.9. mctac Subroutine (Transposing a symmetric matrix)

```
*deck,mctac
      subroutine mctac (a,na,c,nc,nold,nnew)
c  **** function: do  a = c(transpose) * a * c ,  where a is symmetric  **

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


c  input arguments:
c     a        (dp,ar(na,na),inout) matrix to be pre and post multiplied
c                                     (part operated on must be
c                                     square(nold x nold) and symmetric)
c     na       (int,sc,in)          first dimension of the a matrix
c     c        (dp,ar(nc,nnew),in)  matrix to pre and post multiply a by
c                                     (part used may be rectangular(nold x nnew))
c     nc       (int,sc,in)          first dimension of the c matrix
c     nold     (int,sc,in)          size of part of 'A' matrix that is
c                                     to be processed(input size).   maximum = 64
c     nnew     (int,sc,in)          size of part of 'A' matrix that
c                                     results from this operation(output size).
c                                                                    maximum = 64

c  output arguments:
c     a        (dp,ar(na,na),inout) resulting matrix
c                                     (still square(nnew x nnew) and symmetric).
```

4.4.10. 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 ***

c  input arguments:
c     variable (typ,siz,intent)    description
c     zs       (dp,ar(nz,nz),inout) - matrix to be transformed
c     tr       (dp,ar(ntr,ntr),in)  - transformation matrix
c     nz       (int,sc,in)          - dimensioned size of zs matrix
c     ntr      (int,sc,in)          - dimensioned size of tr matrix
c     nrow     (int,sc,in)          - number of rows of zs matrix to transform
c     irot     (int,sc,in)          - block size to transform(size of tr matrix)

c  output arguments:
c     variable (typ,siz,intent)    description
c     zs       (dp,ar(nz,nz),inout) - transformed matrix
```

4.4.11. symeqn Subroutine (Solving Simultaneous Linear Equations)

```
*deck,symeqn
      function symeqn (a,nd,n,nc,defFlag)
c
c primary function:  solve a set of simultaneous linear equations
c
c secondary functions: invert a matrix
c
c          NOTE:  this routine assumes that the matrix to be solved or
c                 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 ***
```

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
c
c  input arguments:
c     variable (typ,siz,intent)    description
c     a        (dp,ar(nd,*),inout) - matrix to be solved or inverted
c                                       second dimension must be at least:
c                                                                n + abs(nc)
c     nd       (int,sc,in)         - first dimension of the a matrix
c     n        (int,sc,in)         - number of equations
c     nc       (int,sc,in)         - number of additional columns.
c                                      if nc = +n or -n, invert n x n matrix and
c                                      put result in the n+1 to 2xn columns.
c                                      if nc is 0 or negative, nc will be reset to
c                                      n and then symeqn will set up identity
c                                      matrix after the input matrix, where the
c                                      result of the inversion will be put.
c                                      if nc is positive and less than n, do a
c                                      partial inversion. see example 1 below.
c     defFlag  (int,sc,in)         - flag indicating that incoming matrix MUST be:
c                                      -1 - negative definite
c                                       0 - positive or negative definite
c                                       1 - positive definite
c
c  output arguments:
c     variable (typ,siz,intent)    description
c     symeqn   (in,sc,out)         - 0 - non-singular matrix
c                                    1 - singular matrix
c                                    2 - near-singular matrix
c     a        (dp,ar(nd,*),inout) - results or inverted matrix.
c                                      starts in column n+1.
c                                      note:  original information is destroyed.
c
c   example 1:   Solve three simultaneous linear equations:
c                      i = symeqn (a(1,1),3,3,1)
c                   calling routine has a dimensioned as a(3,4)
c                   each equation has its 3 coefficents in the first 3 columns,
c                    and the constant term is in the fourth column.
c                   solution is in fourth column.
c
c   example 2:   Invert a 3x3 matrix:
c                      i = symeqn (a(1,1),3,3,-3)
c                   calling routine has a dimensioned as a(3,6)
c                   input matrix was input in first 3 columns
c                   output matrix in ouput in last 3 columns
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries and affiliates.

Appendix A. Creating External Commands in Linux

External commands allow you to add your own customized extensions to Mechanical APDL without relinking the program. You can create custom routines in C that access any of the Mechanical APDL API functions, link them into shared libraries using the supplied utilities, and execute the routines via the "external command" feature within Mechanical APDL. In addition, the program 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 Linux platforms. Refer to your ANSYS, Inc. 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/v182/ansys/customize/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).

```
#include "cAnsInterface.h"
#include "CAnsQuery.h"
/*
-------------------------  Function Description --------------------
extfnc
  int extfnc(uecmd)
  char *uecmd;

Purpose:
     Demonstrate C API entry function for an external command.

Parameters:
     Input
     ----------------------------
```

```
       uecmd
         The ANSYS external command string.

     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:

```
gen_share [-h] [-64] shared_object_name
```

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
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

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 Mechanical APDL 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 Mechanical APDL. 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_
```

Mechanical APDL also makes use of external commands, and places its own shared libraries and the associated external table file in the `/ansys_inc/v182/ansys/lib/<platform>` directory (where *`<platform>`* is the directory specific to your computing platform, such as `/linx64`).

Mechanical APDL loads external commands as follows:

- Checks the `ans_ext.tbl` file in the `/ansys_inc/v182/ansys/lib/<platform>` directory and loads any external commands referenced there.
- 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. 346)).

If you designate a command name that has the same first four characters as a command listed in the `/ansys_inc/v182/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 external table file to make sure you have no external command name conflicts. *Do not* modify the `/ansys_inc/v182/ansys/lib/<plat-`

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

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 Mechanical APDL is executed.

A.1.4. Setting the ANSYS_EXTERNAL_PATH Environment Variable

Before launching Mechanical APDL, 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 Mechanical APDL 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.

```
External Command Mappings:

 Command   Library                              Function         Accessed?
  *-*-*-*
```

Note

The **/CLEAR** command also closes/resets all external command shared libraries.

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries 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 Visual Studio 2012 Professional Update 4.
- 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\V182\customize\include\cAnsInterface.h`, `cAnsQuery.h`, and `cAnspick.h`. You can then execute these functions via the "external command" feature within Mechanical APDL. In addition, the program 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).

```
#include <windows.h>
#include "cAnsInterface.h"
#include "CAnsQuery.h"

/*
------------------------------  Function Description -----------------------
extfnc
  int extfnc(uecmd)
  char *uecmd;

Purpose:

     Demonstrate C API entry function for an external command.

Parameters:

     Input
```

```
      ----------------------------
        uecmd
          The ANSYS external command string.

      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 Visual Studio Project

The steps for building a Visual Studio project are demonstrated in the example at the end of this appendix. See Example: Creating an External Command Using Visual Studio 2010 (p. 352).

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 Mechanical APDL to access. There is no limit to the number of entries. The file entries have the following format:

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
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 Mechanical APDL. 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
```

Mechanical APDL 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\V182\ansys\lib\<platform>` directory (where *`<platform>`* is the directory specific to your computing platform, such as `\winx64`). The program loads external commands in the following order:

- Checks the `ans_ext.tbl` file in the `C:\Program Files\ANSYS Inc\V182\ansys\lib\<platform>` directory and loads any external commands referenced there.
- 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. 351)).

If you designate a command name that has the same first four characters as a command listed in the `C:\Program Files\ANSYS Inc\V182\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 external table file to make sure you have no external command name conflicts. Do not modify the `C:\Program Files\ANSYS Inc\V182\ansys\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 Mechanical APDL is executed.

B.1.6. Setting the ANSYS_EXTERNAL_PATH Environment Variable

Before launching Mechanical APDL, 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
```

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

B.1.7. Using External Commands

To call an external command, issue it as you would any other Mechanical APDL command. 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.

```
External Command Mappings:
 Command   Library                          Function         Accessed?
  *-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-
 ~excmd    /home/mydir/mycode/mycommand.so     excmd            YES
```

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.

B.1.10. Example: Creating an External Command Using Visual Studio 2012 Professional

An example for setting up an external command using Windows Visual Studio 2012 Professional Update 4 is provided on the installation media. To run this example, perform the following steps.

1. Go to the `Program Files\ANSYS Inc\V182\ANSYS\custom\user\winx64\ExtCmd` directory.
2. Open the Visual Studio 2012 Professional solution file `extcmd.sln` (double click the file).
3. From the Visual Studio 2012 Professional menu, click on `Build->Rebuild Solution`.

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

4. Exit Visual Studio 2012 Professional.

5. Double-click on `runextcmdtest.bat` to run Mechanical APDL and test the external command that was just compiled.

6. In the output window enter `~excmd`. You should see the following:

```
 BEGIN:
 ~excmd
   COMMAND SENT FROM EXTERNAL COMMAND
  Max Node Number = 0

  *** NOTE ***                            CP =       0.625   TIME= 14:24:33
  Command= ~excmd was processed as an external command which is a
  non-standard use of the ANSYS program.
```

7. Enter `/exit,nosave` to exit the program.

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries and affiliates.

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. 187).

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 the **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 (S*xx*, S*yy*) and plastic strain components (EP*xx*, EP*yy*) 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
mat2=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
```

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
! 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,
tb,state,mat2,,8                    ! define 8 state variables

! 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
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
  0.28750       -0.110968     -0.110968       1525.07       1525.07
  0.45625       -0.814415     -0.814415       1536.67       1536.67
  0.66204        -1.73160      -1.73160       1548.95       1548.95
  0.89592        -1.86240      -1.86240       1561.97       1561.97
   1.0000       -0.176924E-01 -0.176924E-01   1569.16       1569.16


         ***** ANSYS POST26 VARIABLE LISTING *****


    TIME          1 EPPLX       2 EPPLX       1 EPPLY       2 EPPLY
                  EPX_BISO      EPX_USER      EPY_BISO      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,
     &                   ldstep,isubst,keycut,
     &                   nDirect,nShear,ncomp,nStatev,nProp,
     &                   Time,dTime,Temp,dTemp,
     &                   stress,ustatev,dsdePl,sedEl,sedPl,epseq,
     &                   Strain,dStrain, epsPl, prop, coords,
     &                   var0, defGrad_t, defGrad,
     &                   tsstif, epsZZ,
     &                   var1, var2, var3, var4, var5,
     &                   var6, var7, var8)
c*************************************************************************
c     *** primary function ***
c
c           user defined material constitutive model
c
c      Attention:
c           User must define material constitutive law properly
c           according to the stress state such as 3D, plane strain
c           and axisymmetry, plane stress and 3D/1D beam.
c
c           a 3D material constitutive model can use for
c           plane strain and axisymmetry cases.
c
c           When using shell elements, a plane stress algorithm
c           must be use.
c
c
c       The following demonstrates a USERMAT subroutine for
c       a plasticity model, which is the same as TB, BISO,
c       for different stress states.
c       See "ANSYS user material subroutine USERMAT" for detailed
c       description of how to write a USERMAT routine.
c
c       This routine calls four routines,
c       usermat3d.F, usermatps.F usermatbm.F and usermat1d.F, w.r.t.
c       the corresponding stress states.
c       Each routine can be also a usermat routine for the specific
c       element.
c
c*************************************************************************
c
c     input arguments
c     ===============
c      matId     (int,sc,i)              material #
c      elemId    (int,sc,i)              element #
c      kDomIntPt (int,sc,i)              "k"th domain integration point
c      kLayer    (int,sc,i)              "k"th layer
c      kSectPt   (int,sc,i)              "k"th Section point
c      ldstep    (int,sc,i)              load step number
```

```
c      isubst    (int,sc,i)              substep number
c      nDirect   (int,sc,in)             # of direct components
c      nShear    (int,sc,in)             # of shear components
c      ncomp     (int,sc,in)             nDirect + nShear
c      nstatev   (int,sc,l)              Number of state variables
c      nProp     (int,sc,l)              Number of material ocnstants
c
c      Temp      (dp,sc,in)              temperature at beginning of
c                                        time increment
c      dTemp     (dp,sc,in)              temperature increment
c      Time      (dp,sc,in)              time at beginning of increment (t)
c      dTime     (dp,sc,in)              current time increment (dt)
c
c      Strain   (dp,ar(ncomp),i)         Strain at beginning of time increment
c      dStrain  (dp,ar(ncomp),i)         Strain increment
c      prop     (dp,ar(nprop),i)         Material constants defined by TB,USER
c      coords   (dp,ar(3),i)             current coordinates
c      defGrad_t(dp,ar(3,3),i)           Deformation gradient at time t
c      defGrad  (dp,ar(3,3),i)           Deformation gradient at time t+dt
c
c     input output arguments
c     ======================
c      stress   (dp,ar(nTesn),io)        stress
c      ustatev   (dp,ar(nstatev),io)     user state variables
c      sedEl    (dp,sc,io)               elastic work
c      sedPl    (dp,sc,io)               plastic work
c      epseq    (dp,sc,io)               equivalent plastic strain
c      tsstif   (dp,ar(2),io)            transverse shear stiffness
c                                        tsstif(1) - Gxz
c                                        tsstif(2) - Gyz
c                                        tsstif(1) is also used to calculate hourglass
c                                        stiffness, this value must be defined when low
c                                        order element, such as 181, 182, 185 with uniform
c                                        integration is used.
c      var?     (dp,sc,io)               not used, they are reserved arguments
c                                        for further development
c
c     output arguments
c     ================
c      keycut   (int,sc,io)              loading bisect/cut control
c                                        0 - no bisect/cut
c                                        1 - bisect/cut
c                                        (factor will be determined by ANSYS solution control)
c      dsdePl   (dp,ar(ncomp,ncomp),io)  material jacobian matrix
c      epsZZ    (dp,sc,o)                strain epsZZ for plane stress,
c                                        define it when accounting for thickness change
c                                        in shell and plane stress states
c
c*************************************************************************
c
c      ncomp   6   for 3D  (nshear=3)
c      ncomp   4   for plane strain or axisymmetric (nShear = 1)
c      ncomp   3   for plane stress (nShear = 1)
c      ncomp   3   for 3d beam      (nShear = 2)
c      ncomp   1   for 1D (nShear = 0)
c
c      stresss and strains, plastic strain vectors
c          11, 22, 33, 12, 23, 13    for 3D
c          11, 22, 33, 12            for plane strain or axisymmetry
c          11, 22, 12                for plane stress
c          11, 13, 12                for 3d beam
c          11                        for 1D
c
c      material jacobian matrix
c        3D
c           dsdePl    |  1111   1122   1133   1112   1123   1113 |
c           dsdePl    |  2211   2222   2233   2212   2223   2213 |
c           dsdePl    |  3311   3322   3333   3312   3323   3313 |
c           dsdePl    |  1211   1222   1233   1212   1223   1213 |
c           dsdePl    |  2311   2322   2333   2312   2323   2313 |
c           dsdePl    |  1311   1322   1333   1312   1323   1313 |
c        plane strain or axisymmetric (11, 22, 33, 12)
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries and affiliates.

```
c           dsdePl    |  1111   1122   1133   1112 |
c           dsdePl    |  2211   2222   2233   2212 |
c           dsdePl    |  3311   3322   3333   3312 |
c           dsdePl    |  1211   1222   1233   1212 |
c        plane stress (11, 22, 12)
c           dsdePl    |  1111   1122   1112 |
c           dsdePl    |  2211   2222   2212 |
c           dsdePl    |  1211   1222   1212 |
c        3d beam (11, 13, 12)
c           dsdePl    |  1111   1113   1112 |
c           dsdePl    |  1311   1313   1312 |
c           dsdePl    |  1211   1213   1212 |
c        1d
c           dsdePl    |  1111 |
c
c***********************************************************************
#include "impcom.inc"
c
      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  ), ustatev  (nStatev),
     &                 dsdePl  (ncomp,ncomp),
     &                 Strain  (ncomp  ), dStrain (ncomp  ),
     &                 epsPl   (ncomp  ), prop    (nProp  ),
     &                 coords  (3),
     &                 defGrad_t(3,3),    defGrad(3,3),
     &                 tsstif  (2)
c
c***************** User defined part *********************************
c
c --- parameters
c
      INTEGER          NEWTON, mcomp
      DOUBLE PRECISION HALF, THIRD, ONE, TWO, SMALL, ONEHALF,
     &                 ZERO, TWOTHIRD, ONEDM02, ONEDM05, sqTiny
      PARAMETER       (ZERO       = 0.d0,
     &                 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,
     &                 ONEHALF    = 1.5d0,
     &                 TWOTHIRD   = 2.0d0/3.0d0,
     &                 NEWTON     = 10,
     &                 mcomp      = 6
     &                 )
c
c --- local variables
c
c      sigElp   (dp,ar(6  ),l)           trial stress
c      dsdeEl   (dp,ar(6,6),l)           elastic moduli
c      sigDev   (dp,ar(6  ),l)           deviatoric stress tensor
c      dfds     (dp,ar(6  ),l)           derivative of the yield function
c      JM       (dp,ar(6,6),l)           2D matrix for a 4 order tensor
c      pEl      (dp,sc     ,l)           hydrostatic pressure stress
c      qEl      (dp,sc     ,l)           von-mises stress
c      pleq_t   (dp,sc     ,l)           equivalent plastic strain at
c                                        beginnig of time increment
c      pleq     (dp,sc     ,l)           equivalent plastic strain at end
c                                        of time increment
c      dpleq    (dp,sc     ,l)           incremental equivalent plastic strain
c      cpleq    (dp,sc     ,l)           correction of incremental
```

```
c                                       equivalent plastic strain
c      sigy_t   (dp,sc     ,l)          yield stress at beginnig of time increments
c      sigy     (dp,sc     ,l)          yield stress at end of time
c                                       increment
c      young    (dp,sc     ,l)          Young's modulus
c      posn     (dp,sc     ,l)          Poiss's ratio
c      sigy0    (dp,sc     ,l)          initial yield stress
c      dsigdep  (dp,sc     ,l)          plastic slope
c      twoG     (dp,sc     ,l)          two time of shear moduli
c      threeG   (dp,sc     ,l)          three time of shear moduli
c      funcf    (dp,sc     ,l)          nonlinear function to be solved
c                                       for dpleq
c      dFdep    (dp,sc     ,l)          derivative of nonlinear function
c                                       over dpleq
c
c --- temporary variables for solution purpose
c      i, j
c      threeOv2qEl, oneOv3G, qElOv3G, con1, con2, fratio
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
       INTEGER          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,    con2
c*************************************************************************
c
       keycut   = 0
       dsigdep  = ZERO
       pleq_t   = ustatev(1)
       pleq     = pleq_t
c *** get Young's modulus and Poisson's ratio, initial yield stress and others
       young    = prop(1)
       posn     = prop(2)
       sigy0    = prop(3)
c *** calculate the plastic slope
       dsigdep  = young*prop(4)/(young-prop(4))
       twoG     = young / (ONE+posn)
       threeG   = ONEHALF * twoG
c
c *** calculate elastic stiffness matrix (3-D)
c
c
       elast1=young*posn/((1.0D0+posn)*(1.0D0-TWO*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)
       dsdeEl(3,3)=(elast1+TWO*elast2)*G(3)*G(3)
       dsdeEl(3,4)=elast1*G(3)*G(4)+elast2*TWO*G(5)*G(6)
       dsdeEl(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))
       dsdeEl(4,6)=elast1*G(4)*G(6)+elast2*(G(4)*G(6)+G(5)*G(2))
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries and affiliates.

```
      dsdeEl(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
c     copy elastic moduli dsdeEl to material Jacobian matrix
      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
      sigy    = sigy0 + dsigdep * pleq
c
      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
      dpleq    = (qEl - sigy) * oneOv3G
      pleq     = pleq_t + dpleq
c
c *** Newton-Raphson 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
c        --- avoid negative equivalent plastic strain
         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.
     &        (abs(dpleq ) .LE. sqTiny         )))  GO TO 100
      END DO
c
c *** Uncovergence, set keycut to 1 for bisect/cut
```

```
      keycut   = 1
      GO TO 990
 100  CONTINUE
c
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
      END DO
      DO i = nDirect + 1 , ncomp
         epsPl(i) = epsPl(i) + TWO * dfds(i) * dpleq
      END DO
      epseq  = pleq
c *** Update state variables
      ustatev(1) = pleq
c *** Update plastic work
      sedPl = sedPl + HALF * (sigy_t+sigy)*dpleq
c
c *** Material Jcobian matrix
c
      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
c
      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))
      END DO
      sedEl    = sedEl * HALF
c
 990  CONTINUE
c
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries and affiliates.

```
      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)
c
c        description
c           function to inquire element and solution information
c        definition
c           inquire - query argument (string)
c           value   - value of query argument
c        variables
c           inquire        - value
c           'LDSTEP'       - load step number
c           'ISUBST'       - substep step number
c           'IEQITR'       - current interation number
c           'NUMINTG'      - number of gauss integration
c           'ELEMID'       - element number
c           'MATID'        - material number of current element
c         'NSVAR'          - number of state variable for current material at
c                            gauss intg.
c           'NCOMP'        - number of vector components, such as stresses
c

c *** subroutine get_ElmData (inquire, elemId, kIntg, nvect, vect)
c
c        description
c           function to get/inquire solution dependent variables
c           such as stress, strains at gauss intg. point.
c        definition
c           inquire        - query argument (string)
c           elemId         - element number
c           kIntg          - gauss intg. number
c           nvect          - number of vector to be inquired
c           vect           - vector to be inquired
c        variables
c          'SIG '          - stress vector
c          'EPTO'          - Total   strain vector (EPEL+EPPL+EPCR+EPTH)
c          'EPPL'          - plastic strain vector
c          'EPCR'          - creep   strain vector
c          'EPTH'          - thermal strain vector
c          'ISIG'          - Initial stress vector
c          'PLEQ'          - accumulated equivalent plastic strain
c          'CREQ'          - accumulated equivalent creep   strain
c          'SVAR'          - State variables (define by tb,state)

c
c *** subroutine put_ElmData (inquire, elemId, kIntg, nvect, vect)
c        description
c           function to put solution dependent variables
c           such as stress, strains at gauss intg. point.
c        !! Use this in caution, it overides ansys database. Usually
c        !! you should only write user defined state variables,
c        !! SVAR
c        definition
c           inquire        - query argument (string)
c           elemId         - element number
c           kIntg          - gauss intg. number
c           nvect          - number of vector to be inquired
c           vect           - vector to be inquired
c
c        variables
c          'SIG '          - stress vector
c          'EPTO'          - Total   strain vector (EPEL+EPPL+EPCR+EPTH)
c          'EPPL'          - plastic strain vector
c          'EPCR'          - creep   strain vector
c          'EPTH'          - thermal strain vector
c          'ISIG'          - Initial stress vector
```

```
c          'PLEQ'          - accumulated equivalent plastic strain
c          'CREQ'          - accumulated equivalent creep   strain
c          'SVAR'          - State variables (define by tb,state)
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries and affiliates.

Appendix D. Fully Coupled Wind Turbine Example in Mechanical APDL

This wind coupling example solution has been implemented in Mechanical APDL to illustrate how to perform an integrated analysis of a wind turbine with its supporting structure. In this solution procedure, both the structural code (i.e., Mechanical APDL) and aeroelastic (e.g. Flex5) code are run simultaneously with continuous data exchange between the two programs at each time step. The data transfer is done through a set of interface routines that put or get data from a shared common data space. The interface is supplied in the form of a DLL which both the structural and aeroelastic programs will be accessing during an analysis. The example includes user programmable functions, and a macro has been developed to facilitate the communication with the DLL from Mechanical APDL. Aeroelastic code developers will need to utilize the same set of routines to establish communications between their code and the DLL.

There is also a sequential coupling method available; see Sequential Coupled Wind Turbine Solution in Mechanical APDL in the *Mechanical APDL Advanced Analysis Guide* for more information about using that method.

D.1. Implementing a Fully Coupled Wind Turbine Analysis

This example implementation of the fully coupled wind solution in Mechanical APDL follows a similar strategy to that used for ANSYS ASAS. In particular, data access is provided through the same set of interface routines. This enables easy adaptation of the new facility by existing ASAS wind turbine users.

The following summarize the modeling characteristics in ANSYS Mechanical APDL for a wind coupling analysis:

- The turbine effect is modeled via the user element USER300. This special user element has 9 nodes, with 6 freedoms (UX, UY, UZ, ROTX, ROTY, and ROTZ) on the first node and 3 freedoms (UX, UY, and UZ) on each of the subsequent nodes, making it capable of having a maximum 30 degrees of freedom on the element. The first node is the connection point between the turbine and the supporting structure and therefore it must be a node in the structural model. The other 8 nodes are created to accommodate the additional freedoms that are internal to the turbine element and are used solely by the aeroelastic code. Therefore, these nodes should not be connected to any other parts of the model.

- Key option 1 (KEYOPT(1)) of the user element is used to specify the damping matrix option. Damping can be obtained from the aeroelastic code alone, or computed from Rayleigh damping in Mechanical APDL based on the turbine mass and stiffness matrices, or both. KEYOPT(1) = 0 indicates that the damping matrix will be taken from the aeroelastic code plus Rayleigh damping, and this is the default. KEYOPT(1) = 1 indicates that only Rayleigh damping will be used. KEYOPT(1) = 2 indicates that only damping from the aeroelastic code will be used.

- The element does not have any material property or real constant data.

- The element mass will not generate any body forces even if accelerations (e.g. ACEL) are defined.

- The only element results available are the element nodal forces.

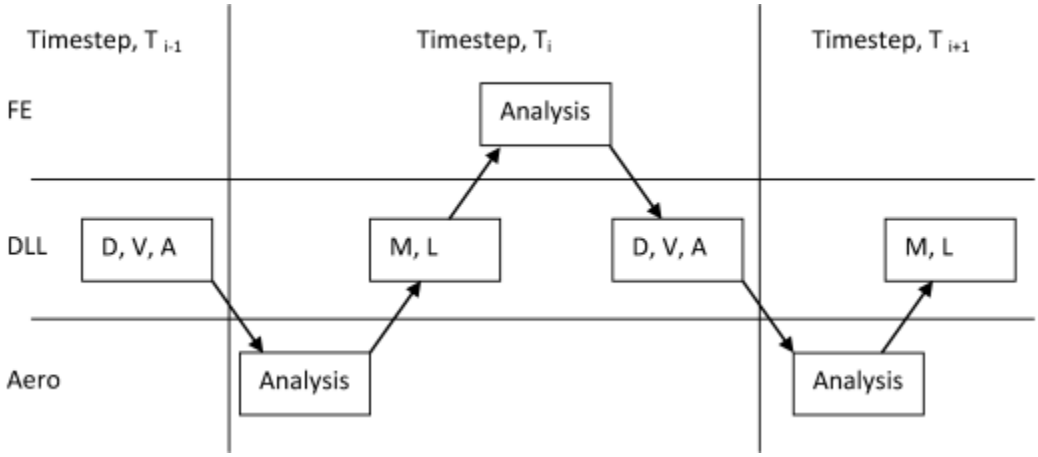
Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

- A Mechanical APDL command macro called **WTBCREATE** is provided to assist with the creation of a wind turbine model. This will automatically generate a turbine element and issue relevant data commands that are necessary to run a wind coupling analysis.
- Special versions of the user subroutines UserElem, USolBeg, USsFin and USolFin are provided to enable a wind coupling analysis. In addition, the shared common DLL `WTBFunctions.dll` is also required. A custom build of ANSYS Mechanical APDL is required during which the aeroelastic linking option should be selected.
- The analysis type should be transient (**ANTYPE**, TRANS) for a wind coupling analysis.

D.2. Theory

The whole offshore wind turbine structure is split into two parts. The upper part is the rotor-nacelle-assembly of the turbine including the tower and this is modeled by a wind turbine aeroelastic code.[1] The lower part is the support structure and this is modeled by structural FE as usual.

In this coupling approach, both the aeroelastic and FE code runs concurrently, with data exchange between the two programs occurring at every time step. The turbine effect as modeled by the aeroelastic code is taken into account in the FE code as a special element, which accepts the wind turbine system matrices (stiffness, damping and mass) and aerodynamic load vector as if it is a superelement.



In the diagram above, the following data items are exchanged:

D, V, A: Displacements, Velocities, Acceleration information at interface point.

M, L: Matricies (Stiffness, Damping and Mass), and/or Load Vectors.

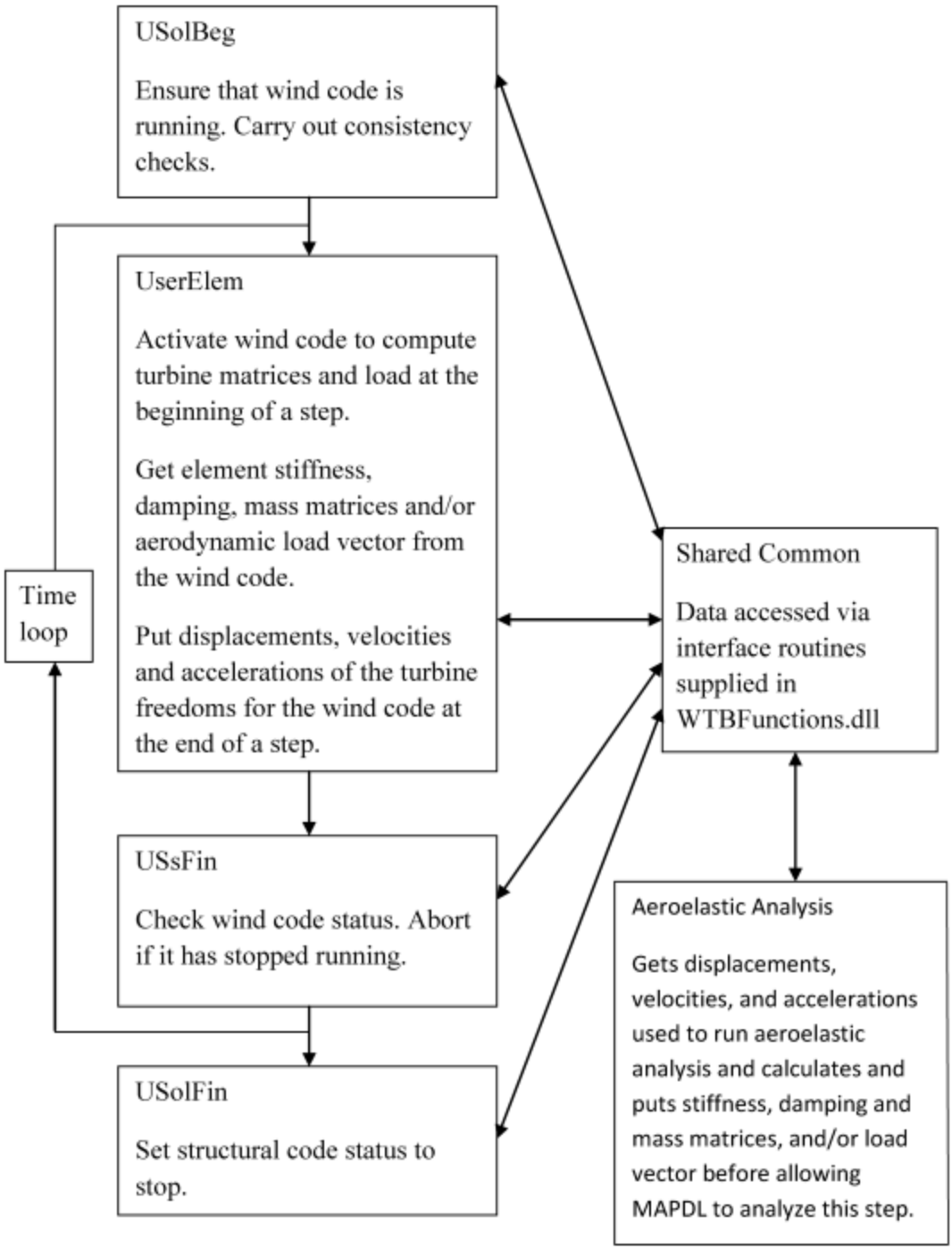
At the beginning of a time step, the aeroelastic code is called to compute the turbine matrices and loading based on the kinematics at the end of the previous step. It is assumed that the turbine data will remain constant during the time step. After transferring the information to the FE code, the full system of equations of motion is solved by the FE code to obtain an updated solution. Finally, the turbine kinematics are passed back to the aeroelastic code to continue the time advancement process.

Further details about the theory of the fully coupled solution can be obtained from the work of Kaufer [1] and Seidal [2].

[1] In the case of Flex5, the maximum number of equations generated by a model is 28.

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Flowchart of Wind Coupling Calculations



D.3. Compiling a Custom Version of Mechanical APDL

To activate the wind coupled analysis feature, it is necessary to create a custom build of Mechanical APDL that includes the wind interface libraries `Aeroelastic.lib` and `WTBFunctions.lib`. The library `Aeroelastic.lib` contains updated user routines to enable a wind coupling analysis in Mechanical APDL. The library `WTBFunctions.lib` contains interface routines for storing and retrieving data to a shared common area. Since the wind coupling solution involves changes to some of the user routines, it is important that the library `Aeroelastic.lib` is linked in before all other user libraries so that the wind library versions are picked up by the linker.

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

The executable of the aeroelastic code should be placed in the same folder as the custom Mechanical APDL executable as both programs need to use `WTBFunctions.dll` to access the shared data. Alternatively, the folder containing the WTBFunctions.dll could be added to the path (on Windows systems) to enable the aeroelastic code to remain elsewhere.

To link a custom version, run ANS_ADMIN from the utilities folder in the start menu. During the re-linking, the user is asked if the aeroelastic functions are to be included.

For further information and compiler requirements regarding the linking procedure and additional settings for using custom compiled executables, see Compiling and Linking UPFs on Windows Systems in the *Mechanical APDL Programmer's Reference*.

D.4. Performing a Wind Coupled Analysis

In order to perform a wind coupling analysis, the aeroelastic software must be modified to provide data communication with Mechanical APDL through the specified interface.

The aeroelastic analysis should be started before the FE analysis Both programs should use the same `WTBFunctions.dll` at run time.

D.4.1. The Wind Coupling Process

Below is a brief description of the wind coupling algorithm implemented in Mechanical APDL:

1. Both Mechanical APDL and the aeroelastic code are run concurrently.
2. Both programs carry out data initialization separately.
3. At the beginning of a time step, set the aeroelastic code to active. The aeroelastic code computes and returns the turbine stiffness, damping and mass matrices to Mechanical APDL together with the aerodynamic force vector for the current time step. Put the aeroelastic code to sleep.
4. Mechanical APDL carries out the time integration to find the new solution at the end of the current time step.
5. The displacements, velocities and accelerations of the turbine freedoms (i.e. freedoms on the USER300 element) at the end of the step are stored to the shared common for the aeroelastic code.
6. Advance to next time step and return to step 3.

D.4.2. Data Exchange Routines

The following routines are used to facilitate data exchange between Mechanical APDL and the aeroelastic code as described above. They can be accessed from the aeroelastic code by linking in the dynamic link library `WTBFunctions.dll`. Coding examples utilizing these routines in Fortran and C++ are available in the folder `Program Files\Ansys Inc\v182\ansys\custom\user\{platform}\Aeroelastic`, where {platform} is a directory that uniquely identifies the hardware platform version: "Winx64" for 64-bit Windows.

```
SUBROUTINE GetWTBParamI(itype,id,ival,ierr)
!****
!****    Routine gets an integer wind turbine parameter to common data area
!****
!****    Arguments
!****    itype  (in ) Data type
!****                 1 - Mechanical APDL run status
```


Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

```
!****                     = -1 not started
!****                     =  0 inactive (waiting)
!****                     =  1 active (running)
!****                     =  2 stopped (finished/aborted)
!****                 2 - Wind code run status
!****                     = -1 not started
!****                     =  0 inactive (waiting)
!****                     =  1 active (running)
!****                     =  2 stopped (finished/aborted)
!****                 3 - Number of active freedoms
!****                 4 - Time step number
!****    id     (in ) element identifier (currently unused, assume 1)
!****    ival   (out) parameter value
!****    ierr   (out) exit code (0 if no error)
integer itype,id,ival,ierr

SUBROUTINE GetWTBParamR(itype,id,rval,ierr)
!****
!****    Routine gets a real wind turbine parameter to common data area
!****
!****    Arguments
!****    itype  (in ) Data type
!****                 1 - Analysis time
!****                 2 - Time step
!****    id     (in ) element identifier (currently unused, assume 1)
!****    rval   (out) parameter value
!****    ierr   (out) exit code (0 if no error)
integer itype,id,ierr
double precision rval

SUBROUTINE GetWTBData(itype,id,array,narray,ierr)
!****
!****    Routine gets wind turbine data from common data area
!****
!****    Arguments
!****    itype  (in ) Data type
!****                 1 - Stif
!****                 2 - Damp
!****                 3 - Mass
!****                 4 - Load
!****                 5 - Disp
!****                 6 - Velo
!****                 7 - Accn
!****    id     (in ) element identifier (currently unused, assume 1)
!****    array  (out) data array
!****    narray (i/o) size of array on input, actual array size on exit
!****    ierr   (out) exit code (0 if no error)
integer itype,id,narray,ierr
double precision array(*)

SUBROUTINE PutWTBParamI(itype,id,ival,ierr)
!****
!****    Routine puts an integer wind turbine parameter to common data area
!****
!****    Arguments
!****    itype  (in ) Data type
!****                 1 - Mechanical APDL run status
!****                     = -1 not started
!****                     =  0 inactive (waiting)
!****                     =  1 active (running)
!****                     =  2 stopped (finished/aborted)
!****                 2 - Wind code run status
!****                     = -1 not started
!****                     =  0 inactive (waiting)
!****                     =  1 active (running)
!****                     =  2 stopped (finished/aborted)
!****                 3 - Number of active freedoms
!****                 4 - Time step number
!****    id     (in ) element identifier (currently unused, assume 1)
!****    ival   (in ) parameter value
!****    ierr   (out) exit code (0 if no error)
integer itype,id,ival,ierr
```

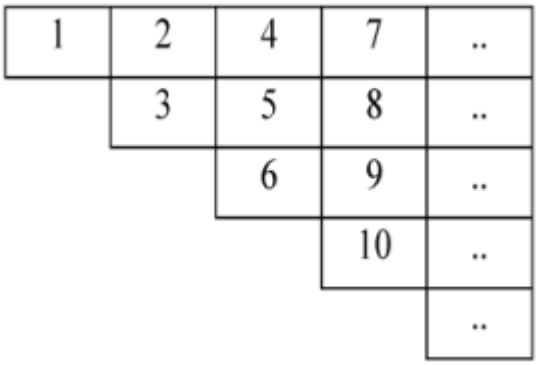
```
SUBROUTINE PutWTBParamR(itype,id,rval,ierr)
!****
!****    Routine puts a real wind turbine parameter to common data area
!****
!****    Arguments
!****    itype  (in ) Data type
!****                 1 - Analysis time
!****                 2 - Time step
!****    id     (in ) element identifier (currently unused, assume 1)
!****    rval   (in ) parameter value
!****    ierr   (out) exit code (0 if no error)
integer itype,id,ierr
double precision rval

SUBROUTINE PutWTBData(itype,id,array,narray,ierr)
!****
!****    Routine puts wind turbine data to common data area
!****
!****    Arguments
!****    itype  (in ) Data type
!****                 1 - Stif
!****                 2 - Damp
!****                 3 - Mass
!****                 4 - Load
!****                 5 - Disp
!****                 6 - Velo
!****                 7 - Accn
!****    id     (in ) element identifier (currently unused, assume 1)
!****    array  (in ) data array
!****    narray (i/o) size of array on input, actual put size on exit
!****    ierr   (out) exit code (0 if no error)
integer itype,id,narray,ierr
double precision array(*)
```

D.4.3. Important Analysis Notes

You must keep in mind the following when performing an aeroelastic analysis:

- After data initialization (e.g. data read in and checking, etc), the aeroelastic code should be put to sleep until the Mechanical APDL run status becomes inactive.
- The number of active freedoms is the number of freedoms in the aeroelastic model. This must be set up and put to the shared common by the aeroelastic code during the data initialization phase of the analysis.
- For the current usage, the aeroelastic code should always only put stiffness, damping, mass, and load data to the shared common, and get displacements, velocities, and accelerations from the shared common.
- The wind turbine array entries must correspond to the order of the element freedoms set up for the wind coupled USER300 element. Thus, freedoms 1 to 6 are UX, UY, UZ, ROTX, ROTY, and ROTZ freedoms of the interface node between the turbine and the support structure. The rest are generalized freedoms internal to the element. All the data must be stated in the structural coordinate axis system.
- The element matrices (i.e. stiffness, damping, and mass) are assumed to be given in packed symmetric form. The order of the packed symmetric matrix form in which the data are specified is defined as follows:

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.



- The units of the wind turbine data values are assumed to be consistent with the analysis units. No units conversion will be carried out by Mechanical APDL.
- It is assumed that identical time step sizes are used in both Mechanical APDL and the aeroelastic code. The solution times are controlled by data in Mechanical APDL since it is the one that solves the complete set of equations of the coupled system.
- The table below shows the explanation of the various exit code values (i.e. ierr):

Code	Meaning
100	Invalid data type integer
101	Specified array size too small to get
102	Specified array size too big to put
103	Number of active freedoms is unset
104	Invalid array size specified
201	Invalid run status specified
202	Invalid number of active freedoms specified
203	Invalid time step number specified
301	Invalid time step value specified

D.5. Example Analysis Using Provided "WindProg" Example for Aeroelastic Coupling

This example uses `WindProg.exe`, which represents a dummy aeroelastic analysis. Source code to allow you to compile this program is provided for both C++ and FORTRAN in `{Installation Folder}\ansys\custom\user\{Platform}\Aeroelastic\WindDemos\{C++|Fortran}`. It assumes that the steps in Compiling a Custom Version of Mechanical APDL (p. 367) have been followed to create a custom executable for Mechanical APDL.

The example Mechanical APDL file is a simple line of pipe elements which is fully supported at node 6 and has the interface node positioned at node 1. The **WTBCREATE** macro is used to set up the user element and define the interface node.

To run the analysis, `WindProg.exe` must be able to find the `WTBFunctions.dll` either by having both files in the same folder, or by adding the folder containing `WTBFunctions.dll` to your Path

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

system environment variable. First run WindProg and enter 3 as the number of freedoms, then run the following example within Mechanical APDL.

```
/FILNAME,wind02
/prep7
/TITLE,wind02, Wind coupling test
/com ************************************************************************
/com Wind coupling testing 02
/com ************************************************************************
antype,trans
nlgeom,off
et,1,pipe288
! define pipe section
secnum,1
sectype,1,pipe
secdata,0.1,0.02
MP,EX,  1,2.1e11
MP,PRXY,1,0.3
MP,ALPX,1,0.0
MP,DENS,1,7850.0
! define the tube
n,         1,        0.0,   0.0
n,         2,        1.0,   0.0
n,         3,        2.0,   0.0
n,         4,        3.0,   0.0
n,         5,        4.0,   0.0
n,         6,        5.0,   0.0
type,1
mat,1
secnum,1
en, 1,   1,  2
en, 2,   2,  3
en, 3,   3,  4
en, 4,   4,  5
en, 5,   5,  6
! define damping factors
alphad,0.3
betad,0.001
! define turbine element
wtbcreate,,1,1  ! use Rayleigh damping
d,  6,all
finish
/SOLU
! CASE 1
F,1,FY,1.0e5
TINTP,0.0
TIME,1.0e-6
nsubst,1
solve
TIME,0.1
nsubst,10,10,10
OUTPR,all,1
OUTRES,all,1
solve
finish
/POST1
FORCE,STATIC
PRESOL,F
PRESOL,M
finish
```

D.6. References

The following references are cited in this appendix:

1. D. Kaufer et. al., Integrated Analysis of the Dynamics of Offshore Wind Trubines with Arbitrary Support Structures, Proc. of EWEC 2009, Marseille: EWEC, 2009.

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

2. M. Seidal et. Al., Validation of Offshore Load Simulations Using Measurement Data from the DOWNVInD Project, Proc. European Offshore Wind 2009, Stockholm, 2009.

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information
of ANSYS, Inc. and its subsidiaries and affiliates.

Index

Symbols

*PREAD, 103
/UPF command, 125, 127
3-D line elements, 332
3-D nodal coordinates, 177
    with UPFs, 119

A

Abbreviations
    file access routines, 65
acelcm.inc file, 123
activating UPFs, 133
aeroelastic analysis, 366
ANSYS
    standard file header, 68
ansyscust file, 127
ansysdef.inc file, 123
arc-length debugging, 136
ARRAY or TABLE parameter
    defining values of, 103
auto time stepping debugging, 136
/AUX2 command, 65
AUX2 process, 65

B

basic element pass debugging, 137
Beam cross sections
    reading, 104
Binary files
    accessing, 63
    characteristics of, 64
    closing or deleting a blocked binary file, 71
    Converting Two Integers into a Pointer, 71
    copying contents, 73
    opening a blocked binary file, 67
    viewing contents, 65
binclo subroutine, 71
binhed subroutine, 68
binini subroutine, 65
biniqr function, 66
binlib library, 63
binrd8 subroutine, 69
binset function, 67
bintfo subroutine, 68
bintrd subroutine, 72
bintst program, 72
bintwr subroutine, 73
binwrt8 subroutine, 70
Buffered binary I/O systems
    initializing, 65
Buffered file
    reading data from, 69
    writing data to, 70

C

C, 119, 122
    Fortran shell for, 122
.c extension, 125
calculating
    rotation caused by internal pressure, 232
Calculating
    dot product of two vectors, 333
    dot product of two vectors with increments, 333
    sum of vector components, 333
CDWRITE
    using the command, 83
CE, 85
ceget, 275
ceinqr, 274
ceput, 275
cesel, 275
Character string
    converting to integer string, 71
Characters retrieved from a double precision variable, 327
CMBLOCK, 86
cmopt.inc file, 123
cnvget subroutine, 179
COMBIN37 parameters, 182
commands
    creating, 239
common blocks, 123
component entities
    defining, 86
constitutive material models
    creating, 187, 197
Constraint equation
    defining constant term in, 85
Constraint equations
    deleting or selecting, 275
    getting , 275
    getting information about, 274
    storing, 275
constraints
    deleting at a node, 276
    getting from the database, 276-277
    getting information, 275
    storing at a node, 276, 300
Contact
    defining, 94

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

contact
  user-defined contact interfacial behaviors, 213
contact interfacial behavior, 213
convection loads
  defining, 179
converting nodal data to fixed format data blocks, 138
Coordinate system
  deleting a coordinate system, 272
  getting a coordinate system, 271
  getting information about a coordinate system, 271
  storing a coordinate system, 272
Coordinate Systems
  defining, 96
Coupled nodes
  defining, 85
Coupled set
  getting a coupled set, 273
  getting information about, 272
  selecting or deleting, 274
  storing a coupled set, 273
Coupling and constraint routines, 272
CP, 85
cpget, 273
cpinqr, 272
cpput, 273
cpsel, 274
creep material behavior, 203
creep1 subroutine, 172
creep3 subroutine, 172
csydel, 272
csyget, 271
csyiqr, 271
csyput, 272
CURVE
  defining, 95

D

debugging
  elements, 137
  solutions, 136
decks, 123
Degree of freedom labels
  customizing, 83
Demonstration routines, 72
/DFLAB, 83
disdel, 276
disget, 275-276
displacement and coordinate debugging, 137
displacement values, 247
displacement vector debugging, 136
disput, 276
DOF pointers, 123
Double precision vectors, 334
dptoch subroutine, 327
dspdel, 300
dspget, 300
dspiqr, 299
dspput, 300

E

ecddel, 323
ecdget, 322
ecdiqr, 322
ecdput, 322
echprm.inc file, 123
echrtr, 266
ecrdel, 314
ecrget, 312
ecriqr, 312
ecrput, 313
ecvdel, 288
ecvget, 287
ecviqr, 286
ecvput, 287
EDCADAPT, 93
edcdel, 294
EDCGEN, 94
edcget, 293
edciqr, 293
edcput, 293
EDCURVE, 95
EDDRELAX, 95
EDLCS, 96
EDLOAD, 96
EDPREAD, 97
EDWELD, 98
eeldel, 310
eelget, 308
eeliqr, 308
eelput, 309
efsdel, 298
efsget, 298
efsiqr, 298
efsput, 299
efudel, 292
efuget, 292
efuiqr, 291
efuput, 292
efxdel, 319
efxget, 318
efxiqr, 318
efxput, 318
egen function, 211
egrdel, 308

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

egrget, 307
egriqr, 307
egrput, 307
ehcdel, 325
ehcget, 325
ehciqr, 324
ehcput, 325
ehgdel, 291
ehgget, 290
ehgiqr, 290
ehgput, 291
eimdel, 296
eimget, 296
eimiqr, 295
eimput, 296
elccmt subroutine, 159
elccmt.inc file, 123
eldwrnL subroutine, 176
eldwrtL subroutine, 176
elecom.inc file, 123
element array data pointers, 123
element attribute routines, 263
element calculated heat generation
  deleting, 325
  getting, 325
  getting information about, 324
  storing, 325
element convection
  deleting, 288
  getting, 287
  getting information about, 286
  storing, 287
element current densities
  deleting, 323
  getting, 322
  getting information about, 322
  storing, 322
element current density
  deleting, 294
  getting, 293
  getting information about, 293
  storing, 293
element data
  converting to fixed format blocks, 138
  writing to a file, 176
element energy
  deleting, 306
  getting, 306
  getting information about, 305
  storing, 306
element euler angle
  deleting, 317
  getting, 316
  storing, 317
element euler angles
  getting information about, 316
element face flagged surface
  deleting, 298
  getting, 298
  getting information about, 298
  storing, 299
element face impedance
  getting, 296
element face pressure
  getting, 285
  storing, 286
element fluence
  deleting, 292
  getting, 292
  getting information about, 291
  storing, 292
element fluxes
  deleting, 319
  getting information about, 318
  storing, 318
element Fluxes
  getting, 318
element heat generation
  deleting, 291
  getting, 290
  getting information about, 290
  storing, 291
element impedance
  deleting, 296
  getting information about, 295
  storing, 296
element initial strain
  deleting, 316
element loading routines, 285
element local force
  deleting, 320
  getting, 319
  storing, 320
element local forces
  getting information about, 319
element matrices, 161
  accessing, 182
  reusing, 166
element miscellaneous non-summable data
  deleting, 321
  getting, 321
  getting information about, 320
  storing, 321
element miscellaneous summable data

deleting, 302
getting, 301
getting information about, 301
storing, 301
element nodal creep strain
deleting, 314
getting, 312
getting information about, 312
storing, 313
element nodal elastic strain
deleting, 310
getting, 308
getting information about, 308
storing, 309
element nodal force
getting, 302
getting information about, 302
storing, 303
element nodal forces
deleting, 303
element nodal gradients
deleting, 308
getting, 307
getting information about, 307
storing, 307
element nodal plastic strain
deleting, 312
getting, 310
getting information about, 310
storing, 311
element nodal stress
deleting, 305
getting, 304
getting information about, 303
storing, 304
element nodal thermal strain
getting, 314
getting information about, 314
storing, 315
element nonlinear tables
deleting, 324
getting, 323
getting information about, 323
storing, 324
element pressure/convection
deleting, 286
getting informatin about, 285
element surface stress data
deleting, 297
getting, 297
getting information about, 296
storing, 297
element swelling strain
deleting, 316
element temperature
deleting, 289
getting, 289
getting information about, 288
storing, 289
element thermal strain
deleting, 316
element virtual displacement
getting, 294
getting information about, 294
storing, 295
Elements
defining, 99
elements, 119
accessing information, 163
changing face
charge density surface information, 230
convection surface information, 229
heat flux surface information, 230
changing pressure information, 228
debugging, 137
deleting an element, 260
deleting an element type, 267
evaluating material properties, 168
example of element attribute routines, 272
finding coordinates, 248
getting information about an element, 263
getting information about an element type, 265-266
getting information about element characteristics, 266
getting the number of the next defined element, 260
getting the number of the next element, 259
getting the number of the previous selected element, 259
inverting an element, 260
inverting an element type, 267
load vectors, 167
matrix debugging, 137
modifying, 180
monitoring, 180
non-summable miscellaneous data
writing to results file, 176
overriding characteristic defaults, 160
parameters, 123
particle velocities and accelerations
due to waves and current, 234
returning reference names, 164
rotation vectors: calculating, 176
rotational pseudovector
updating, 177

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

routines for selecting and retrieving, 258
selecting an element, 260
selecting an element type, 267
storing
user-supplied output, 180
storing an element, 265
storing element type data, 266
subroutines, 122
for element characteristics, 159
for element creation, 163
transformation matrix: calculating, 177
types, 159
unselecting an element, 260
unselecting an element type, 267
updating
3-D nodal coordinates, 177
creep history, 172
plastic history, 170
swelling history, 173
updating rotational pseudovector, 177
user-defined: creating, 141
writing saved variable set, 165
ElemGetMat subroutine, 155
elfdel, 320
elfget, 319
elfiqr, 319
elfput, 320
ElInterp subroutine, 248
elLenPsvrBuf function, 174
elmiqr, 263
elmput, 265
elnext, 259
elnxdf, 260
elparm.inc file, 123
elprev, 259
ElResultGet subroutine, 248
ElResultStrt subroutine, 248
elsel, 260
elucom.inc file, 123
emndel, 321
emnget, 321
emniqr, 320
emnput, 321
emsdel, 302
emsget, 301
emsiqr, 301
emsput, 301
EN, 99
enfdel, 303
enfget, 302
enfiqr, 302
enfput, 303
engdel, 306
engget, 306
engiqr, 305
engput, 306
enldel, 324
enlget, 323
enliqr, 323
enlput, 324
ensdel, 305
ensget, 304
ensiqr, 303
ensput, 304
epldel, 312
eplget, 310
epliqr, 310
eplput, 311
eprdel, 286
eprget, 285
epriqr, 285
eprput, 286
erhandler subroutine, 330
erinqr subroutine, 328
errors
displaying, 330
obtaining information, 328
esfdel, 297
esfget, 297
esfiqr, 296
esfput, 297
ethdel, 316
ethget, 314
ethiqr, 314
ethput, 315
etpdel, 289
etpget, 289
etpiqr, 288
etpput, 289
etycom.inc file, 123
etyget, 266
etyiqr, 265
etyput, 266
etysel, 267
euldel, 317
eulget, 316
euliqr, 316
eulput, 317
evaluating material properties for 1-D elements, 170
evdget, 294
evdiqr, 294
evdput, 295
exinc4 subroutine, 70

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

F

.F extension, 125
faces
  convection surface information, 229
fAnsMemAlloc function, 250
fAnsMemFree subroutine, 250
File access routines
  abbreviations, 65
file headers, 138
File status
  retrieving, 66
Files
  ansyscust, 127
  binary
    accessing, 63
    characteristics of, 64
files, 125, 127
  START.ANS, 239
/FILNAM, 83
fordel, 278
forget, 277
foriqr, 277
forput, 278
Fortran, 119-120, 122
Full transposed matrix, 338
functions
  miscellaneous, 254
  vector, 333

G

Geometry
  relaxing, 95
getnod, 261
GetRForce function, 246
GetStackDisp function, 247
Graphics files
  neutral graphics file format, 108
  pixmap fromat for graphic display files , 107
  z-buffered graphics, 107

H

heat generation
  defining loads, 179
hgnget subroutine, 179
hyperelastic material models
  creating, 202

I

I/O systems, 65
imove function, 334
impcom.inc file, 123
.inc extension, 123
files
  .inc, 123
include files, 123
inexc4 subroutine, 71
Initializing paging space, 67
Integer string
  converting to character string, 70
interpolation, 331
intrp subroutine, 331
issuing a command from a user subroutine, 255
izero function, 334

J

Jobname.cdb
  defining, 97

L

large data files, 138
largeIntGet subroutine, 71
lastv function, 334
Libraries, 63
linking UPFs, 119
linking UPFs to Mechanical APDL, 124
Linux systems
  compiling and linking UPFs on, 125
  Fortran programming on, 122
load data
  getting from analysis results, 248
  reading for a substructure generation run, 167
load steps
  accessing program logic, 249
  containing information about, 123
load vectors, 161
  accessing, 182
Loads
  defining, 96
loads, 119
  complex load vector calculation, 231
  convection, 179
  customizing, 227
  debugging, 137
  heat generation, 179
  pressure, 178
  temperature, 178
LOCAL, 100
Local coordinate system
  defining, 100

M

M, 100
maat subroutine, 339
Master degree of freedom

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

defining, 100
matba subroutine, 340
material linear elastic properties: defining, 205
material properties, 119
checking user-defined data, 211
customizing, 186
debugging, 137
modifying orientation of, 181
nonlinear information, 174
update user-defined field variables, 211
Material properties
deleting a material property table, 269
getting a material property table, 268
getting information about a material property, 267
storing a material property table, 268
Material property data table
defining, 100
materials
defining material linear elastic properties, 205
user-defined: creating, 187
matrix
changing a matrix value, 339
debugging, 136
multiplying a vector by, 337
multiplying a vector by full transposed, 338
subroutines, 337
transposing, 339
transposing symmetric, 341
value updating via addition, 340
value updating via transposition, 340
Matrix
filling upper triangle from lower triangle, 340
value updating via multiplications, 340
matsym subroutine, 340
matxb subroutine, 339
matxv subroutine, 338
matxv1 subroutine, 338
maxv subroutine, 337
maxv1 subroutine, 338
mctac subroutine, 341
Mechanical APDL
running as a subroutine, 239
memory management subroutines, 250
Meshing
defining, 93
meshing debugging, 138
miscellaneous functions, 254
monitoring elements, 180
Moving one vector into another, 334
MPDATA, 100
mpdel, 269
mpget, 268
mpinqr, 267
mpput, 268
MPTEMP, 101
mreuse subroutine, 166
multifield debugging, 136

N

N, 101
ndcdel, 283
ndcget, 282
ndciqr, 282
ndcput, 283
ndgall, 262
ndgxyz function, 337
ndinqr, 260
ndnext, 258
ndnxdf, 258
ndprev, 258
ndpxyz subroutine, 337
ndsel, 259
ndspgt, 262
Neutral graphics file
character set used in, 108
color specification for, 111
decoding , 112
examples of , 112
format, 108
general format of, 108
plot directives , 108
Anno Size, 110
Area-Color, 110
Draw, 110
End-Plot, 111
Font Control, 110
Graph-Color , 110
Line-Type, 110
Line-Width, 110
Marker Size, 110
Move, 110
No-Opt , 111
Normal , 110
parameter types for directives , 109
Pixmap, 111
Pixmap Style, 110
Point, 110
Polygon , 110
Start_Plot, 109
Text , 110
Text Justification, 110
Text-Color, 110
Text-Size, 110
Window, 110

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Newton-Raphson debugging, 136
nfudel, 282
nfuget F, 281
nfuiqr, 281
nfuput, 281
nhgdel, 280
nhgget, 280
nhgiqr, 279
nhgput, 280
nlget function, 174
nminfo subroutine, 164
nmisc record
  storing data in, 175
nodal coordinate system transformation debugging, 138
nodal current density
  deleting, 283
  getting, 282
  getting information about, 282
  storing, 283
nodal data conversion to fixed format data blocks, 138
nodal fluence
  deleting, 282
nodal fluences
  getting, 281
  getting information about, 281
  storing, 281
nodal heat generation
  deleting, 280
  getting, 280
  getting information, 279
  storing, 280
nodal load
  deleting at a node, 278
  storing at a node, 278
nodal loading routines, 275
nodal loads
  getting information about, 277
nodal magnetic virtual displacement
  getting, 284
  getting information about, 283
nodal orientation matrix
  adjusting, 163
nodal reaction force values
  getting, 246
nodal results
  getting from the database, 300
  getting information about, 299
nodal temperature
  deleting, 279
  getting, 279
  getting information about, 278
  storing, 279
nodal virtual displacement
  deleting, 284
  storing, 284
node information routines, 260
Nodes
  defining, 101
  getting XYZ vector for, 337
  storing XYZ vector for, 337
nodes
  deleting a node, 259
  getting a nodal point, 261
  getting information about a node, 260
  getting the next node number, 258
  getting the nodal solution for a node of an element, 262
  getting the number of the next defined node, 258
  getting the number of the previous selected node, 258
  getting the xyz/rotation coordinates vector for a node, 262
  inverting a node, 259
  routines for selecting and retrieving, 258
  selecting a node, 259
  storing a node, 262
  unselecting a node, 259
nonlinear debugging, 137
ntpdel, 279
ntpget, 279
ntpiqr, 278
ntpput, 279
nvddel, 284
nvdget, 284
nvdiqr, 283
nvdput, 284

O

.o extension, 125
.obj extension, 127
object files, 125, 127
optimization variables, 123
outpcm.inc file, 123
output
  generating, 138
  obtaining information about, 328
output control, 123

P

parallel processing and UPFs, 121
Parameters
  retrieving system-dependent parameters, 66
parameters

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

adding, 253
creating a dimensioned parameter, 252
defining in subroutines, 159
element characteristics, 123
finding and evaluating, 253
performing user operations on COMBIN37, 182
subroutines to process, 252
pardef subroutine, 253
pardim subroutine, 252
plast1 subroutine, 170
plast3 subroutine, 171
pointers to element data array, 123
POST1 postprocessor, 159
pressure loads
defining, 178
printer output, 162
Printing file contents, 72
program logic
accessing, 249
tracking, 135
programmer debugging, 136
programming UPFs, 122
prope1 subroutine, 169
propev subroutine, 168
prsget subroutine, 178
pstev1 subroutine, 170
putnod, 262

R

R, 103-104
rdfull, 74
rdsubs subroutine, 74
Reading a results file, 75
Reading an ANSYS substructure file, 74
Reading and reformatting the .FULL file, 74
reading large data files rapidly, 138
Real constant set
defining, 103-104
real constants
debugging, 137
fetching, 168
Real constants
getting information about a real constant set, 269
getting real constant data, 270
selecting or deleting a real constant set, 270
ResRdBegin function, 76
ResRdCsys function, 78
ResRdDemo, 75
ResRdDisp function, 80
ResRdElem function, 78
ResRdEstr function, 81
ResRdFix function, 81
ResRdForc function, 81
ResRdGeomBegin function, 77
ResRdMat function, 79
ResRdNode function, 78
ResRdReal function, 77
ResRdRfor function, 80
ResRdSect function, 79
ResRdSectMatBegin function, 79
ResRdSolBeginl function, 80
ResRdType function, 77
restrictions on UPF creation, 119
results
deleting at a node, 300
Results file
opening the file and retrieving global information, 76
retrieving applied nodal constraints, 81
retrieving applied nodal loads solution, 81
retrieving coordinate systems, 78
retrieving element solutions, 81
retrieving element types, 77
retrieving elements, 78
retrieving geometry information, 77
retrieving material data, 79
retrieving nodal coordinates, 78
retrieving nodal solution, 80
retrieving reaction solution, 80
retrieving real constants, 77
retrieving result set location, 80
retrieving section and material numbers, 79
retrieving section data, 79
Results files
retrieving information from, 75
results information routines, 299
results values
getting at selected points, 248
ResWrDemo, 75
Retrieving
maximum vector value for a location, 334
position of last nonzero term in a vector, 334
reusing element matrices, 166
RLBLOCK, 103
rlget, 270
rlinqr, 269
rlsel, 270
rottr subroutine, 177
RunCommand function, 255
rvrget subroutine, 168

S

saved variables
debugging, 137

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

fetching, 164
Scalar constants
assigning to vectors, 335
scalar fields to user-defined values, 227
SECBLOCK, 104
SFBEAM, 105
SFE, 106
Shell sections
reading, 104
simultaneous linear equations, 341
solution control debugging, 136
solutions
accessing program logic, 249
debugging, 136
soptcm.inc file, 123
source files, 125
stack storage definition, 123
stack.inc file, 123
Standard ANSYS file header, 68
start-up
accessing program logic at, 249
START.ANS file, 239
stepcm.inc file, 123
stopping the program, 332
storing
user-supplied element output, 180
subrd subroutine, 167
subroutine userfld, 211
subroutine userthstrain, 213
Subroutines
accessing results files, 75
subroutines
for elements, 122
for memory management, 250
for processing parameters, 252
matrix, 337
running Mechanical APDL as, 239
supporting for user-created elements, 163
supporting user-defined commands, 246
tracking calls to, 330
subwrt subroutine, 167
Surface load
defining, 106
defining as beams, 105
svgidx subroutine, 164
svpidx subroutine, 165
svrget subroutine, 164
svrput subroutine, 165
swell1 subroutine, 173
swell3 subroutine, 173
swelling laws, 210
symeqn subroutine, 341
sysiqr function, 66
system dependencies, 122
systop subroutine, 332

T

table data
retrieving, 170
tbuser subroutine, 170
temperature loads
defining, 178
Temperature table
defining, 101
thermal materials
user-defined: creating, 197
tmpget subroutine, 178
TrackBegin subroutine, 135, 330
TrackEnd subroutine, 135, 330
tran subroutine, 341
transient debugging, 136
transposing a matrix, 339
tranx3 subroutine, 332
trrot subroutine, 176

U

UAnBeg subroutine, 249
UAnFin subroutine, 249
uec100 subroutine, 159-160
uel100 subroutine, 159, 161
uel101 subroutine, 122
UElMatx subroutine, 182
uep100 subroutine, 159, 162
uex100 subroutine, 159-160
UItBeg subroutine, 249
UItFin subroutine, 249
ULdBeg subroutine, 249
ULdFin subroutine, 249
UPFs (see user programmable functions (UPFs))
user programmable features (UPFs), 119
activating, 133
and loads, 119
and material properties, 119
before you begin using, 120
capabilities of, 119
compiling and linking
on Linux systems, 125
on Windows systems, 127
considerations for using, 121
debugging, 135
definition of, 119
development strategy, 122
include files, 123
linking subroutines to user routines, 124

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

planning, 121
running your executable, 134
understanding, 122
verifying your routines, 135
user subroutines
for creating new elements, 141, 159
for creating new materials, 187
for creating new thermal materials, 197
for customizing contact interfacial behaviors, 213
for defining material linear elastic properties, 205
User subroutines, 159
user-defined commands, 239
making available to all sessions, 239
user-defined contact interfacial behaviors, 213
user-defined elements, 141
user-defined materials, 187
user-defined thermal materials, 197
User-defined variable storage, 123
user-programmable features
defining material linear elastic properties, 205
user-defined contact interfacial behaviors, 213
user-defined elements, 141
user-defined materials, 187
user-defined thermal materials, 197
user01 command, 239-240
user02 command, 241
user03 command, 242
USER300 user-defined element, 143
user_tbelastic subroutine, 205
userac, 272
userac subroutine, 159, 163
useran subroutine, 181
userch subroutine, 230
userck subroutine, 211
usercnprop subroutine, 213
UserCreep subroutine, 203
usercv subroutine, 229
UserElem subroutine, 144
usereo subroutine, 175
userfc subroutine, 208
userfd subroutine, 231
userfric subroutine, 217
userfx subroutine, 230
UserHyper subroutine, 202
userinter subroutine, 220
UserMat subroutine, 187
UserMatTh subroutine, 197
userOceanRead subroutine, 238
userou subroutine, 180
userPanelHydFor subroutine, 238
userPartVelAcc subroutine, 234
userpe function, 232
userpr subroutine, 228
userrc subroutine, 182
userswstrain subroutine, 210
usertr subroutine, 159, 163
userwear subroutine, 226
USolBeg subroutine, 249
USolFin subroutine, 249
USRCAL command, 227, 249
usrefl subroutine, 227
UsrFictive subroutine, 184
usrsurf116 subroutine, 232
USsBeg subroutine, 249
USsFin subroutine, 249
usvrcm.inc file, 123
uthick subroutine, 183

V

vamb subroutine, 335
vamb1 subroutine, 335
vapb subroutine, 335
vapb1 subroutine, 335
vapcb1 subroutine, 335
variables
defining additional, 138
fetching the index for saved, 164
optimization, 123
writing saved element index, 165
Variables
retrieving characters from, 327
vcross subroutine, 336
vdot function, 333
vector functions, 333
vectors
assigning equal values to two vectors, 334
assigning scalar constants to, 335
assigning scalar constants to incrementally, 335
combining two in one, 335
defining by a cross product, 336
getting the difference of two, 335
getting XYZ vector for a node, 337
initializing to zero, 334
moving one into another, 334
moving one into another incrementally, 334
multiplying a matrix by, 338
multiplying by a full transposed matrix, 338
multiplying one by a constant, 336
multiplying to a constant, 335
normalizing a three-component vector, 336
normalizing to unit length, 336
setting integer vector to zero, 334
storing XYZ vector for a node, 337
subtracting one from another, 335

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.

Vectors
  setting one to be the sum of two vectors, 335
velocity information, 123
vidot function, 333
viinit subroutine, 335
vimove subroutine, 334
vinit subroutine, 335
vmax function, 334
vmove subroutine, 334
vmult subroutine, 336
vmult1 subroutine, 336
vnorm subroutine, 336
vnorme subroutine, 336
vsum function, 333
vzero subroutine, 334

W

wavefront reordering debugging, 138
Weld
  defining, 98
wind turbine analysis
  fully coupled, 365
Windows systems
  compiling and linking UPFs on, 127
wrinqr function, 328
Writing an ANSYS substructure file, 73
wrtsub subroutine, 73

X

xyzup3 subroutine, 177

Z

Z-buffered graphics, 107

Release 18.2 - © ANSYS, Inc. All rights reserved. - Contains proprietary and confidential information of ANSYS, Inc. and its subsidiaries and affiliates.