*PARAMETER

The *PARAMETER family of commands assign numerical values or expressions to named parameters. The parameter names can be used subsequently in the input in place of numerical values.

PARAMETER_*OPTION

*PARAMETER_DUPLICATION

*PARAMETER_EXPRESSION

*PARAMETER_TYPE

PARAMETER_*{OPTION}_{OPTION}_{OPTION}

The available options are

<BLANK>

LOCAL (see Remark 5)

MUTABLE (see Remark 6)

NOECHO (see Remark 7)

The keyword name may include any combination of the above keyword options in any order.

Purpose: Define the numerical values of parameter names referenced throughout the input file. The parameter definitions, if used, should be placed at the beginning of the input file following *KEYWORD or at the beginning of an include file if the LOCAL option is specified.

Parameter Cards. Include as many cards as necessary.

Card 1	1	2	3	4	5	6	7	8
Variable	PRMR1	VAL1	PRMR2	VAL2	PRMR3	VAL3	PRMR4	VAL4
Type	A	I, F or C	A	I, F or C	A	I, F or C	A	I, F or C
Default	none	none	none	none	none	none	none	none

VARIABLE	DESCRIPTION
PRMR*n*	PRMR*n* sets both the *n*th parameter and its storage type. $$\mathrm{PRMR} = \mathtt{T}\underbrace{\mathtt{xxxxxxxxx}}_{\text{9 character name}}$$ The first character, "T", is decoded as follows: T.EQ."R": Parameter is a real number; T.EQ."I": Parameter is an integer; T.EQ."C": Parameter is a character. The remaining 9 characters specify the name of the parameter. A parameter name of "time" (case insensitive) is disallowed. Appendix U lists reserved names. Parameter names can only include numbers, letters, and "_". The first character in a name cannot be

VARIABLE	DESCRIPTION
	a number. For example, to define a shell thickness named, "SHLTHK", the input "RSHLTHK", "R␣␣␣SHLTHK", or "R␣␣SHLTHK␣" are all equivalent 10 character strings ("␣" is space). For instructions regarding how to use the variable "SHLTHK," see Remark 1.
VAL*n*	Define the value of the n^{th} parameter as either a real or integer number, or a character string consistent with preceding definition for PRMR*n*.

Remarks:

1. **Syntax for using parameters.** Parameters can be referenced anywhere in the input by placing an "&" immediately preceding the parameter name. If a minus sign "-" is placed directly before "&", i.e., "-&", with no space the sign of the numerical value will be switched.

2. **Including character parameters in larger strings.** A character parameter can be included as part of a larger string. The included character parameter should start with '&' and end with a delimiter. The default delimiter is '^'. The delimiter will be substituted with a replacement string in the resultant larger string. The default replacement string is '_'. For example, a string definition of "&TORSO^lower arm" will result in "P101_lower arm" if parameter "TORSO" is defined as "C" type with a value of "P101".

   To define a customer delimiter and replacement string, use the special character parameter "CPINLSDNR" with the delimiter and replacement separated by '/'. For instance, the following definition includes the file "/home/PARAM/test.inc":

```
*PARAMETER
CCPINLSDNR      ^^//CINCDIR   PARAM
*INCLUDE
file_&TST^.k
/home/&incdir^^test.inc
```

   If no '/' is found in the definition of CPINLSDNR, the delimiter in the large string is skipped, and no string substitution is applied. Taking the above case as an example, instead of "^^//", if CPINLSDNR is defined as "^^", the file to be included becomes /home/PARAMtest.inc. Only one CPINLSDNR will be used. If more than one is specified in input deck, only the last one is used.

Note that prior to R12.1, specifying an include file name with a character parameter in a larger string did not work properly. LS-DYNA added erroneous spaces to the name. For instance,

```
*PARAMETER
C TST             01
*INCLUDE
file_&TST^.k
```

led to an error termination in earlier versions since LS-DYNA interpreted the name as “file_01␣␣.k” (“␣” is a space).

3. **Comma-delimited format for character parameters**. Note that prior to R12.1, the comma-delimited input format for character parameters did not work correctly. You should use the fixed format for prior LS-DYNA versions.

4. **Reserved variable names.** See Appendix U for a list of variable names reserved internally by LS-DYNA.

5. **LOCAL option.** *PARAMETER_LOCAL behaves like the *PARAMETER keyword with one difference. A parameter defined by *PARAMETER without the LOCAL option is visible and available at any later point in the input processing. Parameters defined with the LOCAL versions disappear when the input parser finishes reading the file in which they appear. LOCAL variables can temporarily mask non-LOCAL variables.

   For example, suppose you have the following input files:

   main.k:
```
*PARAMETER
R  VAL1    1.0
*PARAMETER
R  VAL2    2.0
*PARAMETER
R  VAL3    3.0
*CONTROL_TERMINATION
   &VAL1
*INCLUDE
file1
```

   file1:
```
*PARAMETER
R  VAL1    10.0
*PARAMETER_LOCAL
R  VAL2    20.0
*PARAMETER_LOCAL
R  VAL4    40.0
*INCLUDE
```

```
file2
⋮
```

Then, inside file2 we will see VAL1 = 10.0, VAL2 = 20.0, VAL3 = 3.0 and VAL4 = 40.0. In main.k, after returning from file1, we will see VAL1 = 10.0, VAL2 = 2.0, and VAL3 = 3.0. VAL4 will not exist. This allows for include files that can set all their own parameters without clobbering the parameters in the rest of the input.

6. **MUTABLE option for redefining.** The MUTABLE option indicates that an integer or real parameter may be redefined at some later point in the input processing (it is ignored for character parameters). Redefinition is allowed regardless of the setting of *PARAMETER_DUPLICATION. The MUTABLE qualifier must appear for the first definition of the parameter. It is not required for any later redefinition.

7. **NOECHO option.** The NOECHO keyword option tells LS-DYNA to not echo the defined parameters to the d3hsp file. This feature is useful for indicating which parameters in an encrypted file should not be echoed.

*PARAMETER_DUPLICATION

Purpose: The purpose is to control how the code behaves if a duplicate parameter definition is found in the input. This command must appear in the keyword deck before all *PARAMETER definitions.

Card 1	1	2	3	4	5	6	7	8
Variable	DFLAG							
Type	I							
Default	1							

VARIABLE	DESCRIPTION
DFLAG	Flag to control treatment of duplicate parameter definitions: EQ.1: issue a warning and ignore the new definition (default) EQ.2: issue a warning and accept the new definition EQ.3: issue an error and ignore (terminates at end of input) EQ.4: accept silently EQ.5: ignore silently

Remarks:

1. **LOCAL and non-LOCAL Variables.** A LOCAL variable appearing in a file, which masks a non-LOCAL parameter, won't trigger these actions; however, a LOCAL that masks another LOCAL or a non-LOCAL that masks a non-LOCAL will.

2. **Multiple Cards.** Only one *PARAMETER_DUPLICATION card is allowed. If more than one is found, a warning is issued and any after the first are ignored.

*PARAMETER_EXPRESSION_{OPTION}_{OPTION}_{OPTION}

The available options are

<BLANK>

LOCAL

MUTABLE (see Remark 4)

NOECHO (see Remark 5)

The keyword name may include any combination of the above keyword options in any order.

Purpose: Define the numerical values of parameter names referenced throughout the input file. Like the *PARAMETER keyword, but allows for general algebraic expressions, not simply fixed values. The LOCAL option allows for include files to contain their own unique expressions without clobbering the expressions in the rest of the input. See the *PARAMETER keyword.

Parameter Cards. Include as many cards as necessary.

Card 1	1	2	3	4	5	6	7	8
Variable	PRMR1	EXPRESSION1						
Type	A	A						
Default	none	none						

VARIABLE	DESCRIPTION
PRMR*n*	Define the *n*th parameter in a field of 10. Within this field the first character must be either an "R" for a real number, "I" for an integer, or "C" for a character string. Lower or upper case for "I/C/R" is okay. Following the type designation, define the name of the parameter using up to, but not exceeding nine characters. For example, when defining a shell thickness named, "SHLTHK", both inputs "RSHLTHK" or "R SHLTHK" can be used and placed anywhere in the field of 10. When referencing SHLTHK in the input file, see Remark 1 below.
EXPRESSION*n*	General expression which is evaluated, having the result stored in PRMR*n*. The following functions are available (see Remarks 2

VARIABLE	DESCRIPTION
	and 3): sin, cos, tan, csc, sec, ctn, asin, acos, atan, atan2, sinh, cosh, tanh, asinh, acosh, atanh, min, max, sqrt, mod, abs, sign, int, aint, nint, anint, exp, log, log10, float, pi, and general arithmetic expressions involving +, -, *, /, and **. The standard rules regarding operator precedence are obeyed, and nested parentheses are allowed. The expression can reference previously defined parameters (with or without the leading &). The expression can be continued on multiple lines simply by leaving the first 10 characters of the continuation line blank. For type "C" parameters, the expression is not evaluated in any sense, just stored as a string.

Remarks:

1. **Referencing parameters.** Parameters can be referenced anywhere in the input by placing an "&" immediately preceding the parameter name. Expressions can be included in the input when placed between brackets "<>" as long as the total line length does not exceed 80 columns and fields are comma-delimited. For example, this…

```
*parameter
rterm, 0.2, istates,  80
*parameter_expression
rplot,term/(states-30)
*DATABASE_BINARY_D3PLOT
&plot
```

is equivalent to

```
*parameter
rterm, 0.2, istates,  80
*DATABASE_BINARY_D3PLOT
<term/(states-30)>,
```

2. **Expression properties.** Properties used when evaluating the expressions are listed below.

   a) The integer and real properties of constants and parameters are honored when evaluating expressions. So 2/5 becomes 0, but 2.0/5 becomes 0.4.

b) The sign, atan2, min, max, and mod functions all take two arguments. The others all take only 1. The mod function supports integers only, and any real arguments are rounded.

c) Functions that use an angle as their argument, e.g., sin or cos, assume the angle is in radians.

d) The unary minus has higher precedence than exponentiation, that is, the formula -3**2 is interpreted as $(-3)^2 = 9$.

3. **Expression functions.** Descriptions for the available functions in parameter expressions are as follows:

sin	The sine function. It takes one argument, sin(x).
cos	The cosine function. It takes one argument, cos(x).
tan	The tangent function. It takes one argument, tan(x).
csc	The cosecant function. It takes one argument, csc(x).
sec	The secant function. It takes one argument, sec(x).
ctn	The cotangent function. It takes one argument, ctn(x).
asin	The inverse sine function. It takes one argument, asin(x).
acos	The inverse cosine function. It takes one argument, acos(x).
atan	The inverse tangent function. It takes one argument, atan(x).
atan2	Two-argument arctangent (or inverse tangent) function. It takes two arguments, atan(y,x) where $x + iy$ is a complex number.
sinh	The hyperbolic sine function. It takes one argument, sinh(x).
cosh	The hyperbolic cosine function. It takes one argument, cosh(x).
tanh	The hyperbolic tangent function. It takes one argument, tanh(x).
asinh	The inverse hyperbolic sine function. It takes one argument, asinh(x).
acosh	The inverse hyperbolic cosine function. It takes one argument, acosh(x).
atanh	The inverse hyperbolic tangent function. It takes one argument, atanh(x).
min	The minimum function. It takes two arguments, min(x,y), and returns the minimum value from the arguments.
max	The maximum function. It takes two arguments, max(x,y), and returns the maximum value from the arguments.
sqrt	Computes the square root of the argument. It takes one argument, sqrt(x).

mod	The modulo operation. It takes two arguments, mod(x,y), and gives the remainder of the division of x by y. This function only supports integers. Real arguments are rounded.
abs	Computes the absolute value of the argument. It takes one argument, abs(x).
sign	Function that returns the value of the first argument with the sign of the second argument. It takes two arguments, sign(x,y). For example, sign(-4,8) is 4.
int	Converts the argument into a value that has type integer. It takes one argument, int(x). The magnitude of the returned integer does not exceed the magnitude of the argument. For example, int(-48.1) is -48, and int(0.9) is 0.
aint	Function that returns a whole number by truncating the argument. The type of the value returned is real. It takes one argument, abs(x). If the magnitude of the argument is less than one, then it returns 0. The magnitude of the returned whole number does not exceed the magnitude of the argument. For example, aint(-48.1) is -48.
nint	Rounds the magnitude of the argument to the nearest whole number while retaining the sign. The type of the returned value is integer. It takes one argument, nint(x). For example, nint(-48.8) is -49.
anint	Rounds the magnitude of the argument to the nearest whole number while retaining the sign. The type of the returned value is real. It takes one argument, anint(x). For example, anint(-48.8) is -49.
exp	Exponential function, e^x, where x is the argument. It takes one argument, exp(x).
log	Computes the natural logarithm of the argument. It takes one argument, log(x).
log10	Computes the base 10 logarithm of the argument. It takes one argument, log10(x).
float	Converts the argument into a real value. It takes one argument, float(x).
pi	Number π.

4. **Redefining parameters.** The MUTABLE option indicates that an integer or real parameter may be redefined at some later point in the input processing (it is ignored for character parameters). Redefinition is allowed regardless of the setting of *PARAMETER_DUPLICATION. The MUTABLE qualifier must appear for the first definition of the parameter. It is not required for any later redefinition.

5. **NOECHO option.** The NOECHO keyword option tells LS-DYNA to not echo the defined parameters to the d3hsp file. This feature is useful for indicating which parameters in an encrypted file should not be echoed.

*PARAMETER_TYPE

*PARAMETER_TYPE is a variation on the *PARAMETER keyword command. In addition to its basic function of associating a parameter name (PRMR) with a numerical value (VAL), the *PARAMETER_TYPE command also includes information (PRTYP) about how the parameter is used by LS-DYNA, such as a Part ID or as a segment set ID.

*PARAMETER_TYPE is useful only when (1) the parameter is used to represent an integer ID number, and (2) LS-PrePost is used to combine two or more models (keyword decks) into a larger model.

Only by knowing how the parameter is used by LS-DYNA is LS-PrePost able to increment the parameter value by the proper "offset" when LS-PrePost combines two or more input decks together into a larger deck. These offsets are necessary so that IDs of a certain type, such as Part IDs, are not duplicated in the assembled model. Figure 36-1 (b) shows the offset input dialog box of LS-PrePost where offset values for specific ID types are assigned.

Background:

This command is designed to support workflows involving models that are built up from discrete subassemblies created by independent workgroups. As the subassembly models evolve through the design process, Part IDs, material IDs, etc. in the models may change with each design iteration; therefore, it is advantageous to parameterize those IDs. In this way, though the parameter values may change, the parameter names remain the same. When the subassembly models are combined by LS-PrePost to create a larger model of an assembly or of a complete system, for instance, an aircraft engine model, parameter values assigned using *PARAMETER_TYPE are incremented by the proper ID offset value as prescribed when LS-PrePost imports each keyword file.

Card Format:

Parameter Cards. For each parameter with type information, include an additional card. This input ends at the next keyword ("*") card.

Card 1	1	2	3	4	5	6	7	8
Variable	PRMR	VAL	PRTYP					
Type	A	I	A					
Default	none	none	none					

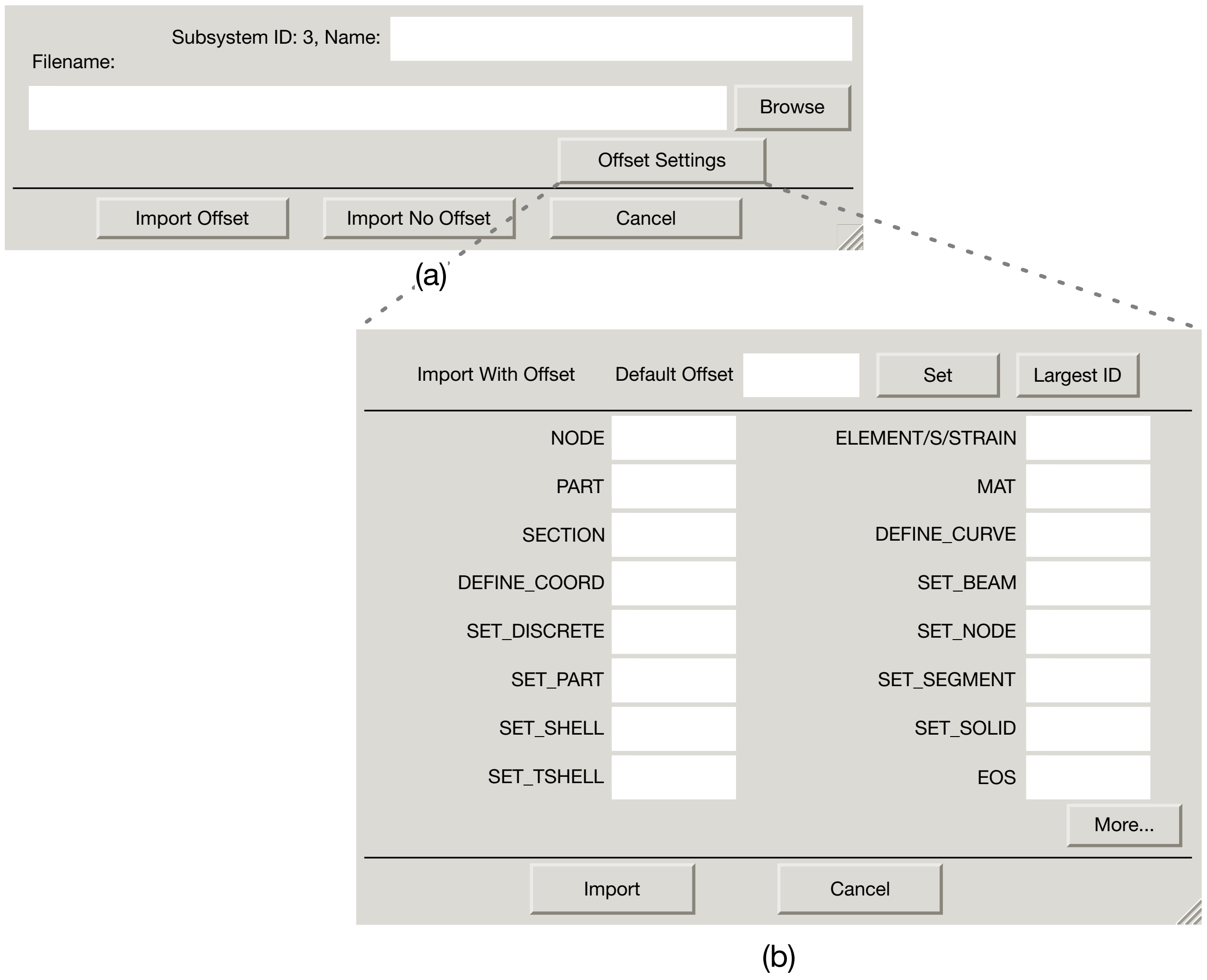

Figure 36-1. (a) the *file → import → keyword dialog box*; (b) the LS-PrePost dialog that takes the offset as a function of ID type.

VARIABLE	**DESCRIPTION**
PRMR	PRMR must be in the following format $$\text{PRMR} = \text{I}\,\underbrace{\text{xxxxxxxxx}}_{\text{9 character name}}$$ The first character is the type indicator and must be set to "I" for integer. The remaining 9 characters specify the name of the parameter. For example, to define a part ID "WHLPID", the input "IWHLPID", "I␣␣␣WHLPID", or "I␣␣WHLPID␣" are all equivalent 10 character strings ("␣" is space). For instructions regarding how to use the variable "WHLPID" see Remark 1.

VARIABLE	DESCRIPTION
VAL	Define the value of the parameter. The VAL field must contain an integer.
PRTYP	Describes, for the benefit of LS-PrePost only, how the parameter PRMR is used by LS-DYNA. PRTYP is ignored by LS-DYNA. For example, if VAL represents a Part ID, then PRTYP should be set to "PID". Knowing how the parameter is used by LS-DYNA, LS-PrePost can apply the appropriate offset to VAL when input decks are combined using LS-PrePost.

EQ.NID:	Node ID,
EQ.NSID:	Node set ID,
EQ.PID:	Part ID,
EQ.PSID:	Part set ID,
EQ.MID:	Material ID,
EQ.EOSID:	Equation of state ID,
EQ.BEAMID:	Beam element ID,
EQ.BEAMSID:	Beam element set ID,
EQ.SHELLID:	Shell element ID,
EQ.SHELLSID:	Shell element set ID,
EQ.SOLIDID:	Solid element ID,
EQ.SOLIDSID:	Solid element set ID,
EQ.TSHELLID:	Tshell element ID,
EQ.TSHELLSID:	Tshell element set ID,
EQ.SSID:	Segment set ID

Remarks:

1. **Referencing Parameters.** Parameters can be referenced anywhere in the input by placing an "&" at the first column of its field followed by the name of the parameter without blanks. For example, if PRMR is set to "I␣␣WHLPID␣", then the appropriate reference is "&WHLPID".

   Example:

```
*PARAMETER_TYPE
I WHLPID  100       PID
I WHLMID  300       MID
```

```
*PART
Wheel
&WHLPID,200,&WHLMID
```

2. **LS-PrePost.** *PARAMETER_TYPE is only supported by LS-PrePost 4.1 or later.

3. **Use Limitations.** Combining *INCLUDE_TRANSFORM with *PARAMETER_-TYPE is unsupported. This will introduce conflicting parameter offset values, and offset values specified in *INCLUDE_TRANSFORM will override offset values associated with *PARAMETER_TYPE.