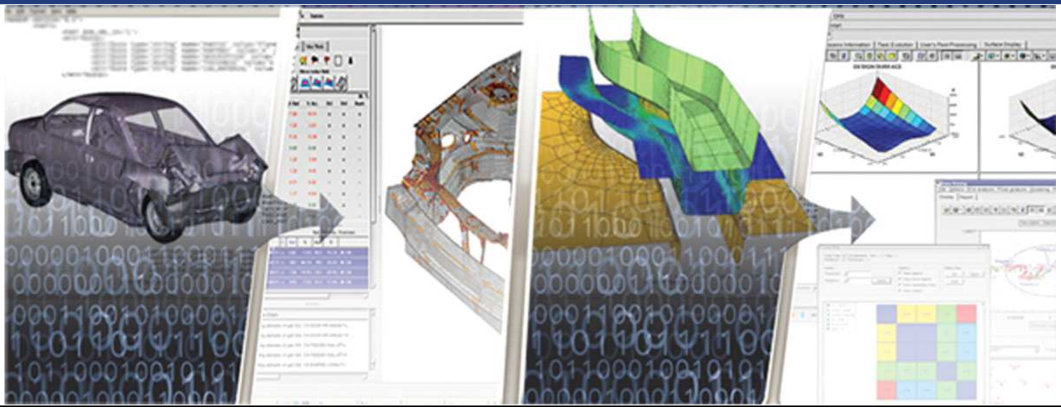


RADIOSS User's Code Interface

2017 version – January 2017

User Sensors

Chapter 5

Altair

Altair Engineering, Inc., World Headquarters: 1820 E. Big Beaver Rd., Troy MI 48083-2031 USA
Phone: +1.248.614.2400 • Fax: +1.248.614.2411 • www.altair.com • info@altair.com

TABLE OF CONTENTS

5.0 User Sensors

In RADIOSS, up to three types of user sensors can be defined.

Two subroutines are needed to define a new sensor: one must be linked with RADIOSS Starter and the other must be linked with RADIOSS Engine.

The Starter subroutines are called LECSEN_USR1, LECSEN_USR2, or LECSEN_USR3. They are used to read sensor data and to initialize parameters.

The Engine subroutines are called USER_SENS1, USER_SENS2 or USER_SENS3 and they perform a user-defined action.

The only argument passed to the user routines in RADIOSS Engine is the sensor identifier.

User sensors can be used in RADIOSS in the same way as standard sensors. They may activate or deactivate other sensors, airbags, springs, boundary conditions, imposed velocities/displacements, concentrated loads, etc. They specify the user-defined activation condition and sensor action. User sensor may retrieve RADIOSS variables using special access functions. All data may be stored in sensor buffer for user-defined processing or for passing information to other parts of user code.

5.1 Starter Subroutine LECSEN_USRn

This subroutine read the user's sensor input data. The number of cards and their formats are free.

The argument list of LECSEN_USRn and its individual arguments and descriptions are as follows:

```
C-----------------------------------------------------------------------
      SUBROUTINE LECSEN_USRn(IIN, IOUT)
C-----------------------------------------------------------------------
```

Argument	**Format**	**Description**
IIN	Integer read only scalar	Input file unit (Starter input file) on which the data are read.
IOUT	Integer read only scalar	Output file unit (Starter listing file).

5.1.1 Storage Functions for Sensor Parameters (Starter Only)

To store data in a sensor's parameters buffers, use the following functions:

- `integer` **`ierr = SET_U_SENS_IPAR(`**`integer` **`isens,`** `integer` **`sens_index,`** `integer` **`var)`**
- `integer` **`ierr = SET_U_SENS_FPAR(`**`integer` **`isens,`** `integer` **`sens_index,`** `float` **`var)`**

These functions store an integer or float variable in respective buffers, at a position indicated by `sens_index`, which is independent in each buffer. The length of sensor parameter buffers is limited to 12 integer variables and 20 float variables.

5.2 Engine Subroutine USER_SENSn

This subroutine performs a user-defined action with a given sensor.

```
C-----------------------------------------------------------------------
       SUBROUTINE USER_SENS1(SENS_ID)
C-----------------------------------------------------------------------
```

5.3 Functions for User Sensors

5.3.1 Restore Functions for Sensor Parameters

You can access the integer and float sensor parameter buffers using two functions, similar to those used in the Starter subroutine:

- `integer` **`ierr = GET_U_SENS_IPAR(`**`integer` **`isens, sens_index`**`, integer` **`var)`**
- `integer` **`ierr = GET_U_SENS_FPAR(`**`integer` **`isens, sens_index`**`, float` **`var)`**

The internal RADIOSS sensor numbers `isens` are not sensor identifiers that can be used to access the parameter buffers. The additional function that translates a sensor ID into an internal sensor number is below.

- `integer` **`isens = GET_U_NUMSENS(`**`integer` **`sens_id)`**

The function returns 0 if the sensor ID is not found in the RADIOSS database. The value can be stored in a local variable for multiple access to parameter buffers, or call the buffer access functions directly with the sensor ID, using the GET_U_NUMSENS function:

- **`ierr = GET_U_SENS_IPAR(GET_U_NUMSENS(sens_id),sens_index,var)`**

5.3.2 Read/Write Functions of Working Array Buffers

To store and read current information, use a user array (float). Its length is limited to 100 variables and can be accessed with the following functions.

- `integer` **`ierr =SET_U_SENS_VALUE (`**`integer` **`isens,`**`integer` **`sens_index,`**`float` **`var)`**
- `integer` **`ierr =GET_U_SENS_VALUE(`**`integer` **`isens,`**`integer` **`sens_index,`**`float` **`var)`**

The user variables are stored and retrieved from a buffer of sensor `isens`, using buffer index `sens_index`.

Use the function inverse `GET_U_NUMSENS` to translate a sensor number into its ID, or if a sensor identifier is needed.

- `integer` **`isens = GET_U_SENS_ID(`**`integer` **`sens_id)`**

5.3.3 Setting and Checking Global Activation Flags

The same mechanisms used to activate and deactivate standard RADIOSS sensors can be used with user sensors. This way, they can activate springs, interfaces, rigid walls, or other sensors using a specific user condition. Additionally, the user sensor may form part of a hierarchy of RADIOSS sensors (ex: logical sensors) or use standard sensors as input.

Use the following functions with standard activation flags.

- `integer` **`ierr = SET_U_SENS_ACTI(`**`integer` **`isens)`**

  This function activates user sensors `isens`. The activation time is set as current time.

  The sensor activation will actually be delayed one cycle to preserve a hierarchical sensor activation order in parallel processing.

- `float` **`dtime = GET_U_SENS_ACTI(`**`integer` **`isens)`**

  This function returns a time delay sine the first activation of sensor number `isens`. If `dtime` > 0, the sensor is `actif`. Use this to check the state of standard sensors or other user sensors that were activated by `GET_U_SENS_ACTI`.

  User sensors may activate RADIOSS variables only by specialized functions, which include:

  - **Access functions to RADIOSS time and cycle variables.**

`float` **`Time = GET_U_TIME()`**	Returns the actual simulation time.
`integer` **`Ncyc = GET_U_CYCLE()`**	Returns the actual cycle number.

  - **Accelerator access functions.**

`integer` **`ierr =`** **`GET_U_ACCEL(`**`integer` **`iacc`**`, float` **`ax`**`, float` **`ay`**`, float` **`az)`**	Provides access to RADIOSS accelerometer values. **Note:** ax, ay, az are acceleration components. `iacc` is the accelerometer number.
`integer` **`iacc =`** **`GET_U_NUMACC(`**`integer acc_id`**`)`**	Restores accelerometer number from its identifier.

  - **Nodal value access functions.**
    These allow access to node coordinates (x, y, z); node displacement componentss (dx, dy, dz); node velocity components (vx, yx, vx); and node acceleration components (ax, ay, az). All use an internal node number (not a node identifier) as an argument.

    - `integer` **`ierr = GET_U_NOD_X(`**`integer` **`inode`**`, float` **`x`**`, float` **`y`**`, float` **`z)`**
    - `integer` **`ierr = GET_U_NOD_D(`**`integer` **`inode`**`, float` **`dx`**`, float` **`dy`**`, float` **`dz)`**
    - `integer` **`ierr = GET_U_NOD_V(`**`integer` **`inode`**`, float` **`vx`**`, float` **`vy`**`, float` **`vz)`**
    - `integer` **`ierr = GET_U_NOD_A(`**`integer` **`inode`**`, float` **`ax`**`, float` **`ay`**`, float` **`az)`**

  Additionally, the function below restores an internal number `inode` from a node identifier `node_id`. Call this function once and store the result in a local variable.

    - `integer` **`inode = GET_U_NUMNOD(node_id)`**

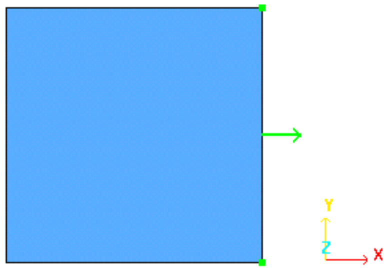
5.4 Example of a User Sensor

A sensor stores a maximum Von Mises stress value (F).

CR =1 sensor parameter (read in Starter by sensor user's subroutine LECSEN_USRm)
F = 1 sensor variable (written to sensor buffer by material user's subroutine SIGESPnn)

Sensor is activated in F > CR.
The activation time of the sensor is displayed.

Mesh example (4-node-shell element)
Loading case: imposed velocities applied on two nodes.



5.4.1 User's Input Data (/SENSOR/USERm_option)

```
[…]
#---1----|----2----|----3----|----4----|----5----|----6----|----7----|----8----|----9----|---10----|
#-  4.  MATERIALS:
#---1----|----2----|----3----|----4----|----5----|----6----|----7----|----8----|----9----|---10----|
/MAT/USER1/1
User's elastic material law
#        Init. dens.          Ref. dens.
            0.00785                    0
#                 E                   Nu
            3102.75                  0.3
#---1----|----2----|----3----|----4----|----5----|----6----|----7----|----8----|----9----|---10----|
#- 15. SENSOR:
#---1----|----2----|----3----|----4----|----5----|----6----|----7----|----8----|----9----|---10----|
/SENSOR/USER1/1
User's sensor 1
#             Tdelay
                 0.0
#                    CR
                 5.1
#---1----|----2----|----3----|----4----|----5----|----6----|----7----|----8----|----9----|---10----|
 […]
```

5.4.2 Starter Sensor User's Subroutine LECSEN_USR1

```
C=========================================================
C  This subroutine reads the user sensor parameters
C=========================================================
      SUBROUTINE LECSEN_USR1(IIN,IOUT)
C
C---------------------------------------------------------
C   D u m m y   A r g u m e n t s
C---------------------------------------------------------
      INTEGER IIN, IOUT
      INTEGER SET_U_SENS_IPAR
      DOUBLE PRECISION SET_U_SENS_FPAR
      EXTERNAL SET_U_SENS_IPAR,SET_U_SENS_FPAR
C
C---------------------------------------------------------
C   L o c a l   V a r i a b l e s
C---------------------------------------------------------
      INTEGER IERROR
      DOUBLE PRECISION CR
C
C---------------------------------------------------------
C   R e a d  U s e r ' s  P a r a m e t e r s
```

```
C-----------------------------------------------------------
      READ(IIN,ERR=999,FMT='(F20.0)') CR
C
C-----------------------------------------------------------
C   W r i t e  in  S e n s o r  B u f f e r
C-----------------------------------------------------------
C     Write float sensor parameter CR in sensor buffer 1:
      IERROR = SET_U_SENS_FPAR(1,CR)
C
C-----------------------------------------------------------
C   O u t p u t  P r i n t
C-----------------------------------------------------------
      WRITE(IOUT,1000) CR
C
      RETURN
C
 999  CONTINUE
        WRITE(IOUT,*)' **ERROR IN USER SENSOR INPUT'
C
 1000 FORMAT(/' CHARGE CRITIQUE. . . . . . . . . . . ',E12.4/)
C
C-----------------------------------------------------------
      RETURN
      END
```

5.4.3 Starter Material User's Subroutine LECm29

```
C===========================================================
C     This subroutine read the user material parameters.
C===========================================================
      SUBROUTINE LECM29(IIN  ,IOUT ,UPARAM ,MAXUPARAM,NUPARAM,
     .                  NUVAR,IFUNC,MAXFUNC,NFUNC    ,PARMAT )
C
C-----------------------------------------------
C   I m p l i c i t   T y p e s
C-----------------------------------------------
      IMPLICIT NONE
C-----------------------------------------------
C   D u m m y   A r g u m e n t s
C-----------------------------------------------
      INTEGER IIN,IOUT,MAXUPARAM,NUPARAM,NUVAR,MAXFUNC,NFUNC,
     .        IFUNC(MAXFUNC)
      DOUBLE PRECISION   UPARAM(MAXUPARAM),PARMAT(*)
C-----------------------------------------------
C   L o c a l   V a r i a b l e s
C-----------------------------------------------
      DOUBLE PRECISION E,NU,A11,A12,A44
C
C-----------------------------------------------
C     INPUT FILE READING (USER DATA)
C-----------------------------------------------
      READ(IIN,'(2F20.0)')E,NU
      A11 = E * (1.-NU) / (1.+NU) / (1.-2.*NU)
      A12 = E * NU / (1.+NU) / (1.-2.*NU)
      A44 = E / 2. / (1.+NU)
C
C-----------------------------------------------
C     DATA CHECKING
C-----------------------------------------------
      IF(NU.LT.0.0.OR.NU.GE.0.5)THEN
        WRITE(IOUT,*)' ** ERROR : WRONG NU VALUE'
      ENDIF
      NUPARAM = 6
      IF(NUPARAM.GT.MAXUPARAM)THEN
```

```
          WRITE(IOUT,*)' ** ERROR : NUPARAM GT MAXUPARAM'
          WRITE(IOUT,*)'      NUPARAM =',NUPARAM,
     .                 ' MAXUPARAM =',MAXUPARAM
        ELSE
C-----------------------------------------------
C     USER MATERIAL PARAMETERS DEFINITION
C-----------------------------------------------
C used in sigeps29c (shell 3n,4n)
          UPARAM(1) = E/(1.-NU*NU)
          UPARAM(2) = NU*E/(1.-NU*NU)
          UPARAM(3) = 0.5*E/(1.+NU)
        ENDIF
C
C-------------------------------------------------
C     NUMBER OF USER ELEMENT VARIABLES AND CURVES
C-------------------------------------------------
        NUVAR = 0
        NFUNC = 0
C
C-----------------------------------------------
C     USED FOR SHELLS
C-----------------------------------------------
        PARMAT(2) = E
        PARMAT(3) = NU
C
C-------------------------------------------------
C     OUTPUT FILE PRINT
C-------------------------------------------------
        WRITE(IOUT,1000)
        WRITE(IOUT,1100)E,NU
C
 1000 FORMAT(
     & 5X,'  ELASTIC USER LAW 29',/,
     & 5X,'  ----------------- ',//)
 1100 FORMAT(
     & 5X,'E . . . . . . . . . . . . . . . . . . .=',E12.4/
     & 5X,'NU. . . . . . . . . . . . . . . . . . .=',E12.4//)
C
C-------------------------------------------------
        RETURN
        END
```

5.4.4 Engine Sensor User's Subroutine USER_SENSm

```
C==========================================================
C  This subroutine activates sensor
C==========================================================
      SUBROUTINE USER_SENS1(ID)
C
C-----------------------------------------------------------
C   L o c a l   V a r i a b l e s
C-----------------------------------------------------------
      INTEGER IERR,NSENS
      DOUBLE PRECISION CR,F,DTIME,TIME,ACTI
C
C-----------------------------------------------------------
C   R e s t o r e  F u n c t i o n s
C-----------------------------------------------------------
C     Get the sensor number:
      NSENS = GET_U_NUMSENS(ID)
C
C     Retrieve a float sensor parameter from sensor buffer 1:
      IERR = GET_U_SENS_FPAR(NSENS,1,CR)
C
C     Read material variable F (see SIGEPS29) from sensor buffer 1:
      IERR = GET_U_SENS_VALUE(NSENS,1,F)
C
C-----------------------------------------------------------
C   S e n s o r   A c t i v a t i o n
C-----------------------------------------------------------
      IF (F.GT.CR) THEN
C     Activate sensor number nsens:
      IERR = SET_U_SENS_ACTI(NSENS)
C     Time delay since the first activation of sensor number nsens:
      DTIME = GET_U_SENS_ACTI(NSENS)
C     Current time:
      TIME = GET_U_TIME()
C     Activation time:
      ACTI = TIME - DTIME
C
C-----------------------------------------------------------
C   C h e c k   S e n s o r   S t a t e
C-----------------------------------------------------------
        IF (TIME.EQ.ACTI) THEN
      print *,'    --> Activation time of sensor id',NSENS,'=',ACTI
      print *,'    [Critical Von Mises value reached =',F,'MPa]'
        ENDIF
C
C-----------------------------------------------------------
      ENDIF
C-----------------------------------------------------------
      RETURN
      END
```

5.4.5 Engine Material User's Subroutine SIGEP29C for Shell Elements

```
C===========================================================
C     This subroutine computes elastic stresses
C     A variable is stored in sensor buffer
C===========================================================
      SUBROUTINE SIGEPS29C(
     1     NEL    ,NUPARAM,NUVAR   ,NFUNC   ,IFUNC  ,
     2     NPF    ,NPT    ,IPT     ,IFLAG   ,
     2     TF     ,TIME   ,TIMESTEP,UPARAM  ,RHO0   ,
     3     AREA   ,EINT   ,THKLY   ,
     4     EPSPXX ,EPSPYY ,EPSPXY  ,EPSPYZ  ,EPSPZX ,
     5     DEPSXX ,DEPSYY ,DEPSXY  ,DEPSYZ  ,DEPSZX ,
     6     EPSXX  ,EPSYY  ,EPSXY   ,EPSYZ   ,EPSZX  ,
     7     SIGOXX ,SIGOYY ,SIGOXY  ,SIGOYZ  ,SIGOZX ,
     8     SIGNXX ,SIGNYY ,SIGNXY  ,SIGNYZ  ,SIGNZX ,
     9     SIGVXX ,SIGVYY ,SIGVXY  ,SIGVYZ  ,SIGVZX ,
     A     SOUNDSP,VISCMAX,THK     ,PLA     ,UVAR   ,
     B     OFF    ,NGL    ,SHF)
C
C-----------------------------------------------
C   I m p l i c i t   T y p e s
C-----------------------------------------------
           IMPLICIT NONE
C-----------------------------------------------
C   I N P U T   A r g u m e n t s
C-----------------------------------------------
C
      INTEGER NEL, NUPARAM, NUVAR, NPT, IPT,IFLAG(*),
     .   NGL(NEL)
      DOUBLE PRECISION TIME,TIMESTEP,UPARAM(NUPARAM),
     .   AREA(NEL),RHO0(NEL),EINT(2,NEL),
     .   THKLY(NEL),PLA(NEL),SHF(NEL),
     .   EPSPXX(NEL),EPSPYY(NEL),
     .   EPSPXY(NEL),EPSPYZ(NEL),EPSPZX(NEL),
     .   DEPSXX(NEL),DEPSYY(NEL),
     .   DEPSXY(NEL),DEPSYZ(NEL),DEPSZX(NEL),
     .   EPSXX(NEL) ,EPSYY(NEL) ,
     .   EPSXY(NEL) ,EPSYZ(NEL) ,EPSZX(NEL) ,
     .   SIGOXX(NEL),SIGOYY(NEL),
     .   SIGOXY(NEL),SIGOYZ(NEL),SIGOZX(NEL)
C-----------------------------------------------
C   VARIABLES FOR FUNCTION INTERPOLATION
C-----------------------------------------------
      INTEGER NPF(*), NFUNC, IFUNC(NFUNC)
      DOUBLE PRECISION FINTER ,TF(*)
      EXTERNAL FINTER
C        Y = FINTER(IFUNC(J),X,NPF,TF,DYDX)
C        Y       : y = f(x)
C        X       : x
C        DYDX    : f'(x) = dy/dx
C        IFUNC(J): FUNCTION INDEX
C              J : FIRST(J=1), SECOND(J=2) ..
C        NPF,TF  : FUNCTION PARAMETER
C-----------------------------------------------
```

```
C   O U T P U T   A r g u m e n t s
C-----------------------------------------------
      DOUBLE PRECISION
     .    SIGNXX(NEL),SIGNYY(NEL),
     .    SIGNXY(NEL),SIGNYZ(NEL),SIGNZX(NEL),
     .    SIGVXX(NEL),SIGVYY(NEL),
     .    SIGVXY(NEL),SIGVYZ(NEL),SIGVZX(NEL),
     .    SOUNDSP(NEL),VISCMAX(NEL)
C-----------------------------------------------
C   I N P U T   O U T P U T   A r g u m e n t s
C-----------------------------------------------
      DOUBLE PRECISION UVAR(NEL,NUVAR), OFF(NEL),THK(NEL)
C
C-----------------------------------------------------------
C   F u n c t i o n   F o r   U s e r ' s   S e n s o r
C-----------------------------------------------------------
      INTEGER ID
      INTEGER GET_U_NUMSENS
      DOUBLE PRECISION SET_U_SENS_VALUE
      EXTERNAL SET_U_SENS_VALUE,GET_U_NUMSENS
C
C-----------------------------------------------
C   L o c a l   V a r i a b l e s
C-----------------------------------------------
      INTEGER I,IERR
      DOUBLE PRECISION A1,A2,G,F
C
C-----------------------------------------------
C     ELASTIC SOLUTION
C-----------------------------------------------
      DO I=1,NEL
C
        A1  = UPARAM(1)
        A2  = UPARAM(2)
        G   = UPARAM(3)
C
        SIGNXX(I)=SIGOXX(I)+A1*DEPSXX(I)+A2*DEPSYY(I)
        SIGNYY(I)=SIGOYY(I)+A2*DEPSXX(I)+A1*DEPSYY(I)
        SIGNXY(I)=SIGOXY(I)+G *DEPSXY(I)
        SIGNYZ(I)=SIGOYZ(I)+G *DEPSYZ(I)
        SIGNZX(I)=SIGOZX(I)+G *DEPSZX(I)
C
        SOUNDSP(I) = SQRT(A1/RHO0(I))
        VISCMAX(I) = 0.
C
C      Von Mises stress (used in user's subroutine USER_SEN1):
        F = SQRT(SIGNXX(I)*SIGNXX(I)
     .     +SIGNYY(I)*SIGNYY(I)
     .     -SIGNXX(I)*SIGNYY(I)
     .     +3.*SIGNXY(I)*SIGNXY(I))
C
C-----------------------------------------------
C     S t o r e   i n   S e n s o r   B u f f e r
C-----------------------------------------------
C       Write the variable F in the sensor buffer number 1:
```

```
          IERR = SET_U_SENS_VALUE(1,1,F)
C
        ENDDO
C-----------------------------------------------
        RETURN
        END
```