*MODULE

The keyword *MODULE provides a way to load user compiled libraries at runtime, to support user defined capabilities such as material models, equations of state, element formulations, etc. The following keywords implement this capability:

*MODULE_LOAD

*MODULE_PATH

*MODULE_USE

MODULE_LOAD_*{OPTION}

Available options:

<BLANK>

SP

DP

Purpose: Load a dynamic library for user subroutines. When using the SP / DP keyword option, LS-DYNA only loads the dynamic library when running a single / double precision binary.

Card Sets. Repeat as many sets of data cards as desired (Cards 1 and 2) to load multiple libraries. This input ends at the next keyword ("*") card.

Card 1	1	2	3	4	5	6	7	8
Variable	MDLID		TITLE					
Type	A20		A60					
Default	none		none					

Card 2	1	2	3	4	5	6	7	8
Variable	FILENAME							
Type	A80							
Default	none							

VARIABLE	DESCRIPTION
MDLID	Module identification. A unique string label must be specified.
TITLE	Description of the module
FILENAME	File name of the library to be loaded, 80 characters maximum. If the file name has no path component, LS-DYNA will search in all directories specified in *MODULE_PATH first. If not found and

VARIABLE	DESCRIPTION
	the file name starts with "+" (a plus sign), LS-DYNA will continue to search all directories specified in the system environment variable LD_LIBRARY_PATH.

Remarks:

1. **Library disclaimer.** The MODULE capability described here and in *MODULE_USE is still under development. We consider this capability experimental.

2. **Execution line option.** If loading only one dynamic library that requires no rules (*MODULE_USE), you can use the execution line option "module=*dll*" to specify this dynamic library. If the file name starts with +, LS-DYNA searches for the dynamic library using the system environment variable LD_LIBRARY_PATH. We implemented this execution line option to support the classic user-subroutine subroutines without modifying the input deck.

MODULE_PATH_*{OPTION}

Available options:

<BLANK>

RELATIVE

LS-DYNA's default behavior is to search for modules in the current working directory. If a module file is not found and the filename has no path component, LS-DYNA will search in all directories specified on the cards following a *MODULE_PATH keyword. Multiple paths can be specified using one *MODULE_PATH keyword card, that is,

```
*MODULE_PATH
Directory_path1
Directory_path2
Directory_path3
```

Directory paths are read until the next keyword ("*") card is encountered. A directory path can have up to 236 characters. See Remark 2.

When the RELATIVE option is used, all directories are relative to the location of the input file. For example, if "i=/home/test/problems/input.k" is given on the command line, and the input contains

```
*MODULE_PATH_RELATIVE
lib
../lib
```

then the two directories /home/test/problems/lib and /home/test/lib will be searched for module files.

Remarks:

1. **Input Format.** Filenames and pathnames are limited to 236 characters spread over up to three 80 character lines. When 2 or 3 lines are needed to specify the filename or pathname, end the preceding line with "␣+" (space followed by a plus sign) to signal that a continuation line follows. Note that the "␣+" combination is, itself, part of the 80 character line; hence the maximum number of allowed characters is 78 + 78 + 80 = 236.

*MODULE_USE

Purpose: Define the rules for mapping the user subroutines loaded in dynamic libraries to the model. The rules can be applied to:

*MAT_USER_DEFINED_MATERIAL_MODELS	(MAT 41 - 50)
*MAT_THERMAL_USER_DEFINED	(MAT T11 - T15)
*EOS_USER_DEFINED	(EOS 21 – 30)
*SECTION_SOLID	(ELFORM 101 - 105)
*SECTION_SHELL	(ELFORM 101- 105)
*USER_INTERFACE_FRICTION	
*CONTACT_..._MORTAR_TIED_WELD	(TEMP.LT.0)

as well as other subroutines in the LS-DYNA user-subroutine package.

LS-DYNA requires that subroutines in modules (see *MODULE_LOAD) be named as if they were part of the traditional user-subroutine framework. Each module, however, can contain a complete set of those subroutines. It is, therefore, possible to import from different modules the same subroutines of the same name several times. Each module is, essentially, an independent copy of the traditional user-subroutine framework. This keyword, *MODULE_USE, deals with namespace conflicts by defining how each subroutine in the module is presented to the other keywords.

The rules defined in *MODULE_USE are applied to only one dynamic library.

Card 1	1	2	3	4	5	6	7	8
Variable	MDLID							
Type	A20							
Default	none							

Rule Cards. Card 2 defines the rules for the module specified in Card 1. Include one instance of this card for each subroutine to be mapped. If two rules conflict, new rules override existing rules. Input ends at the next keyword ("*") card.

Card 2	1	2	3	4	5	6	7	8
Variable	TYPE		PARAM1		PARAM2			
Type	A20		A20		A20			
Default	none		blank		blank			

VARIABLE	DESCRIPTION
MDLID	Module identification defined in *MODULE_LOAD
TYPE	Rule type. See TYPE definitions below.
PARAM1	Type-dependent parameter
PARAM2	Type-dependent parameter.

User-Defined Materials:

TYPE	DESCRIPTION
UMAT	Implements the user-material type PARAM1 in the model via a possibly different type PARAM2 in the dynamic library. Types in the extended range from 1001 to 2000 can be used as material types in the model. For example, if PARAM1 = 1001 and PARAM2 = 42, then model material 1001 will use subroutine `umat42` from this library. If PARAM1 is blank and PARAM2 is not, then all user materials in the model will use the indicated subroutine from this library. If PARAM2 is blank and PARAM1 is not, the material type PARAM1 in the model will use the traditional subroutine from this library. For example, if PARAM1 = 43 and PARAM2 is blank, material type 43 in the model will use the subroutine `umat43` from this library.

TYPE	DESCRIPTION
	If PARAM1 and PARAM2 are blank, then all user-defined materials in the model will use the traditional subroutines from this library (41 → `umat41`, 42 → `umat42`, etc.).
MATID	Implements the user material having ID = PARAM1 via a possibly different material type PARAM2 in the dynamic library. PARAM2 may be blank if the material type defined in the model is the same as the one in the dynamic library. PARAM1 can be a numerical ID or label of the material as defined in the model. Materials beyond user material models can be overloaded.

Thermal User-Defined Materials:

TYPE	DESCRIPTION
TUMAT	Implements the user thermal material type PARAM1 in the model (PARAM1 = 11-15) via type PARAM2 in the dynamic library. See type UMAT for the default rules.
TMATID	Implements the user thermal material having ID = PARAM1 via thermal material type PARAM2 in the dynamic library. PARAM2 may be blank if the material type defined in the model is the same as the one in the dynamic library. PARAM1 can be a numerical ID or label of the thermal material as defined in the model. Materials beyond user material models can be overloaded.

User-Defined EOS:

TYPE	DESCRIPTION
UEOS	Implements the user EOS model PARAM1 in the model (PARAM1 = 21-30) via EOS model PARAM2 in the dynamic library. See type UMAT for the default rules.

User-Defined Elements:

TYPE	DESCRIPTION
UELEM	Implements the user element type PARAM1 in the model (PARAM1 = 101-105) via element type PARAM2 in the dynamic library. See type UMAT for the default rules.
SECTIONID	Implements the user element type with section ID = PARAM1 via element type PARAM2 in the dynamic library. PARAM2 may be blank if the element type defined in the model is the same as the one in the dynamic library.

Note: Solid element types are mapped to subroutine `usrsld`, and shell element types are mapped to subroutine `usrshl`. These interfaces are documented in the Fortran user-subroutine files.

Other User Subroutines:

If TYPE = UTIL, the model uses the following subroutines implemented in the dynamic library. LS-DYNA ignores any subroutines with the same name in other dynamic libraries. PARAM1 must be unique in the model, meaning two libraries cannot use the same PARAM1.

PARAM1	DESCRIPTION
UMATFAIL	matusr_24 matusr_103
UFRICTION	usrfrc mortar_usrfrc mortar_usrtie
UWEAR	userwear
UADAP	useradap uadpnorm uadpval
UWELDFAIL	uweldfail uweldfail12 uweldfail22
USPH	hdot

PARAM1	DESCRIPTION
UTHERMAL	usrhcon usrflux
ULOAD	ujntfdrv loadsetud loadud
UELEMFAILCTL	matfailusercontrol matuserfail
UMATFPERT	usermatfpert
UREBAR	rebar_bondslip_get_nhisvar rebar_bondslip_get_force
ULAGPOROUS	lagpor_getab_userdef
UAIRBAG	airusr user_inflator
UALE	al2rfn_criteria5 al2rmv_criteria5 alerfn_criteria5 alermv_criteria5 shlrfn_criteria5 shlrmv_criteria5 sldrfn_criteria5 sldrmv_criteria5 f3dm9ale_userdef1
UCOU- PLE2OTHER	couple2other_boxminmax couple2other_comm couple2other_dt couple2other_extra couple2other_getf couple2other_givex couple2other_reader chkusercomm usercomm usercomm1

PARAM1	DESCRIPTION
UOUTPUT	ushout
UEM	*EM user subroutines
UICFD	*ICFD user subroutines
UCESE	*CESE user subroutines
UDUALCESE	*DUALCESE user subroutines
UCTRL	uctrl1 uctrl2
UTIME	utimestep

Remarks:

1. **Omitting *MODULE_USE.** To simplify the development of user-defined materials via modules, the *MODULE_USE keyword can be omitted in one case. If only a single module library will be used and no remapping of material types is desired, then a *MODULE_LOAD keyword may appear with a single library and no other *MODULE keywords. In this case, all the user-defined subroutines in this library will be used in the usual way. For example, if the library contains `umat41` and `umat43`, those routines will be used for all materials of type 41 and 43, respectively. The *MODULE_USE keyword describing this library may be omitted.

EXAMPLES:

The following examples demonstrate the input for the MODULE keywords:

*MODULE_PATH

*MODULE_LOAD

*MODULE_USE

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
$  Using *MODULE_PATH to define additional directories where
$  dynamic libraries are saved.
$
$------------------------------------------------------------------------------
$
*MODULE_PATH
/home/lsdynauser/lib
$
```

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
$  Using *MODULE_LOAD to load all dynamic libraries for this model.
$
$  three libraries are loaded here for demonstration:
$
$  M_LIB: my library, which is under development.  A debug version is built
$         in my local directory.
$         it contains: UMAT41, UMAT42, and UMAT45
$
$  LIB_A: a hypoelastic model, provided by a third party, for shell & solid.
$          it contains UMAT41 only, with an optional FLUID
$
$  LIB_B: contains two material models provided by another company, with
$         UMAT41: a elasto-plastic model
$         UMAT45: a hyper-elastic model for rubber
$
$-------------------------------------------------------------------------------
*MODULE_LOAD
$...>....1....>....2....>....3....>....4....>....5....>....6....>....7....>....8
M_LIB               My library
/my_development_path/mylib_r123_dbg.so
LIB_A               library from company A
Lib_hypoelastic.so
LIB_B               library from company B
Lib_plastic.so
$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$
$  Using *MODULE_USE to map to user subroutines
$
$  CASE 1:
$
$  M_LIB is used for UMAT41,UMAT42,UMAT45 in the model, as default
$  UMAT41 in LIB_A is used for MT=1001 in the model for shell, and MT=1002 for solid
$  UMAT45 in LIB_B is used for MATID=202, which also happens to hvae MT=1002
$
$-------------------------------------------------------------------------------
*MODULE_USE
M_LIB
$...>....1....>....2....>....3....>....4....>....5....>....6....>....7....>....8
UMAT

*MODULE_USE
LIB_A
$...>....1....>....2....>....3....>....4....>....5....>....6....>....7....>....8
UMAT                1001                41
UMAT                1002                41
*MODULE_USE
LIB_B
$...>....1....>....2....>....3....>....4....>....5....>....6....>....7....>....8
MATID               202                 45
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$  CASE 2:
$
$  M_LIB is used for UMAT41,UMAT42 as,
$         MATID=101 with MT=41
$         MATID=102 with MT=42
$  UMAT45 in LIB_B is used with different material properties, as,
$         MATID=201 with MT=1001
$         MATID=202 with MT=1002
$         MATID=203 with MT=1003
$
*MODULE_USE
LIB_B
$...>....1....>....2....>....3....>....4....>....5....>....6....>....7....>....8
```

```
UMAT                                    45
*MODULE_USE
M_LIB
$...>....1....>....2....>....3....>....4....>....5....>....6....>....7....>....8
MATID               101
MATID               102
```