*CONSTRAINED_BEAM_IN_SOLID_{OPTION1}_{OPTION2}

This keyword could take the following two forms:

*CONSTRAINED_BEAM_IN_SOLID

*CONSTRAINED_BEAM_IN_SOLID_PENALTY

To define a CONSTRAINED_BEAM_IN_SOLID ID and heading the following options are available:

ID

TITLE

Purpose: This keyword provides either constraint-based or penalty-based coupling between beams embedded in solid or thick shell elements. For shells embedded in solid or thick shell elements; see *CONSTRAINED_SHELL_IN_SOLID.

*CONSTRAINED_BEAM_IN_SOLID invokes constraint-based coupling. It constrains beam structures to move with Lagrangian solids/tshells, which serve as the master component. Both acceleration and velocity are constrained. This feature is intended to sidestep certain limitations in the CTYPE = 2 implementation of *CONSTRAINED_LAGRANGE_IN_SOLID. Notable features of this keyword include:

1. **CDIR = 1 Feature.** With the CDIR = 1 option, coupling occurs only in the normal directions. This coupling allows for releasing the constraints along the beam axial direction.

2. **Axial Coupling Force.** Debonding processes can be modeled with a user defined function or user provided subroutine giving the axial shear force based on the slip between rebar nodes and concrete solid elements. This feature is invoked by setting the AXFOR flag to a negative integer which refers to the *DEFINE_FUNCTION ID or a number greater than 1000. In the latter case, the user must modify the subroutine rebar_bondslip_get_getforce() in dyn21.F to add in one or more debonding laws, each tagged with a "lawid." An AXFOR value greater than 1000 will call the user subroutine and pass AXFOR in as "lawid." CDIR must be set to 1 in this case to release the axial constraints.

3. **NCOUP Feature.** NCOUP is used to invoke coupling not only at beam nodes but also at coupling points between the two beam nodes of each beam element. This implementation resolves the errors in energy balance observed when using the alternative implementation, *CONSTRAINED_LAGRANGE_IN_SOLID, CTYPE = 2.

4. **Tetrahedral and Pentahedral Solid Elements.** These element shapes are supported and are treated as degenerated hexahedra in the *CONSTRAINED_LAGRANGE_IN_SOLID CTYPE = 2 implementation.

5. **Constraints on Nodes.** The *CONSTRAINED_LAGRANGE_IN_SOLID CTYPE = 2 implementation does not constrain beam nodes embedded in elements whose nodes have prescribed motion or other constraints.

6. **R-Adaptivity Support.** Severely deformed solid elements could terminate the simulation prematurely. One way to address this problem is to use LS-DYNA's 3D tetrahedral r-adaptivity method to stop, remesh, and restart. This keyword supports r-adaptivity and can be used to model manufacturing processes with severe deformations, such as compression molding with embedded fibers.

7. **Implicit Support.** Both SMP and MPP are supported.

8. **Optimized Sorting.** The sorting subroutine is optimized for larger problems to achieve better performance with less memory usage.

9. **Thermal Support.** In case the implicit thermal solver is invoked to perform either a thermal analysis (SOLN = 1 in *CONTROL_SOLUTION) or a coupled structural thermal analysis (SOLN = 2 in *CONTROL_SOLUTION), the temperature field is also constrained between beam and solid nodes.

*CONSTRAINED_BEAM_IN_SOLID_PENALTY invokes penalty-based coupling. A penalty spring is attached between coupling points on the beam and in the solid/tshell element. Penalty spring stiffness is calculated based on the geometric mean of beam and solid's bulk modulus. The magnitude of this coupling force can be controlled through field PSSF (penalty spring stiffness scale factor). This penalty coupling conserves kinetic energy much better in transient problems, such as blast loading. Please note, the CDIR and AXFOR fields only work with constraint-based coupling and are not available with penalty-based coupling.

If COUPID is not defined on the Title Card, LS-DYNA will automatically create an ID value for the coupling.

Title Card. Additional card for TITLE and ID keyword options.

Title	1	2	3	4	5	6	7	8
Variable	COUPID	TITLE						
Type	I	A70						

Card 1	1	2	3	4	5	6	7	8
Variable	SLAVE	MASTER	SSTYP	MSTYP			NCOUP	CDIR
Type	I	I	I	I			I	I
Default	none	none	0	0			0	0

Card 2	1	2	3	4	5	6	7	8
Variable	START	END		AXFOR		PSSF		XINT
Type	F	F		I		F		F
Default	0	10^{10}		0		0.1		10^{16}

VARIABLE	DESCRIPTION
COUPID	Coupling (card) ID number (I10). If not defined, LS-DYNA will assign an internal coupling ID based on the order of appearance in the input deck.
TITLE	A description of this coupling definition.
SLAVE	Slave set ID defining a part or part set ID of the Lagrangian beam structure (see *PART, *SET_PART).
MASTER	Master set ID defining a part or part set ID of the Lagrangian solid elements or thick shell elements (see *PART or *SET_PART). See Remark 1.
SSTYP	Slave set type of SLAVE: EQ.0: part set ID (PSID). EQ.1: part ID (PID).
MSTYP	Master set type of MASTER: EQ.0: part set ID (PSID). EQ.1: part ID (PID).

VARIABLE	DESCRIPTION
NCOUP	Number of coupling points generated in one beam element. If set to 0, coupling only happens at beam nodes. Otherwise, coupling is done at both the beam nodes and those automatically generated coupling points.
CDIR	Coupling direction. Only available in constraint form. EQ.0: Constraint applied along all directions. EQ.1: Constraint only applied along normal directions; along the beam axial direction there is no constraint.
START	Start time for coupling.
END	End time for coupling.
AXFOR	ID of a user defined function that calculates the coupling force in the beam axial direction. Only available in constraint form. GE.0: Off LT.0: |AXFOR| is a function ID; see *DEFINE_FUNCTION. GT.1000: Debonding law ID, "lawid," in the user defined subroutine rebar_bondslip_get_force().
PSSF	Penalty spring stiffness scale factor. Only available in penalty form.
XINT	Interval distance. This field is designed to deal with beam elements having a wide variation in lengths. Coupling points are generated at an interval of length equal to XINT. Hence the number of coupling points in a beam element is no longer a fixed number (NCOUP), but rather variable, depending on the length of the beam element. This field can be used together with NCOUP. In that case, in each element, we will take the larger number of coupling points from these two options.

Remarks:

1. **Rigid Body.** The solid part cannot be defined as rigid body. If the solid part shares nodes with rigid bodies and constraint-based coupling is used, make sure rebars are not buried in elements which contain rigid body nodes. This is because constraint-based coupling distribute/collet momentum/force between rebar and solid nodes.

Examples:

1. The example below shows how to define a function and use it to prescribe the debonding process. User defined functions are supported. The function computes the debonding force and has two internally calculated arguments: slip and leng. Slip is the relative axial displacement between the beam node (or coupling point) and the material in which the beam is embedded. Leng is the tributary length of the beam node or coupling point. Implicit calculations require a third argument which is output by the function: stiff. Stiff is the debonding spring stiffness. The asterisk in front of stiff (*stiff) is required to indicate that it is called-by-reference, meaning that its value is returned after the function is evaluated. Please note inside the function body this asterisk could not be placed at column 1 as it would be treated as the start of next keyword by LS-DYNA keyword reader.

```
$...|....1....|....2....|....3....|....4....|....5....|....6....|....7....|....8
*CONSTRAINED_BEAM_IN_SOLID
$#   slave    master     sstyp     mstyp                           ncoup      cdir
         2         1         1         1                               2         1
$#   start       end               axfor
     0.000     0.000                 -10
*DEFINE_FUNCTION
        10
float force(float slip,float leng, float *stiff)
{
  float force,pi,d,area,shear,pf;
  pi=3.1415926;
  d=0.175;
  area=pi*d*leng;
  pf=1.0;
  if (slip<0.25) {
    shear=slip*pf;
  } else {
    shear=0.25*pf;
  }
  force=shear*area;
  *stiff=pf*area;
  return force;
}
```

2. The example below shows how to define a user subroutine and use it to prescribe the debonding process.

```
$...|....1....|....2....|....3....|....4....|....5....|....6....|....7....|....8
*CONSTRAINED_BEAM_IN_SOLID
$#   slave    master     sstyp     mstyp                           ncoup      cdir
         2         1         1         1                               2         1
$#   start       end               axfor
     0.000     0.000                1001
*CONSTRAINED_BEAM_IN_SOLID
$#   slave    master     sstyp     mstyp                           ncoup      cdir
         3         1         1         1                               2         1
$#   start       end               axfor
     0.000     0.000                1002
*USER_LOADING
$    parm1     parm2     parm3     parm4     parm5     parm6     parm7     parm8
       1.0       6.0
$...|....1....|....2....|....3....|....4....|....5....|....6....|....7....|....8
```

The user debonding law subroutine:

```
      subroutine rebar_bondslip_get_force(slip,dl,force,hsv,
     .  userparm,lawid)
      real hsv
      dimension hsv(12),cm(8),userparm(*)
c
```

```
c in this subroutine user defines debonding properties and
c call his own debonding subroutine to get force
      cm(1)=userparm(1)
      cm(2)=userparm(2)
      cm(3)=2.4*(cm(2)/5.0)**0.75
      cm(8)=0.
c
      pi=3.1415926
      d=0.175
      area=pi*0.25*d*d*dl
      pf=1.0
c
      if (lawid.eq.1001) then
        if (slip.lt.0.25) then
          shear=slip*pf
        else
          shear=0.25*pf
        endif
        force=sign(1.0,slip)*shear*area
      elseif (lawid.eq.1002) then
        if (slip.lt.0.125) then
          shear=slip*pf
        else
          shear=0.125*pf
        endif
      endif
      return
      end
```