Programmer's Manual for ANSYS

ANSYS Release 11.0

002328
January 2007

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

Copyright and Trademark Information

Disclaimer Notice

U.S. Government Rights

Third-Party Software

Published in the U.S.A.

Table of Contents

List of Figures

List of Tables

Preface

About the Programmer's Manual for ANSYS

The Programmer's Manual for ANSYS provides information about the various programming interfaces available to customers. This manual assumes that you have at least a basic knowledge of programming (a working knowledge of Fortran 90 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 90 routines that you can use to directly access the ANSYS database. You can also use these capabilities to access data in any of the binary files that ANSYS writes or uses.

Part II - Guide to ANSYS User Programmable Features

ANSYS provides a set of Fortran 90 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 90). 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 (APDL)* 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 90. 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.

Part I, Guide to Interfacing with ANSYS

Chapter 1: Format of Binary Data Files

1.1. What Are ANSYS Binary 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. ANSYS-written binary files include the following:

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

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

1.1.1. Conventions Used to Describe Binary Files

In the information describing the binary file formats:

- Record ID is the identifier for this record. Not all records will have identifiers; they're indicated only for records whose record pointers are stored in a header.
- Type indicates what kind of information this record stores.
- Number of records indicates how many records of this description are found here.
- Record length indicates the number of items stored in the record.

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

1.1.2. The Standard Header for ANSYS Binary Files

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

Item 1	The file number
Item 2	The file format. This item has a value of 1 if the file is small format, -1 if large format.
Item 3	The time, in compact form (i.e., 130619 is 13:06:19)

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

1.2. Format of the Results File

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

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

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

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

1.2.1. Nomenclature

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

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

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

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

1.2.2. Standard ANSYS File Header

See *Section 1.1.2: The Standard Header for ANSYS Binary Files* for a description of this set. File number (Item 1) is 12.

1.2.3. Results File Format

```
*comdeck,fdresu
c *** copyright(c) 2006 SAS IP, Inc.  All rights reserved.
c *** ansys, inc

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

      LONGINT         resufpL
      integer         resubk, resuut, resuLong, resuSpare(3)
      common /fdresu/ resufpL, resubk, resuut, resuLong, resuSpare

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
co  resuLong     0, old 32 bit integer form   1, 64 bit form (8.1)

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       40      .RST FILE HEADER

c                                       12,  maxn,  nnod,resmax,numdof,
c                                     maxe,  nelm,   kan, nsets,ptrend,
c                                    ptrDSI,ptrTIM,ptrLSP,ptrELM,ptrNOD,
c                                    ptrGEO,ptrCYC,CMSflg, csEls, units,
c                                   nSector,csCord,ptrEnd8,ptrEnd8,fsiflag,
c                                    pmeth,noffst,eoffst,nTrans,ptrTRAN,
c                                    kLong, csNds,cpxrst,extopt,nlgeom,
```

```
c                                       0,     0,     0,     0,     0

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                                               1000; 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                                      ptrDSI - pointer to the data steps index
c                                               table
c                                      ptrTIM - pointer to the table of time values
c                                               for a load step
c                                      ptrLSP - pointer to the table of load step,
c                                               substep, and cumulative iteration
c                                               numbers
c                                      ptrELM - pointer to the element equivalence
c                                               table
c                                      ptrNOD - pointer to the nodal equivalence
c                                               table
c                                      ptrGEO - pointer to the beginning of
c                                               geometry information
c                                      ptrCYC - 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                                      units  - unit system used
c                                               = 0 - user defined units
c                                               = 1 - SI (MKS)
c                                               = 2 - CSG
c                                               = 3 - U.S. Customary, using feet
c                                               = 4 - U.S. Customary, using inches
c                                               = 6 - MPA
c                                               = 7 - uMKS
c                                      nSector - number of sectors for cyclic sym
c                                      csCord  - Cyclic symmetry coordinate system
c                                      csEls   - Cyclic sym # eles in master sector
c                                      ptrEnd8  23,24  64 bit file length
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                                      ptrTRAN - pointer to SE transformation vects
c                                      kLong   - 1, 64 bit integer form
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   ---      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, SP01=27, SP02=28, SP03=29, SP04=30, SP05=31, SP06=32
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   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
c                                   the load step, substep, and cumulative
c                                   iteration numbers for each data set.

c   TRAN     dp    nTran     25     Substructure transformation vectors

c   GEO      i       1       40     Geometry data header(was 20 in 32 bit vers)

c                                        0,maxety, maxrl, ndnod,  nelm,
c                                   maxcsy,ptrETY,ptrREL,ptrNOD,ptrCSY,
c                                   ptrEID,     0,     0,     0,     0,
c                                   ptrMAS,csysiz,elmsiz,etysiz, rlsiz,
c                                         ptrETYL,      ptrRELL,
c                                   ptrCSYL,     ptrNODL,      ptrEIDL,
c                                         ptrMASL,     0,     0,     0,
c                                         0,    0,     0,     0,     0

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                                   ndnod  - 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                                   ptrNOD - 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                                   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                                  ptrETYPL - 64 bit pointer to TYPE
c                                  ptrRELL  - 64 bit pointer to REAL
c                                  ptrCSYL  - 64 bit pointer to CSYS
c                                  ptrNODL  - 64 bit pointer to NODES
c                                  ptrEIDL  - 64 bit pointer to ELEMENTS

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
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   NOD     dp       1     7*ndnod  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   NOD     dp     ndnod     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,     0,
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                                    NODES  - node numbers defining the element.
c                                             (See the ANSYS Elements Reference
c                                             for nodal order of an element).

c   MAS     dp       1  nnod*numdof  Diagonal mass matrix.



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,
c                                     ptrNSL,ptrESL, ptrRF,ptrMST, ptrBC,
c                                     rxtrap,  mode,  isym, kcmplx,numdof,
c                                       DOFS,
c                                     positions 51-70 - title,
c                                     positions 71-90 - stitle1,
c                                     dbmtim,dbmdat,dbfncl,soltim,soldat,
c                                     ptrOND,ptrOEL,nfldof,ptrEXA,ptrEXT
c                                     101-102  ptrEXA (was in 99)
c                                     103-104  ptrEXT (was in 100)
c                                     105-106  ptrNSL (was in 11)
c                                     107-108  ptrRF  (was in 13)
c                                     109-110  ptrMST (was in 14)
c                                     111-112  ptrBC  (was in 15)
c                                     113-114  ptrTRF (was in EXT 125)
c                                     115-116  ptrOND (was in 96)
c                                     117-118  ptrOEL (was in 97)
c                                     119-120  ptrESL (was in 12)
c                                     131-132  ptrVSL (was in EXT 196)
c                                     133-134  ptrASL (was in EXT 197)
c                                     139      numRotCmp
c                                     141-142  ptrRCM
c                                     143      nNodStr
c                                     145-146  ptrNODSTR

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 - pointer to nodal solution
c                                     ptrESL - pointer to element solution
c                                     ptrRF  - pointer to reaction forces
c                                     ptrMST - pointer to the masters
c                                     ptrBC  - 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 - pointer to the ordered node list
c                                           (load case files only)
c                                  ptrOEL - pointer to the ordered element list
c                                           (load case files only)
c                                  nfldof - number of extra Flotran DOFs/nodes
c                                           for this data set
c                                  ptrEXA - pointer to header extension for
c                                           FLOTRAN DOF/extra DOF list.
c                                  ptrEXT - pointer to header extension
c                                  numRotCmp - number of rotating components
c                                  ptrRCM - 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                                  ptrNODSTR  pointer to nodal component str
c
c                                  Note: ptrXXX are relative to ptrDSI

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

c                                   timfrq,lfacto,lfactn,cptime,  tref,
c                                    tunif, tbulk, volbase, tstep, 0.0,
c                                   velocity-acceleration-center of gravity
c                                    terms (positions 11-28)
c                                   if pmeth=0: load data (positions 51-100)
c                                   if pmeth=1: p convergence values
c                                    (positions 31-100)

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                                   0.0    - position not used

c                                   positions 11-13 -Linear acceleration terms
c                                   positions 14-16 - Angular velocity
c                                   positions 17-19 - Angular acceleration
c                                   positions 20-22 - Angular velocity about
c                                                     the center of gravity
c                                   positions 23-25 - Angular acceleration
c                                                     about the center of
c                                                     gravity
c                                   positions 26-28 - Center of gravity
c                                                     location
```

```
c                                    if pmeth=1:
c                                       positions 31-100 - P convergence values
c                                    if pmeth=0:
c                                       positions 51-100 - Load data

c                                       position 53 - Convergence key (if 1,
c                                                     substep converged)

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 principles 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
```

```
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   RF       i       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      i       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,     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                                         0,     0,     0,     0,     0

c                                  each item is described below:

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                                    positions 7-40 are unused.
```

```
c   DIX      i       1      numdis   Nodal constraint DOF.  This index is
c                                    calculated as N*32+DOF, where N is the node
c                                    number and DOF is the DOF reference number.
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   Nodal input force DOFs.  This index is
c                                    calculated as N*32+DOF, where N is the node
c                                    number and DOF is the DOF reference number.
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   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
c                                    the order in the element equivalence table.
c                                      (bit 12 (PDBE) in mask)

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 23 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,     0,ptrETB,
c                                    ptrECT,ptrEXY,ptrEBA,ptrESV,     0
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                                       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                                       0      - position not used

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
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,S1,
c                                       S2,S3,SI,SIGE.  Nodal order corresponds to
c                                       the connectivity defined in the element
c                                       description.  Stresses can be nodal values
c                                       extrapolated from the integration points or
c                                       values at the integration points moved to
c                                       the nodes.  If an element is nonlinear,
c                                       integration point values always will be
c                                       written.  (See item rxtrap in the solution
c                                       header for the setting.)  An element is
c                                       considered nonlinear when either plastic,
c                                       creep, or swelling strains are present.
c
c                                       Definition of common terms referred here
c                                       and in subsequent EEL, EPL, ECR, ETH,
c                                       ENL, EUL and EPT 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 =  11 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                                     NL     - number of layers in layered
c                                              elements
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
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), and
c                                     magnetic (AZ).  Gradients can be nodal
c                                     values extrapolated from the integration
c                                     points or values at the integration points
c                                     moved to the nodes.  See item rxtrap in the
c                                     solution header for the setting.  The
c                                     number of items in this record is
c                                     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).

c                                    * For lower-order elements, angles are
c                                      at the centroid and the number of items
c                                      in this record is 3.
c                                    * For higher-order elements, angles
c                                      are at each corner node and the number of
c                                      items in this record is nodstr*3.
c                                    * For layered shells, higher-order layered
c                                      solid elements, and layered SOLSH190 and
c                                      SOLID185, angles are at each corner node,
c                                      plus the layer orientation angle for each
c                                      layer. The number of items in this record is
c                                      (nodstr*3)+NL.
c                                    * For other lower-order layered solid elements,
c                                      Euler angles are at the centroid, plus
c                                      the layer orientation angle for each layer.
c                                      Therefore, the number of items in this record
c                                      is 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, PSV,
c                                   PLWK, CRWK, and ELENG 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                                     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
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   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   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 element types 2,42,45,
c                                    82,92,95, and 180 to 187, 190, 208, and 209.

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 element types 2,42,45,
c                                    82,92,95, and 180 to 187.

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

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


c *** Nodal Component Stresses (unused)

c  NODSTR   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 *Section 1.1.2: The Standard Header for ANSYS Binary Files* for a description of this set. File number (Item 1) is 10.

1.3.2. RDSP File Format

```
*comdeck,fdrdsp
c *** copyright(c) 2006 SAS IP, Inc.  All rights reserved.
c *** ansys, inc


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

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

c  write:  lnfrcl,lnfrin,lnfrwr
c  write:  rdtrcl,rdtrin,rdtrwr
c  read:   rdtrrs,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       40    .RDSP FILE HEADER
c
c                                    fun10,  nmrow, nmatrx,  nmode, numdof,
c                                     maxn,  wfmax, lenbac,  ngaps, ncumit,
c                                       kan,   nres,   ndva,      0,      0,
c                                         0,      0,      0,      0,      0,
c                                   ptrDOF, ptrDNC, ptrSTF, ptrMAS, ptrDMP,
c                                   ptrFRQ, ptrDSP,ptrSTFh,ptrMASh,ptrDMPh,
c                                  ptrFRQh,ptrDSPh, ptrDVA,ptrDVAh,      0,
c                                        0,      0,      0,      0,      0

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                                   ndva   - length of DVA
c                                   0      - position not used
c                                   0      - position not used
c                                   0      - position not used
c                                   0      - position not used
c                                   0      - position not used
c                                   0      - position not used
c                                   0      - position not used
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 calculated
c                                            displacements
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
c                                ptrDVAh- High part of modal disp, velo and acc
c                                0      - position not used
c                                0      - position not used
c                                0      - position not used
c                                0      - position not used
c                                0      - position not used
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  This table equates the actual node number to
c                                the number used for storage.
c                                  (Back(i),i=1,lenbac)

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                                  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                                  timend - the final time of the analysis

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 (using
c                                    the subspace mode extraction 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 (using the subspace
c                                    extraction technique). 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 (using the subspace mode extraction
c                                    technique), 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
c                                    analysis. This record is present only if the
c                                    analysis uses 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 2 records are repeated (as a pair) 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+5   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). For the last five
c                                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                                    (u(i),i=1,nmrow),time,itime,iter,ncumit,
c                                    scale
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  ---      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 *** The next 3 records are kept for possible restart using mode superposition
c *** method. They are overwritten upon restarting. They are written once (last
c *** loadstep).

c  DVA      dp      1     ndva+5  Calculated modal displacements
c                                   The first ndva entries are the modal
c                                   displacements. For the last five
c                                   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                                       (um(i),i=1,ndva),time,itime,iter,ncumit,
c                                       scale

c  ---      dp      1     ndva    Calculated modal velocities
c                                        (vm(i),i=1,ndva)

c  ---      dp      1     ndva    Calculated modal accelerations
c                                        (am(i),i=1,ndva)
```

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 *Section 1.1.2: The Standard Header for ANSYS Binary Files* for a description of this set. File number (Item 1) is 10.

1.4.2. RFRQ File Format

```
*comdeck,fdrfrq
c *** copyright(c) 2006 SAS IP, Inc.  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  write:  harmcl,harmin,harmwr
c  write:  hrfrcl,hrfreq
c  read:   harstr

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,      0, ncumit,
c                                        kan,      0,      0,      0,      0,
c                                          0,      0,      0,      0,      0,
c                                    ptrDOF, ptrDNC, ptrSTF, ptrMAS, ptrDMP,
c                                    ptrFRQ, ptrDSP,ptrSTFh,ptrMASh,ptrDMPh,
c                                   ptrFRQh,ptrDSPh,      0,      0,      0,
c                                         0,      0,      0,      0,      0

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                                   0      - position not used
c                                   ncumit - total number of iterations done
c                                            during analysis
c                                   kan    - analysis type
c                                            = 6  - reduced harmonic
c                                   0      - position not used
c                                   0      - position not used
c                                   0      - position not used
c                                   0      - position not used
c                                   0      - position not used
c                                   0      - position not used
c                                   0      - position not used
c                                   0      - position not used
c                                   0      - position not used
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                                  0      - position not used
c                                  0      - position not used
c                                  0      - position not used
c                                  0      - position not used
c                                  0      - position not used
c                                  0      - position not used
c                                  0      - position not used
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  This table equates the actual node number to
c                                  the number used for storage.
c                                   (Back(i),i=1,lenbac)

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.

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 (using
c                                   the subspace mode extraction 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 (using the subspace
c                                   extraction technique). 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 (using the subspace mode extraction
c                                   technique), 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 (using the
c                                   reduced mode extraction technique).
c                                    (freq(i),i=1,nmrow)

c  DSP      cmp  ncumit  nmrow+5   Calculated complex 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). For the last five
c                                   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                                     (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.
```

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 *Section 1.1.2: The Standard Header for ANSYS Binary Files* for a description of this set. File number (Item 1) is 9.

1.5.2. MODE File Format

```
*comdeck,fdmode
c *** copyright(c) 2006 SAS IP, Inc.  All rights reserved.
c *** ansys, inc.

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

c *** mpg fdmode < modspc romstr lire_freq_mode lire_nb_mode: mode file desc

      character*8  MODENM
      parameter  (MODENM='mode    ')
      LONGINT         modefpL, modefp
      integer         modebk,  modeut

      common /fdmode/ modefpL, modebk, modeut
      equivalence  (modefp,modefpL)

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


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       60      .MODE FILE HEADER
c
c                                    fun09,  nmrow, nmatrx,  nmode, numdof,
c                                     maxn,  wfmax, lenbac,      0,  nontp,
c                                   lumpms, extopt, SvCode,    kan, ldstep,
c                                   numitr, expbeg, expend, nspect, nSPdat,
c                                   ptrRDF, ptrFRQ, ptrPRT, ptrSHP, ptrLOD,
c                                   ptrSTF, ptrMAS, ptrDMP, ptrCOF, ptrDCF,
c                                   ptrLPM, ptrSP1,ptrSHPh,ptrLODh,ptrSTFh,
c                                 ptrMASh,ptrDMPh,ptrLPMh,ptrSP1h,ptrIRHSl,
c                                ptrIRHSh,PowerDyn,     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                                   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.
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                                   0      - position not used
c                                   nontp  - number of equations on the .TRI
c                                            file (Does not apply if extopt =
c                                            0.)
c                                   lumpms - lumped mass key
c                                            = 0 - default matrix type
c                                            = 1 - lumped
c                                            (Does not apply if extopt = 3 or
c                                            4.)
c                                   extopt - mode extraction method
c                                            = 0 - reduced
c                                            = 1 - subspace
c                                            = 3 - unsymmetric Lanczos
c                                            = 4 - damped Lanczos
c                                            = 6 - block Lanczos
c                                            = 7 - QR damped
c                                            = 8 - AMLS
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
c                                   numitr - total number of cumulative
c                                            iterations done during analysis
c                                            (Does not apply if extopt = 3 or
c                                            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
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                                  ptrPRT - pointer to the participation
c                                           factors
c                                  ptrSHP - pointer to the mode shapes
c                                           (eigenvectors)
c                                  ptrLOD - pointer to the load vectors
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
c                                           (if extopt=7 : pointer to the modal
c                                             damping matrix)
c                                  ptrCOF - pointer to the mode coefficients
c                                  ptrDCF - pointer to the modal damping
c                                           coefficients
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 (currently only set in subout.F)
c                                           = 0 Subspace, Block Lanczos, etc... methods
c                                           = 1 Powerdynamics method
c                                  0      - position not used
c                                  0      - position not used
c                                  0      - position not used
c                                  0      - position not used
c                                  0      - position not used
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  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  FRQ      dp      1     nmode     Frequencies (eigenvalues). Frequencies are
c                                   complex if extopt=3 or 4. 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                                    (freq(i),i=1,nmode)

c  PRT      dp      1     nmode     Participation factors. Factors are complex
c                                   if EXTOPT=3 or 4.
c                                    (pfact(i),i=1,nmode)

c  COF      dp      1     nmode     Mode coefficients
c                                    (modecf(i),i=1,nmode)

c  DCF      dp      1     nmode     Modal damping coefficients
c                                    (dampmd(i),i=1,nmode)

c  SHP      dp    nmode   nmrow     Mode shapes (eigenvectors). Mode shapes are
c                                   complex if extopt=3 or 4. 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
```

```
c                                   nodal equivalence table
c                                    (psi(i,j),i=1,nmrow)

c  LOD      dp      1     nmrow     Load vector. This record is present only if
c                                   extopt=0 or 1.
c                                    (f(i),i=1,nmrow)

c IRHS      dp      1     nmrow     Imaginary Load vector.  This record is present
c                                   only if extopt = 6.

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 damping matrix. Each row of the
c                                   matrix is stored as a record. The matrix is
c                                   present only if nmatrx > 2. Row order is the
c                                   same as the DOF order stored at position
c                                   ptrRDF.
c                                    (ac(i,j),i=1,nmrow)

c for each spectrum (nspect records):
c  SP1      dp      1     nmode     Mode coeff for this spectra
c  ---      dp      1     nmode     Modal damping values
c  ---      dp      1     130       svcom: freqtb, etc.
c  ---      dp      1     20        misc. spectra data
```

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 *Section 1.1.2: The Standard Header for ANSYS Binary Files* for a description of this set. File number (Item 1) is 2.

1.6.2. EMAT File Format

```
*comdeck,fdemat
c *** copyright(c) 2006 SAS IP, Inc.  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
```

```
      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,      0,      0, nodref,  lumpm,
c                                     kygst,   kygm,   kycd,  kygss,  kygaf,
c                                     kygrf,      0,      0,      0,      0,
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                                    0      - position not used
c                                    0      - position not used
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
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                                  0      - position not used
c                                  0      - position not used
c                                  0      - position not used

c                                  ptrELMh- Highpointer to element equivalence
c                                           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
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  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,      0,      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                                  0       - position not used
c                                  0       - position not used
c                                  nmrow   - numbers/columns in matrices. If the
c                                          number is negative, the matrices
c                                          will be written in lower 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, and
c                                   stress stiffening matrix. 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 lower
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 psolve
c      command, such as the case of a prestressed nonlinear modal 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
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 *Section 1.1.2: The Standard Header for ANSYS Binary Files* for a description of this set. File number (Item 1) is 8.

1.7.2. SUB File Format

```
*comdeck,fdsub
c *** copyright(c) 2006 SAS IP, Inc.  All rights reserved.
c *** ansys, inc

c     **********  description of substructure matrix file  **********
      character*8  SUBNM
      parameter  (SUBNM='sub    ')
      LONGINT        subfpL, lenSubL
      integer        subbk, subut
      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,
c                                  kstf,  kmass,  kdamp,    kss,  nvect,
c                                nWorkL,  lenU1, sesort, lenlst,ptrLodL,
c                                ntrans, ptrMtx, ptrXFM, ptrHED,  name1,
c                                 name2,      0,      0,  name3,  name4,
c                                ptrDOF, ptrDST, ptrBAC, ptrTIT, ptrNOD,
c                                ptrXYZ, ptrEDG, ptrGDF, thsubs, ptrPOS,
c                                ptrORG, stfmax,ptrLodH, nmodes, keydim,
c                              cmsMethod,  name5,  name6,  name7,  name8,
c                                nvnodes,ptrCTXM, nWorkh,  ptrCG,      0,
c                                      0,      0,      0,      0,      0

c  HED      i       1       60      .SUB FILE HEADER  (SPARSE MATRICES)
c
c                                     9,   nEqn, nmatrx,       , numdof,
c                                  maxn,       , lenbac,   nnod, kunsym,
c                                  kstf,  kmass,  kdamp,       ,  nvect,
```

```
c                               nTermL,       ,       ,       ,ptrLodL,
c                                     ,ptrMtxL,       , ptrHED,  name1,
c                                name2,       ,       ,  name3,  name4,
c                               ptrDOF,       , ptrBAC, ptrTIT, ptrNOD,
c                               ptrXYZ,       ,       , thsubs,       ,
c                                     , stfmax,ptrLodH,       ,       ,
c                                     ,  name5,  name6,  name7,  name8,
c                                     ,       , nTermH,ptrMtxH,ptrColL,
c                              ptrColH,ptrCofL,ptrCofH,      0,      0

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
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 - thermal key
c                                        = 0 - structural
c                                        = 1 - thermal
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                              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                             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
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      i       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  CG      dp       1      10    total mass,CGx,CGy,CGz,6 moments of inertia


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 *Section 1.1.2: The Standard Header for ANSYS Binary Files* for a description of this set. File number (Item 1) is 8.

1.8.2. CMS File Format

```
*comdeck,fdcms
c --- description of cms(component modal synthesis) transformation file
```

```
      character*8  CMSNM
      parameter  (CMSNM='cms     ')
      LONGINT         cmsfpL, cmsfp
      integer         cmsbk, cmsut

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

c --- common variable desciption ------------------------------------------------
co  cmsfp       file position on file mode
co  cmsbk       block number for file mode
co  cmsut       file unit for file mode


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
c     --- type tells what kind of information is stored in this record:
c     i   - integer
c     dp  - double precision
c     cmp - complex
c
c     --- nrec tells how many records of this description are found here
c
c     --- lrec tells how long the records are (how many items are stored)
c
c -------------------------------------------------------------------------------
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                                        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                                   ptrIRFS,ptrNORS,ptrCSTS,      0,ptrIRFL,
c                                   ptrNORL,ptrCSTL,      0,      0,      0,

c                                   each item in header is described below:
cc                                   fun45  - unit number
c                                    neqn   - number of equation in BCS
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 NA in file
c                           ptrIRFS,ptrIRFL  - pointer to inertia relief modes
c                           ptrNORS,ptrNORL  - pointer to normal modes
c                           ptrCSTS,ptrCSTL  - pointer to constraint modes
c
c                                    0      - position not used

c   ---      i       1    neqn      BCS to ANS mapping (lBCStoANS(i), i= 1,neqn)
c
c   NOR     dp   nnorm    neqn      Normal Modes
c
c   IRF     dp   nirfm    neqn      Inertia Relief Modes
c
c   CST     dp   ncstm    neqn      Constraint Modes
```

1.9. Description of the Triangularized Stiffness File

This section explains the contents of the triangularized stiffness file (*jobname*.tri).

1.9.1. Standard ANSYS File Header

See *Section 1.1.2: The Standard Header for ANSYS Binary Files* for a description of this set. File number (Item 1) is 11.

1.9.2. TRI File Format

```
*comdeck,fdtri
c *** copyright(c) 2006 SAS IP, Inc.  All rights reserved.
c *** ansys, inc
c
c
c     **********  description of triangularized stiffness file  **********
c
c *** mpg fdtri.inc < stff10 slvstr romstr modstr: tri file description
c
      character*8  trinm, TriName
      parameter  (trinm='tri     ',TriName='Tri Bufr')

      LONGINT  trifp
      integer  tribk,  triut
      common /fdtri/   trifp,  tribk,  triut

c  open:     slvstr                      subspc
c  write:    eqsol,eqclos,lfout,cfout    subtri
c  read:     stff10,bacfil               subfwd,subbac
c  close:    bacfil,svkan2               subspc
c
c   ********** common variable descriptions ***********
co  trifp       file position on file tri  (LONGINT)
co  tribk       block number for file tri
co  triut       file unit for file tri
c
c   See fddesc for documentation of how binary files are stored.
c
c   The TRI file is generated for static and reduced[modal,harmonic,transient,
c    substructure] analyses
c
c     **********  file format  **********
c
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
c        type tells what kind of information is stored in this record:
c             i - integer
c             dp - double precision
c             cmp - complex
c
c        nrec tells how many records of this description are found here
c
c        lrec tells how long the records are (how many items are stored)
c
c recid    type    nrec    lrec     contents
c
c  ---      i       1      100      standard ANSYS file header (see binhed for
c                                   details of header contents)
c
c  ---      i       1       20      .TRI FILE HEADER
c
c                                    fun11,  nontp,  nmast,      1,    kan,
c                                    wfmax, lenbac, numdof, ptrMST, ptrend,
c                                    lumpm, keyuns, ptrMS1, ptrMS2, ptrEN1,
c                                   ptrEN2,      0,      0, ptrTRI,      0
```

```
c
c                                  each item in header is described below:
c
c                                   fun11  - unit number (tri file is 11)
c                                   nontp  - number of equations on file
c                                   nmast  - number of master dofs
c                                   1      - position not used, always = 1
c                                   kan    - analysis type
c                                   wfmax  - maximum wavefront
c                                   lenbac - number of nodes
c                                   numdof - number of degrees of freedom (DOF)
c                                            per node
c                                   ptrMST - 32 bit pointer to the master dof
c                                            list, only here for backward
c                                            compatibility.  Do not use if
c                                            ptrMS1 or ptrMS2 are non-zero
c                                   ptrend - 32 bit pointer to the end of file
c                                            only here for backward
c                                            compatibility.  Do not use if
c                                            ptrEN1 or ptrEN2 are non-zero
c                                   lumpm  - lumped mass key
c                                            = 0 - default matrix type
c                                            = 1 - lumped
c                                   keyuns - unsymmetric key
c                                            = 0 - the matrix is not unsymmetric
c                                            = 1 - the matrix is unsymmetric
c                                   ptrMS1,
c                                   ptrMS2 - These two values are two halves of
c                                            a 64 bit pointer that points to the
c                                            master dof list
c                                   ptrEN1,
c                                   ptrEN2 - These two values are two halves of
c                                            a 64 bit pointer that points to the
c                                            end of file
c                                   0      - position not used
c                                   0      - position not used
c                                   ptrTRI - pointer to the beginning of the
c                                            triangularized matrix data
c                                   0      - position not used
c
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
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
c  TRI   At this point in the file, the triangularized matrix information is
c        stored.  The info is written row by row, and there are two different
c        storage options for writing a row.  If the row being written does not
c        have a constraint equation associated with it, then two records are
c        written to describe the row.  If the row being written has a constraint
c        equation associated with it, then five records are written to describe
c        the row.  Both formats are shown below.  These groupings of two or five
c        records per row will be written a total of nontp times (to include all
c        rows)
c
c  The next two descriptions show the format for a row that does not have a
c  constraint equation associated with it:
c
c  ---   dp/cmp     1     varies    A row of the triangularized matrix.
c
c                                   If keyuns=0, this record will contain the
c                                    non-diagonal terms of this column, the
c                                    diagonal term itself, the normalized F
c                                    term, followed by the reciprocal of the row
```

```
c                                    pivot.
c
c                                   If keyuns=1, this record will contain the
c                                    non-diagonal terms of this column, the
c                                    diagonal term itself, the normalized F
c                                    term, the reciprocal of the row pivot,
c                                    followed by the non-diagonal terms of this
c                                    row.
c
c                                   The length of this record will vary (actual
c                                   length is returned from routine BINRD8). If
c                                   kan=3, this record contains complex
c                                   information, otherwise it contains double
c                                   precision information.
c                                    (ktri(i),i=1,n),vload,diag (symmetric)
c                                    (ktri(i),i=1,n),vload,diag,(utri(i),i=1,n)
c                                     (unsymmetric)
c
c  ---      i       1     varies    Triangular matrix row indices. The first
c                                   item signifies what term in the row belongs
c                                   to the pivot. The second term signifies what
c                                   DOF is being eliminated, and the remaining
c                                   items signify the new DOFs being introduced
c                                   (if any). The length of this record will
c                                   vary (actual length is returned from routine
c                                   BINRD8).
c                                    (lll(i),i=1,m)
c
c  The next five descriptions show the format for a row that has a constraint
c  equation associated with it.
c
c  ---      dp      1        2      A flag record, indicating that constraint
c                                   equations are being stored, and the storage
c                                   is as shown here.  Both values are TINY.
c
c  ---      dp      1     varies    Coefficients of the constraint equation.
c                                   The length of this record will vary (actual
c                                   length is returned from routine BINRD8).
c                                    (coeff(i),i=1,n+2)
c
c  ---   dp/cmp     1     varies    A row of the triangularized matrix.
c
c                                   If keyuns=0, this record will contain the
c                                    non-diagonal terms of this column, the
c                                    diagonal term itself, the normalized F
c                                    term, followed by the reciprocal of the row
c                                    pivot.
c
c                                   If keyuns=1, this record will contain the
c                                    non-diagonal terms of this column, the
c                                    diagonal term itself, the normalized F
c                                    term, the reciprocal of the row pivot,
c                                    followed by the non-diagonal terms of this
c                                    row.
c
c                                   The length of this record will vary (actual
c                                   length is returned from routine BINRD8). If
c                                   kan=3, this record contains complex
c                                   information, otherwise it contains double
c                                   precision information.
c                                    (ktri(i),i=1,n),vload,diag (symmetric)
c                                    (ktri(i),i=1,n),vload,diag,(utri(i),i=1,n)
c                                     (unsymmetric)
c
c  ---      i       1     varies    Triangular matrix row indices. The first
c                                   item signifies what term in the row belongs
c                                   to the pivot. The second term signifies what
c                                   DOF is being eliminated, and the remaining
c                                   items signify the new DOFs being introduced
c                                   (if any). The length of this record will
c                                   vary (actual length is returned from routine
c                                   BINRD8).
```

```
c                                    (lll(i),i=1,m)
c
c  ---      i       1        2      This record indicates the end of the 5
c                                   record storage for this row.  It is included
c                                   for situations when the file is being read
c                                   from the bottom up.  Its contents:
c
c                                   -cenum,    n
c
c                                   cenum  - the constraint equation number for
c                                            the constraint equation stored
c                                            above
c                                   n      - the length of a row of the matrix
c                                            being written
c
c  MST      i       1      nmast    This record is present only if nmast > 0.
c   or                              This index is calculated as
c  MS1,                             (N-1)*NUMDOF+DOF, where N is the position
c  MS2                              number of the node in the nodal equivalence
c                                   table, and DOF is the DOF reference number
c                                   given above
```

1.10. Description of the Full Stiffness-Mass File

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

1.10.1. Standard ANSYS File Header

See *Section 1.1.2: The Standard Header for ANSYS Binary Files* for a description of this set. File number (Item 1) is 4.

1.10.2. FULL File Format

```
*comdeck,fdfull
c *** copyright(c) 2006 SAS IP, Inc.  All rights reserved.
c *** ansys, inc.

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

c *** mpg fdfull.inc < stff10 slvstr: full file description

      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=60)

      LONGINT         fullfpL, fullfp
      integer         fullbk,  fullut, wrLdstep, wrSbstep, wrEqiter
      common /fdfull/ fullfpL, fullbk, fullut,
     x                wrLdstep,wrSbstep,wrEqiter
      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       60      .FULL FILE HEADER
c
c                                    fun04,   neqn,  nmrow, nmatrx,    kan,
c                                    wfmax, lenbac, numdof,jcgtrmL,jcgtrmH,
c                                    lumpm, jcgeqn, jcgtrm, keyuns, extopt,
c                                    keyse, sclstf, nxrows,ptrIDXl,ptrIDXh,
c                                   ncefull,ncetrm,ptrENDl,ptrENDh,      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                                        0,      0,      0,      0,      0

c NOTE: If fun04 > 0, then the file was created with frontal assembly
c       If fun04 < 0, then the file was created with symbolic assembly;see below
c                     for its format

c -------------------------- frontal assembled file --------------------------

c                                  each item in header is described below:

c                                    fun04  - unit number (full file is 4)
c                                    neqn   - number of equations on file
c                                    nmrow  - number of rows in matrices
c                                    nmatrx - number of matrices on file
c                                    kan    - analysis type
c                                    wfmax  - maximum wavefront
c                                    lenbac - number of nodes
c                                    numdof - number of dofs per node
c                                    jcgtrmL,jcgtrmH - number of coefficients
c                                    lumpm  - lumped mass key
c                                             = 0 - default matrix type
c                                             = 1 - lumped
c                                    jcgeqn - number of jcg equations
c                                    jcgtrm - pre-8.1 this is the number of
c                                             coefficients in sparse jcg
c                                             storage (otherwise this value
c                                             must be 0 and jcgtrmL,jcgtermH
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                                    keyse  - superelement key; set if at least
c                                             one superelement
c                                    sclstf - scale factor for matrices
c                                    nxrows - the maximum rank for this solution
```

```
c                                   ptrIDXl- pointer to the matrix row indices.
c                                   ptrIDXh- high part of row index pointer
c                                   ncefull- Number of constraint equations on
c                                            the full file
c                                   ncetrm - Total number of terms in the
c                                            constraint equations
c                                   ptrENDl- Low part of 64 bit end of file ptr
c                                   ptrENDh- High part of 64 bit end of file ptr
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. This table equates
c                                   the number used for storage to the actual
c                                   node number
c                                    (Back(i),i=1,lenbac)

c  NOTE: The next five records are repeated as a group neqn times.
c   When the matrices get written, one row of each matrix is written to the file
c   at a time. i.e. the first row of each matrix is written, then the second row
c   of each matrix, etc. this pattern continues until all the rows of each
c   matrix have been written to the file.  If kan=3, the matrix rows will be
c   complex valued, otherwise they will be double precision values.


c  IDX      i       1      varies   Matrix row indices. The first
c                                   item signifies what term in the row belongs
c                                   to the pivot. The second term signifies what
c                                   DOF is being eliminated, and the remaining
c                                   items signify the new DOFs being introduced
c                                   (if any). The length of this record will
c                                   vary (actual length is returned from routine
c                                   BINRD8)
c                                    (lll(i),i=1,m)

c  ---      i       1      varies   A second level of indexing for the
c                                   matrix. Indicates positions and
c                                   values of terms to be reduced. The length of
c                                   this record will vary (actual length is
c                                   returned from routine BINRD8)
c                                    (index(i),i=1,n) for compressed rows

c  ---   dp/cmp     1      varies   Stiffness matrix.

c                                   If keyuns=0, this record will contain the
c                                    non-diagonal terms of this column, the
c                                    diagonal term itself, followed by the
c                                    normalized F term.

c                                   If keyuns=1, this record will contain the
c                                    non-diagonal terms of this column, the
c                                    diagonal term itself, the non-diagonal terms
c                                    of this row, followed by the normalized F
c                                    term.

c                                   If lumpm = 1, then the mass for this node is
c                                   located after the F term.  The length of
c                                   this record will vary (actual length is
c                                   returned from routine BINRD8)
c                                    (krow(i),i=1,n),vload,(mass) (symmetric)
c                                    (n-1 column) diag (n-1 row) load (dmass)
c                                     (unsymmetric)

c  ---   dp/cmp     1      varies   Mass matrix.  This record exists only if
c                                   nmatrix > 1.
```

```
c                                  If keyuns=0, this record will contain the
c                                   non-diagonal terms of this column, the
c                                   diagonal term itself, followed by the
c                                   normalized F term.

c                                  If keyuns=1, this record will contain the
c                                   non-diagonal terms of this column, the
c                                   diagonal term itself, followed by the
c                                   non-diagonal terms of this row.

c                                  The length of this record will vary (actual
c                                  length is returned from routine BINRD8)
c                                   (mrow(i),i=1,n) (symmetric)
c                                   (n-1 column) diag (n-1 row) (unsymmetric)
c
c                                  If lumpms=1, this record contains one double
c                                  array with diag values

c  ---   dp/cmp     1      varies   Damping matrix.  This record exists only if
c                                  nmatrx > 2.

c                                  If keyuns=0, this record will contain the
c                                   non-diagonal terms of this column, the
c                                   diagonal term itself, followed by the
c                                   normalized F term.

c                                  If keyuns=1, this record will contain the
c                                   non-diagonal terms of this column, the
c                                   diagonal term itself, followed by the
c                                   non-diagonal terms of this row.

c                                  The length of this record will vary (actual
c                                  length is returned from routine BINRD8)
c                                   (ceqn(i),i=1,n) (symmetric)
c                                   (n-1 column) diag (n-1 row) (unsymmetric)

c -------------------------- symbolic assembled file --------------------------

c  ---      i       1       60      .FULL FILE HEADER
c
c                                     fun04,    neqn,   nmrow,  nmatrx,     kan,
c                                     wfmax,  lenbac,  numdof, ntermKl, ntermKh,
c                                     lumpm,   nmrow,  ntermK, keyuns,   extopt,
c                                     keyse,  sclstf,  nxrows, ptrSTFl, ptrSTFh,
c                                   ncefull, ntermMh, ptrENDl, ptrENDh,ptrIRHSl,
c                                  ptrIRHSh, ptrMASl, ptrMASh, ptrDMPl, ptrDMPh,
c                                    ptrCEl,  ptrCEh,  nNodes, ntermMl, ntermDl,
c                                   ptrDOFl, ptrDOFh, ptrRHSl, ptrRHSh, ntermDh,
c                                   ngMaxNZ,ptrNGPHl,ptrNGPHh,minKdiag,maxKdiag,
c                                  minMdiag,maxMdiag,minDdiag,maxDdiag, ngTerml,
c                                   ngTermh,ngTermCl,ngTermCh,       0,       0,
c                                         0,       0,       0,       0,       0

c                                 each item in header is described below:

c                                   fun04  - negative of the unit number (-4)
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
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 - AMLS (not implemented)
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 internal Nodes
c                                                considered by symbolic 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)
c                                   ngTermCl,h - total number of nonzeroes in nodal graph
c                                                (compressed graph based value)
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  ---      i       1      lenbac   Nodal equivalence table. This table equates
c                                   the number used for storage to the actual
c                                   node number

c Stiffness Matrix. The next two records are repeated as a group neqn times.

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.

c                                  If keyuns=1, this record will contain the
c                                   the 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 DOF information

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

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

c  ---      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  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)

c  ---     dp       1      varies   Matrix terms

c    if lumpm = 1:
c  ---     dp       1        neqn   Matrix diagonals

c Damping Matrix. The next two records are repeated as a group neqn times.

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 G matrix if ncefull > 0.

c  CE       i       1     ncefull   List of slave DOFs

c  ---     dp       1     ncefull   g vector (constant terms)

c  ---     dp       1     ncefull   imaginary g vector (constant terms). This
c                                   vector only exists for harmonic analyses.
c                                   (Antype == 3).

c  ---      i       1           4   Header; 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

c   Repeat for each row:

c  ---      i       1      varies   Column indices

c  ---     dp       1      varies   Column values
```

```
c Nodal graph vectors

c  NGPH     i       1      nNodes   number of nonzeroes for each node

c   Repeat for each node

c           i       1      varies   Index vector.


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)
```

Chapter 2: Accessing Binary Data Files

2.1. Accessing ANSYS Binary Files

The following section 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 BINLIB library is in the dynamic link library `\Program Files\Ansys Inc\V110\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/v110/ansys/customize/misc/<platform>/libbin.so` on UNIX systems (`libbin.sl` on HP systems).

Your distribution media also includes sample FORTRAN source files which employ the BINLIB library:

`bintrd.F`	The bintrd subroutine reads and prints the contents of an ANSYS binary file
`bintwr.F`	The bintwr subroutine copies an ANSYS binary file to a new file
`bintst.F`	The bintst program calls the bintwr and bintrd subroutines as an example of how to use the binlib library to print the contents of a file, copy the original file to a new file, and then print the contents of the new file. Routine bintst has no inputs or outputs. It requires use of the bintcm common. (For more information, see the descriptions of the bintrd and bintwr routines later in this chapter.)

These files reside in the subdirectory `\Program Files\Ansys Inc\V110\ANSYS\custom\misc\<platform>` (on Windows systems) or `/ansys_inc/v110/ansys/customize/misc` (on UNIX systems). To compile and link the BINTST program, a `makefile` procedure has been provided in the `\Program Files\Ansys Inc\V110\ANSYS\custom\misc\<platform>` subdirectory on windows or run the `bintst.link` procedure in the `/ansys_inc/v110/ansys/customize/misc` subdirectory on UNIX.

2.1.1. Access Routines to Results and Substructure Files

Demonstration programs for reading and writing ANSYS results and substructure files are included on the installation media:

- `ResRdDemo`
- `ResWrDemo`
- `rdsubs`
- `wrtsub`
- `rdfull`

On Windows systems:

The FORTRAN source for these programs is located in `\Program Files\Ansys Inc\V110\ANSYS\custom\misc` and the files are named `ResRdDemo.F`, `ResWrDemo.F`, `rdsubs.F`, `wrtsub.F`, and `rdfull.F`.

To link these demonstration programs, use the `\Program Files\Ansys Inc\V110\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\V110\ANSYS\custom\misc\<platform>\rdrwrt  ResRdDemo
```

Appropriate files will then be copied from \Program Files\Ansys Inc\V110\ANSYS\custom\misc\<*platform*> to your working directory, compiled, and linked. The resulting executable will also be placed in your current working directory.

Use the userprog command line option when writing your own customized program, naming the routine userprog.F. The resulting executable will be named userprog.exe. When userprog is used, no files are copied from \Program Files\Ansys Inc\V110\ANSYS\custom\misc\<*platform*> to your working directory.

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

On UNIX systems:

The FORTRAN source for these programs is located in /ansys_inc/v110/ansys/custom/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/v110/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/v110/ansys/customize/misc/rdrwrt.link  ResRdDemo
```

Appropriate files will then be copied from /ansys_inc/v110/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/v110/ansys/bin directory to run these programs, once linked. The procedure files are named ResRdDemo110, ResWrDemo110, rdsubs110, wrtsub110, and rdfull110.

Use the userprog command line option when writing your own customized program, naming the routine userprog.F. The resulting executable will be named userprog.e110. When userprog is used, no files are copied from /ansys_inc/v110/ansys/customize/misc to your working directory. The procedure file is named userprog110.

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.

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.

AUX2 does not use the data pointers discussed in item 5 above. It uses record numbers to locate the binary file data to dump. Although the information that AUX2 provides includes the pointer, using the pointer alone will not get you that information. To get it, you must correlate the pointer and the record number by trial and error.

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
```

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 blocks 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
c                                            (in words)
c                                    = 9 - return starting word for this page
c                                            in buffer

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,
     x                 pname,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     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
```

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 (MKS)
c                                    = 2 - CSG
```

```
c                                 = 3 - U.S. Customary, using feet
c                                 = 4 - U.S. Customary, using inches
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 (MKS)
c                    = 2 - CSG
c                    = 3 - U.S. Customary, using feet
c                    = 4 - U.S. Customary, using inches
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                            (as defined with subroutine bioset)
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                            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
c
```

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
```

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 *Section 2.1: Accessing ANSYS Binary Files* 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 *** ansys(r) copyright(c) 2000
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
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 *** ansys(r) copyright(c) 2000
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(c) 2006 SAS IP, Inc.  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(c) 2006 SAS IP, Inc.  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 if the **PSOLVE**,ELFORM, **PSOLVE**,ELPREP, **PSOLVE**,TRIANG sequence is used. You can also use the **WRFULL** command.

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

```
*deck,rdfull
      program  rdfull
c *** primary function:    demonstrates use of binary access routines
c *** secondary function:  Read and reformat full file
c
c *** copyright(c) 2006 SAS IP, Inc.  All rights reserved.
c *** ansys, inc.
c
c  NOTICE - A new assembly process, termed 'symbolic assembly', has
c           replaced the old assembly process, termed 'frontal
c           assembly', and is now the default assembly process for
c           most analyses.  This program demonstrates how to read
c           and reformat the .FULL file that was created using
c           frontal assembly or symbolic assembly.  ANSYS writes the
c           .FULL file if the PSOLVE,ELFORM
c                             PSOLVE,ELPREP
c                             PSOLVE,TRIANG
c           sequence is used.  ANSYS will also write the .FULL file
c           when the sparse, ICCG, or JCG solver is used, as well as
c           when most mode extraction methods are used.

c    Be sure to set up for modal  ANTYPE,2
c    and full subspace            MODOPT,SUBSP,nmode,0,0, ,OFF
c       (nmode is not used - it can be any value

c      If the free-free stiffness and mass matrices are desired,
c        make sure there are no constraints on the model.
```

2.2.7. Program ResRdDemo (Demonstrates Reading a Results File)

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

This file resides in the subdirectory `\Program Files\Ansys Inc\V110\ANSYS\custom\misc\<platform>` (on Windows systems) or `/ansys_inc/v110/ansys/customize/misc` (on UNIX systems).

2.2.8. Program ResWrDemo (Demonstrates Writing a Results File)

Program `ResWrDemo` demonstrates how to write an ANSYS-readable results file. This file resides in the subdirectory `\Program Files\Ansys Inc\V110\ANSYS\custom\misc\<platform>` (on Windows systems) or `/ansys_inc/v110/ansys/customize/misc` (on UNIX systems).

2.3. Results File Access Routines

You can use the low-level routines in described in *Section 2.1: Accessing ANSYS Binary Files* 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 routines are included on the installation CD (compressed); you can install them on your system by performing a custom installation (as described in the *ANSYS Installation and Configuration Guide* for your platform).

These files reside in the subdirectory `\Program Files\Ansys Inc\V110\ANSYS\custom\misc\<platform>` (on Windows systems) or `/ansys_inc/v110/ansys/customize/misc` (on UNIX systems). See *Section 2.1.1: Access Routines to Results and Substructure Files* 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
  function ResRdSolBegin (key,lstep,substep,ncumit,kcmplx,time,
    subroutine ResRdDispBegin
      function ResRdDisp (node,Disp)
    subroutine ResRdDispEnd
    subroutine ResRdRforBegin (nRForce)
      function ResRdRfor (node,idof,value)
    subroutine ResRdRforEnd
    subroutine ResRdBCBegin (BCHeader)
      subroutine ResRdFixBegin (BCHeader,nFixed)
        function ResRdFix (node,idof,value)
      subroutine ResRdFixEnd
      subroutine ResRdForcBegin (BCHeader,nForces)
        function ResRdForc (node,idof,value)
      subroutine ResRdForcEnd
    subroutine ResRdBCEnd
    subroutine ResRdEresBegin
      function ResRdEstrBegin  (iElem)
        function ResRdEstr (iStr,Str)
      subroutine ResRdEstrEnd
    subroutine ResRdEresEnd
  subroutine ResRdSolEnd
subroutine ResRdEnd
```

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

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

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

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

```
*deck,ResRdBegin
      function ResRdBegin (Nunit, Lunit, Fname, ncFname, Title, JobName,
     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*8,sc,out)     - Jobname from file
c     Units    (int,sc,out)      - 0, unknown  1, SI  2, CSG
c                                  3, U.S. Customary - foot
c                                  4, U.S. Customary - inch
c                                  6, MPA
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 used
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)
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
```

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:
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 NumNode)

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
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. 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)       - 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     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     (int,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
```

2.3.10. ResRdDisp (Retrieving Nodal Solution)

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

c object/library: ResRd

c  input arguments:
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
```

2.3.11. 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)
```

2.3.12. ResRdFix (Retrieving Applied Nodal Constraints)

```
*deck,ResRdFix
      function ResRdFix (node,idof,value)
c primary function:    Retrieve a constraint value

c object/library: ResRd

c  input arguments:
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)
```

2.3.13. ResRdForc (Retrieving Applied Nodal Loads Solution)

```
*deck,ResRdForc
      function ResRdForc (node,idof,value)
c primary function:    Retrieve an applied force value

c object/library: ResRd

c  input arguments:
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)
```

2.3.14. 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
```

Chapter 3: Using CDREAD and CDWRITE

3.1. Using the CDREAD Command

The **CDREAD** command and its GUI equivalent, **Main Menu> Preprocessor> Archive Model> Read**, read a file of model and database information into the ANSYS database. The commands and menu paths listed below define the input data that the ANSYS program requires to solve a model. If the file you are reading into the database via **CDREAD** or **Main Menu> Preprocessor> Archive Model> Read** does not contain all of the required input data, you can use these commands or menu paths to define that data in the preprocessor. For detailed information about the commands, see the *Commands Reference*.

In the following list, commands or menu paths shown with an asterisk (*) are the minimum requirements for a solution.

Command	Equivalent Menu Path	Definition
/PREP7*	**Main Menu> Preprocessor***	Enters the general preprocessor
ET*	**Main Menu> Preprocessor> Element Type> Add/Edit/Delete***	Defines the element types
MP*	**Main Menu> Preprocessor> Material Props***	Defines the material properties
MPTEMP	**Main Menu> Preprocessor> Material Props> Material Models**	Defines a table of material temperatures
MPDATA	**Main Menu> Preprocessor> Material Props> Material Models**	Defines a table of material properties
TB	**Main Menu> Preprocessor> Material Props> Material Models**	Defines nonlinear material data, some element data, or both
R	**Main Menu> Preprocessor> Real Constants> Add/Edit/Delete**	Defines element real constants
LOCAL	**Utility Menu> WorkPlane> Local Coordinate Systems> Create Local CS> At Specified Loc**	Defines a local coordinate system
N*	**Main Menu> Preprocessor> Modeling> Create> Nodes> In Active CS** or **Main Menu> Preprocessor> Modeling> Create> Nodes> On Working Plane**	Defines a nodal location
E or **EN***	**Main Menu> Preprocessor> Modeling> Create> Elements> Auto Numbered> Thru Nodes** or **Main Menu> Preprocessor> Modeling> Create> Elements> User Numbered> Thru Nodes**	Defines an element. (You must set the correct TYPE, MAT, REAL, and ESYS pointers before defining the element.)
ANTYPE	**Main Menu> Preprocessor> Loads> Analysis Type> New Analysis**	Defines the analysis type. See the *Commands Reference* for details.
M	**Main Menu> Preprocessor> Loads> Master DOFs> User Selected> Define**	Defines master degrees of freedom
ACEL	**Main Menu> Preprocessor> Loads> Define Loads> Apply> Structural> Inertia> Gravity**	Defines the acceleration vector
D*	**Main Menu> Preprocessor> Loads> Define Loads> Apply>** ***constraint***	Defines degree of freedom constraints
F	**Main Menu> Preprocessor> Loads> Define Loads> Apply>** ***force type*****> On** ***entity***	Defines nodal forces

Command	Equivalent Menu Path	Definition
SF or **SFE**	**Main Menu> Preprocessor> Loads> Define Loads> Apply>** ***surface load type*** **> On** ***entity***	Defines surface loads on element faces
BF or **BFE**	**Main Menu> Preprocessor> Loads> Define Loads> Apply>** ***load type*** **> On** ***entity***	Defines body forces on elements

3.1.1. Tips for Reading Files with CDREAD

The following list describes practices to follow when reading files with **CDREAD** or its menu path equivalent:

- If you place the command **/NOPR** at the beginning of the file and **/GOPR** at the end of the file, you can suppress printing of the file and speed up the time to read it.
- Use the **/COM** command to add comments to the file. If the file contains the **/NOPR** command, **/COM** provides status information about what is being read in. For example:

```
/COM, READING NODES
N,1,.....
 ,
 ,

/COM, READING ELEMENTS
EN,1,.....
```

- Group elements with the same attributes (TYPE, REAL, MAT, ESYS). *Do not* use a separate **MAT**, **REAL**, **TYPE**, and **ESYS** command for each element.
- Use **Utility Menu> File> Change Title** or the **/TITLE** command to add a title to graphics displays and printouts.
- Use **Utility Menu> File> Read Input From** or the **/INPUT** command at the Begin level to input the file.

3.2. 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 **/FILNAM** 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 *Commands 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.

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

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

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

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

The format for the **/DFLAB** command is:

```
/DFLAB,NDOF,LabD,LabF
```

NDOF
ANSYS degree of freedom number (1 to 32)

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

LabF
Force label to be assigned (up to four characters)

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

DOF 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
26	CURR	VLTG

3.3. Coded Database File Commands

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

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

3.3.1. 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*.

3.3.2. CP Command

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

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

LENGTH
The total number of nodes in the coupled set

NCP
The coupled node reference number

Dlab
The degree of freedom label for the set

N1,*N2*,*N3*,*N4*,*N5*,*N6*,*N7*
The next seven node numbers in the coupled set

3.3.3. 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.

3.3.4. EBLOCK Command

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

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

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

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

Format
Data descriptors defining the format. This must be 19i7.

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

- Field 1 - The material number.
- Field 2 - The element type number.
- Field 3 - The real constant number.
- Field 4 - The section ID attribute (beam section) number.
- Field 5 - The element coordinate system number.
- Field 6 - The birth/death flag.
- Field 7 - The solid model reference number.
- Field 8 - The element shape flag.
- Field 9 - The number of nodes defining this element if *Solkey* = SOLID; otherwise, Field 9 = 0.
- Field 10 - The exclude key (p-elements).
- Field 11 - The element number.
- Fields 12-18 - The node numbers. The next line will have the additional node numbers if there are more than eight.

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

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

3.3.5. 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

ADPTH
Absolute shell thickness level below which adaptivity should begin.

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

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

3.3.6. EDCGEN Command

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

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

Option
The label identifying the contact behavior.

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

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

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

FS
The static friction coefficient.

FD
The dynamic friction coefficient.

DC
The exponential decay coefficient.

VC
The coefficient of viscous friction.

VDC
The viscous damping coefficient in percent of critical damping.

V1,V2,V3,V4
Additional input for some contact types. See **EDCGEN** in the *Commands 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

BOXID2
Target volume as defined using EDBOX

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

3.3.7. EDCURVE Command

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

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

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

LCID
The curve ID.

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

Par1
The abcissa values, repeat *Length* number of times.

Par2
The ordinate values, repeat *Length* number of times.

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

3.3.8. EDDRELAX Command

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

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

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

NRCYCK
The number of iterations between the convergence checks.

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

EDTTL
The convergence tolerance when automatic control is used.

DRTOL
The convergence tolerance.

DFFCTR
The dynamic relaxation factor.

DRTERM
The termination time for dynamic relaxation.

TSSFDR
The scale factor for each computed time step.

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

3.3.9. EDLCS Command

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

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

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

CID
The coordinate system ID.

X1,*Y1*,*Z1*,
The X,Y,Z coordinate of a point on the local X-axis.

X2,*Y2*,*Z2*,
The X,Y,Z coordinate of a point on the local X-Y plane.

X3,*Y3*,*Z3*,
The X,Y,Z coordinate of the local origin.

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

3.3.10. EDLOAD Command

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

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

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

Lab
The load labels.

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

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

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

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

Par1
The time values, with the number of values in the string defined by the *Length* argument (above).

Par2
The load values, with the number of values in the string defined by the *Length* argument (above).

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.

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

3.3.11. EDPREAD Command

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

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

Nmat
The number of materials.

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

Type
The element type number.

Mat
The material number.

Real
The real constant set number.

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

For Release 8.0 and beyond, the command format is:

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

Nmat
The number of materials.

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

PartID
The part number.

Type
The element type number.

Mat
The material number.

Real
The real constant set number.

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

3.3.12. EDWELD Command

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

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

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

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

NWELD
The spotweld ID number.

N1
The node number of the first node connected by the spotweld.

N2
The node number of the second node connected by the spotweld.

SN
The normal force at the spotweld failure.

SS
The shear force at the spotweld failure.

EXPN
The exponent for spotweld normal force.

EXPS
The exponent for spotweld shear force.

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

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

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

NWELD
The generalized weld ID number.

CNAME
The name of the node component.

SN
The normal force at the weld failure.

SS
The shear force at the weld failure.

EXPN
The exponent for weld normal force.

EXPS
The exponent for weld shear force.

EXPF
The effective plastic strain at ductile failure.

TFAIL
The time of failure of the weld.

NSW
The number of spotwelds for the generalized weld.

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

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

3.3.13. EN Command

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

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

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

NUMN
The number of nodes.

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

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

3.3.14. LOCAL Command

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

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

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.

3.3.15. 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

3.3.16. MPDATA Command

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

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

LENGTH
The total number of temperatures in the table.

Lab
The material property label. See the **MP** command description in *Commands Reference* for valid labels.

MAT
The material reference number.

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

VAL1,*VAL2*,*VAL3*
Property values assigned to three locations in the table starting at *STLOC*.

3.3.17. MPTEMP Command

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

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

LENGTH
The total number of temperatures in the table

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

TEMP1,*TEMP2*,*TEMP3*
Temperatures assigned to three locations in the table starting at *STLOC*

3.3.18. N Command

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

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

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.

3.3.19. NBLOCK Command

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

```
NBLOCK, NUMFIELD, Solkey
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.

Format
Data descriptors defining the format. This must be (3i8,6e16.9).

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.
- Field 3 - The line location (if the node exists on a line).
- Field 4 - 6 - The nodal coordinates.
- Field 7 - 9 - The rotation angles.

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
(i8,6e16,9)
       1   6.21299982    0.625999987  -1.019883480E-07
       2   6.14472103    0.625999987       0.156284466
       3   6.21271753    0.625999987   1.096193120E-02
.
.
.
  151464    0    0  5.85640764   -0.442010075   1.911501959E-02
  151465    0    0  5.88715029   -0.442010075   7.201258256E-08
  151466    0    0  5.85541868   -0.442010075   7.201258256E-08
   N, R5.3,LOC, -1
```

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

3.3.20. R Command

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

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

NSET
The real constant set reference number.

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

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

VAL1,*VAL2*,*VAL3*
Real constant values assigned to three locations in the table starting at *STLOC*.

3.3.21. RLBLOCK Command

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

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

NUMSETS
The number of real constant sets defined

MAXSET
Maximum real constant set number

MAXITEMS
Maximum number of reals in any one set

NPERLINE
Number of reals defined on a line

Format1
Data descriptor defining the format of the first line. For the **RLBLOCK** command, this is always (2i8,6g16.9). The first i8 is the set number, the second i8 is the number of values in this set, followed by up to 6 real constant values.

Format2
Data descriptors defining the format of the subsequent lines (as needed); this is always (7g16.9).

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

3.3.22. SECBLOCK Command

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

```
SECBLOCK
Format1
```

```
Format2
Format3
```

Format1
The First Line section. The first value is the number of nodes, and the second is the number of cells.

Format2
The Cells Section. The first 9 values are the cell connectivity nodes. The 10th (last) value is the material ID (**MAT**).

Format3
The Nodes Section. This section contains as many lines as there are nodes. In this example, there are 27 nodes, so a total of 27 lines would appear in this section. Each node line contains the node's boundary flag, the Y coordinate of the node, and the Z coordinate of the node. Currently, all node boundary flags appear as 0s in a cell mesh file. Because all node boundary flags are 0, **SECBLOCK** ignores them when it reads a cell mesh file.

Sample User Section Cell Mesh File

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

First Line:
```
27   4
```

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
...
```

3.3.23. SFBEAM Command

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

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

ELEM
The element number.

LKEY
The load key associated with these surface loads.

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

DIOFFST
Offset distance from node I.

DJOFFST
Offset distance from node J.

3.3.24. SFE Command

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

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

ELEM
The element number.

LKEY
The load key associated with this surface load.

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

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

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

Chapter 4: ANSYS Graphics File Format

4.1. Modifying ANSYS Graphics Files

Some ANSYS users may wish to translate ANSYS graphics files to other formats (such as Encapsulated PostScript or AI). If you plan to translate graphics files, this chapter provides some information to help you:

- Source code, with comments, for a program called PLOT33 which plots all of the plots on the coded plot file. You can link this program with user-supplied Calcomp HCBS software (plot, plots, symbol) or other software that uses the Calcomp subroutine protocol. For example, Hewlett-Packard's ISPP product, Tektronix's Preview product, and Veratec's Versaplot product use the Calcomp protocol.

  To work with other plotter software, you may need to remove the calls to the Calcomp subroutine (plot, plots, and symbol) and substitute the subroutine calls for the plotter software being used.

  Note

  Because ANSYS customers use a wide variety of plotters, ANSYS, Inc. can not verify the use of the PLOT33 program with all plotter types.

- A description of the format for the ANSYS neutral graphics file. This listing excludes format information for z-buffered graphics, but the PLOT33 program *does* include a section on z-buffered graphics.

4.2. Pixmap Format for Graphic Display Files

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

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

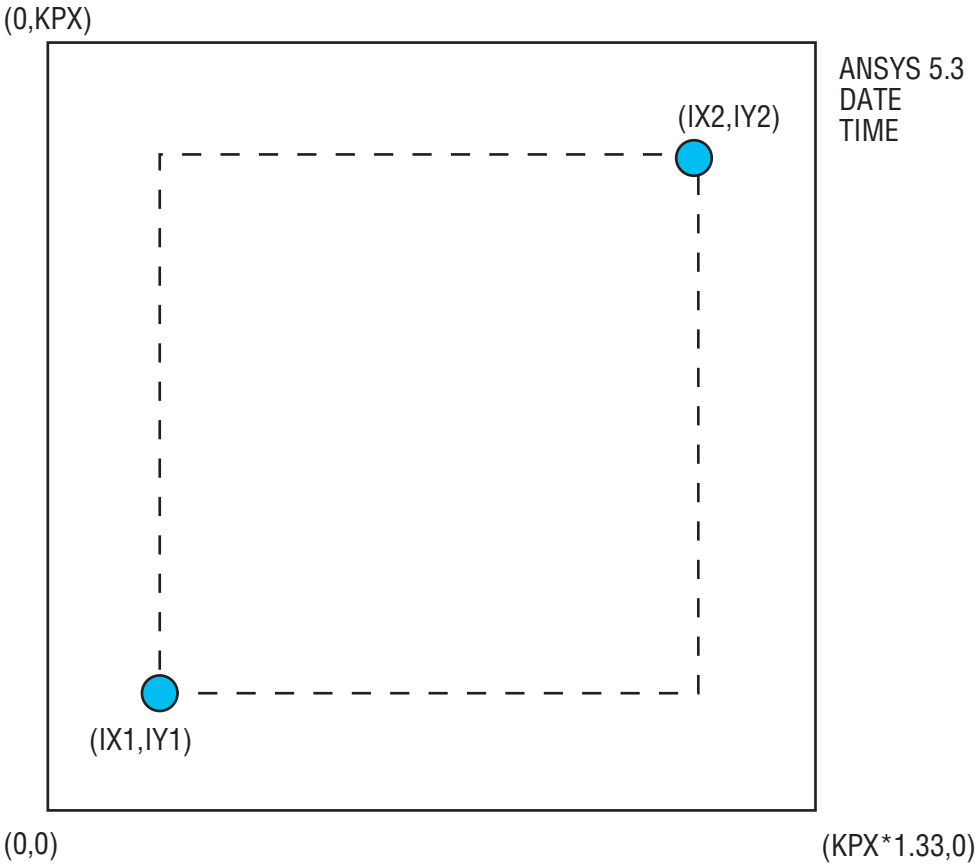
IX1,IY1 is the lower left corner of the z-buffer image.

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

The image should be mapped to the hardcopy device accordingly.

The following graphic illustrates the items described above:

Figure 4.1 Display Format for Z-buffered Graphics



4.3. 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.

4.3.1. 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, " ".

4.3.2. 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).
`>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

4.3.2.1. 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

Parameter Type	Parameter Attributes	Valid Parameter Values
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.

4.3.2.2. 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) 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	Int, Int, Int, Int, Int
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
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	Xy, Long, String

Graphics Directive	Command Character	Parameters	Parameter Types
		string - Defines the text string itself	
Normal	/	inorm - This value, divided by 255, is cos(), where is the viewing direction and the surface normal of subsequent polygons	Long
Polygon	+	n - Defines the number of polygon vertices kedge - Defines whether the polygon edge is displayed (0, = do not display edge, 1 = display it) xy1,...xyn - Defines coordinates for the polygon	Int, Int, Xy...Xy
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, ...

4.3.2.3. 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

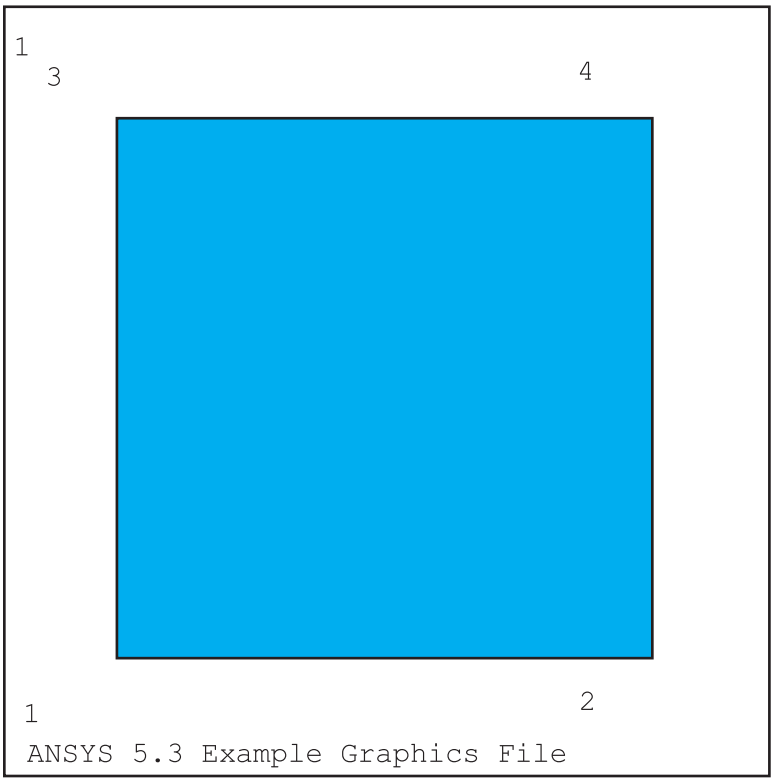
Value	Color
10	Blue
11	Green-Cyan
12	Yellow
13	Dark Gray
14	Light Gray
15	White
16	
:	Reserved for future use
127	
128	Blue
:	:
	Cyan
	:
	Green
	:
	Indices 128 through 255 represent the color spectrum used to display the Low (Blue) to High (Red) contour values.
	:
	Yellow
	:
	Orange
:	:
255	Red

4.4. 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"* below
- The encoded graphics file that these commands produce
- The decoded graphics plot directives

Figure 4.2 Example Display of a Graphics File



4.4.1. The Example Command Stream

To create the graphics display shown in *Figure 4.2, "Example Display of a Graphics File"*, 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
```

4.4.2. Example Graphics File Contents

The commands listed above produce the display shown in *Figure 4.2, "Example Display of a Graphics File"* 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
$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"
`$ABOABOCA ANSYS 5.3` ` Example Graphics File`	Text: 30 30 "ANSYS 5.3 Example Graphics File"
`)`	End-Plot
	No-Op

Part II, Guide to ANSYS User Programmable Features

Chapter 5: Using User Programmable Features (UPFs)

Because the ANSYS program has an open architecture, you can write your own routines or subroutines in C or FORTRAN and either link them to ANSYS or use them as external commands. In fact, some of the ANSYS features you see today as "standard" offerings originated as *user programmable features (UPFs)*. You can take advantage of UPFs if you are licensed for any of the following products:

- ANSYS Multiphysics
- ANSYS Mechanical
- ANSYS Structural
- ANSYS PrepPost
- ANSYS Academic Associate
- ANSYS Academic Research
- ANSYS Academic Teaching Advanced
- ANSYS Academic Teaching Mechanical

Other versions of the ANSYS program do not support UPFs.

5.1. What Are UPFs?

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

UPFs provide the following capabilities:

- To read information into or fetch information from the ANSYS database, you can create subroutines and either link them into the ANSYS program or use them in the external command feature (see *Appendix A:, Creating External Commands in UNIX* for more information about external commands). If you link these subroutines into ANSYS, you are limited to 10 database access commands. Such commands, created through either method, operate at all levels of ANSYS operation, including the begin, preprocessor, general postprocessor, time-history postprocessor, and solution levels. For more information about accessing the ANSYS database, see *Chapter 7: Accessing the ANSYS Database*.
- ANSYS provides a set of routines you can use to specify various types of loads, including BF or BFE loads, pressures, convections, heat fluxes, and charge densities. These routines are described under *Section 6.5: Routines for Customizing Loads*.
- Another set of UPF routines enables you to define the following material properties: plasticity, creep, swelling law, viscoplasticity, hyperelasticity, and layered element failure criteria. To see inputs and outputs for these routines, see *Section 6.4: Routines for Customizing Material Behavior*.
- Several sets of UPFs enable you to define new elements and to adjust the nodal orientation matrix. See *Section 6.1: Creating a New Element* for more information.
- Another group of UPFs enables you to modify and monitor existing elements. For details, see *Section 6.3: Routines for Modifying and Monitoring Existing Elements*.

- You can customize UPF `userop` to create a custom design optimization routine. For more information, see *Section 6.10: Creating Your Own Optimization Routine*.
- You can call the ANSYS program as a subroutine in a program you have written. To learn how, see *Section 6.6: Running ANSYS as a Subroutine*.

5.2. What You Should Know Before Using UPFs

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

The UPF subroutines are written in FORTRAN 90; 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 word "user" in the first line of the routine. Other routines and functions described in this document that do not have "user" in the first line cannot be modified and must be used as is.

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

- The ANSYS program.
- The UPF subroutines themselves. Study the UPF subroutines before customizing them, and make sure that you fully understand the subroutines, as well as any applicable functions. Unless you review them carefully, a few UPF subroutines may seem like a maze with many logic paths to consider. You may have to set special variables correctly in order to run your customized ANSYS program without errors. Even if you have in-depth knowledge of the ANSYS input and your desired outputs, you still need to ensure that everything that needs to be done in the UPF subroutines is done properly in your custom version.
- FORTRAN 90. Besides knowing how to write FORTRAN 90 subroutines, you must be sure that the level of the FORTRAN 90 compiler is as least as high as the level mentioned in your ANSYS installation manual. 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. You should never use the **/CONFIG** command or a `config.ans` file to activate parallelization on a system with UPFs. For additional information, consult your ANSYS installation manual or your on-site ANSYS system support person
- Carefully consider whether you wish to use UPFs, especially if you are linking them into ANSYS (rather than into a shared library for use as external commands). When you add your own routines to ANSYS by either method, you are creating a customized, site-dependent version of the program. ANSYS, Inc. considers the use of UPFs a nonstandard use of the program, one that the ANSYS Quality Assurance verification testing program does not cover. Therefore, you are responsible for verifying that the results produced are accurate and that your customizations do not adversely affect unchanged areas of the ANSYS program.
- Although the flexibility that UPFs offer can be highly attractive, UPF usage is a complicated process that can introduce errors. Consider what you want your customizations to accomplish. You may be able to customize ANSYS more easily and safely with macros than with UPFs.

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

5.3. Planning Your UPFs

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

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

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

5.4. Studying the ANSYS User Routines

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

- If you have a UNIX version of ANSYS, the source code for the UPF routines resides in directory `/ansys_inc/v110/ansys/customize/user/<platform>`.
- If you are running the ANSYS program under Windows, the UPF source code resides in directory `Program Files\Ansys Inc\V110\ansys\custom\user\<platform>`.

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.

5.5. Programming in Languages Other than FORTRAN

If you access UPFs by compiling and linking a custom version of ANSYS, the preferred method is to design and program your custom routine in FORTRAN 90. Although you can use languages other than FORTRAN 90, in each case FORTRAN 90 must provide the interface to the rest of the ANSYS program. If you do use a language other than FORTRAN 90, such as the C programming language, your code may require a FORTRAN shell.

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

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

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

5.6. Developing UPFs: a Suggested Strategy

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

The example procedure below illustrates this type of gradual process. The example assumes that you are creating a new element for the ANSYS program using the method described in *Section 6.1.2: Creating a New Element by Directly Accessing the ANSYS Database*. You develop and test it by performing these steps:

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

5.7. Include Decks

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

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

Include File	Description
`acelcm.inc`	Contains accelerations and angular velocities
`ansys-def.inc`	Defines general ANSYS parameters. You must include this common to retrieve the parameter values of *`MEM_INTEGER`*, *`MEM_DOUBLE`*, *`MEM_COMPLEX`*, or *`MEM_REAL`*.
`cmopt.inc`	Contains optimization variables
`echprm.inc`	Defines parameters for element characteristics
`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 ANSYS program
`outpcm.inc`	Defines output control information
`soptcm.inc`	Contains solution options and keys

Include File	Description
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

5.8. Linking User Routines

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

- Link your routines into shared libraries (as discussed starting in *Appendix A:, Creating External Commands in UNIX*).
- Compile and link your custom routines into the ANSYS program itself. This is discussed for UNIX systems in *Section 5.9: Compiling and Linking UPFs on UNIX Systems* and for Windows systems in *Section 5.10: Compiling and Linking UPFs on Windows Systems*. You may need superuser or root privileges to run the procedure that does the linking.

5.9. Compiling and Linking UPFs on UNIX Systems

As mentioned previously, the source files for the user routines reside in subdirectory /ansys_inc/v110/ansys/customize/user/<*platform*>. If you modify any of these subroutines, select the **Relink ANSYS** option from **ANS_ADMIN** utility to link these changes.

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

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

The **Relink ANSYS** option compiles all FORTRAN files (files ending with .F) and all C files (files ending with .c) in the current working directory. The procedure then loads all object files (files ending with .o) along with the default ANSYS objects and libraries in /ansys_inc/v110/ansys/customize/user/<*platform*> (where *platform* is a directory that uniquely identifies the hardware platform version). The new executable file created will be named ansyscust.e110 and will reside in the current directory.

FORTRAN files are assumed to be FORTRAN 90 (some extensions are allowed), and C files are assumed to be ANSI C.

When relinking on UNIX systems, you can choose to link a distributed version of ANSYS as well. If you choose to link a distributed version, the executable (ansyscustdis.e110) 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. You cannot link a distributed version on Windows systems.

The Installation and Configuration Guide lists the compilers you will need to use UPFs.

Creating a Shared Library You can also set up UPFs on some UNIX and Linux systems through a shared library as an alternative to creating a custom ANSYS executable. Copy the ANSUSERSHARED script from /ansys_inc/v110/ansys/customize/user/<*platform*> into your working directory. All Fortran (*.F) and C (*.c) files that you want to include in your shared library should also reside in your working directory. To compile all *.F and *.c routines, issue the following command:

```
sh ./ANSUSERSHARED
```

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

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

```
setenv ANS_USER_PATH workingdirectory
```

or

```
export ANS_USER_PATH=workingdirectory
```

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

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

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

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

Note

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

5.10. Compiling and Linking UPFs on Windows Systems

As mentioned previously, the source files for the user routines reside in subdirectory `Program Files\Ansys Inc\V110\ansys\custom\user\<platform>` (where <*platform*> is a directory that uniquely identifies the hardware platform version.)

Note

If you intend to modify any of the user routines, make a duplicate copy of the `Program Files\Ansys Inc\V110\ansys\custom\user\<platform>` directory to preserve the original files for later use, if necessary.

If you modify any of these subroutines, select the **Relink ANSYS** option from the **ANS_ADMIN** utility to link these changes into the ANSYS program. This procedure compiles all FORTRAN files (files ending with `.F`) and all C files (files ending with `.c`) in the `Program Files\Ansys Inc\V110\ansys\custom\user\<platform>` directory. The procedure then loads all object files (files ending with `.obj`), along with the default ANSYS objects and libraries. The executable file created will be named `ansys.exe` and will reside in `Program Files\Ansys Inc\V110\ansys\custom\user\<platform>`.

Note

ANS_ADMIN does not exist on Windows 64-bit systems. In order to relink on Itanium systems, you must open a command prompt window, set your current directory to `Program Files\Ansys Inc\V110\ansys\custom\user\win64`, and run ANSCUST.

When relinking on a Windows system, you can choose to link with the small export list ("Do you want to link with the small export list?").

- Answer "No" to link ANSYS with all symbols exported (which may be required by third party interfaces). The link time will be significantly longer. The ANSYS executable provided on the media is linked using this option, and ANSYS recommends that you select this option.
- Answer "Yes" to link ANSYS with the minimum amount of symbols needed to be exported for ANSYS to run. Select this option for the fastest link time.

Caution

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

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

1. To execute the relinked version of the ANSYS program:
   - Click **Start>Programs>ANSYS 11.0> ANSYS 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:

     ```
     ansys110 -custom <path> -p <product variable> -g
     ```

   - Batch:

     ```
     ansys110 -custom <path> -b -p <product variable>-j jobname -i <input file> -o <output file>
     ```

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

Note

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

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

Note

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

5.11. Activating UPFs

The ANSYS program activates many UPFs through a specific user action. This can be through a command option or a user selection. For example, to activate a user element created using the method described in *Section 6.1.2: Creating a New Element by Directly Accessing the ANSYS Database*, all you need to do is select it as one of the element types in a model (using the **ET** command). You then set the element attribute pointer (**TYPE** command), and define elements using the solid modeling or direct generation method.

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 *Commands Reference*.

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

5.12. Running Your Custom Executable

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

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

When run from the command prompt, if no path is specified after the `-custom` argument, the `ansys110` script searches the current working directory for the custom ANSYS executable (`ansyscust.e110` by default on UNIX or `ansys.exe` on Windows). If the custom ANSYS executable resides in a separate directory (or has a name other than `ansyscust.e110` on UNIX), 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 ANSYS executable, the executable must be named `ansys.exe`. This requirement is due to shared library usage.

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

```
ansys110 -custom /pathname/ansyscust.e110
```

5.13. Verifying Your Routines

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

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

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

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

Note

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

5.14. Debugging Commands

To debug errors in your user routines, you can use commands and other features not documented in the *Commands Reference*. Use these commands only for extremely small models with few solution iterations (otherwise, they will generate an excessive amount of output). **/TRACK** and **/DEBUG** are described in detail below. Two other useful commands are **OUTEQ** and **/NERR**. The command **OUTEQ**,ON can be used to output results from all equilibrium iterations. The command **/NERR**,,,-1 causes errors to be reported as before, but the run continues anyway, normally terminating with either a) system abort or b) incorrect answers. The **/NERR**,,,-1 command is intended for program debugging and may generate erroneous results. You should remove this statement before generating solutions for production use.

5.14.1. Tracking the Path of Program Logic

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

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

MonLevel
The level for timing monitoring.

PrintLevel
The level for enter/exit printout.

SumLevel
The level at which the timing sum is output.

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

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

5.14.2. Debugging Elements and Solutions

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

5.14.2.1. Solution Debug Format

Issue the command using this format:

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

F1

1 (provides basic solution control debugging)

F2

1 (provides transient debugging using Newmark constants)

2 (provides transient debugging using velocities and accelerations)

F3

1 (provides element matrix debugging and prints matrix + load vectors, before going into solve)

2 (provides element matrix debugging with load vectors only, before going into solve)

3 (provides element matrix debugging with matrix diagonals and load vectors, before going into solve)

F4

1 (provides auto time stepping debugging)

F5

1 (provides multifield debugging)

F6

1 (provides arc-length debugging)

F7

1 (provides basic Newton-Raphson debugging)

2 (provides Newton-Raphson debugging and prints out-of-balance forces or incremental displacement or each DOF)

3 (provides Newton-Raphson debugging and prints applied loads and n-r restoring force for each DOF)

F8

1,2 (provides displacement vector debugging with displacement pointers)

2 (provides displacement vector debugging with incremental displacement)

3 (provides displacement vector debugging with contact database)

F9

1 (provides temporary programmer debugging)

5.14.2.2. Element Debug Format

Issue the command using this format:

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

G1

1 (provides basic element pass debugging)

G2

1 (provides element displacement and coordinate debugging)

G3

1 (provides element matrix debugging and prints matrix + load vectors, after the element routines)

2 (provides element matrix debugging with load vectors only, after the element routines)

3 (provides element matrix debugging with matrix diagonals and load vectors, after the element routines)

G4
1 (provides element load information debugging)

G5
1 (provides element real constant debugging)

G6
1 (provides element saved variable debugging)

G7
1 (provides element material property debugging with linear material properties)

2 (provides element material property debugging with nonlinear properties)

G8
1,2 (provides element nonlinear debugging with plasticity)

2 (provides element nonlinear debugging with large deformation)

3 (provides element nonlinear debugging with contact database)

G9
1 (provides temporary programmer debugging)

5.14.2.3. General Debug Format

Issue the command using this format:

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

H1
1 (provides file header record information)

2 (provides input line (character))

3 (provides input line (decoded))

H2
1 (provides wavefront 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)

5.15. Other Useful Commands

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

Issue the **/UCMD** command to make a user routine into a custom command. For more information, see *Section 6.7: Defining Your Own Commands*.

5.16. Generating Output

You can generate output controlled by the **/OUTPUT** command by using the FORTRAN write statement. The output unit for this statement is usually called IOTT. IOTT may be defined with the function `wrinqr`. See the discussion on the function `wrinqr` in *Chapter 8: Subroutines for Users' Convenience* for more details.

5.17. Reading Large Data Files More Rapidly

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

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

Chapter 6: UPF Routines and Functions

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

User routines that can be modified have the word "user" in the first line of the routine. For example, the first line of the `userop` routine looks like this:

```
*deck,userop                                                    user
```

User routines that do not have "user" in the first line cannot be modified and must be used as is.

6.1. Creating a New Element

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

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

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

This table highlights the differences between the two methods:

Interface	User-defined element API	Accessing ANSYS database and files directly
Description	Offers a simpler interface while preserving much of the underlying user-element capability. An understanding of the database routines and the file structure is rarely necessary to use the interface.	No special interface. If an element capability exists for an ANSYS element, it will exist here (with a few exceptions). The logic necessary for using this interface effectively is more complex.
Relative level of difficulty	Medium	High
Expected compatibility between versions	High	Medium
Element names	USER300	USER100 to USER105
Demonstration logic included on the ANSYS distribution media	4-node quad and 20-node brick elements	MASS21 and LINK8 elements
Typical linear material access routine	`getMatProp`	`propev`
New nonlinear material properties	Program in `usermat`.	No special programming has been set up.
Existing nonlinear material properties	All ANSYS standard structural materials are accessible via `ElemGetMat`.	Limited capability. Accessible via `plastx`, `creepx`, and `swellx`.
Non-structural material properties	No special programming has been implemented.	No special programming has been implemented.
Number of different element types allowed	Effectively, no limit.	Effectively, no limit.

Interface	User-defined element API	Accessing ANSYS database and files directly
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 routines to be programmed	`UserElem`	`uec100`, `uel100`, and rarely `uex100` and `uep100`. Routines `uec101` to `uec105` are analogous.
Access to database information	Generally through the interface.	Through routines (such as `tmpget`, `prsget`, `svgidx`, `svrget`, `svpidx`, `svrput`).
Access to file information	Through the interface.	Through pointers and routines (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.

Neither method is supported by Distributed ANSYS.

6.1.1. 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.

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 DOFs.	Issue the **USRDOF** command. You can specify a maximum of 10 DOFs per **USRDOF** command; to define additional DOFs, issue the command again. Each node will have the same DOFs. Although you can specify any valid DOFs, the total number of DOFs for your element cannot exceed 480, and the number of DOFs for each node cannot exceed 32.
4.	Define real constants.	If needed.
5.	Create finite element models.	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 create an element with a topology different from that of any standard ANSYS element.

		• *Meshing commands* -- This method is available only if your element has the same topology as that of a standard ANSYS element *and* you have specified any standard element shape (**USRELEM** *KeyShape* value) except ANYSHAPE.
6.	Apply boundary conditions and loads.	As needed.
7.	Specify solution options.	If your element has multi-field DOFs (displacements and temperatures), disable default solution settings (**SOLCONTROL**,OFF).
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.

Note

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

Recommendations and Restrictions

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

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

6.1.1.1. Subroutine UserElem (Writing Your Own Elements)

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

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

Argument	Input (I) *or* Output (O)	Definition	Purpose	How Defined
`elId`	I	Element number	Output information	At FE model creation

matId	I	Material number	Output information Call material routines	At FE model creation
keyMtx	I	Formulation request	Specifying which matrices and load vectors to form	ANSYS code
lumpm	I	Mass matrix format: 0 = Consistent mass matrix 1 = Lumped mass matrix	Specifying how to form the mass matrix	**LUMPM** 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 DOFs	Element coding -- The DOFs are ordered in the way in which they are listed via the **USRDOF** command for each node and repeated for all nodes All element matrices -- DOF values and load vectors must be arranged in the same way	**USRELEM** and **USRDOF** commands
kEStress	I	Element stress state	Element coding Calling material routines if requested	**USRELEM** command
keyAnsMat	I	Element formulation key: 0 -- Write your own material formulation 1 -- Use standard ANSYS material routines and call `ElemGetMat` routine	Specifying how to create material data	**USRELEM** command
keySym	I	Flag for symmetricity of element matrices	Element coding ANSYS assembly logic	**USRELEM** command
nKeyOpt	I	Maximum number of element key options allowed *Example:* If **nKeyOpt** = 2, only KEYOPT(1) and KEYOPT(2) are allowed.	Element coding	ANSYS 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 routine)
temperB	I	Nodal temperatures at the end of the last substep	Temperature dependence and thermal loads	ANSYS code
tRef	I	Reference temperature	Temperature dependence and thermal loads	**TREF** command
kTherm	I	Key indicating whether a thermal load exists: 1 = Calculate the thermal load 0 = No thermal load calculation	Element coding	---
nPress	I	Number of pressure values	Element coding The size of the press vector	**USRELEM** command At FE model creation
Press	I	Pressures at nodes of element facets (available only when **keyShape** is specified via the **USRELEM** command) The pressure vector is ordered in the element with the same topology as that in the standard element library. Refer to that element for details.	Pressure load and pressure load stiffness	**SF** and **SFE** commands
kPress	I	Key indicating whether a pressure load exists: 1 = Calculate the pressure load 0 = No pressure load calculation	Element coding	ANSYS code
nReal	I	Number of real constants	Element coding	**USRELEM** command
RealConst	I	The list of real constants	Element coding 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 The size of **saveVars**	**USRELEM** command
saveVars	I/O	The data saved in the .esav file ANSYS saves the data after exiting the UserElem routine and retrieves it immediately before entering UserElem again. It should include kinematic related variables only when the ANSYS material routine is called; otherwise, it should include both kinematic and	Element coding	UserElem routine

		material data. History dependent variables can only be saved/updated when the substep is converged (**keyHisUpd** = 1).		
xRef	I	Initial coordinates of the element nodes Values in global Cartesian coordinates	Element coding	At FE model creation
xCur	I	Current (deformed) coordinates of element nodes Values in global Cartesian coordinate system, equal to **xRef** when **nlgeom** = off	Element coding	ANSYS code
TotValDofs	I	Total values of DOFs (displacements for structural analysis) Values in global Cartesian coordinate system	Element coding	ANSYS code
IncValDofs	I	Increment values of DOFs occurring at the current substeps Values in global Cartesian coordinate system	Element coding	ANSYS code
ItrValDofs	I	Iteration values of DOFs occurring at the last iteration Values in global Cartesian coordinate system	Element coding	ANSYS code
VelValDofs	I	First time derivatives of DOFs	Velocities	ANSYS code
AccValDofs	I	Second time derivatives of DOFs	Accelerations	ANSYS code
kfstps	I	Key indicating first time entering the element routine: 1 = First time 0 = Other than first time	Initializing data	ANSYS code
nlgeom	I	Flag indicating whether large displacement/deformation is in effect	Element coding	**NLGEOM** command
nrkey	I	Newton-Raphson algorithm key: 1 -- Any nonlinear analysis 0 -- Pure linear analysis	Output Element coding	---
outkey	I	Key indicating output result type: 1 -- This is an output call, the substep is converged, and you can print/save element results 0 -- All other cases	Element coding	ANSYS code
elPrint	I	Key indicating whether any element output should appear in the print file: 0 = No 1 = Yes	Element coding	ANSYS code **OUTPR** command
iott	I	Output file number	The FORTRAN output file number. All information written in the specified	ANSYS code

			file appears in the ANSYS 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	ANSYS 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	ANSYS code
isubst	I	Current substep number	Output	ANSYS code
ieqitr	I	Current iteration number	Output	ANSYS code
timval	I	Current time	Output	ANSYS code
keyEleErr	I/O	Formulation error key: 0 = No error (preset value) 1 = Error occurred in element formulation, possibly due to excessive deformation. (ANSYS will lessen deformation if possible by cutback.)	Element coding	ANSYS 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 constraint for mixed u-P)	ANSYS code
End of special-purpose variable group				
eStiff	I	Small-deformation stiffness matrix In global Cartesian coordinate system	Solution	Requested when **keyMtx(1)** = 1
eMass	I	Mass matrix In global Cartesian coordinate system	Solution	Requested when **keyMtx(2)** = 1
eDamp	I	Damping matrix In global Cartesian coordinate system	Solution	Requested when **keyMtx(3)** = 1
eSStiff	I	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 routine
elMass	O	Element mass	Output	UserElem routine
elCG	O	Element centroid coordinates In global Cartesian coordinate system	Postprocessing	UserElem routine
nRsltBsc	I	Number of basic result data saved in result file	Specifying the size of **RsltBsc**	ANSYS code
RsltBsc	O	Basic result data saved in ANSYS result file This variable is accessible via the **PRESOL** and **PRNSOL** commands in the standard way and can also be plotted if you specify a *Key-Shape* value via the **USRELEM** command.	Postprocessing	UserElem routine
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 routine
nElEng	I	Number of energies Fixed at 3	Solution	UserElem routine
elEnergy	O	Element energy vector: **elEnergy(1)** -- Strain energy **elEnergy(2)** -- Plastic energy **elEnergy(3)** -- Creep energy	Output	UserElem routine

```
*deck,UserElem                              user
      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 *** Notice - This file contains ANSYS Confidential information ***
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
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

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
```

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

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

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

Argument	Input (I) *or* Output (O)	Definition	Purpose	How Defined
elId	I	Element number	Output	At FE model creation
matId	I	Material number	Output information Getting material data	At FE model creation
nDim	I	Number of dimensions of element geometry 2 = 2-D element geometry 3 = 3-D element geometry	Material calculation	At FE model creation

		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 routine
nDirect	I	Number of direct component of stress/strain tensors **nDirect** < or = **nTens**	Specifying the data size	UserElem routine
intPnt	I	Current integration point number	Output Data handling	UserElem routine
xCurIP	I	Coordinates of current integration point Values in global Cartesian coordinate system	Material calculation	UserElem routine
TemperIP	I	Integration point temperatures at the current time	Evaluating temperature-dependent material data	UserElem routine
TemperIPB	I	Integration point temperatures at the end of the last incremental step	Evaluating temperature-dependent material data	UserElem routine
IncStrain	I	Strain components [*1*] Incremental strain of the current substep when **nlgeom** = on Total strain when **nlgeom** = off	Material calculation	UserElem routine
defG0	I	Deformation gradient tensor at the end of previous substep [*1*]	Material updating	UserElem routine
defG	I/O	Total deformation gradient tensor at the current time [*1*]	The component in thickness direction is updated by material routines for plane stress and shell elements	UserElem routine
kTherm	I/O	Thermal loading key: 0 = No thermal loading 1 = Has thermal loading	Thermal load calculation	UserElem routine
cMat	O	Material Jacobian [*1*]	Forming stiffness	Material routine
MatProp	O	Material data for element formulation	Forming mass matrix Handling transverse shear Output	Material routine
Stress	O	Cauchy stress [*1*]	Forming geometric stiffness Calculating internal forces	Material routine
Strain	O	Total strain components [*1*]	Output	Material routine
StressTh	O	Total thermal stress components [*1*]	Output Calculating thermal loading	Material routine
StrainTh	O	Total thermal strain components [*1*]	Output	Material routine

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

1. All tensor component values in the routine 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
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
*************************************************************************
c
```

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

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

6.1.2.1. Input and Output Abbreviations

The descriptions of the 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

6.1.2.2. User Routines

Routines uec100 through uec105 describe the element characteristics. Routine elccmt (on the distribution medium) describes the input for these routines in detail. You can use subroutines uex100 through uex105 to override default logic. Routines 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.

Routines 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 routines available for manipulating element information include the following:

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

6.1.2.3. Subroutine uec100 (Defining Characteristics of the usr100 Routine)

```
*deck,uec100                              user
      subroutine uec100 (elcdn,ielc,kerr)
c     ***** this subroutine defines the characteristics of user100.
c
c         *** copyright(c) 2006 SAS IP, Inc.  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
```

6.1.2.3.1. Subroutines uec101 through uec105

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

6.1.2.4. Subroutine uex100 (Overriding Element Characteristic Defaults)

```
*deck,uex100                            user
      subroutine uex100 (ielc,kerr)
c     *** subroutine to override element characteristic defaults ***
c     *** hence, this routine is needed only in rare cases.
c
c         *** copyright(c) 2006 SAS IP, Inc.  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
c     *** the debug accessed with /debug,,, ,,1
c
```

6.1.2.4.1. Subroutines uex101 through uex105

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

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

```
*deck,uel100                            user
      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(c) 2006 SAS IP, Inc.  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
```

6.1.2.5.1. Subroutines uel101 through uel105

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

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

```
*deck,uep100                              user
      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(c) 2006 SAS IP, Inc.  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
```

6.1.2.6.1. Subroutines uep101 through uep105

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

6.1.2.7. Subroutine usertr (Adjusting the Nodal Orientation Matrix)

```
*deck,usertr                              user
      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(c) 2006 SAS IP, Inc.  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
```

6.1.2.8. Subroutine userac (Accessing Element Information)

This subroutine is provided for demonstration purposes.

```
*deck,userac                              user
      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(c) 2006 SAS IP, Inc.  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
```

6.2. Supporting Subroutines for Element Creation

The subroutines described on the next few pages support the user routines used to create new elements (using the database-access method described in *Section 6.1.2: Creating a New Element by Directly Accessing the ANSYS Database*).

6.2.1. Subroutine nminfo (Returning Element Reference Names)

```
*deck,nminfo
      subroutine nminfo (ielc,rname)
c *** primary function:    set element reference names
c *** secondary functions: none
c
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
```

6.2.2. 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
```

6.2.3. 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
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
```

6.2.4. 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
```

6.2.5. 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
```

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

```
*deck,mreuse
      subroutine mreuse (kelrqq,kelfil,elem,ielc,kmasrt,knlmg,kconve,
     x kpheno,kprop,nprop,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
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
```

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

```
*deck,subrd
      subroutine subrd (iel,key,nd,vect,ka)
c *** primary function:    read element load data from file for substructure
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                                   =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
```

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

```
*deck,subwrt
      subroutine subwrt (iel,nvect,key,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     key      (int,sc,in)       - type of load vect
c                                  = 1 temperature
c                                  = 2 fluences
c                                  = 3 heat generation rates
c                                  = 4 current densities
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
```

6.2.9. 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
```

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

```
*deck,propev
      subroutine propev (iel,mtr,lp,tem,prop,n)
c *** primary function:   to evaluate a group of material properties
```

```
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)

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, BM  =58, QRAT=59, REFT=60, CTEX=61, CTEY=62, CTEZ=63, THSX=64,
c        THSY=65, THSZ=66, DMPR=67, LSSM=68,     =69,     =79,     =71,     =72,
c            =73,     =74,     =75,     =76,     =77,     =78,     =79,     =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
```

6.2.11. 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, BM  =58, QRAT=59, REFT=60, CTEX=61, CTEY=62, CTEZ=63, THSX=64,
c        THSY=65, THSZ=66, DMPR=67, LSSM=68,     =69,     =79,     =71,     =72,
c            =73,     =74,     =75,     =76,     =77,     =78,     =79,     =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
```

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

```
*deck,pstev1
      subroutine pstev1 (elem,matin,tem,prop)
c *** primary function:    to evaluate material properites for 1-d elements
c
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
```

6.2.13. Subroutine tbuser (Retrieving User Table Data)

```
*deck,tbuser                              user
      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
```

6.2.14. 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
```

```
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
```

6.2.15. 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
```

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

```
*deck,creep1
      subroutine creep1 (option,elem,intpt,mat,kstartL,epel,e,epcrp,
     x statev,usvr,tem,dtem,fluen,dflu,sig)
c *** primary function:    to update the creep history for 1-d elements
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)
```

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

```
*deck,creep3
      subroutine creep3 (option,elem,intpt,mat,kstartL,ncomp,epel,e,
     x posn,d,epcrp,statev,usvr,tem,dtem,fluen,dflu,kplst,sig,hsig)
c *** primary function:    to update the creep history for 3-d elements
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
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)
```

6.2.18. 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
```

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

```
*deck,swell3
      subroutine swell3 (option,elem,intpt,mat,kstartL,ncomp,epswel,
     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
```

6.2.20. 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**********************************************************************
```

6.2.21. Function nlget (Retrieving Material Nonlinear Property Information)

```
*deck,nlget
      function nlget (mat,iprop,prop)
c *** primary function:    get a material non-linear property (TB) table.

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

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
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
```

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

```
*deck,usereo                              user
      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
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
```

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

```
*deck,eldwrtL
      subroutine eldwrtL (ielem,edtype,lcerstL,edindxL,nval,value)
c *** primary function:    output element data to result file.

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

c     input arguments:
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(25),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)
```

6.2.24. 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(25),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)
```

6.2.25. Subroutine trrot (Computing the Rotation Vector)

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

6.2.26. Subroutine rottr (Computing the Transformation Matrix)

```
*deck,rottr
      subroutine rottr (rot,tr)
c primary function: compute transformation matrix from rotation vector *****
c *** Notice - This file contains ANSYS Confidential information ***
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
```

6.2.27. 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
```

6.2.28. Subroutine updrot (Updating the Rotation Pseudovector)

```
*deck,updrot
      subroutine updrot (v2,w1)
c primary function: update the rotation pseudovector for 3-d large rotations *****
c
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
```

6.2.29. 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
```

6.2.30. 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
```

6.2.31. 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
```

6.2.32. 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
```

6.2.33. Subroutine prinst (Computing 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
```

6.3. Routines for Modifying and Monitoring Existing Elements

The next few pages describe the user routines you use to modify or monitor existing ANSYS elements. These routines enable you to perform tasks including:

- Computing load vectors for frequency domain logic

- Storing element output that users supply
- Modifying the orientation of material properties and stresses
- Modifying the orientation of material properties and stresses of layers within an element
- Performing a user-defined operation on a parameter for the COMBIN7 and COMBIN37 elements
- Providing a user-defined initial thickness for SHELL181
- Providing a user-defined initial stress for PLANE2, PLANE42, SOLID45, PLANE82, SOLID92, SOLID95, LINK180, SHELL181, PLANE182, PLANE183, SOLID185, SOLID186, SOLID187, SOLSH190, BEAM188, BEAM189, SHELL208, and SHELL209
- Providing a user-defined fictive temperature relationship for VISCO88 and VISCO89
- Providing viscoelastic computation in the stiffness pass for VISCO88 and VISCO89
- Modifying SURF151 and SURF152 film coefficients and bulk temperatures based on information from FLUID116

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

```
*deck,userfd                                user
      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(c) 2006 SAS IP, Inc.  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
```

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

```
*deck,userou                              user
      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(c) 2006 SAS IP, Inc.  All rights reserved.
c         *** ansys, inc.
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
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    udbdat  (dp,ar(maxdat),out)   - user output items to be placed at the end
c                                    of the nmisc record
c
```

6.3.3. Subroutine useran (Modifying Orientation of Material Properties)

```
*deck,useran                                user
      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(c) 2006 SAS IP, Inc.  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
```

6.3.4. Subroutine usanly (Modifying Orientation of Material Properties and Stresses of Layers)

```
*deck,usanly                                user
      subroutine usanly  (vn,vref,elem,thick,xyzctr,ln,bsangl)
c
c     user written routine to modify orientation of material properties
c     and stresses of layers within an element **************************
c      applicable to shell91,99, and solid46,191
c       accessed with keyopt(4)
c
c         *** copyright(c) 2006 SAS IP, Inc.  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  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    = thickness of layer ln at this point (see note below)
c     xyzctr   = location of element centroid or integration point
c     ln       = layer number
c
c  output---
c     bsangl = output from this subroutine.  it represents the angle(s)
c              between vref and the desired orientation of the layer.
c              it may have the default orientation coming in to useran.
c
c              this angle does not affect the angle given in the printout.
c              users may want to add their own angle printout in usanly.
c
c              also, this angle does not affect the angle shown using the
c              layplot or laylist commands
c
c           use 1 angle for 2-d elements and shells
```

```
c           use 3 angles for 3-d solids
c
c     the programmer of usanly can base their logic on the location
c     of the point in space, or an element attributes, such as thick
c
```

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

```
*deck,userrc                              user
      subroutine userrc (elem,ireal,type,nusvr,usvr,parm,parmld,
     x c1,c2,c3,c4,fcon)
c     primary function: user operation on parameter for combin7 and combin37
c      accessed with keyopt(9) = 1
c
c         *** copyright(c) 2006 SAS IP, Inc.  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    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
```

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

```
*deck,userpe                              user
      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(c) 2006 SAS IP, Inc.  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
```

6.3.7. Subroutine UElMatx (Accessing Element Matrices and Load Vectors)

```
*deck,UElMatx                                 user
      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.
c                      May be used to monitor and/or modify the element matrices
c                                                      and load vectors.

c         *** copyright(c) 2006 SAS IP, Inc.  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,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     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.
```

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

```
*deck,uthick                                  user
      SUBROUTINE uthick (elemId, elemType, matId, realId,
     $                   numDomIntPts, curCoords, thickness)
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c     *** primary function: get the user defined thickness
c
c         *** copyright(c) 2006 SAS IP, Inc.  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
```

6.3.9. Subroutine USTRESS (Getting User-defined Initial Stress)

```
*deck,ustress                              user
      SUBROUTINE ustress (elemId, elemType, matId, basis,
     $                    curIntPt, curLayer, curSecPt,
     $                    numDirect, numShear, curCoords,
     $                    tLocation, iniData)
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c     *** primary function: get the user supplied stress state for
c                           initial stress analysis
c
c         *** copyright(c) 2006 SAS IP, Inc.  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 (42, 181 etc.)
c      matId          (int,sc,i)     material number
c      basis          (dp,ar(3,3),i) basis directions (for the element)
c      curIntPt       (int,sc,i)     current Integration point
c      curLayer       (int,sc,i)     current Layer number (for shell
c                                           and layered solids)
c      curSecPt       (int,sc,i)     current point through the thickness in the
c                                           current layer
c      numDirect      (int,sc,i)     number of direct stress comp
c                                           (s_xx,s_yy,s_zz)
c      numShear       (int,sc,i)     number of shear stress comp
c                                           (s_xy,s_yz,s_zx)
c      curCoords      (dp,ar(3),i)   current coordinates
c      tLocation      (dp,sc,i)      normalized thickness location
c                                          (shell elements only)
c                                      -1 = bottom or first sec pt
c                                      +1 = top or last sec pt
c                                       0 = mid surface
c
c     output arguments
c     ================
c      iniData        (dp,ar(numDirect+numShear),o)
c                                  user input stress values
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c --- parameters
c
```

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

```
*deck,UsrFictive                             user
      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(c) 2006 SAS IP, Inc.  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
```

6.3.11. Subroutine UsrViscEl (Performs Viscoelastic Computation)

```
*deck,UsrViscEl                              user
      subroutine UsrViscEl (veinpt,ncomp,tem,dtem,ex,gxy,eex,egxy,phil,
     x   zil,g1l,g2l,hsm,hbm,smcm,bmcm,epsl,epst,ftl,dftl,tref,tvc,dvc,
     x   dsig,rsig,cm,kerr,cml)
c    ****** perform visco-elastic computation - stiffness pass
c
c ***** notice- this routine contains ansys, inc. confidential information ****
c
c        *** copyright(c) 2006 SAS IP, Inc.  All rights reserved.
c        *** ansys, inc.
c
c  input arguments:
c       veinpt (dp,ar(95),in)          - viscoelastic input data
c       ncomp  (int,sc,in)             - number of components (4 or 6)
c       tem    (dp,sc,in)              - temperature
c       dtem   (dp,sc,in)              - incremental temperature
c       ex     (dp,sc,in)              - bulk modulus at infinite time
c       gxy    (dp,sc,in)              - shear modulus at infinite time
c       smcm   (dp,ar(ncomp,ncomp),in) - material matrix for shear modulus
c       bmcm   (dp,ar(ncomp,ncomp),in) - material matrix for bulk modulus
c       epsl   (dp,ar(ncomp),in)       - strain for previous iteration
c       epst   (dp,ar(ncomp),in)       - total strain for current iteration
c       tref   (dp,sc,in)              - reference temperature
c
c  output arguments:
c       eex    (dp,sc,out)             - effective bulk modulus
c       egxy   (dp,sc,out)             - effective shear modulus
c       phil   (dp,sc,inout)           - previous shift factor
c       zil    (dp,sc,inout)           - previous pseudo time
c       g1l    (dp,ar(ncomp,10),out)   - recursive shear relaxation
c       g2l    (dp,ar(ncomp,10),out)   - recursive bulk relaxation
c       hsm    (dp,ar(10),out)         - recursive shear relaxation
c       hbm    (dp,ar(10),out)         - recursive bulk relaxation
c       ftl    (dp,sc,inout)           - previous fictive temperature
c       dftl   (dp,ar(10),out)         - incremental fictive temperature
c       tvc    (dp,sc,inout)           - total volume change
c       dvc    (dp,sc,out)             - incremental volume change
c       dsig   (dp,ar(ncomp),out)      - stress change
c       rsig   (dp,ar(ncomp),out)      - stress relaxation
c       cm     (dp,ar(ncomp,ncomp),out)- total material matrix
c       kerr   (int,sc,out)            - error key
c
c  argument of convenience:
c       cml    (dp,ar(ncomp,ncomp),none)- no value (used only to avoid simplify
c                                         logic due to variable array sizes)
c
c
```

6.3.12. Subroutine usrsurf116 (Modifying SURF151 and SURF152 Film Coefficients and Bulk Temperatures)

```
*deck,usrsurf116                              user
      subroutine usrsurf116 (elem,ielc,center,jdim,kaxis,time,nr,u,
     x                omeg,ndat,         temvel,hc,tb,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
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(c) 2006 SAS IP, Inc.  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.
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),inout) - bulk temperature
c                                      (has input values for each corner
c                                      of element)
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        tb     (dp,ar(ndat),inout) -bulk temperature(includes any modification)
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) = 0 = no new bulk temperature
c                                      key(2) = 1 = define new bulk temperature
c                                       (if key(2) = 1, the adiabatic wall
c                                                temperature logic is not used).
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.
```

6.3.13. Subroutine User116Cond (Computes the conductance coefficient for FLUID116)

```
*deck,User116Cond                               user
      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(c) 2006 SAS IP, Inc.  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
```

6.3.14. Subroutine User116Hf (Computes the film coefficient for FLUID116)

```
*deck,User116Hf                                 user
      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(c) 2006 SAS IP, Inc.  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
```

6.3.15. Subroutine Us_Surf_Str (Captures surface stresses)

```
*deck,Us_Surf_Str                              user
      subroutine Us_Surf_Str (elem,face,area,temp,pressure,ep,stress)
c *** primary function:    User routine to capture surface stresses
c                          called by PLANE2, 42, 82 and SOLID45, 92, 95

c *** Notice - This file contains ANSYS Confidential information ***
c         *** copyright(c) 2006 SAS IP, Inc.  All rights reserved.
c         *** ansys, inc.

c  input arguments:
c     variable (typ,siz,intent)    description
c    elem    (int,sc,in)       element number
c    face    (int,sc,in)       face number
c    area    (dp, sc,in)       face area (or length)
c    temp    (dp, sc,in)       face temperature
c    pressure(dp, sc,in)       face pressure
c    ep      (dp,ar(4),in)     face strains
c    stress  (dp,ar(11),in)    face stresses

c  output arguments:  none
```

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

```
*deck,usflex                                user
      subroutine usflex (etype,elem,rvrm,kff,prs,ex,  flexi,flexo)
c *** primary function:   to (re)compute the flexibility factor
c                           for pipe16, pipe17, pipe18, and pipe60
c                         this is accessed by inputting the flexibility factor
c                           as any negative number.
c *** secondary functions: none
c
c *** Notice - This file contains ANSYS Confidential information ***
c
c         *** copyright(c) 2006 SAS IP, Inc.  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     etype    (int,sc,in)       - pipe element type (16, 17, 18 or 60)
c     elem     (int,sc,in)       - element number
c     rvrm     (dp,ar(*),in)     - real constants
c     kff      (int,sc,in)       - keyopt for flexibility factor
c                                    (not used for pipe16 or pipe17)
c     prs      (dp,ar(5),in)     - pressures
c     ex       (dp,sc,in)        - young's Modulus
c     flexi    (dp,sc,inout)     - effective in-plane flexibility factor
c     flexo    (dp,sc,inout)     - effective out-of-plane flexibility factor
c                                    (not used for pipe16 or pipe17)
c
c  output arguments:
c     variable (typ,siz,intent)    description
c     flexi    (dp,sc,inout)     - effective in-plane flexibility factor
c     flexo    (dp,sc,inout)     - effective out-of-plane flexibility factor
c                                    (not used for pipe16 or pipe17)
c
```

6.3.17. Subroutine UsrShift (Calculates pseudotime time increment)

```
*deck,                              user
      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**********************************************************************
```

6.4. Routines for Customizing Material Behavior

This section describes the user routines you use to modify or monitor how materials behave. These routines enable you to perform tasks including:

- Writing your own material constitutive models
- Writing your own plasticity, creep, or swelling laws
- Writing your own hyperelasticity laws
- Updating the nonlinear strain history for a user-defined material
- Checking material data you have defined
- Computing the derivatives of the strain energy density "W" with respect to its invariants
- Controlling hygrothermal growth.

Note

If you write a material behavior routine using any of the ANSYS commands **MPDATA**, **MPDELE**, **TB**, or **TBDELE**, be aware that these commands interpret the string "_MATL" to mean the current active material when it appears in their *MAT* field. The "_MATL" is used in conjunction with the library (LIB) option of the **MPREAD** and **MPWRITE** commands. **MPWRITE** inserts "_MATL" in lieu of the specified material number as these commands are written to the material library file. This occurs only when you use the LIB option. When ANSYS reads a material library file written in this format, it interprets "_MATL" to mean the currently active material (as defined by the **MAT**,*MAT* command). Do not use the "_MATL" string outside the scope of the **MPREAD** command.

6.4.1. Subroutine usermat (Writing Your Own Material Models)

This subroutine is applicable to the following elements: LINK180, SHELL181, PLANE182, PLANE183, SOLID185, SOLID186, SOLID187, SOLSH190, BEAM188, BEAM189, SHELL208, and SHELL209.

```
*deck,usermat                       user
      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,
     &                  rotateM, 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, plain strain
c          and axisymmetry, plane stress and 3D/1D beam.
c
c          a 3D material constitutive model can use for
c          plain 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      rotateM  (dp,ar(3,3),i)           Rotation matrix for finite deformation update
c                                        Used only in 5.6 and 5.7
c                                        Unit matrix in 6.0 and late version
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)
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*************************************************************************
```

6.4.2. Subroutine userpl (Writing Your Own Plasticity Laws)

This subroutine is applicable to the following elements: LINK1, PLANE2, LINK8, PIPE20, BEAM23, BEAM24, PLANE42, SHELL43, SOLID45, SHELL51, PIPE60, SOLID62, SOLID65, PLANE82, SHELL91, SOLID92, SHELL93, SOLID95.

```
*deck,userpl                              user
      subroutine userpl (elem,intpt,mat,ncomp,kfirst,kfsteq,e,nu,dens,
     x prop,d,ktform,timval,timinc,tem,dtem,toffst,flu,dflu,epel,eppl,
     x statev,usvr,epeq,plwork,sigepl,sigrat,depeq,dt)
c
c *** primary function:   allow users to write their own plasticity laws.
c                           this logic is accessed with tb,user.
c                           the below demonstration logic is the same as using
c                           tb,bkin, without adaptive descent (nropt,,,off).
c                           Other plasticity rules may require internal
c                           iterations and/or the more general definition of
c                           plasticity theory, discussed in the Theory
c                           Manual.
c *** secondary function:    demonstrate the use of user-written plasticity laws
c                  in this routine:
c                     a.    update the nonlinear strain history
c                     b.    compute the material tangent matrix if requested
c
c *** Notice - This file contains ANSYS Confidential information ***
c
c
c         *** ansys(r) copyright(c) 2004
c         *** ansys, inc.
c
c  input arguments:
c     variable (type,sze,intent)    description
c
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     ncomp    (int,sc,in)         - no. of stress/strain components (1,4 or 6)
c                                    1 - x
c                                    4 - x,y,z,xy
c                                    6 - x,y,z,xy,yz,xz
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
c     e        (dp,sc,in)          - average elastic modulus
c     nu       (dp,sc,in)          - average poisson ratio
c     dens     (dp,sc,in)          - current material density (mass/volume)
c     prop                         - linear material property array
c              (dp,ar(9),in)         (ex,ey,ez, gxy,gyz,gxz, nuxy,nuyz,nuxz)
c              (dp,ar(1),in)         if ncomp=1 (ex)
c     d        (dp,ar(ncomp,ncomp),in)- elastic stress-strain matrix
c     ktform   (int,sc,in)         - request key for tangent matrix formation
c                                    (=1, form tangent  .ne.1, do not form)
c
c     timval   (dp,sc,in)          - current time value
c     timinc   (dp,sc,in)          - time increment over this substep
c
c     tem      (dp,sc,in)          - temperature at the end of this substep
c     dtem     (dp,sc,in)          - temperature increment over this substep
c     toffst   (dp,sc,in)          - temperature offset from absolute zero
c     flu      (dp,sc,in)          - fluence at the end of this substep
c     dflu     (dp,sc,in)          - fluence increment over this substep
c
c     epel     (dp,ar(ncomp),inout)- modified total strain (trial strain)
c                                    epel = eptot - eppl - eptherm - ...
c                                    if a large strain analysis, epel is
c                                    rotation neutralized and is the hencky
c                                    (i.e. log) strain
c     eppl     (dp,ar(ncomp),inout)- plastic strain from previous substep
```

```
c
c     statev   (dp,ar(ncomp,6),inout)- state variables from previous substep
c     usvr     (dp,ar(nuval,nintp),inout)- additional state variables from
c                                    previous equilibrium iteration (saved
c                                    if the nsvr command is used)
c
c     epeq     (dp,sc,inout)       - effective plastic strain from prev substep
c     plwork   (dp,sc,inout)       - accumulated plastic work from prev substep
c
c  output arguments:
c     variable (type,sze,intent)    description
c
c     epel     (dp,ar(ncomp),inout)- elastic strain
c     eppl     (dp,ar(ncomp),inout)- updated plastic strain
c
c     statev   (dp,ar(ncomp,6),inout)- updated state variables
c     usvr     (dp,ar(nuval,nintp),inout)- updated additional state variables
c
c     epeq     (dp,sc,inout)       - updated effective plastic strain
c     plwork   (dp,sc,inout)       - updated accumulated plastic work
c
c     sigepl   (dp,sc,out)         - stress value on stress-strain curve at epeq
c     sigrat   (dp,sc,out)         - ratio of trial stress to yield stress
c     depeq    (dp,sc,out)         - increment in plastic strain (equivalent)
c                                    (used for auto time stepping - time step
c                                    is reduced if it exceeds .05)
c
c     dt       (dp,ar(ncomp,ncomp),out)- material tangent modulus
c
c  fortran parameters (to be defined by the user):
c     variable (type)              description
c     numinp   (int)               - number of data items in the user-defined
c                                    data table (tbdat commands)
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     b        (dp,ar(6,6))    - 2nd derivative of the yield function
c     c        (dp,ar(6,12))   - part of deff
c     con      (dp,sc)         - temporary variable
c     deppl    (dp,ar(6))      - plastic strain increment
c     dfds     (dp,ar(6))      - derivative of the yield function (normal)
c     dlamb    (dp,sc)         - plastic multiplier
c     ep       (dp,ar(6))      - shifted strain
c     epshfo   (dp,ar(6))      - initial shift strain (center of the yield surf)
c     epshft   (dp,ar(6))      - shift strain (center of the yield surface)
c     et       (dp,sc)         - tangent modulus (stress/total strain)
c     h        (dp,sc)         - plastic tangent modulus (stress/plastic strain)
c     n2       (int,sc)        - ncomp squared, ncomp*ncomp
c     seqtr    (dp,sc)         - equivalent (von mises) trial stress
c     sigtr    (dp,ar(6))      - trial stress
c     sigy     (dp,sc)         - yield stress
c     vect     (dp,ar(6))      - temporary vector
c
```

6.4.3. Subroutines usercreep and usercr (Defining Viscoplastic/Creep Material Behavior)

ANSYS provides two subroutines to allow you to specify your own creep equations to define the time-dependent viscoplastic/creep behavior of materials. The two subroutines are `usercreep` and `usercr`. Use the subroutine `usercreep` when you issue the **TB** command with the CREEP option, and with *TBOPT* = 100. Use the subroutine `usercr` when you issue the **TB** command with the CREEP option, with *TBOPT* = 0, and data constant C6 = 100. Subroutine `usercreep` is incorporated with an implicit time integration algorithm, while subroutine `usercr` is incorporated with an explicit time integration algorithm. In general, the implicit time integration algorithm is more effective for long time periods. The explicit creep algorithm is more effective for short time periods such

as transient analyses with very small time increments, or when the creep behavior of materials is not that significant. In the finite deformation analysis, you should interpret the strain variables in the subroutines as logarithmic strains, and you should interpret the stresses as true stresses.

6.4.3.1. Creep Subroutine usercreep

Use the subroutine `usercreep` to define viscoplastic/creep behavior of materials when you issue the **TB** command with the CREEP option, and with *TBOPT* = 100. The subroutine will be called at all integration points of elements for which the material is defined by this command. ANSYS always uses implicit time integration for this creep option. You can use plasticity options (BISO, MISO, NLISO) to define the plastic behavior of materials. Creep and plastic strain will be calculated simultaneously when both creep and plasticity are defined for a material. Through this subroutine, you can specify a "uniaxial" creep law that will be generalized to the multi-axial state by the general time-dependent viscoplastic material formulation implemented in ANSYS. You can use and update internal state variables in the subroutine. The number of state variables has to be defined by **TB**,STATE.

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

```
*deck,usercreep                                 user
      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
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
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
```

6.4.3.2. Creep Subroutine usercr

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

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

6.4.4. Subroutine usersw (Writing Your Own Swelling Laws)

```
*deck,usersw                              user
      subroutine usersw (option,elem,intpt,mat,proptb,ncomp,epswel,
     x epel,e,nuxy,fluen,dfluen,tem,dtem,toffst,timvll,timvnc,usvr)
c
c *** primary function:   allow users to write their own swelling laws.
c                           this logic is accessed with c72 = 10
c *** secondary function:    demonstrate the use of user-written swelling laws
```

```
c
c
c         *** copyright(c) 2006 SAS IP, Inc.  All rights reserved.
c         *** ansys, inc.
c *** Notice - This file contains ANSYS Confidential information ***
c
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     proptb   (dp,ar(*),in)     nonlinear material table (tb commands)
c     ncomp    (int,sc,in)       number of strain components (=1, 4, or 6)
c                                   1 - truss or beam elements
c                                   4 - 2-d solids and pipe elements
c                                   6 - 3-d solids and most shells
c     epswel   (dp,sc,inout)     total accumulated swelling strain
c                                     before this substep
c     epel     (dp,ar(ncomp),inout) elastic strain
c     e        (dp,sc,in)        elastic modulus
c     nuxy     (dp,sc,in)        poisson'S RATIO
c     fluen    (dp,sc,in)        total fluence (bf or bfe commands)
c     dfluen   (dp,sc,in)        increment in fluence for this substep
c     tem      (dp,sc,in)        temperature (bf or bfe commands)
c     dtem     (dp,sc,in)        increment in temperature for this substep
c     toffst   (dp,sc,in)        offset of temperature scale from absolute zero
c                                          (toffst command)
c     timvll   (dp,sc,in)        time at the end of this substep
c     timvnc   (dp,sc,in)        the increment of time this substep
c     usvr     (dp,ar(*),inout)  user-defined state variables(optional)
c
c   output arguments:
c     epswel   (dp,sc,inout)     total accumulated swelling strain
c                                     after this substep
c     epel     (dp,ar(ncomp),inout) elastic strain adjusted
c                                            for swelling increment
c     usvr     (dp,ar(*),inout)  updated user-defined state variables
c
```

6.4.5. Subroutine UserHyper (Writing Your Own Hyperelasticity Laws)

```
*deck,UserHyper                                user
      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
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
```

6.4.6. Subroutine uservp (Updating Nonlinear Strain History for Materials)

```
*deck,uservp                              user
      subroutine uservp (elem,mat,kfirst,kfsteq,g,prop,timval,timinc,
     x tem,toffst,delepn,svrn,svri,usvr,epeln,epeli,strn,stri,depdt,
     x cee,eta,norm)
c
c *** primary function:    update the nonlinear strain history for a
c                          user-defined material for the viscoxxx elements
c *** secondary functions: compute the material tangent terms
c                     accessed with tb,user and keyopt(1) = 1
c
c *** Notice - This file contains ANSYS Confidential information ***
c
c
c         *** copyright(c) 2006 SAS IP, Inc.  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     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
c     g        (dp,sc,in)          - shear modulus
c     prop     (dp,ar(13),in)      - linear material property array
c                                    (ex,ey,ez, gxy,gyz,gxz, nuxy,nuyz,nuxz,
c                                     alpx,alpy,alpz, dens)
c
c     timval   (dp,sc,in)          - current time value
c     timinc   (dp,sc,in)          - time increment over this substep
c
c     tem      (dp,sc,in)          - temperature at the end of this substep
c     toffst   (dp,sc,in)          - temperature offset from absolute zero
c
c     delepn   (dp,ar(3,3),in)     - hencky strain increment over the substep
c
c     svrn     (dp,sc,in)          - state variable from previous substep
c     svri     (dp,sc,inout)       - state variable from previous iteration
c     usvr     (dp,ar(*),inout)    - additional state variables from previous
c                                    iteration (saved if the nsvr command is
c                                    used)
c
c     epeln    (dp,ar(3,3),in)     - elastic strain from previous substep
c     epeli    (dp,ar(3,3),inout)  - elastic strain from previous iteration
c
c     strn     (dp,ar(3,3),in)     - stress from previous substep
c     stri     (dp,ar(3,3),inout)  - stress from previous iteration
c
c     depdt    (dp,sc,inout)       - effective inelastic deformation rate
c                                    (d(deppl)/dt) from previous iteration
c
c  output arguments:
c     variable (type,sze,intent)    description
c
c     svri     (dp,sc,inout)       - updated state variable
c     usvr     (dp,ar(*),inout)    - updated additional state variables
c
c     epeli    (dp,ar(3,3),inout)  - updated elastic strain
```

```
c     stri     (dp,ar(3,3),inout)  - updated stress
c
c     depdt    (dp,sc,inout)       - effective inelastic deformation rate
c
c     cee      (dp,sc,out)         - scalar linearization modulus (see below)
c     eta      (dp,sc,out)         - radial return factor (see below)
c     norm   (dp,ar(3,3),out)      - direction of plastic flow
c
c  fortran parameters (to be defined by the author):
c     variable (type)              description
c     numinp   (int)               - number of data items in the user-defined
c                                    data table (tbdat commands)
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                                    (8 is the maximum)
c         note:  nuval x nintp cannot exceed 840!
c
```

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

```
*deck,userck                              user
      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(c) 2006 SAS IP, Inc.  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
```

6.4.8. Subroutine usermc (Controlling Hygrothermal Growth)

```
*deck,usermc                              user
      subroutine usermc (elem,time,msci,dmsci,mscr,mscra)
c *** primary function:     user control of hygrothermal growth
c *** secondary functions:  none
c     presently, called only by shell91
c
c         in order to activate this user programmable feature,
c         the user must enter the usrcal command.
c
c         *** copyright(c) 2006 SAS IP, Inc.  All rights reserved.
c         *** ansys, inc.
c *** Notice - This file contains ANSYS Confidential information ***
c
c  input arguments:
c     elem     (int,sc,in)       - element number
c     time     (dp,sc,in)        - time
c     msci     (dp,sc,in)        - moisture content
c     dmsci    (dp,sc,in)        - change of moisture content
c     mscr     (dp,ar(3),in)     - input material properties for growth
c
c  output arguments:
c     mscra    (dp,ar(3),out)    - output material properties for growth
c
```

6.4.9. Subroutine usrfc6 (Defining Custom Failure Criteria)

```
*deck,usrfc6                              user
      subroutine usrfc6 (elem,matlay,iott,keyer,tem,elim,slim,
     x                   eps, sig, fc6)
c     primary function: user subroutine for defining your own failure criterion
c *** secondary functions: none
c     --- accessed with c6 = -1
c *** user programmable functions may not be used in parallel processing ***
c     this is currently only available with
c
c         *** copyright(c) 2006 SAS IP, Inc.  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                                    (tbdata items 1-9)
c     slim     (dp,ar(12),in)    - failure stresses and coupling coefficients
c                                    at the current temperature
c                                    (tbdata items 10-21)
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     keyer    (int,sc,inout)    - error flag (0 = ok, 1 = error or routine not defined)
c
c  output arguments:
c     variable (typ,siz,intent)    description
c     keyer    (int,sc,inout)    - error flag (0 = ok, 1 = error or routine not defined)
c     fc6      (dp,sc,out)       - failure criterion to be processed by solid46,
c                                    shell91, shell99, or solid191
c
```

6.4.10. Subroutines usrfc1 through usrfc5

The source code for subroutines `usrfc1`, `usrfc2`, `usrfc3`, `usrfc4`, and `usrfc5` is identical to subroutine `usrfc6` shown above.

6.4.11. Subroutine UserVisLaw (Defining Viscosity Laws)

```
*deck,UserVisLaw                              user
       subroutine UserVisLaw
     x (dudx,dudy,dudz,
     x  dvdx,dvdy,dvdz,
     x  dwdx,dwdy,dwdz,
     x  u,v,w,x,y,z,kGeom,
     x  Vis,Temp,Tref,Pres,Pref,Cf,
     x  MFrac,DfNSpec,Time,VisNew,toffst)

C      Primary function: to provide a user defined viscosity
C                        relationship in terms of the following:
C                        pressure, temperature, position, time,
C                        velocity, & velocity-gradient
C                        This routine is for use with the FLOTRAN
C                        elements, Fluid141 and Fluid142 only.
C
C      In order to activate this subroutine the user must issue
C      FLDA,PROT,VISC,USRV command.
C
C      In addition the initial value of viscosity must be specified via
C      FLDA,PROP,IVIS,value. This value is not available in this routine.
C
C      Optionally the user may specify 4  additional coefficients
```

```
C      which are available in this routine by the commands:
C      FLDA,NOMI,VISC,value1
C      FLDA,COF1,VISC,value1
C      FLDA,COF2,VISC,value2
C      FLDA,COF3,VISC,value3
C
c         *** copyright(c) 2006 SAS IP, Inc.  All rights reserved.
c         *** ansys, inc.
C
C      input arguments:
C      variable   (typ,siz,intent)    description
C      dudx       (dp,sc,in)          velocity gradient component
C      dudy       (dp,sc,in)          velocity gradient component
C      dudz       (dp,sc,in)          velocity gradient component
C      dvdx       (dp,sc,in)          velocity gradient component
C      dvdy       (dp,sc,in)          velocity gradient component
C      dvdz       (dp,sc,in)          velocity gradient component
C      dwdx       (dp,sc,in)          velocity gradient component
C      dwdy       (dp,sc,in)          velocity gradient component
C      dwdz       (dp,sc,in)          velocity gradient component
C      u          (dp,sc,in)          velocity component
C      v          (dp,sc,in)          velocity component
C      w          (dp,sc,in)          velocity component
C      x          (dp,sc,in)          position component
C      y          (dp,sc,in)          position component
C      z          (dp,sc,in)          position component
C      kGeom      (int,sc,in)         analysis type
C      Vis        (dp,sc,in)          old viscosity
C      Temp       (dp,sc,in)          absolute temperature
C      Tref       (dp,sc,in)          reference temperature (Absolute also)
C      Pres       (dp,sc,in)          pressure
C      Pref       (dp,sc,in)          reference pressure
C      Cf         (dp,ar(4),in)       input coefficients
C      Mfrac      (dp,ar(6),in)       species mass fractions
C      DfNSpec    (int,sc,in)         defined number of species
C      Time       (dp,sc,in)          time
C      toffst     (dp,sc,in)          Temperature offset for absolute scale.
C      output arguments:
C      variable   (typ,siz,intent)    description
C      VisNew     (dp,sc,out)          new viscosity
C
```

6.4.12. 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
```

6.5. Routines for Customizing Loads

This section describes the user routines you use to modify or monitor existing ANSYS elements. These routines enable you to perform tasks including:

- Setting custom values for scalar fields (temperatures, heat fluxes, etc.)
- Changing element pressure information
- Changing information about element face convection surfaces
- Changing information about element face heat flux surfaces
- Changing information about element face charge density surfaces
- Changing information about element acceleration/angular velocity.

Activate these routines by issuing the **USRCAL** command or by choosing an equivalent menu path.

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

```
*deck,usrefl                              user
      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(c) 2006 SAS IP, Inc.  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
```

6.5.2. Subroutine userpr (Changing Element Pressure Information)

```
*deck,userpr                              user
      subroutine userpr (ielc,elem,time,ndat,dat)
c *** primary function:    change element pressure information.

c         *** copyright(c) 2006 SAS IP, Inc.  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
```

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

```
*deck,usercv                              user
      subroutine usercv (elem,ielc,time,nr,u, ndat,hc,tb)
c *** primary function: change element face convection surface info
c
c         *** copyright(c) 2006 SAS IP, Inc.  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
```

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

```
*deck,userfx                              user
      subroutine userfx (ielc,elem,time,nr,u, ndat,dat)
c *** primary function: change element face heat flux surface info
c
c         *** copyright(c) 2006 SAS IP, Inc.  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
```

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

```
*deck,userch                                user
      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(c) 2006 SAS IP, Inc.  All rights reserved.
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
```

6.6. Running ANSYS as a Subroutine

To call the ANSYS program, use the following:

```
program ansys
```

If you are running ANSYS on a UNIX system (but not under Windows), you also can call the ANSYS program as a subroutine in a program you've written. To do so, use the following:

```
subroutine ansys
```

For multiple calls to subroutine ANSYS, you must open and close standard input in the calling routine. (Usually, input and output are FORTRAN units 5 and 6, respectively.) The calling routine can't use the database access routines. But, other user programmable features can use the database access routines freely.

There may be times when ANSYS exits abnormally. Check the `file.err` file to see if ANSYS wrote an exit code to the file before ending. These error codes may help you to understand the problem ANSYS had:

Table 6.1 ANSYS Exit Codes

Code	Explanation	Code	Explanation
0	Normal Exit	14	XOX Error
1	Stack Error	15	Fatal Error
2	Stack Error	16	Possible Full Disk
3	Stack Error	17	Possible Corrupted or Missing File
4	Stack Error	18	Possible Corrupted DB File
5	Command Line Argument Error	21	Authorized Code Section Entered
6	Accounting File Error	25	Unable to Open X11 Server
7	Auth File Verification Error	30	Quit Signal
8	Error in ANSYS or End-of-run	31	Failure to Get Signal
11	User Routine Error	>32	System-dependent Error
12	Macro STOP Command		

6.7. Defining Your Own Commands

ANSYS, Inc. supplies a set of user routines, named `user01` through `user10`, which you can use to define custom ANSYS commands. To do so, follow these steps:

1. Insert the code for the functions you want to perform into routine `user01` (or `user02`, etc.).
2. Link the routine into the ANSYS program.
3. Issue the ANSYS command **/UCMD** to define a name for a custom command that calls and executes your routine. Use the command format shown below:

```
/UCMD,Cmd,SRNUM
```

Cmd
The name for your new command. It can contain any number of characters, but only the first four are significant. The name you specify can not conflict with the name of any ANSYS command or the names of any other commands or macros.

SRNUM

The number of the routine your command should call; that is, a value between 01 and 10. For example, suppose that you create and link in a user routine for a parabolic distribution of pressure, and you name that routine `user01`. Issuing the command shown below creates a new command, PARB, that when issued calls your parabolic pressure distribution routine:

```
/UCMD,PARB,1
```

To make these "custom command" routines available in all your ANSYS sessions, include the **/UCMD** commands in your start-up file (START . ANS).

You also can use **/UCMD** to remove a custom command. To do so, simply use a blank value for *Cmd*, as shown below:

```
/UCMD,,1
```

This command removes the PARB command. To list all user-defined command names, issue the command **/UCMD**,STAT.

6.7.1. Function user01

```
*deck,user01           user
      function  user01()
c *** primary function:    user routine number  01

c         *** copyright(c) 2006 SAS IP, Inc.  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
c    ************************************************************

      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
```

6.7.2. Function user02 (Demonstrates Offsetting Selected Nodes)

```
*deck,user02           user
      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(c) 2006 SAS IP, Inc.  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    ************************************************************

      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)

#include "ansysdef.inc"

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
```

6.7.3. Function user03 (Demonstrates Using ANSYS Memory)

```
*deck,user03            user
      function  user03()
c *** primary function:    user routine number 03. Gives example of
c                          ANSYS Memory usage

c         *** copyright(c) 2006 SAS IP, Inc.  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"

      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
      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
```

6.7.4. Function user04

```
*deck,user04           user
      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(c) 2006 SAS IP, Inc.  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     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    ************************************************************


      external  wrinqr, ndkpnt, ndline, ndarea, ch4infun, intinfun
      integer   wrinqr, ndkpnt, ndline, ndarea, intinfun
      character*4  ch4infun

      integer  user04, iott, listk(20),listl(20),lista(20),
     x          i,  num,ktype,     nkpnts,   nlines,   nareas
      character*4  type, lab2
```

```
#include "ansysdef.inc"

      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)

      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 = ndkpnt (num,  listk(1))
         write (iott,2110) nkpnts
 2110    format (' number of nodes on keypoint = ',i4)
         write (iott,2115) (listk(i),i=1,nkpnts)
 2115    format (' node on keypoint = ',i4)

      elseif (lab2 .eq. 'LN  ' ) then
         nlines = ndline (num,ktype,listl(1))
         write (iott,2120) nlines
 2120    format (' number of nodes on line = ',i4)
         write (iott,2125) (listl(i),i=1,nlines)
 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
```

6.7.5. Functions user05 through user10

The source code for user routines `user05`, `user06`, `user07`, `user08`, `user09`, and `user10` is identical to function `user01` shown above.

6.8. Supporting Subroutines

6.8.1. Function GetRForce (Getting Nodal Reaction Force values)

```
*deck,GetRForce
      function GetRForce (Node,Label,Value)
c primary function:    Get the K * u - F at a node from the rfsum vector.
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:
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)
```

6.8.2. Function GetStackDisp (Getting Current Displacement Values)

```
*deck,GetStackDisp
      function GetStackDisp (Node,Label,Value)

c primary function:    Get the displacement at a node from the disp vector
c secondary functions: Return pointer for fast access

c object/library:  usr
```

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

6.8.3. Subroutine ElResultStrt (Getting Load Data from Analysis Results)

```
*deck,ElResultStrt
      subroutine ElResultStrt (Label,Comp,LabAvg,TypeData,nVal,iLoc)
c *** primary function:    (post1) Load data for later ElResultGet

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

6.8.4. 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
```

6.8.5. 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)
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
```

6.9. Access at the Beginning and End of Various Operations

You can access the logic just before an ANSYS run begins or just after a run ends, and at many other intermediate points, by using the ANSYS routines listed below. These routines can perform actions such as evaluating results or performing calculations. (None of the routines have input or output arguments.)

Issue the **USRCAL** command (or use an equivalent menu path) to activate or deactivate these routines.

User Subroutine	Is Called
UAnBeg	At ANSYS start-up
USolBeg	Before solution
ULdBeg	Before a load step
USsBeg	Before a substep
UItBeg	Before an iteration
UItFin	After an iteration
USsFin	After a substep
ULdFin	After a load step
USolFin	After solution
UAnFin	At the end of an ANSYS run

Subroutines USSBeg, UItBeg, UItFin and USSFin default to reading a command macro file from the current working directory whose name is subroutine.mac (that is, ussfin.mac is read by USSFin.F). No user action to relink the ANSYS program is required for the command macro to be read except that the calling sub-

routine must be activated by the **USRCAL** command. The design of the command reading ability of these subroutines is limited to APDL parameter setting commands (***GET**, ***SET**, a = value, etc) and testing for general ANSYS commands is limited. Commands which are known to work include ***DIM**, ***STATUS**. Commands which require another line (***MSG**, ***VWRITE**) are not allowed. Other commands which are known to not work are the solution loading commands (**D**, **F**, **SFE**, and so on). If these capabilties are required, the user will need to create a Fortran subroutine and link this subroutine into ANSYS, as described in *Chapter 5: Using User Programmable Features (UPFs)*.

While parameter substitution into commands is not permitted, USSBeg, and so on were designed to be used in conjunction with dynamic tables and parameter substitution from the user subroutine. As an example, consider an ANSYS table defined as d5 = f(par1), If d5 contains values of displacement as a function of PAR1, then d5 may be used as a constraint, as

```
*dim,d5,table,10,1,1,PAR1
d5(1)=0,.1,.25,

/solu
d,5,ux,%d5%
```

Modify the value of PAR1 in USSBeg.MAC and the constraint on node 5, ux can then be modified in the middle of a load step.

The following is an example of a valid input that may be read by USSBeg, UItBeg, UItFin and USSFin.

```
/COM, SAMPLE ussfin.mac
a=5
b=nx(1)                         ! *get function is ok
*get,c,active,solu,Time,cpu     ! *get is ok
*dim,array,,6                   ! array parameters are ok
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)
```

6.10. Creating Your Own Optimization Routine

One way to customize the ANSYS program for your site's specific needs is to substitute your own external optimization for the standard ANSYS optimization algorithm. You can do so using either of these methods:

- Link a user routine within the ANSYS program.
- Run a stand-alone optimization program using the ANSYS optimization "save" file, Jobname.OPT.

This chapter describes both methods. You can find additional information on design optimization techniques and procedures in the *Advanced Analysis Techniques Guide*.

6.10.1. Linking a Custom Optimization Program to ANSYS

If you are performing iterative analyses for design optimization, and you wish to use the standard ANSYS optimizer, you simply choose GUI path **Main Menu>Design Opt** or issue the ANSYS command **/OPT**.

However, if you plan to use an optimization routine of your own design, you must do the following:

1. Define parameters for your external optimization routine, using either the **OPUSER** command or GUI path **Main Menu>Design Opt>Method/Tool**. (For more information about **OPUSER**, see the *Commands Reference*.)
2. Either issue the **OPTYPE**,USER command or choose menu path **Main Menu>Design Opt>Method/Tool** and specify **User Optimizer**.
3. Issue the **OPEXE** command or choose GUI path **Main Menu>Design Opt>Run**.

Issuing the **OPTYPE**,USER command (or its GUI equivalent) instructs the ANSYS program to bypass the standard ANSYS design optimizer logic and instead execute your custom optimization routine.

The `userop` routine (below) includes a flow chart showing how a user-defined optimization routine interacts with the ANSYS program. It also contains a commented example that shows you sample ANSYS command input for design optimization, done with a user-customized version.

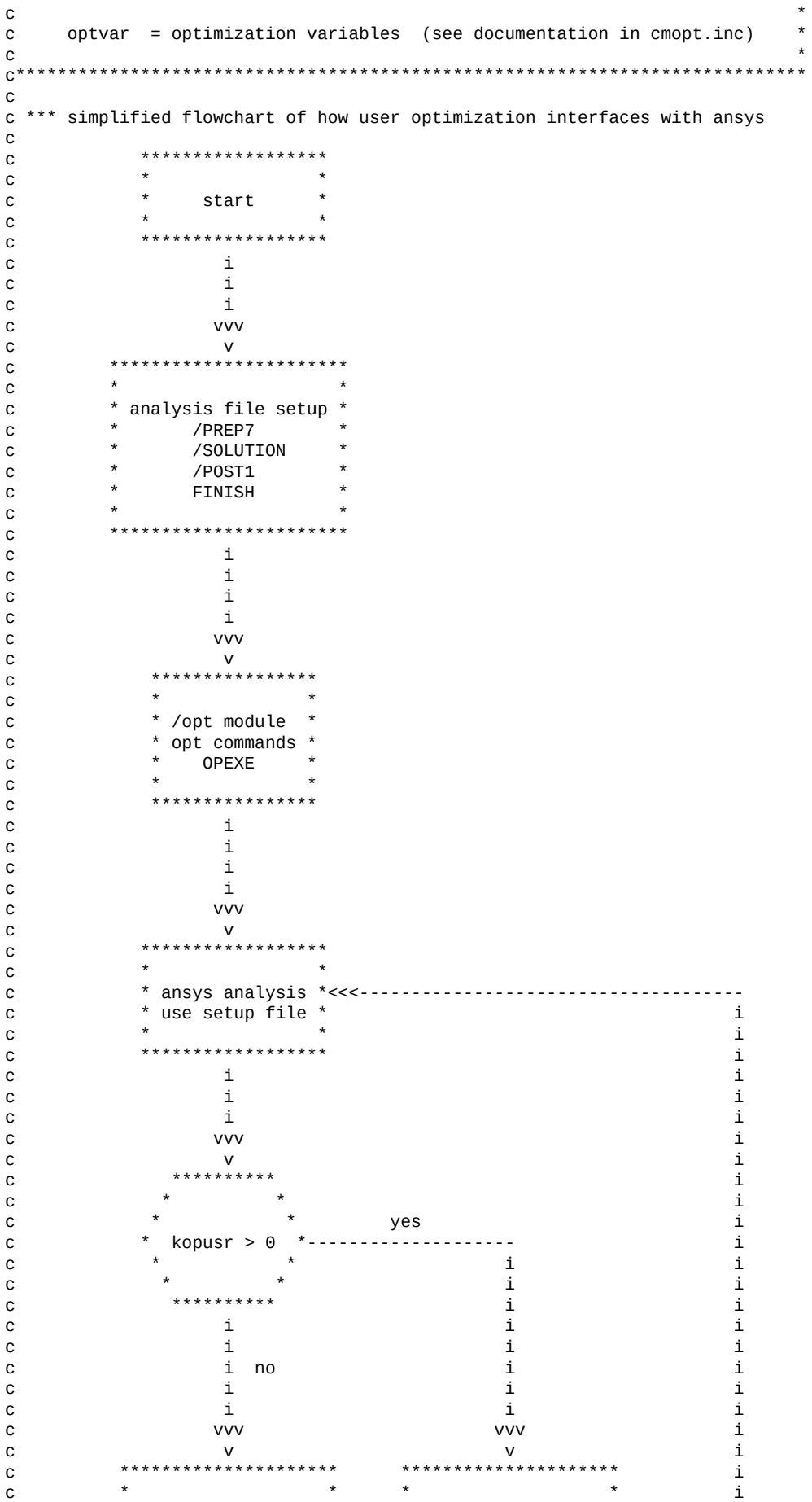
For information about the *kopusr* variable and the `userop` routine, see the next section.

6.10.2. Subroutine userop (Defining a Custom Optimization Routine)

Instead of creating your custom design optimization routine from scratch, you can simply tailor the `userop` routine available on your ANSYS distribution medium. Defined within this routine are a number of variables to which you assign the values you want.

Below is a listing of the header information for`userop` which includes the inputs and outputs and an example user optimization:

```
*deck,userop                              user
      subroutine userop (iott,nterm,maxparm,optvar)
c        primary purpose:  user-supplied optimization routine
c          accessed with optype,user
c           other input comes from the opuser command
c
c         *** copyright(c) 2006 SAS IP, Inc.  All rights reserved.
c         *** ansys, inc.
c *** Notice - This file contains ANSYS Confidential information ***
c
c*************************************************************************
c                                                                        *
c incoming arguments:                                                    *
c                                                                        *
c                                                                        *
c     iott  = ansys output unit                                          *
c                                                                        *
c     nterm = looping control parameter                                  *
c              passed back to routine opterm. this variable should be set   *
c              as follows:                                               *
c                        nterm = 0  if optimization looping should stop     *
c                        nterm = 1  if optimization looping should continue *
c                                                                        *
c     maxparm = maximum number of parameters (first dimension on optvar)    *
```



```
c          *       ansys       *      *      userop       *          i
c          *    optimization   *      *    optimization   *          i
c          *                   *      *    call userop    *          i
c          *********************      *********************          i
c                    i                          i                    i
c                    i                          i                    i
c                    i                          i                    i
c                    ----------------------------                    i
c                                 i                                  i
c                                vvv                                 i
c                                 v                                  i
c                                @@@                                 i
c  ****************            @     @                               i
c  *              *   yes     @         @                             i
c  *     exit     *<<<-----@  terminate  @                            i
c  *              *(nterm=0) @ looping?@                             i
c  ****************            @     @                               i
c                                @@@                                 i
c                                 i                                  i
c                                 i  no                              i
c                                 i (nterm=1)                        i
c                                vvv                                 i
c                                 v                                  i
c                   ****************************                    i
c                   *                          *                    i
c                   * move optimization results *----- to new -----<<<i
c                   *   into parameter vector  *     analysis
c                   *                          *
c                   ****************************
c
c
c *** sample input for user optimization ***
c
c!!!----   minimize  y = x1**2 + x2**2
c!!! ---   subject to :  1 < x1 < 10
c!!! ---                 1 < x2 < 10
c!!! ---                 g = x1 - x2 > 1
c
c     /batch,list         ! use batch mode
c     x1=5                ! initial value for dv x1
c     x2=5                ! initial value for dv x2
c     /prep7              ! enter prep
c     y = x1*x1 + x2*x2   ! define function y
c     g = x1 - x2         ! define function g
c     finish              ! finish prep
c     /opt                ! enter opt
c     opvar,x1,dv,1,10    ! define 1st dv (range 1 to 10)
c     opvar,x2,dv,1,10    ! define 2nd dv (range 1 to 10)
c     opvar,y,obj,,,1     ! define objective function (tol = 1)
c     opvar,g,sv,1        ! define state variable (lower limit 1)
c     optype,user         ! opt method is user supplied
c     opuser,10,1.1,2.0   ! defime user opt modifiers (max iterations=10)
c     opexe               ! execute opt
c     oplist,all          ! list results
c     finish
```

When you finish customizing the `userop` routine, you relink it using either of the methods described in the Installation and Configuration Guide. You must include the **cmopt** common block in your routine, to allow it to pass common data (such as design variables, state variables, and objective function values) to and from the ANSYS program.

6.10.3. Structuring Your Input

When your `userop` optimization routine finishes executing, program control returns to the ANSYS design optimizer, so that the ANSYS commands on `Jobname.LOOP` can execute for the next optimization loop. To use your own customization routine, you should issue the following ANSYS commands (in the order shown):

```
/OPT
OPTYPE,USER   ! ANSYS optimization logic bypassed
OPUSER,       ! user-defined optimization to be done
OPVAR         ! design and state variables & objective function values defined
.
.
.
OPEXE         ! looping begins
```

6.11. Memory Management Routines

ANSYS provides UPF routines you can use for memory management.

6.11.1. Using the Memory Management Routines

ANSYS uses a dynamic memory manager that overlays the system *malloc* and *free* functions and provides a mechanism for accessing the memory from FORTRAN as well as c and c++. Since the UPF routines are provided in FORTRAN, we will be discussing the FORTRAN access routines.

You may certainly use the system *malloc* and *free* functions or, for FORTRAN, the *allocate* system function. However, you may end up competing with ANSYS for memory, and for large problems there may be insufficient system memory to perform the function.

Dynamic memory is done through *Cray-style* pointers, where a dynamically allocated array is defined via the construct

```
pointer (piArray,Array)
integer Array(*)
```

and memory space for the array is allocated by assigning the pointer, in this case piArray, to the allocated memory space:

```
piArray = fAnsMemAlloc (size,...)
```

To use the ANSYS memory manager in a UPF, follow these steps:

1. 1. Define the dynamically allocated arrays:

   ```
   pointer (piArray,Array), (pdData,Data)
   integer Array(*)
   double precision Data(*)
   ```

2. Initialize the pointers as follows:

   ```
   piArray = PTRFTNNULL
   pdData = PTRFTNNULL
   ```

3. Allocate space for an array or arrays, as follows:

   For integer numbers:

   ```
   piArray = fAnsMemAlloc(ileng,MEM_INTEGER,C16Label)
   ```

   For double precision numbers:

   ```
   pdArray = fAnsMemAlloc(dleng,MEM_DOUBLE,C16Label)
   ```

   For complex numbers:

   ```
   pcArray = fAnsMemAlloc(cleng,MEM_COMPLEX,C16Label)
   ```

   For real numbers:

```
prArray = fAnsMemAlloc(rleng,MEM_REAL,C16Label)
```

Where the arguments are:

- *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 routines.

For an example, see *Section 6.7.3: Function user03 (Demonstrates Using ANSYS Memory)*, which appears earlier in this chapter.

6.11.2. Function fAnsMemAlloc (Allocating Space and Returning a Pointer)

```
*deck,fAnsMemAlloc
      function fAnsMemAlloc (iLen, key, c16Label)

c primary function:    Get A Block of Space from mem manager and Return Pointer

c keywords:  integer function for mem allocate

c object/library:  mem

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

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
```

6.11.3. 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
```

6.12. Parameter Processing Routines

The ANSYS distribution medium contains three routines you can use for parameter processing: `pardim`, `parevl`, and `pardef`.

6.12.1. Subroutine pardim (Creating a Dimensioned Parameter)

```
*deck,pardim
      subroutine pardim (cName,labl4,nDim,nxyz,cLabels)
c *** primary function:    create a dimensioned parameter

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
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
```

6.12.2. Function parevl (Finding and Evaluating a Parameter)

```
*deck,parevl
      subroutine parevl (ParName,nDim,subc,lvl,dpValue,chValue,kerr)
c *** primary function:    find and evaluate a parameter
c
c *** Notice - This file contains ANSYS Confidential information ***
c
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
c  output arguments:
c     dpValue (dp,sc,out)           - the value of the parameter (may be a
c                                     packed character*8
c     chValue (ch*128,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
c
c
```

6.12.3. Subroutine pardef (Adding a Parameter)

```
*deck,pardef
      subroutine pardef (label,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     label (chr*(PARMSIZE),sc,in) - name of parameter
c                                    label 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                                    "label" (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 "label" (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 "label" or "label(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                                    "label" (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
```

6.13. Miscellaneous Useful Functions

The ANSYS program has several miscellaneous functions you may find useful for working with UPFs:

- The `erhandler` routine displays output messages (notes, warnings, and errors).
- The `RunCommand` function lets you issue an ANSYS command from within a user routine.
- The `GetStackDisp` routine retrieves current displacement values.
- The **/UNDO** command writes an "undo" file at critical points as a user routine executes.
- The**/HOLD** command allows you to synchronize multiple tasks in ANSYS.
- The **/TRACK** command enables you to do program tracing and timing.

For further descriptions of `erhandler` and **/TRACK**, see *Chapter 8: Subroutines for Users' Convenience*. For details about the `GetStackDisp` function, see *Section 6.8.2: Function GetStackDisp (Getting Current Displacement Values)*.

6.13.1. Using Function RunCommand

This function enables you to execute an ANSYS command from within a user routine. 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
```

6.13.2. Using the /UNDO Command

The "undo" file you create by issuing the **/UNDO** command is similar to the `File.DB` file created when you issue the **SAVE** command. The **/UNDO** command format is:

```
/UNDO,Action
```

Action

ON, to write the undo file
OFF, to prevent the undo file from being written
PROMPT, to have ANSYS ask permission before writing the file
STATUS, to restore the file as it existed after executing the last command issued before the **/UNDO** command.

6.13.3. Using the /HOLD command

Issue the **/HOLD** command to synchronize tasks in ANSYS. The ANSYS program can synchronize tasks at the end of each results file set.

```
/HOLD,Filename,TimeInterval,Timeout
```

Filename
The eight-character name of a message file. If the named file exists, the ANSYS program reads a command from the file and then deletes the file.

TimeInterval
The length of time, in seconds, that ANSYS waits before trying to read the message file again.

Timeout
The maximum length of time, in seconds, that ANSYS can wait between attempts to read the file.

Chapter 7: Accessing the ANSYS Database

This chapter describes how you can retrieve information in the ANSYS database (or store information in the database) by linking subroutines you create into the ANSYS program.

You can use the database access routines with any of the user-programmable features. For example, you can create your own ANSYS commands and use them to execute database access routines (or have a database access routine call a user-defined command).

7.1. 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

7.2. Types of Database Access Routines

The rest of this chapter describes the functions and subroutines available for accessing information in the ANSYS database. The function and subroutine descriptions are grouped into sets: nodal information functions, element attribute information functions, and so on.

7.3. Routines for Selecting and Retrieving Nodes and Elements

7.3.1. 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
```

7.3.2. 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
```

7.3.3. 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
c
c
```

7.3.4. 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
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        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
c     output arguments:
c        none.
c
```

7.3.5. 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
```

7.3.6. 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
```

7.3.7. 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
c
```

7.3.8. 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 ***

c     input arguments:
```

```
c        ielei    (int,sc,in)       - element number
c                                     = 0 - all elements
c        ksel     (int,sc,in)       - type of operation to be performed.
c                                     = 0 - delete element.
c                                     = 1 - select element.
c                                     =-1 - unselect element.
c                                     = 2 - invert select status for element

c     output arguments:  none
```

7.4. Node Information Routines

7.4.1. ndinqr Function (Getting Information About a Node)

The primary function of `ndinqr` is getting information about a node. You can also use this function to set the current node pointer to this node.

```
*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      (dp,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.
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.
```

7.4.2. 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
```

7.4.3. 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.
```

7.4.4. 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
```

7.4.5. 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,
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.
```

7.5. Element Attribute Routines

7.5.1. 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 etc. (see elmcmx)
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
```

7.5.2. 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     input arguments:
c        ielem    (int,sc,in)       - element number

c     output arguments:
c        elmget   (int,func,out)    - status of element.
c                                     = 0 - element undefined
c                                     < 0 - number of nodes on unselected
c                                           element
c                                     > 0 - number of nodes on selected element
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                           (element may need to have its p-level increased)
c                                         if set   - exclude
c                           (element does not need to have its p-level increased)
c                  EL_PEXCLUDE is also used for the LSDYNA part number (trh 9/05)

c        nodes    (int,ar(*),out)   - node numbers for element.
c
c
```

7.5.3. 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.
```

7.5.4. 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.
```

7.5.5. 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.
```

7.5.6. 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
```

7.5.7. 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
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
```

7.5.8. 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
```

7.5.9. 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, BM  =58, QRAT=59, REFT=60, CTEX=61, CTEY=62, CTEZ=63, THSX=64,
c        THSY=65, THSZ=66, DMPR=67, LSSM=68,     =69,     =79,     =71,     =72,
c            =73,     =74,     =75,     =76,     =77,     =78,     =79,     =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.
```

7.5.10. 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, BM  =58, QRAT=59, REFT=60, CTEX=61, CTEY=62, CTEZ=63, THSX=64,
c        THSY=65, THSZ=66, DMPR=67, LSSM=68,     =69,     =79,     =71,     =72,
c            =73,     =74,     =75,     =76,     =77,     =78,     =79,     =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
```

7.5.11. mpput Subroutine (Storing a Material Property Table)

```
*deck,mpput
      subroutine mpput (mat,iprop,ntab,temp,prop)
c *** primary function:    store material property tables.

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

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, BM  =58, QRAT=59, REFT=60, CTEX=61, CTEY=62, CTEZ=63, THSX=64,
c        THSY=65, THSZ=66, DMPR=67, LSSM=68,     =69,     =79,     =71,     =72,
c            =73,     =74,     =75,     =76,     =77,     =78,     =79,     =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.
```

7.5.12. 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, BM  =58, QRAT=59, REFT=60, CTEX=61, CTEY=62, CTEZ=63, THSX=64,
c        THSY=65, THSZ=66, DMPR=67, LSSM=68,     =69,     =79,     =71,     =72,
c            =73,     =74,     =75,     =76,     =77,     =78,     =79,     =80
c
c               (see mpinit for uncommented code and for TB command information)

c     output arguments:  none.
```

7.5.13. 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
```

7.5.14. 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
```

7.5.15. 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
```

7.5.16. 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.
```

7.5.17. 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
```

7.5.18. 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
```

7.5.19. 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
```

7.5.20. userac Subroutine (Demonstrates Use of Element Attribute Routines)

See *Section 6.1.2.8: Subroutine userac (Accessing Element Information)* for an example that demonstrates how to use the `userac` subroutine to extract information about an element type and element real constants from the ANSYS database. You can find this subroutine on your ANSYS distribution media.

7.6. Coupling and Constraint Routines

7.6.1. 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
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
```

7.6.2. 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)
```

7.6.3. 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
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)
```

7.6.4. 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
```

7.6.5. 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           = -1             - return master dof for this eqn

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

7.6.6. ceget Function (Getting an Constraint Equation)

```
*deck,ceget
      function ceget (nce,ieqn,deqn)
c *** primary function:    get a constraint equation

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

```
c  input arguments:
c     nce            (int,sc,in)         - constraint equation number

c  output arguments:
c     ceget        (int,func,out)           - number of dof in equation
c     ieqn         (int,ar(ceget+1),out)    - integer info
c                             ieqn(1:ceget) - list of node*32+dof
c                             ieqn(ceget+1) - number of dof in equation
c                                             (copy of return value)
c                                           - negative means internal CE
c     deqn         (dp,ar(ceget+1),out)     - dp info
c                             deqn(1:ceget) - list of coefficients
c                             deqn(ceget+1) - constant term
```

7.6.7. 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  input arguments:
c     nce        (int,sc,in)       - constraint equation set number
c     n          (int,sc,in)       - number of degrees of freedom in set
c     ieqn       (int,ar(n+1),in)  - integer info
c                           ieqn(1:n) - node*32+dof for each dof in set
c                           ieqn(n+1) - number of dof in set (copy of n above)
c                                     - negative means internal CE
c     deqn       (dp,ar(n+1),in)   - dp info
c                           deqn(1:n) - coefficients of each dof in set
c                           deqn(n+1) - constant term
c
c  output arguments:  none
```

7.6.8. 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  input arguments:
c     ncei         (int,sc,in)        - constraint equation number
c     ksel         (int,sc,in)        - select/delete flag
c                                       = 0 - delete equation
c                                       = 1 - select equation

c  output arguments:  none
```

7.7. Nodal Loading Routines

7.7.1. disiqr Function (Getting a Information About Constraints)

```
*deck,disiqr
      function disiqr (node,key)
c *** primary function: get information about constraints

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

c     input arguments:
c        node    (int,sc,in)       - node number for inquire.
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.
```

7.7.2. 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
c    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
```

7.7.3. 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                           PRES=19, TEMP=20, VOLT=21, MAG =22, ENKE=23, ENDS=24
c         EMF =25, CURR=26           (missing entries are spares)

c        value    (dp,ar(2),in)     - (real,imag) values for constraint

c     output arguments:  none.
```

7.7.4. disdel Subroutine (Deleting a Constraint at a Node)

```
*deck,disdel
      subroutine disdel (node,idf)
```

```
c *** primary function:    delete a constraint at a node

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

c     input arguments:
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                           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.
```

7.7.5. 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.
```

7.7.6. 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
```

7.7.7. 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                           FLOW=19, HEAT=20, AMPS=21, FLUX=22, NPKE=23, NPDS=24
c         CURT=25, VLTG=26           (missing entries are spares)

c        value    (dp,ar(2),in)     - (real,imag) values of force

c     output arguments:  none.
```

7.7.8. 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                           FLOW=19, HEAT=20, AMPS=21, FLUX=22, NPKE=23, NPDS=24
c         CURT=25, VLTG=26           (missing entries are spares)

c     output arguments:
c        none.
```

7.7.9. 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
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.
```

7.7.10. 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).
```

7.7.11. 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.
```

7.7.12. 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.
```

7.7.13. 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.
```

7.7.14. 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).
```

7.7.15. 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.
```

7.7.16. nhgdel Subroutine (Deleting a Nodal Heat Generation)

```
*deck,nhgdel
      subroutine nhgdel (node)
c *** primary function:    delete nodal heat generations.

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

7.7.17. 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.
```

7.7.18. 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).
```

7.7.19. nfuput Subroutine (Storing a Nodal Fluence)

```
*deck,nfuput
      subroutine nfuput (node,fluen)
c *** primary function:    store nodal fluence.

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

c     input arguments:
c        node    (int,sc,in)       - node number
c        fluen   (dp ,sc,in)       - nodal fluence

c     output arguments:  none.
```

7.7.20. 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
```

7.7.21. 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.
```

7.7.22. 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

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).
```

7.7.23. 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.
```

7.7.24. 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.
```

7.7.25. 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.
```

7.7.26. 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
```

7.7.27. 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.
```

7.7.28. 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.
```

7.8. Element Loading Routines

7.8.1. 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.
```

7.8.2. 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 (1-6)

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
```

7.8.3. 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 (1-6)
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.
```

7.8.4. 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.
```

7.8.5. 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.
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.
```

7.8.6. 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)
```

7.8.7. 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.
```

7.8.8. 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.
```

7.8.9. 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
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.
```

7.8.10. 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)
```

7.8.11. 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)
```

7.8.12. 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.
```

7.8.13. 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.
```

7.8.14. 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

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)
```

7.8.15. 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)
```

7.8.16. 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
```

7.8.17. 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
```

7.8.18. 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)
```

7.8.19. 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)
```

7.8.20. efudel Subroutine (Deleting an Element Fluence)

```
*deck,efudel
      subroutine efudel (ielem)
c *** primary function:    delete element fluences

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

c     variable (typ,siz,intent)     description
c        ielem    (int,sc,in)       - element number

c     output arguments:  none
```

7.8.21. 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
```

7.8.22. 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)
```

7.8.23. 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)
```

7.8.24. edcdel Subroutine (Deleting an Element Current Density)

```
*deck,edcdel
      subroutine edcdel (ielem)
c *** primary function:    delete element current densities

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

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

c     output arguments:  none
```

7.8.25. 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
```

7.8.26. 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
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)
```

7.8.27. 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 displacments

c     output arguments:  none
c                                     NOTE: If a value is not defined, a very
c                                        small number should be used (2**-100)
```

7.8.28. 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.
```

7.8.29. eimget Function (Getting an Element Face Impedance)

```
*deck,eimget
      function eimget (ielem,iface,value)
c *** primary function:    get an element face impedance

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

c     input arguments:
c        ielem    (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 impedances
c                                     = 0 - this element face has no impedances
c                                     > 0 - number of values defined
c        value    (dp ,ar(*),out)   - the element impedances (real,imag)
```

7.8.30. 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
```

7.8.31. 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
```

7.8.32. 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

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
```

7.8.33. 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.
```

7.8.34. 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
```

7.8.35. 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.
```

7.8.36. 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.
```

7.8.37. 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
```

7.8.38. 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
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.
```

7.8.39. 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.
```

7.9. Results Information Routines

7.9.1. 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
```

7.9.2. 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     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        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                      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        dspget   (int,func,out)    - number of actual results
c        value    (dp,ar(ndf),out)  - result values
```

7.9.3. dspput Subroutine (Storing a Constraint 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
```

7.9.4. 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
```

7.9.5. 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
```

7.9.6. 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
```

7.9.7. 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
```

7.9.8. 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
```

7.9.9. 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
```

7.9.10. 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
```

7.9.11. 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
```

7.9.12. 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
```

7.9.13. 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
```

7.9.14. 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.
```

7.9.15. 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                                       (6*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 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.
```

7.9.16. 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.
```

7.9.17. 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
```

7.9.18. 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) = spares
c                                     (9) = thermal energy
c                                  (10-11) = spares
```

7.9.19. 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) = spares
c                                     (9) = thermal energy
c                                 (10-11) = spares

c     output arguments:  none
```

7.9.20. 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:
c        ielem    (int,sc,in)       - element number
c                                       = 0 - delete for all defined elements

c     output arguments:  none.
```

7.9.21. 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
```

7.9.22. 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
```

7.9.23. 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
```

7.9.24. 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.
```

7.9.25. 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
```

7.9.26. 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 ***

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
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.
```

7.9.27. 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                                       (6*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
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.
```

7.9.28. 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.
```

7.9.29. 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
```

7.9.30. eplget Function (Getting an Element's Nodal Plastic Strains)

```
*deck,eplget
      function eplget (ielem,value)
c *** primary function:    get element nodal plastic strains.

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.
```

7.9.31. 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.
```

7.9.32. 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.
```

7.9.33. 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
```

7.9.34. ecrget Function (Getting an Element's Nodal Creep Strains)

```
*deck,ecrget
      function ecrget (ielem,value)
c *** primary function:    get element nodal creep strains.
```

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

c     input arguments:
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.
```

7.9.35. 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.
```

7.9.36. 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.
```

7.9.37. 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
```

7.9.38. ethget Function (Getting an Element's Nodal Thermal Stresses)

```
*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

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.
```

7.9.39. ethput Subroutine (Storing an Element's Nodal Thermal Stresses)

```
*deck,ethput
      subroutine ethput (ielem,nval,value)
c *** primary function:    store nodal thermal strains at an element.
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                                      (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, 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.
```

7.9.40. 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.
```

7.9.41. 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
```

7.9.42. eulget Function (Getting an Element's Nodal Euler Angles)

```
*deck,eulget
      function eulget (ielem,value)
c *** primary function:    get element nodal euler angles.

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

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

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)
```

7.9.43. 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)
```

7.9.44. 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.
```

7.9.45. 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
```

7.9.46. 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
```

7.9.47. 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
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
```

7.9.48. 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.
```

7.9.49. 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
```

7.9.50. 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.
```

7.9.51. 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
```

7.9.52. 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.
```

7.9.53. 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
c                                        setting of key
```

7.9.54. 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
```

7.9.55. 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
```

7.9.56. 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.
```

7.9.57. 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
```

7.9.58. 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
```

7.9.59. 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
```

7.9.60. 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.
```

7.9.61. 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
```

7.9.62. 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
```

7.9.63. 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
```

7.9.64. 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.
```

7.9.65. 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

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

7.9.66. 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.
```

7.9.67. 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
```

7.9.68. 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.
```

Chapter 8: Subroutines for Users' Convenience

This chapter describes ANSYS routines available to you for use in programming. Using these routines isn't required, but may make your life easier. These routines include a set of general routines that perform utility-type functions, a set of routines supporting vector functions, a set of routines supporting matrix functions, and routines supporting message processing options.

8.1. Input and Output Abbreviations

The descriptions of inputs and outputs for the routines discussed in this chapter use the following abbreviations:

- Argument *type* is one of the following:

  int - integer
  dp - double precision
  log - logical
  chr - character
  dcp - double precision complex

- Argument *size* is one of the following:

  sc - scalar variable
  ar(*n*) - array variable of length *n*
  func - functional return value

- Argument *intent* is one of the following:

  in - input argument
  out - output argument
  inout - both an input and an output argument

8.2. General Subroutines

8.2.1. 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 ***

c !!! NOTICE to progammers: this routine does not convert from a !!!
c !!! machine-independent format! Use dpexttoch if this dp word  !!!
c !!! came from a common or non-char database record             !!!

c  input arguments:
c     dp8      (dp,sc,in)        - dp variable containing characters

c  output arguments:
c     ch8      (ch*8,sc,out)     - characters retreived from the dp word
```

8.2.2. 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_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
```

8.2.3. 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
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                                  21=errhpi,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
```

8.2.4. 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
```

8.2.5. 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
```

8.2.6. erhandler Subroutine (Displaying ANSYS Errors)

```
*deck,erhandler
      subroutine erhandler  (filein,msgid,msglvl,lngstrng,dperr,cherr)

c *** primary function:    Display ANSYS error messages

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

c  input arguments:
c     filein   (ch*40,sc,in)     - Filename used for character portion of
c                                  message ID (this is the file name of the
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
```

8.2.7. intrp Subroutine (Doing Single Interpolation)

```
*deck,intrp
      subroutine intrp (klog,kppx,kstpz,xval,ax,ay,yval,nmax,kyoff)
c *** primary function: **** subroutine for single interpolation ****
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
```

8.2.8. tranx3 Subroutine (Processing Geometry for 3-D Line Elements)

```
*deck,tranx3
      subroutine tranx3 (nnod,xyz,nx,tr)
c *** primary function: geometric processor for 3-d line elements
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
```

8.2.9. systop Subroutine (Stopping an ANSYS Program Run)

```
*deck,systop
      subroutine systop (icode)
c *** primary function:    stop an ansys run
c *** secondary functions: pass an error code to the system
c
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
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
```

8.3. Vector Functions

8.3.1. vdot Function (Computing the Dot Product of Two Vectors)

```
*deck,vdot
      function  vdot (v1,v2,n)
c *** primary function: compute dot product of vectors v1 and v2

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

c  input arguments:
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
```

8.3.2. 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
```

8.3.3. vmax Function (Retrieving the Maximum Vector Value at a Given Location)

```
*deck,vmax
      function vmax (v,n,locmax)
c *** primary function: return the max value and location in a vector

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

c  input arguments:
c     v        (dp,ar(n),in)     - vector v
c     n        (int,sc,in)       - length of vector v

c  output arguments:
c     locmax   (int,sc,out)      - location of max value in vector v
c     vmax     (dp,sc,out)       - max value in vector v
```

8.3.4. lastv Function (Retrieving the Position of the Last Nonzero Term in a Double Precision Vector)

```
*deck,lastv
      function lastv (v,n)
c ********* find position of last non-zero term in a d.p. vector *********
```

8.3.5. izero Function (Setting an Integer Vector to Zero)

```
*deck,izero
      subroutine izero (ivect,n)
c **********  set an integer vector to zero  **********
c
```

8.3.6. imove Function (Assigning Equal Values to Two Integer Vectors)

```
*deck,imove
      subroutine imove (i1,i2,n)
c **********  move a vector from one to another  **********
c
```

8.3.7. 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 ***

c  input arguments:
c     v        (dp,ar(n),inout)  - vector to be zeroed out
c     n        (int,sc,in)       - number of words to zero out

c  output arguments:
c     v        (dp,ar(n),inout)  - zeroed vector
c
```

8.3.8. vmove Subroutine (Moving One Vector into Another)

```
*deck,vmove
      subroutine vmove (v1,v2,n)
c *** primary function: copy a vector into another vector
c
c *** Notice - This file contains ANSYS Confidential information ***
c
c *** Note: This function can have unpredictable results if v1 and v2
c           reference the same array and overlap.
c
c  input arguments:
c     v1       (dp,ar(*),in)     - vector v1
c     n        (int,sc,in)       - length of vectors v1, v2
c
c  output arguments:
c     v2       (dp,ar(*),out)    - vector v2
c
c
```

8.3.9. 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
c *** Notice - This file contains ANSYS Confidential information ***
c
c  input arguments:
c     v1       (dp,ar(inc1,n),in)  - vector v1
c     inc1     (int,sc,in)         - increment on vector v1
c     inc2     (int,sc,in)         - increment on vector v2
c     n        (int,sc,in)         - number of items to be moved
c
c  output arguments:
c     v2       (dp,ar(inc2,n),in)  - vector v2
c
```

8.3.10. 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 ***

c  input arguments:
c     n        (int,sc,in)       - length of vector v
c     const    (dp,sc,in)        - constant to set vector v to

c  output arguments:
c     v        (dp,ar(n),out)    - vector v
c
```

8.3.11. viinit Subroutine (Assigning a Scalar Constant to a Vector Incrementally)

```
*deck,viinit
      subroutine viinit (v,inc,n,const)
c *** primary function: set the components of vector v to const by increments
c
c *** Notice - This file contains ANSYS Confidential information ***
c
c  input arguments:
c     inc      (int,sc,in)         - increment (first dimension) of vector v
c     n        (int,sc,in)         - length (second dimension) of vector v
c     const    (dp,sc,in)          - constant to set components of vector v to
```

```
c
c  output arguments:
c     v        (dp,ar(inc,n),out)  - vector v
c
```

8.3.12. vapb Subroutine (Setting a Vector to Sum of Two Vectors)

```
*deck,vapb
      subroutine vapb (a,b,c,n)
c *** primary function: add vector a to vector b to get vector c

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

c  input arguments:
c     a        (dp,ar(n),in)     - a vector
c     b        (dp,ar(n),in)     - b vector
c     n        (int,sc,in)       - length of vectors a,b,c

c  output arguments:
c     c        (dp,ar(n),out)    - c vector
c
```

8.3.13. vapb1 Subroutine (Combining Two Vectors in One)

```
*deck,vapb1
      subroutine vapb1 (a,b,n)
c *** primary function: add vector b to vector a,
c                      and store in vector a
```

8.3.14. 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
```

8.3.15. vamb Subroutine (Gets a Third Vector by Subtracting One Vector from Another)

```
*deck,vamb
      subroutine vamb (a,b,c,n)
c *** primary function: subtract vector b from vector a to get vector c

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

c  input arguments:
c     a        (dp,ar(n),in)     - vector a
c     b        (dp,ar(n),in)     - vector b
c     n        (int,sc,in)       - length of vectors a,b,c

c  output arguments:
c     c        (dp,ar(n),out)    - vector c
c
```

8.3.16. vamb1 Subroutine (Subtracting One Vector from Another)

```
*deck,vamb1
      subroutine vamb1 (a,b,n)
c *** primary function: subtract vector b from vector a and save in vector a

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

c  input arguments:
c     a        (dp,ar(n),inout)  - a vector
c     b        (dp,ar(n),in)     - b vector
c     n        (int,sc,in)       - length of vectors a,b
```

```
c  output arguments:
c     a        (dp,ar(n),inout)  - a vector
c
```

8.3.17. 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 ***

c  input arguments:
c     v1       (dp,ar(n),in)     - vector v1
c     n        (int,sc,in)       - length of vectors v1, v2
c     const    (dp,sc,in)        - constant to multiply v1

c  output arguments:
c     v2       (dp,ar(n),out)    - vector v2
```

8.3.18. 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 ***

c  input arguments:
c     v1       (dp,ar(n),inout)  - vector v1
c     n        (int,sc,in)       - length of vector n
c     const    (dp,sc,in)        - constant to multiply v1

c  output arguments:
c     v1       (dp,ar(n),inout)  - vector v1
c
```

8.3.19. 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
```

8.3.20. 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
```

8.3.21. vnorm Subroutine (Normalizing a Vector to Unit Length)

```
*deck,vnorm
      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
```

8.3.22. 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
```

8.3.23. 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
```

8.4. Matrix Subroutines

8.4.1. 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
```

8.4.2. maxv1 Subroutine (Multiplying a Vector by a Matrix)

```
*deck,maxv1
      subroutine maxv1 (a,v, nr,nc)
c *** primary function: multiply a vector by a matrix

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

c  input arguments:
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
```

8.4.3. matxv Subroutine (Multiplying a Vector by a Full Transposed Matrix)

```
*deck,matxv
      subroutine matxv (a,v,w, nr,nc)
c *** primary function: multiply vector by full transposed matrix

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

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
```

8.4.4. matxv1 Subroutine (Multiplying a Vector by a Full Transposed Matrix)

```
*deck,matxv1
      subroutine matxv1 (a,v, 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
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
```

8.4.5. matxb Subroutine (Transposing a matrix)

```
*deck,matxb
      subroutine matxb (a,b,c, na,nb,nc, n1,n2,n3)
c *** primary function:  (a)t * (b) = (c)     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 columns 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
```

8.4.6. maat Subroutine (Changing a Matrix Value via Addition, Multiplication, and Transposition)

```
*deck,maat
      subroutine maat(a,c, nc,n, con)
c primary function: does con*a*at and sums the result onto c (a is a vector)

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

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
```

8.4.7. 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
```

8.4.8. 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).
```

8.4.9. 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
```

8.4.10. 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 ***
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
```

Appendix A. Creating External Commands in UNIX

External commands allow you to add your own customized extensions to ANSYS without relinking the program. You can create custom routines in C that access any of the ANSYS API functions, link them into shared libraries using the supplied utilities, and execute the routines via the "external command" feature within ANSYS. In addition, ANSYS provides special commands that list all available external commands and allow you to reset all currently referenced external commands.

External command capability is supported on all UNIX platforms. Refer to your 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/v110/ansys/custom/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 */
    cAnsPrintf("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
```

The utility will produce a Makefile in the current directory. You will be able to generate the shared library by issuing the following command:

```
make
```

For example, to create the shared library for `mylibrary.so`, you would issue the following command:

```
% make
```

You will then find the specified shared library file in the current directory. You may also see warnings from the make process, and you may need to modify the Makefile or your source code.

A.1.3. Creating an External Table File

The external table file (`ans_ext.tbl`) can reside in any directory (but you must specify that directory in the **ANSYS_EXTERNAL_PATH** environment variable). The file contains an entry for each shared library function you wish to allow ANSYS to access. There is no limit to the number of entries. The file entries have the following format:

`/shared/library/path/library.so ~cm_name function_name`

where

`/shared/library/path/library.so`

Is the path to the directory that contains the shared library file. (Remotely mounted file systems are not recommended.)

`~cm_name`

Is the command used to invoke the function within ANSYS. The command name must begin with a tilde (~) and each command name must be unique within the first four characters. The command name must be eight characters or less, including the tilde (~).

`function_name`

Is the name of the function that is referenced by the specified command name. (This must be unique within the first four characters if multiple external commands are specified.)

For example, the following entry references the `/home/mydir/mylibs/myobject.so` shared library and the *`myobject_`* function. It specifies *`~myobj`* as the related command:

```
/home/mydir/mylibs/myobject.so ~myobj myobject_
```

ANSYS also makes use of external commands, and places its own shared libraries and the associated external table file in the `/ansys_inc/v110/ansys/lib/<platform>` directory (where *<platform>* is the directory specific to your computing platform, such as `/sgi64` or `/hppa8000-64`).

ANSYS loads external commands in the following order:

- ANSYS first checks the `ans_ext.tbl` file in the `/ansys_inc/v110/ansys/lib/<platform>` directory and loads any external commands referenced there.
- ANSYS then loads external commands referenced by the external table file in the directory designated with the **ANSYS_EXTERNAL_PATH** environment variable (see section *Section A.2.4: Setting the ANSYS_EXTERNAL_PATH Environment Variable*).

If you designate a command name that has the same first four characters as a command listed in the `/ansys_inc/v110/ansys/lib/<platform>/ans_ext.tbl` file, you will not be able to access your command. Therefore, it is a good practice to check the ANSYS external table file to make sure you have no external command name conflicts. *Do not* modify the `/ansys_inc/v110/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 ANSYS will be executed.

A.1.4. Setting the ANSYS_EXTERNAL_PATH Environment Variable

Before launching ANSYS, you must first set the **ANSYS_EXTERNAL_PATH** to point to the directory containing the external table file. (For convenience, if you distribute your new functionality to other users they should set their `.login` or `.cshrc` files so that it is persistent from session to session.) For example, the following sets the environment variable to point to the `/home/mydir` directory.

```
setenv ANSYS_EXTERNAL_PATH /home/mydir
```

A.1.5. Using External Commands

To call an external command, enter it as you would any other ANSYS command. You can also call external commands through either an APDL macro or UIDL script.

Note

Avoid recursive external commands; that is, avoid situations where an external command calls another external command.

A.1.6. Checking External Command Status

You can check what shared libraries are currently accessible by entering the **~DEBUG** command in the command input window. The following figure shows an example of **~DEBUG** command output.

```
External Command Mappings:

 Command   Library                              Function         Accessed?
  *-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-
 ~excmd    /home/mydir/mycode/mycommand.so      excmd            YES
```

In this example, the output lists the command, the related shared library, the function, and if the command has been accessed.

A.1.7. Resetting External Commands

You can

- Close all shared libraries
- Free memory associated with external commands

by issuing the **~RESET** command. The command issues the following message to confirm that the reset operation was complete.

```
~RESET was processed: The external command buffers have been cleared.
```

Note

The **/CLEAR** command also closes/resets all external command shared libraries.

Appendix B. Creating External Commands in Windows

This section describes the steps required to create external commands on Windows platforms.

B.1. Tasks in Creating an External Command

To create a functional external command, you will need to complete the following general steps:

- Create compatible C source code.
- Create an external definition file (`projname.def`).
- Create a new project in Microsoft Developer Studio.
- Create a shared library.
- Create an external table file (`ans_ext.tbl`), listing the various shared libraries, each function and the related command.
- Set the **ANSYS_EXTERNAL_PATH** environment variable

The following sections detail each of these tasks.

B.1.1. Creating Compatible Code

You can create your functions using any of the API functions described in `Program Files\Ansys Inc\V110\custom\include\cAnsysInterface.h`, `cAnsQuery.h`, and `cAnspick.h`. You can then execute these functions via the "external command" feature within ANSYS. In addition, ANSYS provides special commands that list all available external commands and allow you to reset all currently referenced external commands. The following code segment demonstrates, at a minimal level, how to create functions that can be used as an entry point into a custom coded shared library.

The most important point in the following example is:

- The C program interface is an integer function that has one argument (a char pointer).

```
#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 */
    cAnsPrintf("Max Node Number = %g\n",dblrtn);

    return (i);
```

B.1.2. Creating a New Project

To build and link your code, you will first need to create a project in the Microsoft Developer Studio.

- After launching Developer Studio, press **Ctrl+N** (or choose **New** from the **FILE** menu). In the **New** window, click on the **Projects** tab. Select **Win 32 Dynamic-Link Library**, and specify your project name and directory.
- In the Workspace frame, click on the **File View** tab. Right click on the sub category **projname files** and press **F** or select **Add Files to Project**). Choose your source and definition files, `ansys.lib`, `cAnsInterface.h` and any other necessary header files. The import library file `ansys.lib` is in `/ansys110/lib` and the header file `cAnsysInterface.h` is in `/ansys110/extra/custom/include`.

B.1.3. Creating an External Definition File

For each external command, you must declare it in the external definition file. The naming convention for this file is the name of your project with the `.def` extension; it must be located in your project directory. This file consists of the word *EXPORTS* on the first line, and the name(s) of the functions to be exported on each successive line. For the example function above:

```
EXPORTS

extfunc
```

B.1.4. Creating a Shared Library

Once all of the necessary files have been incorporated into your project, simply compile (**Ctrl+F7**) and build (**F7**) the project. In your project directory, Developer Studio will create a Debug directory and will place the library in that directory (`projname.dll`).

B.1.5. Creating an External Table File

The external table file (`ans_ext.tbl`) can reside in any directory (but you must specify that directory in the **ANSYS_EXTERNAL_PATH** environment variable). The file contains an entry for each shared library function you wish ANSYS to access. There is no limit to the number of entries. The file entries have the following format:

`C:\shared\library\path\projname.dll ~cm_name function_name`

where:

`C:\shared\library\path\projname.dll` is the path to the directory that contains the shared library file. (Remotely mounted file systems are not recommended.)

~cm_name Is the command used to invoke the function within ANSYS. The command name must begin with a tilde (~) and the first four characters of each command name must be unique.

function_name Is the name of the function that is referenced by the specified command name. (This must be unique within the first four characters if multiple external commands are specified.)

For example, the following entry references the `C:\home\mydir\mylibs\myobject.dll` shared library and the *myobject* function, and specifies *~myobj* as the related command:

```
C:\home\mydir\mylibs\myobject.dll ~myobj myobject
```

ANSYS also makes use of external commands, and places its own shared libraries and the associated external table file in the `C:\Program Files\Ansys Inc\V110\lib\platform` directory (where platform is the directory specific to your computing platform, such as `\Intel`). ANSYS loads external commands in the following order:

- ANSYS first checks the `ans_ext.tbl` file in the `C:\Program Files\Ansys Inc\V110\lib\platform` directory and loads any external commands referenced there.
- ANSYS then loads external commands referenced by the external table file in the directory designated with the **ANSYS_EXTERNAL_PATH** environment variable (see *Section B.1.6: Setting the ANSYS_EXTERNAL_PATH Environment Variable*).

If you designate a command name that has the same first four characters as a command listed in the `C:\Program Files\Ansys Inc\V110\lib\platform` file, you will not be able to access your command. Therefore, it is a good practice to check the ANSYS external table file to make sure you have no external command name conflicts. Do not modify the `C:\Program Files\Ansys Inc\V110\lib\platform\ans_ext.tbl` file. You can also use the **~DEBUG** command to verify that no external command name conflicts exist.

Note

The shared library must be consistent with the computer type and OS level on which ANSYS will be executed.

B.1.6. Setting the ANSYS_EXTERNAL_PATH Environment Variable

Before launching ANSYS, you must first set the **ANSYS_EXTERNAL_PATH** to point to the directory containing the external table file. In Windows NT, the environment variables are in System Properties, which can be accessed through the Control Panel. For example, the following string sets the environment variable to point to the `C:\home\mydir` directory.

```
set ANSYS_EXTERNAL_PATH=C:\home\mydir
```

B.1.7. Using External Commands

To call an external command, enter it as you would any other ANSYS command in the ANSYS command window. You can also call external commands through either an APDL macro or UIDL routine.

Note

Avoid recursive external commands; that is, avoid situations where an external command calls another external command.

B.1.8. Checking External Command Status

You can check what shared libraries are currently accessible by entering the **~DEBUG** command in the command input window. The following figure shows an example of **~DEBUG** command output.

```
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.

Index

Symbols

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

R

S

T

U

V

W

X

Z