APPENDIX F: User Defined Interface Control

This subroutine may be provided by the user to turn the interfaces on and off. This option is activated by the *USER_INTERFACE_CONTROL keyword. The arguments are defined in the listing provided below.

```
      subroutine uctrl2(nsi,nty,time,cycle,nsb,nsbn,nsa,nsan,
     1 thsb,thsa,vt,xi,ut,iskip,idrint,numnp,dt2,ninput,ua,
     2 irectsb,nrtsb,irectsa,nrtsa)
c
c*****************************************************************
c|  Livermore Software Technology Corporation  (LSTC)            |
c|  ------------------------------------------------------------ |
c|  Copyright 1987-2008 Livermore Software Tech.  Corp           |
c|  All rights reserved                                          |
c*****************************************************************
c
c     user subroutine for interface control
c
c     note:  ls-dyna uses an internal numbering system to
c            accommodate arbitrary node numbering.  To access
c            information for user node n, address array location m,
c            m = lqf8(n,1).  To obtain user node number, n,
c            corresponding to array address m, set n = lqfinv(m,1)
c
c     arguments:
c          nsi           = number of sliding interface
c          nty           = interface type.
c                            .eq.4:single surface
c                            .ne.4:surface to surface
c          time          = current solution time
c          cycle         = cycle number
c          nsb(nsbn)     = list of surfb nodes numbers in internal
c                            numbering scheme
c          nsbn          = number of surfb nodes
c          nsa(nsan)     = list of surfa nodes numbers in internal
c                            numbering scheme
c          nsan          = number of surfa nodes
c          thsb(nsbn)    = surfb node thickness
c          thsa(nsan)    = surfa node thickness
c          vt(3,numnp)   = nodal translational velocity vector
c          xi(3,numnp)   = initial coordinates at time = 0
c          ut(3,numnp)   = nodal translational displacement vector
c          idrint        = flag for dynamic relaxation phase
c                            .ne.0: dynamic relaxation in progress
c                            .eq.0: solution phase
c          numnp         = number of nodal points
c          dt2           = time step size at n+1/2
c          ninput        = number of variables input into ua
c          ua(*)         = users' array, first ninput locations
c                            defined by user.  the length of this
c                            array is defined on control card 10.
c                            this array is unique to interface nsi.
c          irectsb(4,*)  = list of surfb segments in internal
```

```
c                         numbering scheme
c          nrtsb        = number of surfb segments
c          irectsa(4,*) = list of surfa segments in internal
c                         numbering scheme
c          nrtsa        = number of surfa segments
c
c     set flag for active contact
c       iskip = 0 active
c       iskip = 1 inactive
c
c*******************************************************************
c
      integer cycle
      real*8 ut
      real*8 xi
      dimension nsb(*),nsa(*),thsb(*),thsa(*),vt(3,*),xi(3,*),
     .          ut(3,*),ua(*),irectsb(4,*),irectsa(4,*)
c
c     The following sample code is provided to illustrate how
c     this subroutine might be used.  Here we check to see if the
c     surfaces in the surface to surface contact are separated.  If
c     so, iskip = 1, and the contact treatment is skipped.
c
c     if (nty.eq.4) return
c     dt2hlf = dt2/2.
c     xminsa = 1.e+16
c     xmaxsa = -xminsa
c     yminsa = 1.e+16
c     ymaxsa = -yminsa
c     zminsa = 1.e+16
c     zmaxsa = -zminsa
c     xminsb = 1.e+16
c     xmaxsb = -xminsb
c     yminsb = 1.e+16
c     ymaxsb = -yminsb
c     zminsb = 1.e+16
c     zmaxsb = -zminsb
c     thksa = 0.0
c     thksb = 0.0
c     do 10 i = 1,nsan
c     dsp1 = ut(1,nsa(i))+dt2hlf*vt(1,nsa(i))
c     dsp2 = ut(2,nsa(i))+dt2hlf*vt(2,nsa(i))
c     dsp3 = ut(3,nsa(i))+dt2hlf*vt(3,nsa(i))
c     x1 = xi(1,nsa(i))+dsp1
c     x2 = xi(2,nsa(i))+dsp2
c     x3 = xi(3,nsa(i))+dsp3
c     thksa = max(thsa(i),thksa)
c     xminsa = min(xminsa,x1)
c     xmaxsa = max(xmaxsa,x1)
c     yminsa = min(yminsa,x2)
c     ymaxsa = max(ymaxsa,x2)
c     zminsa = min(zminsa,x3)
c     zmaxsa = max(zmaxsa,x3)
c  10 continue
c     do 20 i = 1,nsbn
c     dsp1 = ut(1,nsb(i))+dt2hlf*vt(1,nsb(i))
c     dsp2 = ut(2,nsb(i))+dt2hlf*vt(2,nsb(i))
c     dsp3 = ut(3,nsb(i))+dt2hlf*vt(3,nsb(i))
c     x1 = xi(1,nsb(i))+dsp1
c     x2 = xi(2,nsb(i))+dsp2
```

```
c      x3 = xi(3,nsb(i))+dsp3
c      thksb = max(thsb(i),thksb)
c      xminsb = min(xminsb,x1)
c      xmaxsb = max(xmaxsb,x1)
c      yminsb = min(yminsb,x2)
c      ymaxsb = max(ymaxsb,x2)
c      zminsb = min(zminsb,x3)
c      zmaxsb = max(zmaxsb,x3)
c   20 continue
c
c      If thksa or thksb equal zero, set them to some reasonable value.
c
c      if (thksa.eq.0.0) then
c        e1=(xi(1,irectsa(1,1))-xi(1,irectsa(3,1)))**2
c     .    +(xi(2,irectsa(1,1))-xi(2,irectsa(3,1)))**2
c     .    +(xi(3,irectsa(1,1))-xi(3,irectsa(3,1)))**2
c        e2=(xi(1,irectsa(2,1))-xi(1,irectsa(4,1)))**2
c     .    +(xi(2,irectsa(2,1))-xi(2,irectsa(4,1)))**2
c     .    +(xi(3,irectsa(2,1))-xi(3,irectsa(4,1)))**2
c        thksa=.3*sqrt(max(e1,e2))
c      endif
c      if (thksb.eq.0.0) then
c        e1=(xi(1,irectsb(1,1))-xi(1,irectsb(3,1)))**2
c     .    +(xi(2,irectsb(1,1))-xi(2,irectsb(3,1)))**2
c     .    +(xi(3,irectsb(1,1))-xi(3,irectsb(3,1)))**2
c        e2=(xi(1,irectsb(2,1))-xi(1,irectsb(4,1)))**2
c     .    +(xi(2,irectsb(2,1))-xi(2,irectsb(4,1)))**2
c     .    +(xi(3,irectsb(2,1))-xi(3,irectsb(4,1)))**2
c        thksb=.3*sqrt(max(e1,e2))
c      endif
c
c      if (xmaxsa+thksa.lt.xminsb-thksb) go to 40
c      if (ymaxsa+thksa.lt.yminsb-thksb) go to 40
c      if (zmaxsa+thksa.lt.zminsb-thksb) go to 40
c      if (xmaxsb+thksb.lt.xminsa-thksa) go to 40
c      if (ymaxsb+thksb.lt.yminsa-thksa) go to 40
c      if (zmaxsb+thksb.lt.zminsa-thksa) go to 40
c      iskip = 0
c
c      return
c   40 iskip = 1
c
       return
       end
```