GETTING STARTED

DESCRIPTION OF KEYWORD INPUT

The keyword input provides a flexible and logically organized database that is simple to understand. Similar functions are grouped together under the same keyword. For example, under the keyword *ELEMENT are included solid, beam, shell elements, spring elements, discrete dampers, seat belts, and lumped masses. Many keywords have options that are identified as follows: "*OPTIONS*" and "*{OPTIONS}*". The difference is that "*OPTIONS*" requires that one of the options must be selected to complete the keyword command. The option <BLANK> is included when {} are used to further indicate that these particular options are not necessary to complete the keyword.

The LS-DYNA User's Manual is alphabetically organized in logical sections of input data. Each logical section relates to a particular input. There is a control section for resetting LS-DYNA defaults, a material section for defining constitutive constants, an equation-of-state section, an element section where element part identifiers and nodal connectivities are defined, a section for defining parts, and so on. Nearly all model data can be input in block form. For example, consider the following where two nodal points with their respective coordinates and shell elements with their part identity and nodal connectivity's are defined:

```
$define two nodes
$
*NODE
10101x y z
10201x y z
$    define two shell elements
$
*ELEMENT_SHELL
10201pidn1n2n3n4
10301pidn1n2n3n4
Alternatively, acceptable input could also be of the form:
$     define one node
$
*NODE
10101x y z
$    define one shell element
$
*ELEMENT_SHELL
10201pidn1n2n3n4
$
$     define one more node
$
*NODE
10201x y z
$ define one more shell element
$
*ELEMENT_SHELL
10301pidn1n2n3n4
```

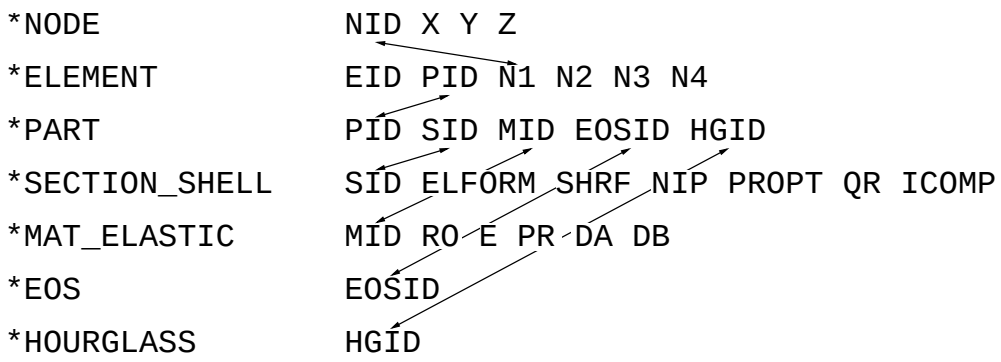

Figure 2-1. Organization of the keyword input.

A data block begins with a keyword followed by the data pertaining to the keyword. The next keyword encountered during the reading of the block data defines the end of the block and the beginning of a new block. A keyword must be left justified with the "*" contained in column one. A dollar sign "$" in column one precedes a comment and causes the input line to be ignored. Data blocks are not a requirement for LS-DYNA, but they can be used to group nodes and elements for user convenience. Multiple blocks can be defined with each keyword if desired as shown above. It would be possible to put all nodal points definitions under one keyword *NODE, or to define one *NODE keyword prior to each node definition. The entire LS-DYNA input is order independent with the exception of the optional keyword, *END, which defines the end of input stream. Without the *END termination is assumed to occur when an end-of-file is encountered during the reading.

Figure 2-1 highlights how various entities relate to each other in LS-DYNA input. In this figure the data included for the keyword, *ELEMENT, is the element identifier, EID, the part identifier, PID, and the nodal points identifiers, the NID's, defining the *element connectivity*: N1, N2, N3, and N4. The nodal point identifiers are defined in the *NODE section where each NID should be defined just once. A part defined with the *PART keyword has a unique part identifier, PID, a section identifier, SID, a material or constitutive model identifier, MID, an equation of state identifier, EOSID, and the hourglass control identifier, HGID. The *SECTION keyword defines the section identifier, SID, where a section has an element formulation specified, a shear factor, SHRF, a numerical integration rule, NIP, among other parameters.

Constitutive constants are defined in the *MAT section where constitutive data is defined for all element types including solids, beams, shells, thick shells, seat belts, springs, and dampers. Equations of state, which are used only with certain *MAT materials for solid elements, are defined in the *EOS section. Since many elements in LS-DYNA use uniformly reduced numerical integration, zero energy deformation modes may develop. These modes are controlled numerically by either an artificial stiffness or viscosity which resists the formation of these undesirable modes. The hourglass control can optionally be user specified using the input in the *HOURGLASS section.

During the keyword input phase where data is read, only limited checking is performed on the data since the data must first be counted for the array allocations and then reordered. Considerably more checking is done during the second phase where the input data is printed out. Since LS-DYNA has retained the option of reading older non-keyword input files, we print out the data into the output file d3hsp (default name) as in previous versions of LS-DYNA. An attempt is made to complete the input phase before error terminating if errors are encountered in the input. Unfortunately, this is not always possible and the code may terminate with an error message. The user should always check either output file, d3hsp or messag, for the word "Error".

The input data following each keyword can be input in free format. In the case of free format input the data is separated by commas, i.e.,

```
*NODE
10101,x ,y ,z
10201,x ,y ,z

*ELEMENT_SHELL
10201,pid,n1,n2,n3,n4
10301,pid,n1,n2,n3,n4
```

When using commas, the formats *must not* be violated. An I8 integer is limited to a maximum positive value of 99999999, and larger numbers having more than eight characters are unacceptable. The format of the input can change from free to fixed anywhere in the input file. The input is case insensitive and keywords can be given in either upper or lower case. *The asterisks "*" preceding each keyword must be in column one.*

To provide a better understanding behind the keyword philosophy and how the options work, a brief review of the keywords is given below.

*AIRBAG

The geometric definition of airbags and the thermodynamic properties for the airbag inflator models can be made in this section. This capability is not necessarily limited to the modeling of automotive airbags, but it can also be used for many other applications, such as tires and pneumatic dampers.

*ALE

This keyword provides a way of defining input data pertaining to the Arbitrary-Lagrangian-Eulerian capability.

*BOUNDARY

This section applies to various methods of specifying either fixed or prescribed boundary conditions. For compatibility with older versions of LS-DYNA it is still possible to specify some nodal boundary conditions in the *NODE card section.

*CASE

This keyword option provides a way of running multiple load cases sequentially. Within each case, the input parameters, which include loads, boundary conditions, control cards, contact definitions, initial conditions, etc., can change. If desired, the results from a previous case can be used during initialization. Each case creates unique file names for all output results files by appending CID*n* to the default file name.

*COMPONENT

This section contains analytical rigid body dummies that can be placed within vehicle and integrated implicitly.

*CONSTRAINED

This section applies constraints within the structure between structural parts. For example, nodal rigid bodies, rivets, spot welds, linear constraints, tying a shell edge to a shell edge with failure, merging rigid bodies, adding extra nodes to rigid bodies and defining rigid body joints are all options in this section.

*CONTACT

There are numerous options in the *CONTACT section for treating interaction between disjoint parts.

*CONTROL

Options available in the *CONTROL section allow the resetting of default global parameters, such as the hourglass type, the contact penalty scale factor, shell element formulation, numerical damping, and termination time.

*DAMPING

Defines damping either globally or by part identifier.

*DATABASE

This keyword with a combination of options can be used for controlling the output of ASCII databases and binary files output by LS-DYNA. With this keyword the frequency of writing the various databases can be determined.

*DEFINE

This section allows the user to define curves for loading, constitutive behaviors, etc.; boxes to limit the geometric extent of certain inputs; local coordinate systems; vectors; and orientation vectors specific to spring and damper elements. Items defined in this section are referenced by their identifiers throughout the input. For example, a coordinate system identifier is sometimes used on the *BOUNDARY cards, and load curves are used on the *AIRBAG cards.

*DEFORMABLE_TO_RIGID

This section allows the user to switch parts that are defined as deformable to rigid at the start of the analysis. This capability provides a cost efficient method for simulating events, such as rollover events. While the vehicle is rotating the computation cost can be reduced significantly by switching deformable parts that are not expected to deform to rigid parts. Just before the vehicle comes in contact with ground, the analysis can be stopped and restarted with the part switched back to deformable.

*ELEMENT

Define identifiers and connectivities for all elements which include shells, beams, solids, thick shells, springs, dampers, seat belts, and concentrated masses in LS-DYNA.

*EOS

This section reads the equations of state parameters. The equation of state identifier, EOSID, points to the equation of state identifier on the *PART card.

*HOURGLASS

Defines hourglass and bulk viscosity properties. The identifier, HGID, on the *HOURGLASS card refers to HGID on *PART card.

*INCLUDE

To make the input file easy to maintain, this keyword allows the input file to be split into sub-files. Each sub-file can again be split into sub-sub-files and so on. This option is beneficial when the input data deck is very large.

*INITIAL

Initial velocity and initial momentum for the structure can be specified in this section. The initial velocity specification can be made by *INITIAL_VELOCITY_NODE card or *INITIAL_VELOCITY cards. In the case of *INITIAL_VELOCITY_NODE nodal identifiers are used to specify the velocity components for the node. Since all the nodes in the system are initialized to zero, only the nodes with non-zero velocities need to be specified. The *INITIAL_VELOCITY card provides the capability of being able to specify velocities using the set concept or boxes.

*INTEGRATION

In this section the user defined integration rules for beam and shell elements are specified. IRID refers to integration rule number IRID on *SECTION_BEAM and *SECTION_SHELL cards, respectively. Quadrature rules in the *SECTION_SHELL and *SECTION_BEAM cards need to be specified as a negative number. The absolute value of the negative number refers to user defined integration rule number. Positive rule numbers refer to the built in quadrature rules within LS-DYNA.

*INTERFACE

Interface definitions are used to define surfaces, nodal lines, and nodal points for which the displacement and velocity time histories are saved at some user specified frequency. This data may then be used in subsequent analyses as an interface ID in the *INTERFACE_LINKING_DISCRETE_NODE as constraining nodes, in *INTERFACE_LINKING_SEGMENT as constraining segments, and in *INTERFACE_LINKING_EDGE as the constraining edge for a series of nodes.

This capability is especially useful for studying the detailed response of a small member in a large structure. For the first analysis, the member of interest need only be discretized sufficiently that the displacements and velocities on its boundaries are reasonably accurate. After the first analysis is completed, the member can be finely discretized in the region bounded by the interfaces. Finally, the second analysis is performed to obtain highly detailed information in the local region of interest.

When beginning the first analysis, specify a name for the interface segment file using the Z=parameter on the LS-DYNA execution line. When starting the second analysis, the name of the interface segment file created in the first run should be specified using the

L=parameter on the LS-DYNA command line. Following the above procedure, multiple levels of sub-modeling are easily accommodated. The interface file may contain a multitude of interface definitions so that a single run of a full model can provide enough interface data for many component analyses. The interface feature represents a powerful extension of LS-DYNA's analysis capabilities. A similar capability using *INTERFACE_SSI may be used for soil-structure interaction analysis under earthquake excitation.

*KEYWORD

Flags LS-DYNA that the input deck is a keyword deck. To have an effect this must be the very first card in the input deck. Alternatively, by typing "keyword" on the execution line, keyword input formats are assumed and the "*KEYWORD" is not required. If a number is specified on this card after the word KEYWORD, it defines the memory size to be used in words. The memory size can also be set on the command line.

> **NOTE:** The memory specified on the execution line overrides memory specified on the *KEYWORD card.

*LOAD

This section provides various methods of loading the structure with concentrated point loads, distributed pressures, body force loads, and a variety of thermal loadings.

*MAT

This section allows the definition of constitutive constants for all material models available in LS-DYNA including springs, dampers, and seat belts. The material identifier, MID, points to the MID on the *PART card.

*NODE

Define nodal point identifiers and their coordinates.

*PARAMETER

This option provides a way of specifying numerical values of parameter names that are referenced throughout the input file. The parameter definitions, if used, should be placed at the beginning of the input file following *KEYWORD. *PARAMETER_EXPRESSION permits general algebraic expressions to be used to set the values.

*PART

This keyword serves two purposes:

1. Relates part ID to *SECTION, *MATERIAL, *EOS and *HOURGLASS sections.

2. Optionally, in the case of a rigid material, rigid body inertia properties and initial conditions can be specified. Deformable material repositioning data can also be specified in this section if the reposition option is invoked on the *PART card, that is, *PART_REPOSITION.

*PERTURBATION

This keyword provides a way of defining deviations from the designed structure, such as buckling imperfections.

*RAIL

This keyword provides a way of defining a wheel-rail contact algorithm intended for railway applications but can also be used for other purposes. The wheel nodes (defined on *RAIL_TRAIN) represent the contact patch between wheel and rail.

*RIGIDWALL

Rigid wall definitions have been divided into two separate sections, PLANAR and GEOMETRIC. Planar walls can be either stationary or moving in translational motion with mass and initial velocity. The planar wall can be either finite or infinite. Geometric walls can be planar as well as have the geometric shapes, such as rectangular prism, cylindrical prism and sphere. By default, these walls are stationary unless the option MOTION is invoked for either prescribed translational velocity or displacement. Unlike the planar walls, the motion of the geometric wall is governed by a load curve. Multiple geometric walls can be defined to model combinations of geometric shapes available. For example, a wall defined with the CYLINDER option can be combined with two walls defined with the SPHERICAL option to model hemispherical surface caps on the two ends of a cylinder. Contact entities are also analytical surfaces but have the significant advantage that the motion can be influenced by the contact to other bodies, or prescribed with six full degrees-of-freedom.

*SECTION

In this section, the element formulation, integration rule, nodal thicknesses, and cross sectional properties are defined. All section identifiers (SECID's) defined in this section must be unique; that is, if a number is used as a section ID for a beam element, then this number cannot be used again as a section ID for a solid element.

*SENSOR

This keyword provides a convenient way of activating and deactivating boundary conditions, airbags, discrete elements, joints, contact, rigid walls, single point constraints, and constrained nodes. The sensor capability is new in the second release of version 971 and will evolve in later releases to encompass many more LS-DYNA capabilities and replace some of the existing capabilities, such as the airbag sensor logic.

*SET

A concept of grouping nodes, elements, materials, etc., in sets is employed throughout the LS-DYNA input deck. Sets of data entities can be used for output. So-called surfa nodes used in contact definitions, surfa segment sets, surfb segment sets, pressure segment sets, and so on can also be defined. The keyword, *SET, can be defined in two ways:

1. Option LIST requires a list of entities in which eight entities are defined per card and as many cards as needed are used to define all the entities.

2. Option COLUMN, where applicable, requires an input of one entity per line along with up to four attribute values which are used by other keywords to specify, for example, the failure criterion input that is needed for *CONTACT_CONSTRAINT_NODES_TO_SURFACE.

*TERMINATION

This keyword provides an alternative way of stopping the calculation before the termination time is reached. The termination time is specified on the *CONTROL_TERMINATION input and will terminate the calculation whether or not the options available in this section are active.

*TITLE

In this section a title for the analysis is defined.

*USER_INTERFACE

This section provides a method to provide user control of some aspects of the contact algorithms including friction coefficients via user defined subroutines.

RESTART

This section of the input is intended to allow the user to restart the simulation by providing a restart file and optionally a restart input defining changes to the model, such as

deleting contacts, materials, elements, switching materials from rigid to deformable, and deformable to rigid.

*RIGID_DEFORMABLE

This section switches rigid parts back to deformable in a restart to continue the event of a part impacting the ground which may have been modeled with a rigid wall.

*STRESS_INITIALIZATION

This is an option available for restart runs. In some cases there may be a need for the user to add contacts, elements, etc., which are not available options for standard restart runs. A full input containing the additions is needed if this option is invoked upon restart.

SUMMARY OF COMMONLY USED OPTIONS

The following table gives a list of the commonly used keywords related by topic.

Topic	Component	Keywords
Geometry	Nodes	*NODE
	Elements	*ELEMENT_BEAM *ELEMENT_SHELL *ELEMENT_SOLID *ELEMENT_TSHELL
	Discrete Elements	*ELEMENT_DISCRETE *ELEMENT_SEATBELT *ELEMENT_MASS

Topic	Component	Keywords
Materials	Part	PART cards glues the model together: *PART → { *MAT, *SECTION, *EOS, *HOURGLASS }
	Material	*MAT
	Sections	*SECTION_BEAM *SECTION_SHELL *SECTION_SOLID *SECTION_TSHELL
	Discrete sections	*SECTION_DISCRETE *SECTION_SEATBELT
	Equation of state	*EOS
	Hourglass	*CONTROL_HOURGLASS *HOURGLASS
Contacts & Rigid walls	Defaults for contacts	*CONTROL_CONTACT
	Definition of contacts	*CONTACT_*OPTION*
	Definition of rigid walls	*RIGIDWALL_*OPTION*
Boundary Conditions & Loadings	Restraints	*NODE *BOUNDARY_SPC_*OPTION*
	Gravity (body) load	*LOAD_BODY_*OPTION*
	Point load	*LOAD_NODE_*OPTION*
	Pressure load	*LOAD_SEGMENT_*OPTION* *LOAD_SHELL_*OPTION*
	Thermal load	*LOAD_THERMAL_*OPTION*
	Load curves	*DEFINE_CURVE
Constraints and spot welds	Constrained nodes	*CONSTRAINED_NODE_SET
	Welds	*CONSTRAINED_GENERALIZED_WELD *CONSTRAINED_SPOT_WELD
	Rivet	*CONSTRAINED_RIVET

Topic	Component	Keywords
Output Control	Items in time history blocks	*DATABASE_HISTORY_*OPTION*
	Default	*CONTROL_OUTPUT
	ASCII time history files	*DATABASE_OPTION
	Binary plot/time history/restart files	*DATABASE_BINARY_OPTION
	Nodal reaction output	*DATABASE_NODAL_FORCE_GROUP
Termination	Termination time	*CONTROL_TERMINATION
	Termination cycle	*CONTROL_TERMINATION
	CPU termination	*CONTROL_CPU
	Degree of freedom	*TERMINATION_NODE

Table 2.1. Keywords for the most commonly used options.

EXECUTION SYNTAX

The execution line for LS-DYNA, sometimes referred to as the command line, is as follows:

LS-DYNA I=**inf** O=**otf** G=**ptf** D3PART=**d3part** D=**dpf** R=**rtf** N=**rtff** F=**thf** T=**tpf** A=**rrd** M=**sif** S=**iff** H=**iff** Z=**isf1** L=**isf2** B=**rlf** W=**root** E=**efl** X=**scl** C=**cpu** K=**kill** V=**vda** Y=**c3d** P=**pfile** BEM=**bof** CPM=**cmpof** DEM=**demof** PBM=**pbmof** ISPH=**isphof** FSILNK=**fsilof** {**KEYWORD**} {**THERMAL**} {**COUPLE**} {**CASE**} {**PGPKEY**} {**FLUIDS**} {**MICROSCALE**} {**HASH**} MEMORY=**nwds** MODULE=**dll** NCPU=**ncpu** PARA=**para** ENDTIME=**time** NCYCLE=**ncycle** JOBID=**jobid** D3PROP=**d3prop** GMINP=**gminp** GMOUT=**gmout** MCHECK=**y** MAP=**map** MAP1=**map1** MAP2=**map2** LAGMAP=**lagmap** LAGMAP1=**lagmap1** SHELL=**shefch** ND3KIL=**ncyclek** STDOUT=**stdout** SRTFLG=**srtflg**

where,

inf = Input file (user-specified)

otf = High speed printer file (default = d3hsp)

ptf = Binary plot file for postprocessing (default = d3plot)

d3part = Binary plot file for subset of parts; see *DATABASE_BINARY_D3PART (default = d3part)

dpf = Dump file to write for purposes of restarting (default = d3dump). This file is written at the end of every run and during the run as requested by *DATABASE_BINARY_D3DUMP. To stop the generation of this dump file, specify "d=nodump" (case insensitive). Sense switch "sw1" triggers writing a dump file and overrides "d=nodump".

thf = Binary plot file for time histories of selected data (default = d3thdt)

rtf = Binary file from which to restart the analysis for restarts (see RESTART ANALYSIS). This applies to simple and small deck restarts. It also applies to full deck restarts for SMP.

rtff = d3full file from which to restart the analysis for full deck restarts in MPP (see RESTART ANALYSIS).

tpf = Optional temperature file

rrd = Running restart dump file (default = runrsf)

sif = Stress initialization file (user-specified)

iff = Interface force file (user-specified). See *DATBASE_BINARY_INTFOR and *DATABASE_BINARY_FSIFOR.

isf1 = Interface segment save file to be created (default = infmak)

isf2 = Existing interface segment save file to be used (user-specified)

rlf = Binary plot file for dynamic relaxation (default = d3drlf)

efl = Echo file containing optional input echo with or without node/element data

root = Root file name for general print option

scl = Scale factor for binary file sizes (default =70)

cpu = Cumulative cpu time limit in seconds for the entire simulation, including all restarts, if cpu is positive. If cpu is negative, the absolute value of cpu is the cpu time limit in seconds for the first run and for each subsequent restart run.

kill = If LS-DYNA encounters this file name it will terminate with a restart file (default = d3kil)

vda = VDA/IGES database for geometrical surfaces

c3d = CAL3D input file

pfile = File containing MPP specific parameters (see Appendix O for details)

bof = *FREQUENCY_DOMAIN_ACOUSTIC_BEM output file

cpmof = CPM interface force file (user-specified). See *DATABASE_BINARY_*OPTION*.

demof = DEM interface force file (user-specified). See *DATABASE_BINARY_*OPTION.*

pbmof = Particle blast method interface force file (user-specified). See *DATABASE_BINARY_*OPTION.*

isphof = Incompressible SPH interface force file (user-specified). See *DATABASE_BINARY_*OPTION.*

fsilof = ALE interface linking database file (user-specified). See *DATABASE_BINARY_*OPTION.*

nwds = Number of words to be allocated. A word is 4 bytes in single precision and 8 bytes in double precision. This number overwrites the memory size specified on the *KEYWORD card at the beginning of the input deck. This option is necessary if the default memory allocation is insufficient, in which case LS-DYNA will write an error message stating such.

dll = The dynamic library for user subroutines. Only one dynamic library can be loaded via "module=dll". See *MODULE_LOAD command for loading multiple dynamic libraries.

ncpu = Overrides NCPU and CONST defined in *CONTROL_PARALLEL. A positive value sets CONST = 2 and a negative values sets CONST = 1. See the *CONTROL_PARALLEL command for an explanation of these parameters. The *KEYWORD command provides an alternative way to set the number of CPUs.

para = Overrides PARA defined in *CONTROL_PARALLEL.

time = Overrides ENDTIM defined in *CONTROL_TERMINATION.

ncycle = Overrides ENDCYC defined in *CONTROL_TERMINATION.

jobid = Character string which acts as a prefix for all output files. Maximum length is 72 characters. Do not include the following characters:) (* / ? \.

d3prop = See *DATABASE_BINARY_D3PROP input parameter IFILE for options.

gminp = Input file for reading recorded motions in *INTERFACE_SSI (default = gmbin).

gmout = Output file for writing recorded motions in *INTERFACE_SSI_AUX (default = gmbin).

map = Output/Input file for writing/reading data in ALE mappings controlled by *INITIAL_ALE_MAPPING or *BOUNDARY_ALE_MAPPING.

map1 = Output file for writing data in ALE mappings controlled by *INITIAL_ALE_MAPPING or *BOUNDARY_ALE_MAPPING if MAP= is used to read data from a different file.

map2 = Input .txt file that lists multiple mapping files for reading data from multiple ALE mapping files controlled by *INITIAL_ALE_MAPPING.

lagmap = Output/Input file for writing/reading data in Lagrangian mappings controlled by *INITIAL_LAG_MAPPING.

lagmap1 = Output file for writing data in Lagrangian mappings controlled by *INITIAL_LAG_MAPPING if LAGMAP= is used to read data from a different file.

shefch = Change the shell element formulation from 16 to 30 or 30 to 16 without changing the input deck. To do this, use either "shell=16to30" or "shell=30to16".

ncyclek = Overrides the default frequency of checking for the d3kil file. The default is every 100 cycles in explicit and every cycle in implicit.

stdout = Control the messages printed to stdout. The available values and their meanings are in the following table. All means that all of that particular type of message is printed, while none means none of that kind of message is printed. The default value of stdout is 0.

stdout	Information	Warning	Error
0	all	all	all
1	none	all	all
2	all	none	all
3	none	none	all
4	all	all	none
5	none	all	none
6	all	none	none
7	none	none	none

srtflg = Flag to process parts, contacts, nodal rigid bodies, and elements in a sequence sorted by their respective user IDs, regardless of the order they appear in the input files. When turned on (srtflg = 1), it ensures consistent results if the user modifies the order of these entities in the input files, such as by changing the order of include files between runs of the same model. srtflg = overrides SRTFLG on *CONTROL_ACCURACY. Thus, srtflg = 0 deactivates this sorting.

These command line options can be *in any order*.

To avoid undesirable or confusing results, each LS-DYNA run should be performed in a separate directory, unless using the command line parameter "jobid" described above. If rerunning a job in the same directory, old files should first be removed or renamed to avoid confusion since the possibility exists that the binary database may contain results from both the old and new run.

By including "keyword" anywhere on the execute line or instead if *KEYWORD is the first card in the input file, the keyword formats are expected; otherwise, the older structured input file will be expected.

To run a coupled thermal analysis the command "couple" must be in the execute line. A thermal-only analysis may be run by including the word "thermal" in the execution line.

The execution line option "pgpkey" will output the current public PGP key used by LS-DYNA for encrypting the input. LS-DYNA writes the public key and some instructions on how to use the key are to the screen and to a file named "lstc_pgpkey.asc".

For licenses restricted to the fluids capabilities of LS-DYNA, you need to include "fluids" as an execution line option.

The execution line option "microscale" is needed for running the microscale process in a multiscale simulation with either *INCLUDE_MULTISCALE or *INCLUDE_MULTISCALE_SPOTWELD. See those keywords for details.

Setting "ncycle=1" on the execution line causes the calculation to run just one cycle followed by termination with restart files. No editing of the input deck is required. The calculation can then be restarted with or without any additional input.

If the word "case" appears on the command line, then *CASE statements will be handled by the built-in driver routines. Otherwise, they should be processed by the external "lscasedriver" program, and if any *CASE statements are encountered, it will cause an error.

If "mcheck=y" is given on the command line, the program switches to "model check" mode. In this mode, the program will run only 10 cycles – just enough to verify that the model will start. For implicit problems, all initialization is performed, but execution halts before the first cycle. If the network license is being used, the program will attempt to check out a license under the program name "LS-DYNAMC" so as not to use up one of the normal DYNA licenses. If this fails, a normal execution license will be used.

If "hash" is given on the command line, then SHA256 hashes are computed and reported at the bottom of the d3hsp file for the following: any *HASH/*HASH_END blocks, all input files, and some of the generated output files (currently glstat, d3plot, and binout files). Additionally, the final d3hsp file is digitally signed by LS-DYNA, generating a d3hsp.sig file, which can be verified using GPG with the 2048-bit LS-DYNA public encryption key.

SENSE SWITCH CONTROLS

The status of an in-progress LS-DYNA simulation can be determined by using the sense switch. On UNIX versions, this is accomplished by first typing a "^C" (Control-C). This sends an interrupt to LS-DYNA which is trapped and the user is prompted to input the sense switch code. LS-DYNA has a number of terminal sense switch controls that are tabulated below:

Type	Response
SW1	A restart (dump) file is written, and LS-DYNA terminates.
SW2	LS-DYNA responds with time and cycle numbers.
SW3	A restart file is written, and LS-DYNA continues.
SW4	A plot state is written, and LS-DYNA continues.
SWA	Flush ASCII file buffers.
SWB	A dynain file is written and LS-DYNA continues. For SWB, SWC and SWD (see below), *INTERFACE_SPRINGBACK_LSDYNA must be defined in the input deck. The file name is dynain.# where # is incremented by 1 each time SWB, SWC, or SWD is issued.
SWC	A restart file and a dynain file are written, and LS-DYNA continues.
SWD	A restart file and a dynain file are written, and LS-DYNA terminates.
SWE	Terminate explicit dynamic relaxation and proceed with the transient analysis.
conv	Temporarily override nonlinear convergence tolerances.
endtime=*t*	Change the termination time to *t* (*t* is a real number).
iter	Enable/Disable output of binary plot database "d3iter" showing mesh after each equilibrium iteration. Useful for debugging convergence problems.
lprint	Enable/Disable printing of equation solver memory, cpu requirements.
nlprint	Enable/Disable printing of nonlinear equilibrium iteration information.

Type	Response
prof	Output current timing information to messag (SMP) or prof.out (MPP).
stop	Halt execution immediately, closing open files.

On UNIX/LINUX and Windows systems the sense switches can still be used if the job is running in the background or in batch mode. To interrupt LS-DYNA simply create a file called d3kil containing the desired sense switch, for example, "sw1". LS-DYNA periodically looks for this file, and if found, the sense switch contained therein is invoked and the d3kil file is deleted. Command line option nd3kil=ncyclek changes the frequency of checking for this file. A null d3kil file is equivalent to a "sw1".

When LS-DYNA terminates, all scratch files are destroyed: the restart file, plot files, and high-speed printer files remain on disk. Of these, only the restart file is needed to continue the interrupted analysis.

PROCEDURE FOR LS-DYNA/MPP

As described above the serial/SMP code supports the use of the SIGINT signal (usually Ctrl-C) to interrupt the execution and prompt the user for a "sense switch." The MPP code also supports this capability. However, on many systems a shell script or front end program (generally "mpirun") is required to start MPI applications. Pressing Ctrl-C on some systems will kill this process, and thus kill the running MPP-DYNA executable. On UNIX/LINUX systems, as workaround, when the MPP code begins execution it creates a file named, “bg_switch”, in the current working directory. This file contains the following single line:

rsh **<machine name>** kill -INT **<PID>**

where < machine name > is the hostname of the machine on which the root MPP-DYNA process is running, and <PID> is its process id. (on HP systems, "rsh" is replaced by "remsh"). Thus, simply executing this file will send the appropriate signal. For Windows, usually the d3kil file is used to interrupt the execution.

For more information about running the LS-DYNA/MPP Version see Appendix O.

Files: Input and Output

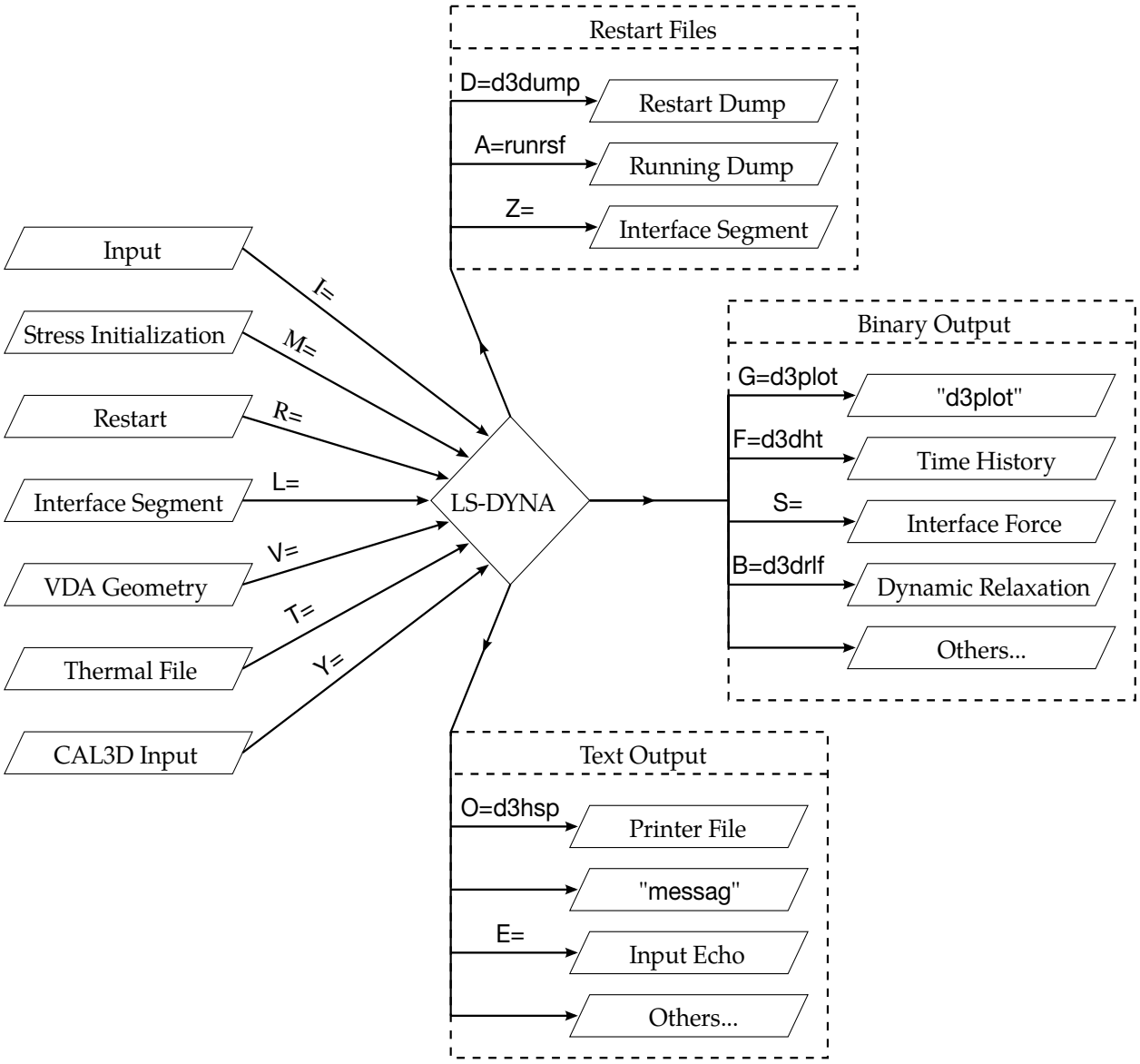

Figure 2-2. Files Input and Output.

FILES

1. **Uniqueness.** File names must be unique.

2. **Interface forces.** The interface force file is created only if it is specified on the execution line "S=**iff**".

3. **File size limits.** For very large models, the default size limits for binary output files may not be large enough for a single file to hold even a single plot state, in

which case the file size limit may be increased by specifying "X=**scl**" on the execution line. The default file size limit (X=70) is 70 times one-million octal words or 18.35 Mwords. That translates into 73.4 Mbytes (for 32-bit output) or 146.8 Mbytes (for 64-bit output).

4. **CPU limits.** Using "C=**cpu**" defines the maximum CPU usage allowed. When the CPU usage limit is exceeded, LS-DYNA will terminate with a restart file. During a restart, **cpu** should be set to the total CPU used up to the current restart plus whatever amount of additional time is wanted.

5. **File usage in restart.** When restarting from a dump file, the execution line (except of a full deck restart in MPP) becomes

   LS-DYNA I=**inf** O=**otf** G=**ptf** D=**dpf** R=**rtf** F=**thf** T=**tpf** A=**rrd** S=**iff** Z=**isf1** L=**isf2** B=**rlf** W=**root** E=**efl** X=**scl** C=**cpu** K=**kill** {KEYWORD}

   where

   rtf=[name of dump file written by LS-DYNA]

   The root names of the dump files written by LS-DYNA = are controlled by dpf (default = d3dump) and rrd (default = runrsf). A two-digit number follows the root name, for example, d3dump01, d3dump02, etc., to distinguish one dump file from another. Typically, each dump file corresponds to a different simulation time.

   The adaptive dump files contain all information required to successfully restart so that no other files are needed except when CAD surface data is used. When restarting a problem that uses VDA/IGES surface data, the vda input file must be specified, e.g.:

   LS-DYNA R=**d3dump01** V=**vda**

   No stress initialization is possible at restart. Also the VDA files and the CAL3D files must not be changed.

6. **Default file names.** File name dropouts are permitted; for example, the following execution lines are acceptable:

   LS-DYNA I=**inf**

   and

   LS-DYNA R=**rtf**

7. **Interface segments.** For an analysis using interface segments, the execution line in the first analysis is given by:

LS-DYNA I=**inf** Z=**isf1**

and in the second by:

LS-DYNA I=**inf** L=**isf1**

8. **Batch execution.** In some installations (e.g., GM) calculations are controlled by a file called "names" on unit 88. The names files consists of two lines in which the second line is blank. The first line of names contains the execution line, for instance:

I=**inf**

For a restart the execution line becomes:

I=**inf** R=**rtf**

RESTART ANALYSIS

The LS-DYNA restart capability allows analyses to be broken down into stages. After the completion of each stage in the calculation, a "restart dump" is written that contains all information necessary to continue the analysis. The size of this "dump" file is roughly the same size as the memory required for the calculation. Results can be checked at each stage by post-processing the output databases in the normal way, so the chance of wasting computer time on incorrect analyses is reduced.

The restart capability is frequently used to modify models by deleting excessively distorted elements, materials that are no longer important, and contact surfaces that are no longer needed. Output frequencies of the various databases can also be altered. Often, these simple modifications permit a calculation that might otherwise not continue on to a successful completion. Restarting can also help diagnose why a model is giving problems. By restarting from a dump that is written before the occurrence of a numerical problem and obtaining output at more frequent intervals, it is often possible to identify where the first symptoms appear and what aspect of the model is causing them.

The format of the restart input file is described in this manual. If, for example, the user wishes to restart the analysis from dump state *nn*, contained in file D3DUMP*nn*, then the following procedure is followed:

1. Create the restart input deck, if required, as described in the Restart Section of this manual. Call this file restartinput.

2. Start dyna from the command line by invoking:

LS-DYNA I=**restartinput** R=**D3DUMP*nn***

3. If no alterations to the model are made, then the execution line:

   LS-DYNA R=**D3DUMP*nn***

   will suffice. Of course, the other output files should be assigned names from the command line if the defaults have been changed in the original run.

The full deck restart option allows the user to begin a new analysis, with deformed shapes and stresses carried forward from a previous analysis for selected materials. The new analysis can be different from the original, such as more contact surfaces and different geometry (of parts which are not carried forward). Examples of applications include:

- Crash analysis continued with extra contact surfaces.
- Sheet metal forming continued with different tools for modeling a multi-stage forming process.

The following procedure is followed for full-deck restart:

1. Create the restart input deck using *STRESS_INITIALIZATION, as described in the Restart Section of this manual. Call this file restartinput.

2. For SMP, start dyna from the command line by invoking:

   LS-DYNA I=**restartinput** R=**D3DUMP*nn***

3. For MPP, start dyna from the command line by invoking:

   LS-DYNA I=**restartinput** N=**D3FULL*nn***

A typical restart file scenario:

LS-DYNA is run using an input file named "job1.inf", and a restart dump named "d3dump01" is created. A new input file, "job2.inf", is generated and submitted as a restart with, "R=d3dump01", as the dump file. The input file job2.inf contains the entire model in its original *undeformed* state but with more contact surfaces, new output databases, and so on.

Since this is a restart job, information must be given to tell LS-DYNA which parts of the model should be initialized in the full deck restart. When the calculation begins, the restart database contained in the file d3dump01 is read, and a new database is created to initialize the model in the input file, job2.inf. The data in file job2.inf is read, and then LS-DYNA proceeds through the entire input deck and initialization. At the end of the initialization process, all the parts selected are initialized from the data saved from d3dump01. This means that the deformed position and velocities of the nodes on the elements of each part, and the stresses and strains in the elements (and, if the material of the part is rigid, the rigid body properties) will be assigned.

It is assumed during this process that any initialized part has the same elements, in the same order, with the same topology, in job1 and job2. If this is not the case, the parts cannot be initialized. However, the parts may have different identifying numbers.

For discrete elements and seat belts, the choice is all or nothing. All discrete and belt elements, retractors, slip rings, pretensioners, and sensors must exist in both files and will be initialized.

Materials which are not initialized will have no initial deformations or stresses. However, if initialized and non-initialized materials have nodes in common, the nodes will be moved by the initialized material causing a sudden strain in the non-initialized material. *This effect can give rise to sudden spikes in loading.*

Points to note are:

- Time and output intervals are continuous with job1; that is, the time is not reset to zero.
- Don't try to use the restart part of the input to change anything since this will be overwritten by the new input file.
- Usually, the complete input file part of job2.inf will be copied from job1.inf, with the required alterations. We again mention that there is no need to update the nodal coordinates since the deformed shapes of the initialized materials will be carried forward from job1.
- Completely new databases will be generated with the time offset.

VDA/IGES DATABASES

VDA surfaces are surfaces of geometric entities which are given in the form of polynomials. The format of these surfaces is as defined by the German automobile and supplier industry in the VDA guidelines, [VDA 1987].

The advantage of using VDA surfaces is twofold. First, the problem of meshing the surface of the geometric entities is avoided and, second, smooth surfaces can be achieved which are very important in metal forming. With smooth surfaces, artificial friction introduced by standard faceted meshes with corners and edges can be avoided. This is a big advantage in springback calculations.

A very simple and general handling of VDA surfaces is possible allowing arbitrary motion and generation of surfaces. For a detailed description, see Appendix L.

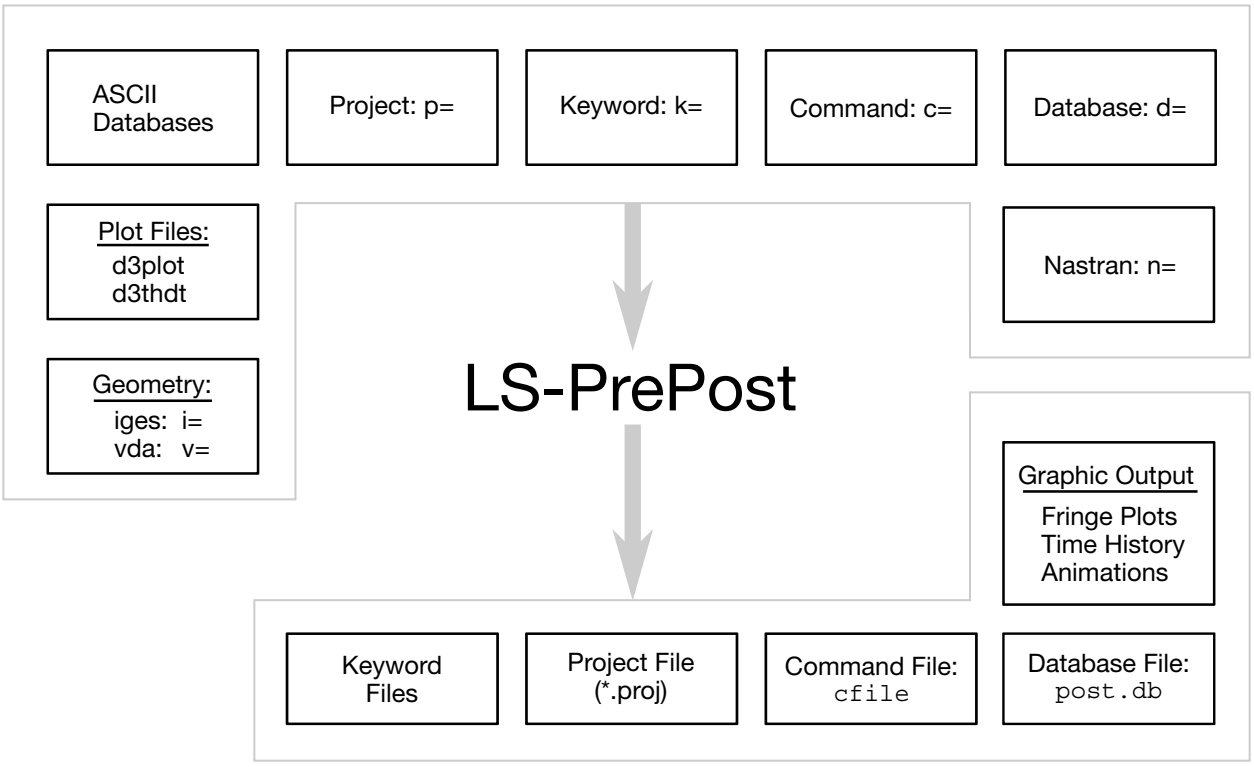

Figure 2-3. File Organization

LS-PrePost®

LS-DYNA is designed to operate with a variety of commercial pre- and post-processing packages. Currently, direct support is available from TRUEGRID, PATRAN, eta/VPG, HYPERMESH, EASi-CRASH DYNA and FEMAP. Several third-party translation programs are available for PATRAN and IDEAS.

Alternately, the pre- and post-processor LS-PrePost is available from Ansys and is specialized for LS-DYNA. LS-PrePost is an advanced pre- and post-processor that is delivered free with LS-DYNA. The user interface is designed to be both efficient and intuitive. LS-PrePost runs on Windows, Linux, and Unix, utilizing OpenGL graphics to achieve fast model rendering and XY plotting.

Some of the capabilities available in LS-PrePost are:

- Complete support for all LS-DYNA keyword data.
- Importing and combining multiple models from many sources (LS-DYNA keyword, IDEAS neutral file, NASTRAN bulk data, STL ASCII, and STL binary formats).
- Improved renumbering of model entities.
- Model Manipulation: Translate, Rotate, Scale, Project, Offset, Reflect

- LS-DYNA Entity Creation: Coordinate Systems, Sets, Parts, Masses, CNRBs, Boxes, Spot welds, SPCs, Rigidwalls, Rivets, Initial Velocity, Accelerometers, Cross Sections, etc.
- Mesh Generation: 2Dmesh Sketchboard, nLine Meshing, Line sweep into shell, Shell sweep into solid, Tet-Meshing, Automatic surface meshing of IGES and VDA data, Meshing of simple geometric objects (Plate, Sphere, Cylinder)
- Special Applications: Airbag folding, Dummy positioning, Seatbelt fitting, Initial penetration check, Spot weld generation using MAT_100
- Complete support of LS-DYNA results data file: d3plot file, d3thdt file, All ASCII time history data file, Interface force file

LS-PrePost processes output from LS-DYNA. LS-PrePost reads the binary plot-files generated by LS-DYNA and plots contours, fringes, time histories, and deformed shapes. Color contours and fringes of a large number of quantities may be interactively plotted on meshes consisting of plate, shell, and solid type elements. LS-PrePost can compute a variety of strain measures and reaction forces along constrained boundaries.

LS-DYNA generates three binary databases. One contains information for complete states at infrequent intervals; 50 to 100 states of this sort is typical in a LS-DYNA calculation. The second contains information for a subset of nodes and elements at frequent intervals; 1000 to 10,000 states is typical. The third contains interface data for contact surfaces.

EXECUTION SPEEDS

The relative execution speeds for various elements in LS-DYNA are tabulated below:

Element Type	Relative Cost
8 node solid with 1 point integration and default hourglass control	4
as above but with Flanagan-Belytschko hourglass control	5
constant stress and Flanagan-Belytschko hourglass control, i.e., the Flanagan-Belytschko element	7
4 node Belytschko-Tsay shell with four thickness integration points	4
4 node Belytschko-Tsay shell with resultant plasticity	3
BCIZ triangular shell with four thickness integration points	7

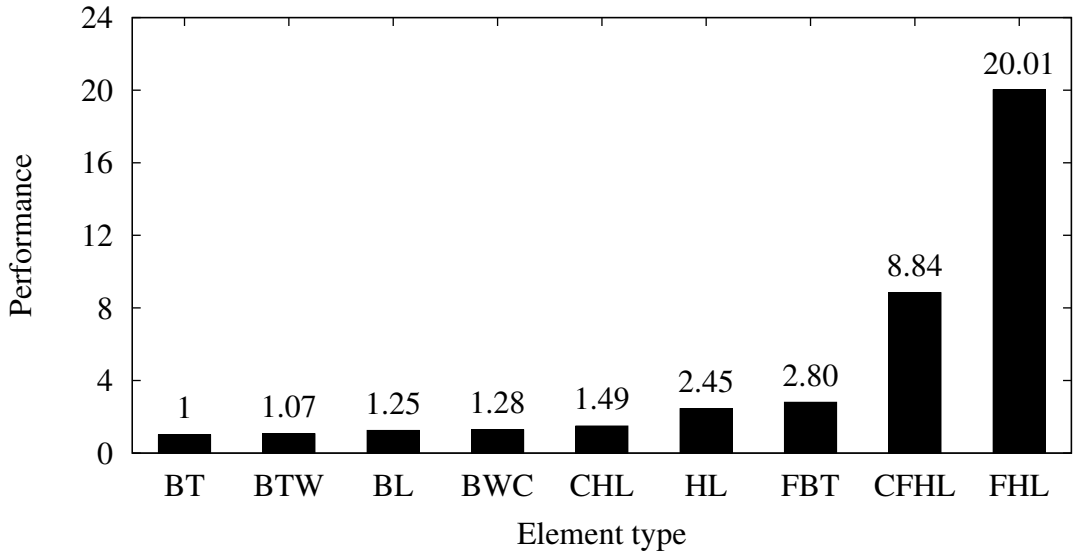

Figure 2-4. Relative cost of the four noded shells available in LS-DYNA where BT is the Belytschko-Tsay shell, BTW is the Belytschko-Tsay shell with the warping stiffness taken from the Belytschko-Wong-Chiang, BWC, shell. The BL shell is the Belytschko-Leviathan shell. CHL denotes the Hughes-Liu shell, HL, with one point quadrature and a co-rotational formulation. FBT is a Belytschko-Tsay like shell with full integration, FHL is the fully integrated Hughes-Liu shell, and the CFHL shell is its co-rotational version.

Element Type	Relative Cost
C^0 triangular shell with four thickness integration points	4
2 node Hughes-Liu beam with four integration points	9
2 node Belytschko-Schwer beam	2
2 node simple truss elements	1
8 node solid-shell with four thickness integration points	11

These relative timings are very approximate. Each interface node of the sliding interfaces is roughly equivalent to one-half zone cycle in cost. Figure 2-4. illustrates the relative cost of the various shell formulations in LS-DYNA.

UNITS

The units in LS-DYNA must be consistent. One way of testing whether a set of units is consistent is to check that:

$$[\text{force unit}] = [\text{mass unit}] \times [\text{acceleration unit}]$$

and that

$$[\text{acceleration unit}] = \frac{[\text{length unit}]}{[\text{time unit}]^2}.$$

Examples of sets of consistent units are tabulated below. For a more comprehensive table, see http://ftp.lstc.com/anonymous/outgoing/support/FAQ/consistent_units .

	(a)	(b)	(c)
Length unit	meter	millimeter	millimeter
Time unit	second	second	millisecond
Mass unit	kilogram	tonne	kilogram
Force unit	Newton	Newton	kiloNewton
Young's Modulus of Steel	210.0E+09	210.0E+03	210.0
Density of Steel	7.85E+03	7.85E–09	7.85E–06
Yield stress of Mild Steel	200.0E+06	200.0	0.200
Acceleration due to gravity	9.81	9.81E+03	9.81E-03
Velocity equivalent to 30 mph	13.4	13.4E+03	13.4

GENERAL CARD FORMAT

The following sections specify, for each keyword command, the cards that must be defined and those cards that are optional. Each card is described in its fixed format form and is shown as a number of fields in an 80 character string. With the exception of "long format input" as described later in this section, most input cards consist of 8 fields with a field length of 10 characters. A sample card is shown below. The card format is clearly stated if the format is different than 8 fields of 10 characters. See also notes on a special "I10 format" at the end of this section.

As an alternative to fixed format, a card may be in free format with the values of the variables separated by commas. When using comma-delimited values on a card, the number of characters used to specify a value must not exceed the field length for fixed format. For example, an I8 number is limited to a value of 99999999 and a larger number with more than 8 characters is unacceptable. A further restriction is that characters beyond column 80 of each line are ignored by the code. Fixed format and free, comma-delimited format can be mixed throughout the deck and even within different cards of a single command but not within a card.

The limits on number of characters per variable and number of characters per line as stated above are raised in the case of long format input. See the description of long format input below.

Example Card.

Card [N]	1	2	3	4	5	6	7	8
Variable	NSID	PSID	A1	A2	A3	KAT		
Type	I	I	F	F	F	I		
Default	none	none	1.0	1.0	0	1		
Remarks	1			2		3		

In the example shown above, the row labeled "Type" gives the variable type and is either F, for floating point or I, for an integer. The row labeled "Default" reveals the default value set for a variable if zero is specified, the field is left blank, or the card is not defined. The "Remarks" row refers to enumerated remarks at the end of the section.

Optional Cards:

Each keyword card (line beginning with "*") is followed by a set of data cards. Data cards are either,

4. *Required Cards.* Unless otherwise indicated, cards are required.

5. *Conditional Cards.* Conditional cards are required provided some condition is satisfied. The following is a typical conditional card:

ID Card. Additional card for the ID keyword option.

ID	1	2	3	4	5	6	7	8
Variable	ABID	HEADING						
Type	I	A70						

6. *Optional Cards.* An optional card is one that may be replaced by the next keyword card. The fields in the omitted optional data cards are assigned their default values.

Example. Suppose the data set for *KEYWORD consists of 2 required cards and 3 optional cards. Then, the fourth card may be replaced by the next keyword card.

All the fields in the omitted fourth and fifth cards are assigned their default values.

> **WARNING:** In this example, even though the fourth card is optional, the input deck may *not* jump from the third to fifth card. The *only* card that card 4 may be replaced with is the next keyword card.

Long Format Input:

To accommodate larger or more precise values for input variables than are allowed by the standard format input as described above, a "long format" input option is available. One way of invoking long format keyword input is by adding "long=**y**" to the execution line. A second way is to add "long=**y**" to the *KEYWORD command in the input deck.

long=**y**: read long keyword input deck; write long structured input deck.
long=**s**: read standard keyword input deck; write long structured input deck.
long=**k**: read long keyword input deck; write standard structured input deck.

The "long=**s**" option may be helpful in the rare event that the keyword input is of standard format, but LS-DYNA reports an input error and the dyna.str file (see *CONTROL_STRUCTURED) reveals that one or more variables is incorrectly written to dyna.str as a series of asterisks due to inadequate field length(s) in dyna.str.

The "long=**k**" option really serves no practical purpose.

When long format is invoked for keyword input, all variables that are less than or equal to 20 characters in standard format are expanded to or remain at 20 characters. There is one exception to the previous rule for alphanumeric variables of 20 characters in length. A20 in standard format becomes A40 in long format. Alphanumeric variables that are longer than A20 in standard format are extended to A160 in long format. The number of input lines in long format is unchanged from standard format, rather only the length of each line changes. For example, long format for the card corresponding to the _ID option of a keyword becomes (I20,A160) whereas standard format is (I10,A70).

Allowing a 20-character field for a numerical variable does not mean that it will allow for 20 significant digits. There are more fundamental limitations at work, particularly when running in single precision. The absolute value of an integer in single precision cannot be larger than $2^{31} - 1 = 2{,}147{,}483{,}647$. The mantissa of a floating point value cannot exceed 2^{23} in single precision and cannot exceed 2^{52} in double precision.

To convert a standard format input deck to a long format input deck, run LS-DYNA with "newformat = long" on the execution line. For example, if standard.k is a standard format input deck,

LS-DYNA i=**standard.k** newformat=**long**

will create a long format input deck called standard.k.long.

To mix the long and standard formats within one input deck, use "+" or "-" signs within the deck. If the execution line indicates standard format, appending "+" to the end of a keyword invokes long format just for that keyword. For example, "*NODE +" in place of "*NODE" invokes a read format of (I20,3F20,2I20).

Similarly, if the execution line indicates long format, appending "-" to the end of a keyword invokes standard format for that keyword. For example, "*NODE –" in place of "*NODE" invokes the standard read format of one line per node (I8,3F16,2I8).

To take this idea a step further, adding a "-" or "+" to the end of the *INCLUDE keyword command signals to LS-DYNA that all the commands in the included file are standard format or long format, respectively.

I10 Format Input:

In some cases, the user may be satisfied in extending 8-character input fields where they exist in standard format input to 10-character fields. If that is all that is required, then invoking the so-called "I10 format" input would suffice.

I10 format input is invoked by appending "i10=**y**" to the execution line or else by adding "i10=**y**" to the *KEYWORD command in the input deck. Or you can invoke I10 format for specific keywords in an input deck by adding the percent sign "%" to the end of the keyword. For example, "*NODE %" in place of "*NODE" invokes a read format of (I10, 3F16,2I10) instead of the standard (I8,3F16,2I8).

To convert a standard format input deck to an I10 format input deck, run LS-DYNA with "newformat = i10" on the execution line. For example, if standard.k is a standard format input deck,

LS-DYNA i=**standard.k** newformat=**i10**

will create an I10 format input deck called standard.k.i10.