10/14/89 M.L.R. (1)

CONTRACTOR REPORT

SAND89—7100
Unlimited Release
UC—32

Users' Guide to SPEEDI: The Sandia Partitioned Engineering Environmental Database Implementation

Topical Report RSI-0330

Elisa M. Kephart, Karen H. Haskell, Grettel M. von Laven
RE/SPEC, Inc.
3815 Eubank NE
Albuquerque, NM 87191

Prepared by Sandia National Laboratories Albuquerque, New Mexico 87185 and Livermore, California 94550 for the United States Department of Energy under Contract DE-AC04-76DP00789

Printed July 1989

SAND--89-7100

DE89 013998

SAND89-7100
Unlimited Release

Users' Guide to SPEEDI: The Sandia Partitioned Engineering Environmental Database Implementation

Topical Report RSI-0330*

by

Elisa M. Kephart
Karen H. Haskell
Grettel M. von Laven

prepared for

Engineering Analysis Department 1520
Sandia National Laboratories
Albuquerque, New Mexico 87185

*Topical Report RSI-0330, prepared by RE/SPEC Inc. under Contract No. 14-2162 with Sandia National Laboratories.

Users' Guide to SPEEDI: The Sandia Partitioned Engineering Environmental Database Implementation

Topical Report RSI-0330

by

Elisa M. Kephart
Karen H. Haskell
Grettel M. von Laven

prepared for

Engineering Analysis Department 1520
Sandia National Laboratories
Albuquerque, New Mexico 87185

June 1989

TABLE OF CONTENTS

LIST OF TABLES

1 INTRODUCTION

This report is an overview of the general use of the *Sandia Partitioned Engineering Environmental Database Implementation*, or SPEEDI. This software implements the evaluation and design which are documented in earlier reports [Haskell and Helman, 1987; Haskell et al., 1987]. We briefly summarize some of the discussion and notation used in those reports for the convenience of the reader.

Sandia currently maintains a large Environmental Databank consisting of data represented in a variety of formats. It was desired to convert from a hard copy system to a digitized system. A two-level system was proposed and implemented. The top level consists of a stable, read-only environment for engineers seeking information from the Environmental Databank. The lower level, the Analysis Databank, is an organized and automated work space to be used by analysts who require read-write access to large quantities of data. As these data are processed and reviewed, some may be selected for permanent retention in the top level. At this point, Quality Assurance (QA) restrictions may be applied.

During Phase 1 the attributes and interrelationships of the data were identified. Several categories of users were identified and their needs were summarized. In addition to the analysts who access the Environmental Databank and use the Analysis Databank, two specific sets of tasks were identified to be the responsibility of the *System Database Administrator* (SDBA) and the *Environmental Database Manager* (EDBM). The SDBA must have a knowledge of the host computer system and the commercial DataBase Management System (DBMS) which manages the data. This individual will maintain the DBMS software, authorize users, maintain the database schema and data dictionary, and perform database backup procedures. The EDBM must have a thorough understanding of the nature of the data being stored, as well as some understanding of the DBMS. This individual will be responsible for entering data into the Environmental Databank and responding to outside requests for data retrieval.

Much of the data stored in the Analysis Databank originates in the *Standard ASCII Test and Data Exchange Format* (SATADEF) [Adams, 1985a]. The analysts currently use GRAFAID [Adams, 1985b] extensively to manipulate these data. A revised SATADEF is the basis for the choice of many of the attribute fields which describe the data stored in the database [Adams, 1989]. The capabilities of GRAFAID have been enhanced by interfacing the SPEEDI and GRAFAID packages. One of these enhancements is that a single data set may be accessed by multiple users via SPEEDI. Previously, multiple copies of data sets were required for use by multiple analysts via GRAFAID.

In order to facilitate access to and effective use of the data stored in the database, each data set is uniquely identified by a *Record Identification and Data Definition Label* (RIDDL). For convenience, the analysts have been provided the capability of defining a RIDDL mask which allows them to efficiently access the data through a

more compact subRIDDL, much as they have used the GRAFAID data identifiers. The methods for using the RIDDL are detailed in this report.

The organizational base for storing and retrieving data is the commercial Database Management System (DBMS) INGRES marketed by Relational Technology Incorporated (RTI). The DBMS stores the relations (tables of data) which define the (logical) databanks. In addition, the DBMS stores most of the organizational material used for the operation of SPEEDI.

The operating system manages all of the file space associated with the SPEEDI system. Since the actual data are stored in VAX/VMS [DEC, 1986] files owned by each user, but hidden from the user, any disk file quotas in effect in the user's data area determine the amount of data each user can store.

This report documents the system which has been implemented from the user's point of view. Chapter 2 presents the notation used and outlines the general capabilities of SPEEDI. Chapter 3 presents the SPEEDI program and the commands available to SPEEDI users. The subcommands available for defining RIDDL Masks and using subRIDDLs are discussed in Chapter 4. Chapter 5 discusses a companion program which facilitates the communication between analysts and the organizations which perform tests and data reduction.

It should be noted that SPEEDI (as currently implemented), must be installed on a VAX/VMS system which also has Version 5 of INGRES installed. The new user authorization procedure and the system resource limits and quotas required are fully documented in this report's companion, *System Guide to SPEEDI* [Kephart et al., 1989]. As with any other analysis code, it is the responsibility of the user to ensure that only unclassified data are loaded into an unclassified SPEEDI installation. Both classified and unclassified data may be loaded into a classified SPEEDI installation.

2 ORGANIZATION OF THE SPEEDI SYSTEM

2.1 NOTATION CONVENTIONS USED IN THIS DOCUMENT

In order to describe the SPEEDI command syntax as concisely as possible, the following notation conventions are used throughout this document.

- SPEEDI command verbs and keywords are shown in uppercase. When entering a command verb or keyword, only enough characters to uniquely specify the word are required.
- Optional command arguments are enclosed in braces ({}).
- A vertical bar (|) separating keywords indicates that only one of the keywords should be used in a single instance of the command.

2.2 SPEEDI SYNTAX

SPEEDI commands consist of a command verb which may be followed by a list of command arguments. Command arguments consist of parameters, parameter lists, key words, or a combination of these. Commands may be entered in lower or upper case. Several special characters play particular roles in the syntax of SPEEDI commands entered by the user, and they are summarized below:

- Major command fields (that is, command verbs and command arguments) are shown separated by commas. A space may replace a comma.
- The colon (:) is used to separate parts of a single command argument. The colon must appear in the appropriate context when the command is entered.
- The slash (/) is used to separate parameters in a parameter list. The slash must appear in a parameter list consisting of more than one parameter.
- The backslash (\) serves as the line continuation character, indicating that the input line ends but the command is continued on the next line.
- The double quote (") is the character string delimiter. If a double quote mark is to be part of a character string, it must be represented by two successive double quote marks.
- The single quote (') is the symbol substitution character.
- The exclamation point (!) serves as a comment delimiter. Anything after the exclamation point on an input line will be ignored by the lexical analyzer.

Whenever SPEEDI encounters one of the seven special characters discussed above, an attempt will be made to interpret the character according to its use within SPEEDI. If a user wants to include one of these characters as part of a character string, the character string must be enclosed by string delimiters (double quotes).

Several other characters have special meanings in the context of SPEEDI commands. Some of these are listed below:

- On a VAX/VMS system, SPEEDI will recognize Ctrl-C entered by the user to cancel execution of certain commands and display the prompt.
- The wildcard characters * and % will be recognized just as they are in a VMS file specification. In addition, ? is recognized as % in VMS. That is to say, * is a wildcard which matches one or more characters, and % and ? are wildcards which match exactly one character.

Other special characters are discussed in the context of their use in later sections and chapters of this report.

2.3 THE SPEEDI KEYBOARD

INGRES forms are used within SPEEDI for all menu-driven utilities. When forms are used, menu options are displayed at the bottom of the user's screen. In addition to the menu options and their associated keys, the INGRES Forms Run-Time System (FRS) defines a default set of keystrokes for moving around on a form. FRS and these default keystrokes are explained in detail in the *INGRES Terminal User's Guide* [RTI, 1986]. As discussed in Section 2.7, the default terminal designation for SPEEDI users is "vt100f." Note that a differentiation should be made between the actual type of terminal and the terminal designation. A brief description of the default keystrokes is presented below for the user's convenience.

- Moving Around on a Form
  - Use the space bar only when incorporating a space within a field or within a command.
  - Use Tab to move the cursor to the next field on a form.
  - Use Ctrl-P to move the cursor to the previous field on a form.
  - Use the left and right arrow keys to move the cursor within a field without altering data already entered in that field. Use Ctrl-H and Ctrl-L to move the cursor left and right, respectively, if the terminal does not support the use of arrow keys.
  - Use the up and down arrow keys to move the cursor within a form. Use Ctrl-K and Ctrl-J to move the cursor up and down, respectively, if the terminal does not support the use of arrow keys.

- Use the arrow keys to move around in a table field if the terminal is designated as a "vt100f." "vt100nk." "vt100k," or "vt220." Otherwise, if arrow keys are not supported, use Ctrl-K and Ctrl-J to scroll up or down one row at a time.
- Use <dash> on the VT100 keypad (to the right of the alphanumeric keyboard) to scroll backward one full page. Equivalently use Prev Screen if the terminal is designated as "vt220," or Ctrl-G if function keys and arrow keys are not activated on the terminal.
- Use <comma> on the VT100 keypad (to the right of the alphanumeric keyboard) to scroll forward one full page. Equivalently use Next Screen if the terminal is designated as "vt220," or Ctrl-F if function keys and arrow keys are not activated on the terminal.
- Use Ctrl-L to scroll a form to the left if the form is too wide to fit on the screen. Use Ctrl-O instead if function and arrow keys are not activated on the terminal.
- Use Ctrl-H to scroll a form to the right if the form is too wide to fit on the screen. Use Ctrl-U instead if function and arrow keys are not activated on the terminal.
- Use PF1 (or Esc if function and arrow keys are not activated on the terminal) to move the cursor to the menu line in order to type the name of the operation desired.

• Entering and Editing Data on a Form

- Use Ctrl-E to toggle between the default overwrite mode and insert mode. Use Insert Here instead of Ctrl-E if the terminal is designated as "vt220."
- Use Tab to move to the next field in a form, but use Ctrl-N to move to the next row in a table field. Use Return to delete characters under and to the right of the cursor in the current field and move to the next field.
- Use Ctrl-U (or Ctrl-B if function and arrow keys are not activated on the terminal) to move the cursor to the next word in a field containing character data.
- Use Ctrl-R (reverse) to move the cursor to the previous word in a field containing character data.
- Use Ctrl-X to clear the contents from a field in a form.
- Use Del or Rubout to delete the character immediately to the left of the cursor.
- Use Ctrl-D on a terminal designated as "vt100" (or Remove if the terminal is designated as "vt220") to delete the character directly under the cursor.
- Use Ctrl-V to edit a text field using a text editor. Upon exit from the text editor, the edited text is returned to its field.

2.4 DATA TYPES

We present the data types with which SPEEDI deals, the format in which they will be stored internally, and the syntax by which they must be specified by the user.

Character. Character data can consist of any printable character, including letters, numbers, and special characters. If a user wishes to include a space or one of the special characters discussed above in a character datum, the character datum must be enclosed by double quote marks. In the absence of quote marks, an attempt will be made to interpret the special characters according to their definitions within SPEEDI.

Integer. Integer data must be specified as a string of digits optionally preceded by a sign.

Real. Real data may be specified in any of the formats recognized by FORTRAN, that is, with or without a sign or decimal point, or in scientific notation.

Complex. Complex numbers are stored within the system in the representation in which they are input. The user may specify that complex values be returned either as they are stored or in either of three numerical representations. The user may specify that representation to be *cartesian*, that is a pair (val_1, val_2) of two real numbers representing the real and imaginary parts of the complex number. Alternatively, (val_1, val_2) will represent *magnitude* and *phase*, respectively, where *phase* is specified in either degrees or radians. The syntax for specifying alternate interpretations of the pair is discussed in Chapter 3.

Date. Dates are stored internally in a consistent format. They can be specified by the user in any of the following formats:

dd-mmm-yy
dd-mmm-yyyy
mmddyy
mm/dd/yy
mm/dd
mm-dd-yy
mm-dd
yy.mm.dd
TODAY

Using the string TODAY will supply the current date in the command being entered.

Time. The time will be stored internally in a consistent format. It may be entered by the user in any of the following formats:

hh:mm:ss
hh:mm
NOW

Using the string NOW will supply the current time in the command being entered. All times specified should be in twenty-four hour format.

2.5 THE ENVIRONMENTAL DATABANK

The Environmental Databank is a sub-databank of the SPEEDI system corresponding to the manual *DOE/DOD Environmental Data Bank Index* [Davidson et al., 1985]. At this time none of the Data Bank data have been digitized and stored on the system. SPEEDI is serving as an automated index tool for retrieving information about what is stored in the Environmental Databank. The SPEEDI entries will direct the user to the appropriate Databank access numbers for the entries desired. Once data sets are retained in the Environmental Databank, the data set itself will be returned rather than its access number.

A subset of the Environmental Databank is the Transportation Databank. A user may limit access to this subset in order to limit retrievals to transportation data only.

2.6 THE ANALYSIS DATABANK

The Analysis Databank is a sub-databank of the SPEEDI system which serves as an automated workspace for analysts. Authorized analysts can store data in the SPEEDI database, retrieve it according to a variety of index methods, move data to secondary storage, and delete data sets. In addition, the analyst has available a variety of tools to manipulate and display the data managed by SPEEDI.

2.7 PREPARING TO RUN SPEEDI

In order to use SPEEDI, a user must first be authorized by the SDBA. To run the program it is necessary to first execute a command procedure which will define a number of logical names and set up the database access. The system manager will probably have already set this up to execute automatically upon login. If not, it is suggested that the regular SPEEDI user execute this command procedure from within the usual LOGIN.COM file. The procedure can be executed as follows:

@SPEEDI$ROOT:[PROD]SPLOGIN

This procedure designates the user's terminal as "vt100f," which will work on VT100, Envision, Lear Siegler, and VT220, and VT320 terminals. A user with another type of terminal should contact the SDBA for the method of designating another terminal type.

2.8 SUMMARY OF SPEEDI SOFTWARE MODULES

The SPEEDI system consists mainly of a program (SPEEDI) which interacts with the user to store and retrieve data from the INGRES database tables. In addition, there is a stand-alone program (DRRF) available to assist analysts in communicating requests to the test and data reduction organizations which provide the data stored in SPEEDI. These capabilities are documented in the remaining chapters of this report.

3 RUNNING SPEEDI

3.1 INVOKING SPEEDI

Once the user has executed the SPEEDI login command procedure (Section 2.7), the software can be invoked from the operating system level with the command

SPEEDI

If SPEEDI is started from within another program (such as GREEDI), its operation will be much the same as if it had been called from the operating system level. There are a few necessary restrictions when operating in this mode; these will be noted in the discussion of the commands affected. Under certain circumstances, the INGRES database engine, and hence SPEEDI, may become inaccessible. In such an event, notify the SDBA to rectify the situation.

3.1.1 The SPEEDI Control File and SPEEDI Procedures

A user can initialize a SPEEDI session by including a sequence of SPEEDI commands in a *control file*. If a control file called SPCONTROL is in the directory from which SPEEDI is invoked, it will be executed during the initialization. On a VMS system this may be any file pointed to by the logical name SPCONTROL. If SPCONTROL is not defined as a logical name, a file called SPCONTROL.SPP in the current default directory will be sought. If no control file is found, SPEEDI will be initialized with default parameters.

The control file is a specific instance of a SPEEDI *procedure*. A procedure is simply a sequence of SPEEDI commands. Any of the SPEEDI commands can be executed from a procedure. SPEEDI procedures can be created and edited from the operating system level. If a file extension other than .SPP is used, the extension must be explicitly specified when the procedure is executed (see the EXECUTE command).

3.1.2 Entering SPEEDI Commands

The user may enter SPEEDI commands in three ways: from a procedure, directly in response to a SPEEDI prompt, or as a choice from a SPEEDI menu. The command prompt displayed during a SPEEDI session will consist of a character string indicating the program level followed by the character >, for example "SPEEDI>". If SPEEDI does not have sufficient information to complete the execution of a command, the user will be prompted to supply that information. Such information must be supplied interactively, even if the command was executed from a SPEEDI procedure.

3.1.3 Key Sequences Recognized from SPEEDI Menus

Each SPEEDI menu indicates which commands are available to the user at the particular level, and which keys should be used to invoke those commands. However, there are certain key sequences which have special meaning to the INGRES FRS and are always available (see Section 2.3).

3.2 GLOBALLY AVAILABLE COMMANDS

The following commands are recognized at any SPEEDI program level.

DIRECTORY	SEND	EXIT
EXECUTE	SET	QUIT
HELP	SHOW	STOP
OSC	TYPE	

3.2.1 The DIRECTORY Command

This command can be used to obtain information about the data in the database without actually retrieving the data. The syntax of the command is

DIRECTORY{, *subRIDDL*, FIELD:$field_1/field_2/\ldots$, OUT:*device*}

The command DIRECTORY alone will display the RIDDLs for all of the data sets which satisfy the active RIDDL mask. If *subRIDDL* is specified, the command will display the RIDDLs of all data sets which satisfy that subRIDDL. Specific data attribute fields will be displayed for fields $field_1$, $field_2$, etc., if they are listed. If a full directory of all fields is desired, the wildcard character * can be used in place of the field names, that is FIELD:*. By default the directory will be displayed on the terminal. Alternatively, it can be sent to the output device designated by *device*.

3.2.2 The EXECUTE Command

This command is used to execute a previously defined and saved SPEEDI procedure. The syntax of the command is

EXECUTE, *procname*

where *procname* is the name of a previously saved SPEEDI procedure. A VAX/VMS directory specification can be given as part of *procname*. Otherwise, it is assumed to be stored in the directory from which the user has invoked SPEEDI. If a file extension is not supplied, it will be assumed to have the default extension .SPP. If the VERBOSE flag has been activated, each command in the procedure will be echoed to the terminal as it is executed. Otherwise, only error messages and prompts for additional input will be displayed.

3.2.3 The HELP Command

SPEEDI provides online help for the commands it supports. To see a list of the SPEEDI commands available from the current program level, enter

HELP

with no parameters. To obtain more detailed information about a particular command, enter

HELP, *command*

where *command* is one of the available SPEEDI commands.

3.2.4 The OSC Command

OSC (Operating System Command) is provided so that a user can execute an operating system command without leaving SPEEDI. The syntax for the command is

OSC, *command*

where *command* is the operating system command which will be executed. It is not necessary to enclose *command* in quotes, even if it contains spaces or special characters. After the command is executed, control will return to SPEEDI.

3.2.5 The SEND Command

Messages can be sent to any SPEEDI user using the SEND command as follows:

SEND, *username*

where *username* is the username of the SPEEDI user to whom the message should be sent. If the *username* is SPEEDI, then the message will be sent to the SPEEDI manager. The prompt MESSAGE> is displayed in response to this command. The user can then enter as many message lines as desired. Entering Crtl-Z at the MESSAGE> prompt terminates message input. The message is stored in a system-wide message table. Whenever a user is running SPEEDI interactively, the message table is checked before each prompt is displayed. If that user has a message in the table, it is immediately displayed and then removed from the table. This message facility is not as powerful as most operating system MAIL facilities, but is provided here as a convenience to the user.

3.2.6 The SET and SHOW Commands

The SET commands modify the operating environment of SPEEDI to suit the user. The SHOW commands display the status of the corresponding values. The following are values which may be set. Their default settings are indicated.

SET, COMPLEX, ASIS|CARTESIAN|POLAR{, DEGREES|RADIANS} — Sets the format of complex data. If ASIS is specified, complex data are loaded into the database as they are stored. This is the default interpretation. If CARTESIAN is specified the data are loaded as (*real,imaginary*) pairs. If POLAR is specified the data are loaded as (*magnitude,phase*) pairs. In the latter case the user can further specify the representation of *phase* as either DEGREES or RADIANS. The default is RADIANS.

SHOW, COMPLEX — Show the current interpretation of a complex pair entered by the user.

SET, DATABANK, EDB|TRA|ADB — Activate the databank indicated. This causes all database retrievals to restrict the search to the databank indicated. The default is ADB for the SD and DD RIDDL types and EDB for the DB RIDDL type. Chapter 4 describes these RIDDL types in detail.

SHOW, DATABANK — Show the current databank.

SET, DERIVATION, TOUCH|INCLUDE|EXACT|NONE — Set the mode by which matches between subRIDDLs and derived data sets are performed. TOUCH retrieves all base data sets and derivations which contain some data set matching the specified subRIDDL. INCLUDE retrieves all base data sets and derivations which contain some subset of the data sets matching the specified subRIDDL. EXACT retrieves all base data sets and derivations which contain exactly the data sets which match the specified subRIDDL. (EXACT is currently not implemented.) NONE will retrieve only base data sets, and no derived data sets. The default is TOUCH, which (except for NONE) is by far the fastest operation to perform. Retrievals using INCLUDE are the next least time-consuming to perform, with EXACT retrievals being quite slow.

The differences among these four modes of retrieval can best be illustrated by an example. Suppose there are the base data sets and derivations illustrated in Figure 3-1. Further, suppose that the user has specified a subRIDDL which matches Base Data #1, Base Data #2, and Base Data #3, but *does not* match Base Data #4. Under the four retrieval modes the following base and derived data sets would be retrieved:

1. TOUCH would retrieve base data sets #1, #2, and #3, and derivations #1, #2, #3, and #4.
2. INCLUDE would retrieve the same three base data sets and derivations #1, #2, and #4.

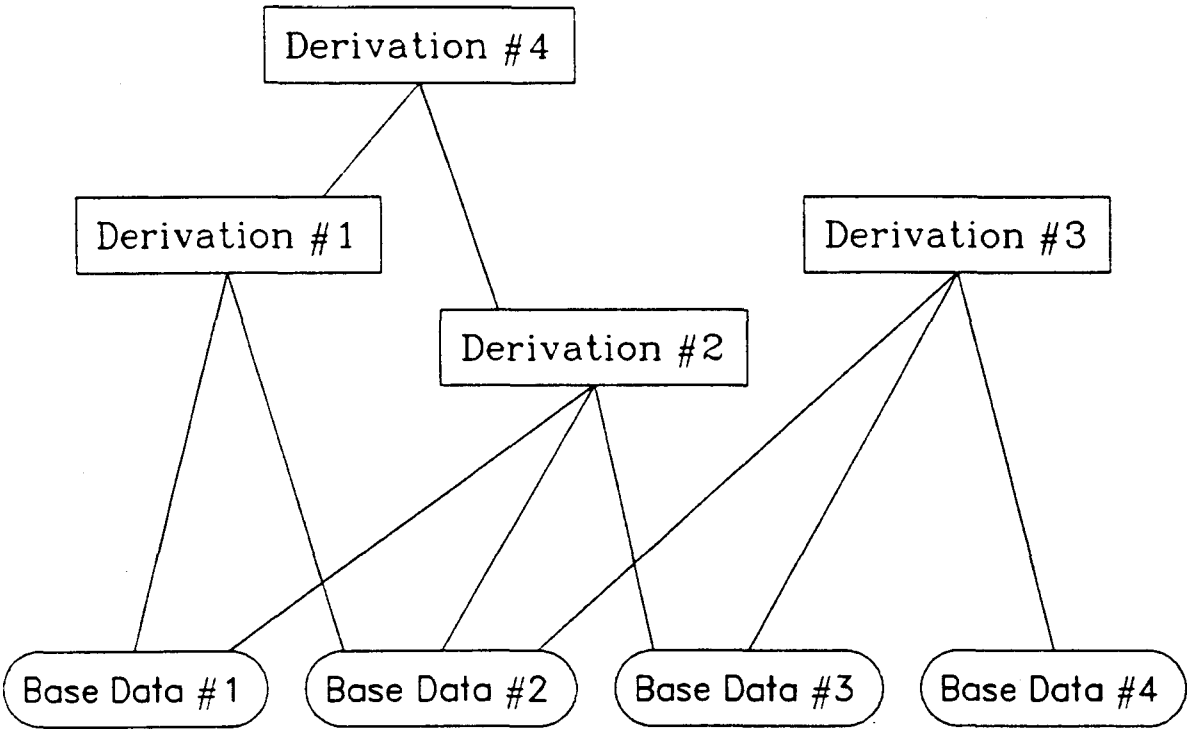

Figure 3-1. Base Data and Associated Derivations for SET DERIVATION example.

3. EXACT would retrieve the same three base data sets and derivation #2.
4. NONE would retrieve only the three base data sets.

SHOW, DERIVATION — Show the current mode of derived data set retrieval.

SET, LINELENGTH, *num* — Set the SPEEDI output line length to *num*. The default value of *num* is 80.

SHOW, LINELENGTH — Show the current value of the SPEEDI output line length.

SET, MENU, ON|OFF — Set the mode by which the user will interact with SPEEDI. If MENU is OFF, the user will enter all input to SPEEDI through commands (like the ones listed here) or in response to prompts from SPEEDI. This is the default mode of operation. If MENU is ON, a menu of possible selections will be displayed from which the user may make a choice. The same SPEEDI capabilities will be available in either case. If the menu option is changed,

that change will take effect the next time the user is at one of the SPEEDI program levels.

SHOW, MENU — Show the current menu mode value.

SET, OUTLEVEL, LOW|MEDIUM|HIGH — Set the output level for RIDDL display. If the output level is LOW, only the values of the explicit fields in the current active mask and those fields overridden in the current subRIDDL are reported. This is the default output level. If the output level is MEDIUM, the values of all RIDDL and overridden fields are reported. If the output level is HIGH, the field names and values are reported for all RIDDL and overridden fields.

SHOW, OUTLEVEL — Show the current output level for RIDDL display.

SET, PAGELENGTH,*num* — Set the SPEEDI output page length to *num*. The default value of *num* is 23.

SHOW, PAGELENGTH — Show the current value of the SPEEDI output page length.

SHOW, PROCEDURES{, DIRECTORY:*directory*} — Show the names of the currently saved SPEEDI Procedures in the directory specified by *directory*. If a directory is not specified in the command, the names of the procedures in the current default directory will be displayed. (To execute procedures, see the EXECUTE command.)

SET, RIDDLTYPE, DB|SD DD — Set the SPEEDI RIDDL type. The default RIDDL type is determined when the user is authorized as a SPEEDI user.

SHOW. RIDDLTYPE — Show current SPEEDI RIDDL type.

SET, SYMBOL, *sym:string* — Set a local symbol *sym* that will be replaced by *string* in any command line which SPEEDI processes. If a defined symbol is used at the beginning of a command line, it will be automatically substituted. If it is used elsewhere in a command line, it must be delimited by the symbol substitution character (single quote). Symbol definitions will remain in effect for the duration of a single SPEEDI session.

SHOW, SYMBOL{, *sym*} — Show the current substitution for the SPEEDI symbol *sym*. If *sym* is not specified, all currently defined SPEEDI symbols will be displayed.

SET, THRESHOLD, *num* — When a general search of the database results in the selection of several data sets, SPEEDI requires the user to explicitly indicate continuation of the command if *num* or more data sets were found. The default value of *num* is five.

SHOW, THRESHOLD — Show the current value of the continuation threshold.

SET, VERBOSE, ON|OFF — If VERBOSE is ON, each command executed from a SPEEDI procedure is echoed to the terminal as it is executed. Otherwise, successfully executed procedures are not echoed. VERBOSE, OFF is the default mode of operation.

SHOW, VERBOSE — Show the current verbose mode value.

3.2.7 The TYPE Command

This command can be used to display a data set to the terminal or to another device. The syntax of the command is

TYPE, *subRIDDL*{,OUT:*device*,NOCONFIRM}

where *subRIDDL* specifies a subRIDDL of the data which are to be retrieved from the database and displayed, and *device* specifies the name of a system device, logical name, or disk file to which the output should be directed. If *device* is not specified, the information is written to the standard output device (e.g., the terminal for an interactive session or the log file for a batch session). If NOCONFIRM is specified and more than a threshold number of data sets match the subRIDDL, only a threshold number of the data sets are displayed. Without NOCONFIRM each matching RIDDL is displayed with the following options:

- YES — Display the data set corresponding to the given RIDDL.
- NO — Do not display the data set.
- QUIT — Terminate execution of the current TYPE command.
- ALL — Display all matching data sets with no further prompting.
- THRESHOLD — Display the threshold number of matching data sets with no further prompting.

In any case the number of data sets matching *subRIDDL* is reported before the display begins. The format of the display is determined by the kind of data being displayed.

3.2.8 The EXIT Command

The EXIT command returns the user to the previous program level. If there are incomplete operations outstanding, the user is prompted to complete them before the exit. For example, before exiting from the RIDDL program level, the user may be prompted to save the active RIDDL mask. Exiting from the SPEEDI program level returns the user to the operating system (or to GRAFAID, if SPEEDI has been invoked from within GREEDI).

3.2.9 The QUIT Command

The QUIT command returns the user to the previous program level without finishing any incomplete operations. For example. when quitting the MODIFY program level, all data attribute fields remain unchanged. If incomplete operations are outstanding, the user is notified and may continue to quit or resume working at the current program level.

3.2.10 The STOP Command

The STOP command returns the user directly to the caller of SPEEDI skipping over previous program levels. If returned to a calling program, the fact that a STOP was requested is indicated so that the caller can in turn exit to the operating system. If there are incomplete operations outstanding the, user is prompted to complete them before the exit.

3.3 SUMMARY OF SPEEDI COMMANDS

The following commands are available at the main SPEEDI program level.

ADMINISTER	MANAGE
EXPORT	MODIFY
GRAFAID	RIDDL
IMPORT	

The syntax and implementation details of these commands will be discussed in the sections which follow.

3.3.1 The ADMINISTER Command

This command is used to access the subcommands which provide capabilities for the SDBA and the EDBM to perform administrative operations in SPEEDI. It can be invoked only by an individual with the privilege to do so. The syntax for the command is simply

ADMINISTER

The subcommands available from the ADMINISTER program level are fully documented in the *System Guide to SPEEDI.*

3.3.2 The EXPORT Command

The *Database Output Utility* is the utility which allows users to obtain data in various output formats from either the Analysis Databank or the Environmental Databank. It can be invoked using the EXPORT command. The syntax for the command is

EXPORT, *output-file*{, *output-format*}

where *output-file* specifies the file or device to which the data are to be written and *output-format* indicates the desired format of the output (either SATADEF2 or GRAFNEU). If *output-format* is not specified, the default SATADEF2 is assumed.

3.3.2.1 Output Data Formats

The following output formats are available.

SATADEF2: (Standard ASCII Tape And Data Exchange Format, Version 2) Nearly all of the data in the Analysis Databank and some of the data in the Environmental Databank contain information in the attribute fields which allow them to be written directly in SATADEF 2.0 format. The output utility is capable of exporting this data outside of SPEEDI to a SATADEF 2.0 tape or file. Data which were not originally compatible with the SATADEF format will be more difficult to output in that format. The software makes every effort to produce SATADEF output for all types of data in the Environmental Databank. In some cases incomplete SATADEF headers may result.

GRAFNEU: (GRAFAID NEUtral File Format) Because many analysis programs which are used to manipulate these data are prepared to accept GRAFAID neutral file input, the ability to output data from SPEEDI in a compatible format is provided. The current GRAFAID neutral file format is documented in the *GRAFAID Code User Manual.* The GRAFAID neutral file format does not currently allow for all of the information that are present in the data attribute fields. Therefore, if such data are reloaded into SPEEDI, the user will be required to supply the missing attribute information.

3.3.3 The GRAFAID Command

In order to provide an integrated analysis and data access environment, GRAFAID and SPEEDI have been combined to form *GREEDI*, a Graphical Resource for an Engineering Environmental Database Implementation. When executing GREEDI, the user enters the GRAFAID partition by default. The user may specify GREEDI/SPEEDI when invoking GREEDI to enter the SPEEDI partition directly.

In order to access the GRAFAID analysis tools from the SPEEDI partition the user may enter

GRAFAID

The user then has access to all of the normal capabilities of the GRAFAID package. The user also has access to the SPEEDI tools from GRAFAID by entering

SPEEDI

The user then has all of the normal capabilities of the SPEEDI package. When the user exits SPEEDI, control is returned to GRAFAID if SPEEDI was invoked by GRAFAID or to the operating system if SPEEDI was invoked directly.

3.3.4 The IMPORT Command

The *Database Load Utility* is the mechanism by which data are loaded into the Analysis Databank. It can be invoked using the IMPORT command. The syntax for the command is

IMPORT{, *input-file*, *input-format*, KEY:*key-file*}

where *input-file* specifies the file or device from which the data are to be read; *input-format* indicates the format of the input data (either SATADEF1, SATADEF2, or GRAFNEU); and *key-file* specifies the name of a SPEEDI procedure which will provide input to the Database Load Utility. If *input-format* is not specified, the default value of SATADEF2 is assumed. The *key-file* is a SPEEDI procedure which can contain certain SPEEDI commands discussed in the next section. If no *key-file* is supplied or the one supplied does not contain all of the required information, the user will be prompted to provide any missing information interactively. Note that if a user is loading data from a batch job and the *key-file* does not supply all of the required information for a data set, that data set will not be loaded into the database. If IMPORT is entered with no command arguments, the user will be placed at the IMPORT program level. The commands available at the IMPORT program level are the same as those permitted in a *key-file* and are discussed in the next section.

Each data set loaded into the Analysis Databank must contain identifying information for the attribute fields which are included in every RIDDL and for the unique attribute fields required for the particular RIDDL type. There is also optional information which might be supplied for other data attributes. If the value supplied for a particular data attribute is not from the permitted vocabulary for that field, the user will be prompted with the choice of adding it as a user-defined value in the permitted vocabulary or supplying a valid value. Chapter 4 presents a

detailed discussion of the data attribute fields and the RIDDL. The RIDDL type will be determined automatically by the format of the input data. The data formats SATADEF1 [Adams, 1985a] and SATADEF2 will use the SATADEF (SD) RIDDL type; GRAFNEU will use the Derived Data (DD) RIDDL type. The RIDDL types which are currently defined are described in Chapter 4.

The database load works in the following way. Once the user has specified the data attribute information and requested that the data be loaded, an attempt is made to load the data into the database. If the attribute field information supplied is not sufficient to uniquely identify the data set in the database, the data set will not be loaded. If the program is being run in batch, the data set will be skipped and the program will try the next data set. Otherwise, the program will return to the user with a series of menus (regardless of whether the user has chosen menu mode) indicating the data attribute information that has been supplied and allowing the user to provide additional attribute data to further identify the data. Another attempt to load the data can then be made. This sequence of events continues until either the data can be successfully loaded or the user enters the QUIT command.

3.3.4.1 Input Data Formats

The data being loaded can originate in three formats. The data formats recognized are as follows:

SATADEF1: (Standard ASCII Test and Data Exchange Format, Version 1) A great deal of data will come into the Analysis Databank from SATADEF tapes. This utility provides the capability to read data written in SATADEF Version 1. Since the SATADEF 1.0 headers do not contain all of the information required to store data in SPEEDI, it is necessary for the individual performing this operation to explicitly input many of the attributes necessary to store the data in the database.

SATADEF2: (Standard ASCII Test and Data Exchange Format, Version 2) Data written in the SATADEF Version 2 format can be loaded into the database with much less input from the user. These revised SATADEF headers provide nearly all of the attributes required to store the data in the database if they have been properly completed. The Data Reduction Request Facility discussed in Chapter 5 can be used by analysts when communicating with test and data reduction organizations to ensure that SATADEF data are written as completely as possible.

GRAFNEU: (GRAFAID NEUtral File Format) The GRAFAID neutral file format is currently used for communicating data among various analysis packages in use by analysts. Thus, much of the data currently stored in other formats can be written into GRAFAID neutral files and subsequently loaded into the database. Since GRAFAID does not supply all of the information required for the database, it must be provided explicitly to the Database Load Utility.

3.3.4.2 Supplying Information Interactively or via the Key File

At the IMPORT program level or in the key file, the user can specify values for any of the attribute fields for a data set being imported. The key file is similar in format to the SATADEF 2.0 Header Definition File. However, there are certain crucial differences which should be noted. Some of the commands listed below are meaningful only within a key file and some only from the IMPORT program level. This will be indicated with each command description. In some key file statements the names of SPEEDI attribute fields must be specified. Each of these field names must be one of the indexable attribute fields listed in Appendix A. Only sufficient characters to uniquely identify the field name need be specified. Care should be taken to distinguish between the names of attribute fields and their values in the following discussion.

DEFINITION, *defname, iversion* — The DEFINITION statement marks the beginning of a key file and assigns the name *defname* to the key file. It is permissible for *defname* to be blank. The *iversion* must be present for SATADEF input to indicate which SATADEF header version is used, either SATADEF1 or SATADEF2.

AUTHOR, *username* — (NOTE: key file only) The AUTHOR statement is used to specify the author of the key file. It is treated as a comment field and is not processed by SPEEDI.

DATE, *keydate* — (NOTE: key file only) The DATE statement is used to specify the date on which the key file was prepared. It is treated as a comment field and is not processed by SPEEDI.

DEFAULT, *fieldname:value* — The DEFAULT statement defines the values for default attribute fields which may or may not vary from data set to data set. Any number of these statements can appear. The values in attribute fields defined by these statements are normally fixed for all data sets or they may serve as a template that may be replaced by variable fields. The field specified by *fieldname* is defined to contain *value* by default. This can be used to override the contents of a particular field in the input file, as well as to supply a value for a missing field.

PACKET, *packet-name, field, bp1, bp2* — The PACKET statement marks the beginning of a packet definition. It is a specification for a group of attribute fields whose values are dependent upon a specific input header field. This statement assigns the name *packet-name* to the packet and defines the packet's independent input header field. The input header *field* is specified in the form H.F, where H is the SATADEF header table number and F is the field number within that header. If the *field* is not specified, then the block pointers *bp1* and *bp2* must be present to specify particular character positions within the SATADEF header to be used as the independent header value.

PARAMETER, *independent-value* — The PARAMETER statement defines the value of the packet's independent input header field, *independent-value*.

ENTRY, *attribute-field, attribute-value* — The ENTRY statements following a particular PARAMETER statement define a series of *attribute-value* values for the dependent fields named by *attribute-field*. The ENTRY statement can be repeated as many times as necessary to define all values of the attribute fields which depend on the independent input header field. The next PARAMETER, PACKET, or END statement marks the end of the ENTRY statements associated with the previous parameter value.

COMMENT, *comment-text* — (NOTE: key file only) The COMMENT statement may be entered in a key file to help document the Key File. Alternatively, a comment statement may be started with the character ! instead of the COMMENT keyword.

ADDVALUE, *fieldname, newval, meaning* — This command can be used to add values to the user-defined permitted vocabulary for a particular attribute field. It is fully documented in Section 3.3.5.

END — (NOTE: key file only) The END statement marks the end of the key file.

The structure of the key file is illustrated as follows:

DEFINITION
AUTHOR
DATE
DEFAULT *first default field*
.
.
.
DEFAULT n^{th} *default field*
! end default definitions and begin first variable packet definition
PACKET *first packet and independent input header specification*
! PARAMETER and ENTRY sets are repeated until all possible
! parameters and their dependent attributes are specified
PARAMETER *first independent input header field value*
ENTRY *first field associated with first* PARAMETER *value*
.
.
.
ENTRY n^{th} *field associated with first* PARAMETER *value*
PARAMETER *second independent input header field value*
ENTRY *first field associated with second* PARAMETER *value*
.
.
.

ENTRY n^{th} *field associated with second* PARAMETER *value*

.

.

.

! end first packet definition

PACKET *second packet and independent input header specification*

.

.

.

END

Note that the key file format is so general that redundant specifications of a given attribute field are possible. Ideally, a proper key file should not have redundancies. However, if they do exist the following interpretation rules will apply:

1. An attribute specified in a DEFAULT statement has priority over the content of the input file.
2. A variable attribute specification defined in a packet has priority over a default field specification, as well as over the content of the input file.
3. The last variable attribute specification packet has priority over all previous variable attribute specification packets.

One additional command is available from the IMPORT program level but not from within a key file.

IMPORT, *input-file*{, *input-format*}

This command initiates the data loading process from the file or device *input-file*. Optionally *input-format* may be specified (see Section 3.3.4.1). After the data have been loaded, the user is returned to the IMPORT program level.

3.3.5 The MANAGE Command

This command is used to enter the *Management Utility*. This utility allows users to move data back and forth from secondary storage, add and delete user-defined values from the permitted vocabulary for a field, and grant various access privileges to other users for data they own. The syntax of the command is

MANAGE

From within the Management Utility, the following commands are available:

STORE, *subRIDDL* — This command transfers to secondary storage all data sets belonging to the user running SPEEDI which match *subRIDDL*.

RETRIEVE, *subRIDDL* — This command retrieves from secondary storage all data sets which match *subRIDDL* and loads them into the Analysis Databank. Only data sets to which the user has been permitted access will be retrieved.

DELDATA, *subRIDDL*{, NOCONFIRM} — This command deletes data sets matching *subRIDDL* from the database. By default the user is asked to confirm each deletion before it is done. If NOCONFIRM is specified, the user is only informed of the data sets which are being deleted. A data set may only be deleted by its owner or by a user with the appropriate access privilege.

ADDVALUE, *fieldname, newval, meaning* — This command adds the value *newval* with definition *meaning* to the user-defined permitted vocabulary of field *fieldname*.

DELVALUE, *fieldname, oldval* — This command deletes the value *oldval* from the user-defined permitted vocabulary of field *fieldname*. A permitted value may only be deleted by its owner.

PERMIT, USER:*username*, ACCESS:*accesslist*, *subRIDDL* — This command permits the users specified by *username* to access the data matching *subRIDDL*, with privileges defined by *accesslist*. The *accesslist* must consist of some combination of the characters R, W, D, and H, and these indicate, respectively, Read, Write, and Delete privileges on the data sets, and read privileges on the Headers (RIDDLs) of the data sets. Using the wildcard character for *username* grants the specified access privileges to all authorized SPEEDI users.

3.3.6 The MODIFY Command

This command allows users to modify the contents of the attribute fields of data sets in the Analysis Databank. Data set attribute fields may only be modified by a user with write permission for that data set. The syntax for this command is

MODIFY, *subRIDDL*

where *subRIDDL* identifies the data sets whose attributes should be modified. The user is notified of the number of data sets which match *subRIDDL*, and their RIDDLs are displayed. At the MODIFY program level, values for data attribute fields can be modified using the key file commands discussed in Section 3.3.4.2.

3.3.7 The RIDDL Command

The *RIDDL Mask Definition Utility* can be invoked using the RIDDL command. This utility defines the user mapping of subRIDDLs into RIDDLs. The commands available within the RIDDL Mask Definition Utility are described in detail in Chapter 4.

4 DATA DEFINITION AND IDENTIFICATION

One of the most important features of the user's interface to SPEEDI is the means by which the user identifies the working data sets. In this chapter we will first discuss data identification and the RIDDL types available in SPEEDI for data identification. We will then present the method of and commands available for defining and using RIDDL Masks to identify data.

4.1 DATA DEFINITION USING THE RIDDL

All data in the Environmental and Analysis Databanks are uniquely defined by a series of data attributes. For each data set entered into the database, each of a number of attribute fields must contain some identifying information for that data set. Many fields may contain information consisting of long character strings. Some attributes will be used strictly for maintaining historical information while others are required for data identification. Some of the fields must contain values which conform to a specific list of permitted values (the *permitted vocabulary*) for the field while others may range over all values of a given type and size. All of SPEEDI's indexable field names, contents, types, widths, and indications of whether or not the fields are tied to a permitted vocabulary are presented in Appendix A.

The *Record Identification and Data Definition Label* (RIDDL) can be thought of as a shorthand representation of a specification of the fields required to uniquely identify a data set stored in the database. A complete RIDDL uniquely identifies a data set, and consists of a string of values for each of the RIDDL fields separated by the delimiter /. Notice that since a field delimiter is used in the RIDDL, the user need not fill in the entire width of any given field. Because the types of data to be stored in the Analysis and Environmental Databanks are so varied in origin and structure, it would be difficult and inefficient to design a single RIDDL which can be used for all data. Therefore, SPEEDI allows for multiple RIDDL types.

The type associated with each RIDDL determines the number of fields, the width of each field, the name of each field, and the ordering of the fields in the RIDDL. Each RIDDL attribute field is represented by a few (from one to twenty) characters. These fields have been chosen such that any of the data attributes on which a user wishes to frequently search the database are represented. In addition, enough attributes are included such that each data set is uniquely identified; that is, if the user were to select a single value from the permitted vocabulary for each field, no more than one data set would be selected from the database. The RIDDL is intended as a shorthand notation for the convenience of the user. Therefore, it is also desirable to minimize the number of fields included.

Four fields are always associated with each RIDDL type. These are RTYPE, which specifies the RIDDL type; GROUP, which identifies a group of users with

access to the data; OWNER, which specifies the owner of the data set; and VERSION which identifies distinct working copies of the same data set. RTYPE is implicitly defined by the format from which the data set is loaded. Four additional attributes have been identified which are required for each data set for quality assurance and security but are not necessary for unique data identification. These four fields are as follows: CLASS (security classification), QA_LEVEL (most recent quality assurance certification level), QA_DATE (date QA_LEVEL was certified), and QA_AUTH (individual authorizing quality assurance).

As data sets are processed and manipulated by users, it is necessary to identify them by specifying values for each and every one of the core attribute fields in the appropriate RIDDL type. In order to meet the needs of the frequent and experienced user, in particular the analysts who are accessing the Analysis Databank, we have implemented an approach which allows the analyst to create a restricted view of the database in order to use an abbreviated form of the RIDDL when accessing data. This method is described in detail in Sections 4.3 and 4.4.

4.2 RIDDL TYPES AVAILABLE IN SPEEDI

A core group of attributes provides a unique specification and identification, a RIDDL, for each type of data in the database. Three types of data have been identified and their corresponding RIDDL types have been implemented. These three types of data are the Environmental Data Bank data, test and simulation data, and derived data. The RIDDLs designed for each of these types of data are described in detail in the following three sections.

4.2.1 The Environmental Data Bank or DB RIDDL Type

The DB RIDDL type is designed to provide for maximal indexing capabilities for the data from the DOE/DOD Environmental Data Bank. These data comprise the SPEEDI Environmental Databank. Fourteen of the indexable fields were chosen as a key into this databank. These fourteen fields are enumerated in Table 4-1 which specifies the order, names, contents, data types, and widths of the fields in the DB RIDDL type. A complete specification of a DB RIDDL requires a maximum of seventy-one characters—fifty-eight characters of value specifications and thirteen field delimiters. Although the SUMMARY and FILE fields form a unique identifier for each data set in the databank, this was deemed insufficient for adequate indexing capabilities. Of course, the RTYPE for the DB RIDDL type is always DB. RIDDLs of the DB RIDDL type may have multiple environment/event pairs associated with them. These pairs will be used when generating Environmental Databank indices, but the permitted value 'VA' for VARIOUS must be used to index these data via the subRIDDL.

Table 4-1. The Environmental Data Bank or DB RIDDL Type

DB Order	Field Name	Field Contents	Field Type	Field Width
1	RTYPE	RIDDL type	C	2
2	GROUP	Identifier of User Group with Access to Data	C	2
3	OWNER	Username of the Data Set Owner	C	9
4	VERSION	Version Number of the Data Set	I	4
5	ROLE	Role of Data in the Test	C	1
6	PROGRAM	Program Designation for Test Item	C	10
7	PHASE	Phase of System Life Cycle to which Data Pertain	C	1
8	CONDITION	Environmental Condition	C	1
9	ENVIRONMENT	Environment Type	C	4
10	EVENT	Event Monitored	C	5
11	CARRIER	Carrier of Unit Tested	C	2
12	MODEL	Carrier Model	C	10
13	SUMMARY	Data Summary Prefix for DB RIDDLs	C	1
14	FILE	File Access Number for DB RIDDLs	I	6
Total Width Including Field Delimiters				**71**

4.2.2 The SATADEF or SD RIDDL Type

Data in the Analysis Databank is usually loaded from the SATADEF files or tapes, so the SD RIDDL type which is to be used for analysis data is based on the SATADEF header. This RIDDL consists of twenty-six of the SPEEDI indexable fields. These fields are enumerated in Table 4-2 which specifies the order, name, contents, data type, and width of each of the fields in the SD RIDDL type. To completely specify a SD RIDDL, up to 161 characters are required. Of those 161 characters, 136 may be needed to specify values while 25 are required for field delimiters. This is necessary to provide a unique RIDDL key for each data set in the Analysis Databank and maximal flexibility in efficiently indexing data. The RTYPE for any RIDDL of the SD RIDDL type will always be SD.

4.2.3 The Derived Data or DD RIDDL Type

The Analysis Databank also contains data sets derived from other data sets. In order to represent these data sets with a minimum of redundant information, the derived data, or DD, RIDDL type has been implemented. The DD RIDDL is composed of twenty-eight fields from the indexable SPEEDI fields. These fields are presented in Table 4-3 which specifies the order, name, contents, data type, and width of each of the fields in the DD RIDDL type. A complete RIDDL specification of the DD RIDDL type requires up to 203 characters, 176 of which represent field values and 27 of which are the field delimiters. In addition to the RIDDL key information, the derivation parameters specific to the given derivation type and the data tags of the data sets involved in the derivation procedure are stored in SPEEDI.

4.3 THE RIDDL MASK AND THE SUBRIDDL

An analyst typically uses well-defined subsets of data during a particular analysis. In the context of the data attributes and the RIDDL, this implies that a subset of data can be defined by fixing the value of a particular field of the RIDDL. By fixing the values of many of the RIDDL fields, the user can define a manageable subset of data with which to work. Once those fields are fixed, it is no longer necessary to specify them when selecting data sets from the database. Only the subset of the RIDDL fields which are not fixed need be specified when selecting data sets. This subset of the RIDDL fields is referred to as the *subRIDDL*.

The *RIDDL mask* is a collection of information which describes what must be specified in the analyst's subRIDDL. The mask also defines how the subRIDDL is to be interpreted and mapped into a full RIDDL. This information includes which fields are to be explicitly specified by the analyst and which are implicitly specified, the ordering of the explicit fields, and logical expressions which define values for each field.

Table 4-2. The SATADEF or SD RIDDL Type (Page 1 of 2)

SD Order	Field Name	Field Contents	Field Type	Field Width
1	RTYPE	RIDDL Type	C	2
2	GROUP	Identifier of User Group with Access to Data	C	2
3	OWNER	Username of the Data Set Owner	C	9
4	VERSION	Version Number of the Data Set	I	4
5	ORIGIN	Origin of the Data Set	C	1
6	FUNCTION	Functional Description of Data	C	3
7	FILTER	Filter Identifier	C	9
8	MEASURE	Data Measured	C	4
9	ROLE	Role of Data in the Test	C	1
10	PROGRAM	Program Designation for Test Item	C	10
11	UNIT	Test Unit, Simulation Model, or Event Site (for Natural Phenomena)	C	10
12	PHASE	Phase of System Life Cycle to which Data Pertain	C	1
13	CONDITION	Environmental Condition	C	1
14	ENVIRONMENT	Environment Type	C	4
15	METHOD	Method of Obtaining Data	C	2
16	EVENT	Event Monitored	C	5
17	SUBEVENT	Stage of Event Tested	C	10
18	CARRIER	Carrier of Unit Tested	C	2
19	SUBCARRIER	Carrier Subgroup	C	2
20	MODEL	Carrier Model	C	10
Subtotal Width Including Field Delimiters				**111**

Table 4-2. The SATADEF or SD RIDDL Type (Page 2 of 2)

SD Order	Field Name	Field Contents	Field Type	Field Width
21	E_AXIS	Event Axis (Axis Being Loaded)	C	10
22	R_ID	Run Identifier (Test Run Recorded)	C	6
23	T_ID	Transducer Identifier	C	10
24	T_AXIS	Transducer Axis	C	10
25	A_ID	Analysis Run Identifer	I	4
26	CHANNEL	Analog to Digital Tape Channel Identifer (Monitored for Data)	C	4
Total Width Including Field Delimiters				161

Table 4-3. The Derived Data or DD RIDDL Type (Page 1 of 2)

DD Order	Field Name	Field Contents	Field Type	Field Width
1	RTYPE	RIDDL type	C	2
2	GROUP	Identifier of User Group with Access to Data	C	2
3	OWNER	Username of the Data Set Owner	C	9
4	VERSION	Version Number of the Data Set	I	4
5	ORIGIN	Origin of the Data Set	C	1
6	FUNCTION	Functional Description of Data	C	3
7	FILTER	Filter Identifier	C	9
8	MEASURE	Data Measured	C	4
9	ROLE	Role of Data in the Test	C	1
10	PROGRAM	Program Designation for Test Item	C	10
11	UNIT	Test Unit, Simulation Model, or Event Site (for Natural Phenomena)	C	10
12	PHASE	Phase System of Life Cycle to Which Data Pertain	C	1
13	CONDITION	Environmental Condition	C	1
14	ENVIRONMENT	Environment Type	C	4
15	METHOD	Method of Obtaining Data	C	2
16	EVENT	Event Monitored	C	5
17	SUBEVENT	Stage of Event Tested	C	10
18	CARRIER	Carrier of Unit Tested	C	2
19	SUBCARRIER	Carrier Subgroup	C	2
20	MODEL	Carrier Model	C	10
Subtotal Width Including Field Delimiters				**111**

Table 4-3. The Derived Data or DD RIDDL Type (Page 2 of 2)

DD Order	Field Name	Field Contents	Field Type	Field Width
21	E_AXIS	Event Axis (Axis Being Loaded)	C	10
22	R_ID	Run Identifier (Test Run Recorded)	C	6
23	T_ID	Transducer Identifier	C	10
24	T_AXIS	Transducer Axis	C	10
25	A_ID	Analysis Run Identifer	I	4
26	CHANNEL	Analog to Digital Tape Channel Identifer (Monitored for Data)	C	4
27	DERNAME	User Defined Derivation Name for Derived Data RIDDL Type	C	20
28	DERTYPE	Type of Derivation Used to Generate Data Set	C	20
Total Width Including Field Delimiters				**203**

4.4 SPECIFYING SUBRIDDLS TO SPEEDI

Throughout SPEEDI the user identifies data by specifying a subRIDDL. In the context of a particular RIDDL mask, the subRIDDL will retrieve zero or more data sets which match the attributes specified in the subRIDDL. The syntax for specifying a subRIDDL must adhere to the following rules:

1. All subRIDDL fields must be separated by the slash (/) character.
2. The wildcard characters *, %, and ? may be used anywhere in a subRIDDL attribute value.
3. The EXPLICIT RIDDL field values must be listed in the order specified in the RIDDL mask. A null EXPLICIT field value is interpreted the same as the wildcard * for that field.
4. Overridden IMPLICIT RIDDL mask field values and nonRIDDL field values should appear after the EXPLICIT fields in the subRIDDL in the format

   fieldname:field-value

   where *fieldname* is the name of an indexable SPEEDI attribute field (see Appendix A) and *field-value* is the value of that attribute which is to be matched. A user need only specify the *fieldname* with sufficient characters to uniquely identify the name of the attribute field. Wildcard characters are not meaningful in the *fieldname*.
5. The logical expression for an EXPLICIT RIDDL mask field may also be overridden using the format specified above, but the override must appear before its actual EXPLICIT field in the subRIDDL. The overridden EXPLICIT field position may not be omitted from the subRIDDL, but the null string may be specified.

4.4.1 Defining RIDDL Masks

One of the capabilities available in SPEEDI is a utility to define a RIDDL mask. Using the *RIDDL Mask Definition Utility*, the analyst may tailor the use of the RIDDL as follows:

- Specify a permissible value or range of permissible values for a field using logical expressions.
- State which fields must be specified explicitly in the subRIDDL when the mask is active and which fields are implicitly specified in the context of the mask.

- Reorder the explicit field sequence.

To assist the analyst in using and maintaining a library of RIDDL masks, the following operations are also available in the RIDDL Mask Definition Utility.

- Save a RIDDL mask and associate a name with it.
- Activate a named mask.
- Display the names of saved masks.
- Display the definition of a mask.
- Display the permitted vocabulary for a particular field.
- Delete a user-defined RIDDL mask.
- Rename a user-defined RIDDL mask.
- Copy a RIDDL mask.

4.4.2 Globally Available RIDDL Masks

There are three RIDDL masks intrinsic to SPEEDI for each RIDDL type. These are the FULL, EMPTY, and UNSPECIFIED masks. These masks may be used as templates for defining new masks. In the FULL mask, all fields must be specified explicitly in the user's subRIDDL and all values in the permitted vocabulary for each field are allowed. Conversely, the EMPTY mask has no fields which must be specified explicitly by the user (that is, all fields are specified implicitly) and no values from the permitted vocabulary are allowed. The UNSPECIFIED mask defines all fields to be implicit with all values from the permitted vocabulary for each field allowed. FULL is the default mask when SPEEDI is used and no mask is specified. There is a fourth global mask available for the DD RIDDL Type. It is the GRAFAID mask, and it defines all fields to be implicit except the DERNAME and DERTYPE fields. (Other masks of general interest to all users can be made global masks through a utility available to the EDBM.)

The FULL RIDDL mask is aimed at assisting new and infrequent users of SPEEDI. The FULL mask will be helpful as a starting point for the definition of a RIDDL mask in which the analyst would like to fix a small number of fields and would like to specify the rest when using the subRIDDL. Similarly, the UNSPECIFIED mask would provide a suitable start in defining a RIDDL mask with few fields fixed, but with the rest of the fields implicitly matching all permitted values. Conversely, the EMPTY mask would serve as a helpful basis for defining a mask in which most of the fields are to be fixed.

4.4.3 Using RIDDL Masks

Once the analyst has activated a mask, access to the database is interpreted through the definition of that mask. When specifying a subRIDDL in the context of a given mask, a list of values separated by the special character / is given. These values are assumed to appear in the order of the explicit fields as defined in the mask. Unless changed in the mask, the order is as specified in Tables 4-1, 4-2, and 4-3. Recall that matching wildcards * and % function as in the VAX/VMS file specifications defined in the *VAX/VMS User's Manual*. The wildcard ? functions as % does. Implicit fields are not to be specified in the subRIDDL but are assumed to range over the values in the mask; that is, they are hidden by the mask. However, it is possible to override the explicit ordering and implicit fields defined in the mask. To override the mask, the analyst must provide both the field name and the value for the overridden field separated by :, one of the special SPEEDI characters. The override capability can also be used to specify values for any of the other indexable fields presented in Appendix A.

4.4.4 Examples of RIDDL Masks and SubRIDDLs

The differences among the FULL, EMPTY, and UNSPECIFIED RIDDL masks can best be shown by example. This discussion will also illustrate some of the ways an analyst might use a subRIDDL to search for data. For this example, a simplified version of the SD RIDDL type is defined as having three fields which uniquely identify the data in the database:

- PROGRAM (program designation) with permitted values X and Y.
- UNIT (test unit) with permitted values EDW, JTA, and JT5.
- CARRIER (mode of transportation) with permitted values AIRCRAFT, SHIP, and TRUCK.

If the FULL mask were active, a query for all matches to */*/* would produce all of the user's data sets in the database with the SD RIDDL type. One unique data set can be accessed by specifying each field as follows:

X/EDW/TRUCK

A query using this subRIDDL would match the set of data in the database with

- RTYPE=SD, PROGRAM=X, UNIT=EDW, and CARRIER=TRUCK,

if that data set exists.

Conversely, if the EMPTY mask were active, a simple query (that is, a query with no overrides) would produce zero sets of data. In order to match a data set, each field must be overridden. For example,

PROGRAM:X/UNIT:EDW/CARRIER:TRUCK

In this case, exactly one data set would again be matched, the one with the correct values in the corresponding fields of the RIDDL, if it exists.

When the UNSPECIFIED mask is active, a simple query would produce all of the user's data sets in the database (with the SD RIDDL type) without the need to use wildcards. The all-inclusive range can be overridden in the same manner as the all-exclusive range in the previous example.

PROGRAM:X/UNIT:EDW/CARRIER:TRUCK

This subRIDDL will match exactly one data set's RIDDL, the one with

- RTYPE=SD, PROGRAM=X, UNIT=EDW, and CARRIER=TRUCK,

if such a data set exists.

To illustrate the use of other RIDDL masks, let the TEST mask be structured such that:

- PROGRAM is implicit with the value X.
- UNIT is explicit with EDW and JTA as values allowed.
- CARRIER is explicit with TRUCK as the value disallowed.
- Ordering of the explicit fields is first CARRIER and second UNIT.

Let the TEST mask be active in the context of the following subRIDDL examples. A simple subRIDDL would specify each explicit field in the prescribed order. One such subRIDDL would be

AIRCRAFT/EDW

which would match the single data set with

- RTYPE=SD, PROGRAM=X, UNIT=EDW, and CARRIER=AIRCRAFT.

A subRIDDL such as

*/%

would match data sets having the following RIDDLs:

- RTYPE=SD, PROGRAM=X, UNIT=EDW, and CARRIER=AIRCRAFT.
- RTYPE=SD, PROGRAM=X, UNIT=EDW, and CARRIER=SHIP.
- RTYPE=SD, PROGRAM=X, UNIT=JTA, and CARRIER=AIRCRAFT.
- RTYPE=SD, PROGRAM=X, UNIT=JTA, and CARRIER=SHIP.

Implicit fields can be overridden by specifying the subRIDDL as follows:

PROGRAM:Y/SHIP/JTA

This matches the data set with

- RTYPE=SD, PROGRAM=Y, UNIT=JTA, and CARRIER=SHIP.

Using wildcards when overriding a field causes the entire permitted vocabulary to be used to find matches, and not just the mask range. For example, the subRIDDL

SHIP/JTA/PROGRAM:%

matches data sets with the following RIDDLs:

- RTYPE=SD, PROGRAM=X, UNIT=JTA, and CARRIER=SHIP.
- RTYPE=SD, PROGRAM=Y, UNIT=JTA, and CARRIER=SHIP.

In addition, the ordering of explicit fields may be overridden as the following examples illustrate. The subRIDDL

UNIT:JTA/AIRCRAFT/""

matches the data set with

- RTYPE=SD, PROGRAM=X, UNIT=JTA, and CARRIER=AIRCRAFT.

The subRIDDL

UNIT:*/AIRCRAFT/""

matches the data sets with

- RTYPE=SD, PROGRAM=X, UNIT=EDW, and CARRIER=AIRCRAFT.

- RTYPE=SD, PROGRAM=X, UNIT=JTA, and CARRIER=AIRCRAFT.
- RTYPE=SD, PROGRAM=X, UNIT=JT5, and CARRIER=AIRCRAFT.

Finally, nonRIDDL fields may be specified in the subRIDDL using the override capability as follows:

SHIP/JTA/CLASS:UNC

This matches the data set having UNC in the nonRIDDL attribute field CLASS and with the following RIDDL:

- RTYPE=SD, PROGRAM=X, UNIT=JTA, and CARRIER=SHIP.

4.5 THE RIDDL MASK DEFINITION UTILITY

The RIDDL Mask Definition Utility can be invoked by entering the SPEEDI command

RIDDL{, *rtype*}

where *rtype* is the desired RIDDL type. If *rtype* is not specified, the current RIDDL type will be assumed. SPEEDI will respond with the "RIDDL>" prompt or a menu presentation. In addition to the globally available commands described in Chapter 3, the following commands are available in the RIDDL Mask Definition Utility:

MASK	ACTIVATE
FIELD	SAVE
EXPLICIT	DELMASK
IMPLICIT	RENAME
REORDER	COPY

4.5.1 Displaying RIDDL Masks

To display saved RIDDL mask definitions, enter

MASK{, *maskspec*}

where *maskspec* is a mask specification. A mask specification is the name of a saved RIDDL mask optionally preceded by the username of the mask owner and :, or one of the intrinsic masks FULL, EMPTY, or UNSPECIFIED. Wildcards *, %, and ? may be used in *maskspec*. If no *maskspec* is given, the names of all available masks belonging to the user are displayed along with an indication of which mask is active.

4.5.2 Displaying the Permitted Vocabulary and the Current Active Mask

This command is used to display the permitted vocabulary. The syntax of the command is

FIELD{, *fieldname*}

where *fieldname* is the name of any SPEEDI indexable field. Wildcards *, %, and ? may appear in *fieldname*. If no *fieldname* is given, the names of all RIDDL fields are displayed, along with their logical expressions and an indication of whether they are implicit or explicit in the active mask.

4.5.3 Specifying Explicit RIDDL Fields in the Active Mask

To define a field to be one which must be expressly specified in the context of the active mask use

EXPLICIT, *fieldname*{:*logicalEXP*}

where *fieldname* is the name of a field in the RIDDL, and *logicalEXP* is a *logical expression* consisting of *relational expression* operands and the logical operators & and | (logical AND and logical OR, respectively). The logical expression is evaluated from left to right. The user can specify precedence by using parentheses. The logical expression is used to restrict the values or specify a range of values for a field.

A relational expression consists of an assumed first operand (the value of the given *fieldname*); one of the relational operators =, <> (not equal to), >, >=, <, or <=; and an explicit second operand. Wildcards *, %, and ? are permitted in the second operand. The operator = is optional. That is, if no operator is specified, then equality is assumed. When no logical expression is specified, the field is made explicit but its logical expression is unchanged.

Warning: The logical expression provides the analyst with a great deal of power in specifying values. This power also provides ample opportunity to specify nonsense in a syntactically correct manner. For example, if an analyst makes the following definition in a mask

EXPLICIT, PROGRAM:X & (= Y)

then no RIDDL will ever match the analyst's simple subRIDDL in the context of that mask. It is impossible for the PROGRAM field to have both values X and Y simultaneously.

4.5.4 Specifying Implicit RIDDL Fields in the Active Mask

This command is used to fix a given field's allowed values in the context of the active mask. The syntax of the command is

IMPLICIT, *fieldname*{:*logicalEXP*}

where *fieldname* is the name of a field in the RIDDL, and *logicalEXP* is a logical expression as defined above. When no logical expression is specified the field is made implicit, but the logical expression is unchanged.

4.5.5 Reordering the Sequence of Explicit Fields in the Active Mask

To reorder the sequence of the explicit fields in the active mask, enter

REORDER, $field_1/field_2/\ldots/field_n$

where $field_1$, $field_2$, ..., $field_n$ are the names of explicit fields which are to be moved to the first n positions in the explicit field ordering. The default ordering of the explicit fields is that of their corresponding EXPLICIT commands. If an implicit field is given in the REORDER command, the analyst is warned of the inconsistency and the field is ignored in the processing of the command.

4.5.6 Activating a RIDDL Mask

The ACTIVATE command activates a mask for use in interpreting subRIDDLs. The syntax for the command is

ACTIVATE, *maskspec*

where *maskspec* is a mask specification. If no specification is given, the analyst will be prompted for one.

4.5.7 Saving the Active Mask

The active mask can be saved with a name associated with it by using the following command:

SAVE, *maskname*

where *maskname* is the name for the mask to be saved. If no name is given, the analyst will be prompted for one. The mask is saved with the analyst's username as the owner. Global masks and intrinsic masks such as FULL, EMPTY, and UNSPECIFIED cannot be saved by the analyst.

4.5.8 Deleting a RIDDL Mask

To delete a saved mask, enter

DELMASK, *maskname*

where *maskname* is the name of a RIDDL mask belonging to the analyst. The analyst will be prompted for a name if none is given.

4.5.9 Renaming a RIDDL Mask

This command is used to change the specification of a saved mask. The syntax of the command is

RENAME, *oldname,newname*

where *oldname* is the name of a saved RIDDL mask which belongs to the analyst, and *newname* is the new name for the mask.

4.5.10 Copying a RIDDL Mask

The COPY command is used to save a new copy of a mask. To do this, enter

COPY, *oldspec*, *newname*

where *oldspec* is the name of a saved mask, and *newname* is the name for the new copy of the mask.

4.5.11 Example of Defining a RIDDL Mask

We illustrate the use of the SPEEDI RIDDL Mask Definition Utility through the following example. We will build the TEST mask used in the examples in Section 4.4.4. First, we wish to create a mask of the SD RIDDL type, so we enter the utility with the following command:

RIDDL, SD

Since we wish to specify a logical expression for each of the three fields in the SD RIDDL, it does not matter which of the three intrinsic masks we use as a starting point, but some mask must be used. We activate the EMPTY mask with the following command.

ACTIVATE, EMPTY

To make the PROGRAM field implicit with the value X, we use the IMPLICIT command.

```
IMPLICIT, PROGRAM:X
```

We make the UNIT field explicit with EDW and JTA as values allowed via the EXPLICIT command.

```
EXPLICIT, UNIT:EDW | =JTA
```

Likewise, we make the CARRIER field explicit with TRUCK as the value disallowed via the EXPLICIT command.

```
EXPLICIT, CARRIER:<> TRUCK
```

Finally, the ordering of the explicit fields is changed to first CARRIER and second UNIT via the REORDER command.

```
REORDER, CARRIER/UNIT
```

The TEST mask as defined in Section 4.4.4 is now complete and may be used in subRIDDL interpretation for the remainder of the SPEEDI session. If the user wishes to save the mask for future sessions, it must be associated with a name and stored in the database using the SAVE command.

```
SAVE, TEST
```

5 THE DATA REDUCTION REQUEST FACILITY

The SPEEDI *Data Reduction Request Facility* is based on the *SATADEF Header Definition File Format*, Appendix B of *SATADEF Version 2*. The objective of this facility is to automate the process of generating SATADEF header information, and therefore, SPEEDI SD RIDDL information, as much as possible. This utility stores the header information provided by the requester of a test and produces a file format for transmission to the data production site. Information known at the data reduction site can be integrated with the information in this file to produce a more complete SATADEF header. This will in turn enable SPEEDI to load data with less human intervention. From within SPEEDI, this utility is fully menu-driven, and the user is prompted for all missing information.

REFERENCES

Adams, C. R., 1985a. "SATADEF (Standard ASCII Test/Analysis Data Exchange Format)," Sandia memorandum from C. R. Adams to Sandia National Laboratories Staff, Albuquerque, NM, February 18.

Adams, C. R., 1985b. *GRAFAID Code User Manual, Version 2.0*, SAND84-1725, Sandia National Laboratories, Albuquerque, NM.

Adams, C. R., 1989. "SATADEF (Standard ASCII Test/Analysis Data Exchange Format) Version 2," Sandia memorandum from C. R. Adams to Sandia National Laboratories Staff, Albuquerque, NM, April 12.

Davidson, C. A., J. T. Foley, and C. A. Scott, 1985. *DOE/DOD Environmental Data Bank Index*, SAND85-0155, Sandia National Laboratories, Albuquerque, NM.

DEC (Digital Equipment Corporation), 1986. VAX/VMS User's Manual, Maynard, MA.

Haskell, K. H. and P. A. Helman, 1987. *Evaluation, Design, and Implementation of an Engineering Database Management System for Sandia National Laboratories Phase 1: Evaluation*, RSI-0312, by RE/SPEC Inc., prepared for Sandia National Laboratories, Albuquerque, NM.

Haskell, K. H., E. M. Kephart, and P. A. Helman, 1987. *Evaluation, Design, and Implementation of an Engineering Database Management System for Sandia National Laboratories, Phase 2: Design*, RSI-0319, by RE/SPEC Inc., prepared for Sandia National Laboratories, Albuquerque, NM.

Kephart, E. M., K. H. Haskell, and G. M. von Laven, 1989. *System Guide to SPEEDI: The Sandia Partitioned Engineering Environmental Database Implementation*, SAND89-7099, prepared by RE/SPEC Inc. for Sandia National Laboratories, Albuquerque, NM.

RTI (Relational Technology, Inc.), 1986. *INGRES Terminal User's Guide*, Release 5.0, VMS, 1080 Marina Village Parkway, Alameda, CA.

APPENDIX A

SPEEDI INDEXABLE FIELDS

APPENDIX A
SPEEDI INDEXABLE FIELDS

There are 88 SPEEDI fields for use in indexing data sets. Any SPEEDI indexable field may be used in any SPEEDI command where a field name is expected except in the RIDDL Mask Definition Utility where only RIDDL fields of the RIDDL type specified when entering the utility may be used. The three RIDDL types, DB, SD, and DD, are comprised of subsets of this set of fields. (The RIDDL types are defined in Tables 4-1, 4-2, and 4-3 of this report.) The following table presents the name, contents, type, width, and an indication of whether a permitted vocabulary exists for each of the SPEEDI indexable fields. The possible field types are 'C' for character, 'I' for integer, 'F' for floating point or real, and 'D' for date. These data types are described in detail in Section 2.4 of this report.

More detailed descriptions of these fields may be found via the FIELD command in the SPEEDI RIDDL Mask Definition Utility. Along with the field description, the permitted vocabulary associated with the given field (or a description of the type of information to be stored in the field if no permitted vocabulary exists) is displayed to the user.

Table A-1. SPEEDI Indexable Fields (Page 1 of 4)

Field Name	Field Contents	Field Type	Field Width	Permitted Vocabulary
A_ID	Analysis Run Identifer	I	2	N
ABSTRACT	Abstract Text	C	80	N
ANALOG_TAPE	Analog Tape Number or Name	C	10	N
CARRIER	Carrier of Unit Tested	C	2	Y
CHANNEL	Analog to Digital Tape Channel Identifer	C	4	N
CHARGE_NUM	Account Charged for the Test	C	10	N
CLASS	Security Classification of the Data	C	3	Y
CONDITION	Environmental Condition	C	1	Y
CREATED	Data and Time Stamp Indicating When the Data Were Entered into SPEEDI	D	25	N
DBTYPE	Name of the Databank to which the Data Belongs	C	3	Y
DELTAX	X Increment of the Data if Nonzero	F	13	N
DERNAME	User Defined Derivation Name for Derived Data RIDDL Type	C	20	N
DERTYPE	Type of Derivation Used to Generate Data Set	C	20	Y
DFNAME	Digital Filter Name	C	18	N
DFTYPE	Digital Filter Type	C	2	N
E_AXIS	Event Axis (Axis Being Loaded)	C	10	Y
ENGINEER	Name of the Engineer Responsible for the Test	C	20	N
ENVIRONMENT	Environment Type	C	4	Y
EVENT	Event Monitored	C	5	Y
FACILITY	Data Reduction Facility	C	20	N
FILE	File Access Number for DB RIDDLs	I	4	N
FILTER	Filter Identifier	C	11	N
FORMAT	Format from which the Data Were Loaded	C	3	Y
FUNCTION	Functional Description of Data	C	3	Y
GROUP	Identifier of User Group with Access to Data	C	2	N
IRIGSTART	Data Start Time in IRIG Format	C	15	N
IRIGZERO	Zero Test Time in IRIG Format	C	15	N

Table A-1. SPEEDI Indexable Fields (Page 2 of 4)

Field Name	Field Contents	Field Type	Field Width	Permitted Vocabulary
KEYWORD	Keyword Associated with the Data Set	C	80	N
MEASURE	Data Measured	C	4	Y
MEDIUM	Medium on which the Data are Stored	C	2	Y
METHOD	Method of Obtaining Data	C	2	Y
MINX	Minimum X Value	F	13	N
MODEL	Carrier Model	C	10	Y
NUMPTS	Number of Data Points in the Data Set	I	11	N
ORIGIN	Origin of the Data Set	C	1	Y
OWNER	Username of the Data Set Owner	C	9	N
PAGES	Number of Pages of Information	I	6	N
PATH	Pointer to the Data Stored on the Given Medium	C	80	N
PDNAME	Name Associated with a Plot Definition	C	20	N
PDOWN	Owner of the Plot Definition	C	9	N
PHASE	Phase of System Life Cycle to which Data Pertains	C	1	Y
PNAME	Name Associated with a Plot Aggregate	C	20	N
PNOTE1	First Note Associated with a Plot Definition	C	80	N
PNOTE2	Second Note Associated with a Plot Definition	C	80	N
PNOTE3	Third Note Associated with a Plot Definition	C	80	N
PNOTE4	Fourth Note Associated with a Plot Definition	C	80	N
PNOTE5	Fifth Note Associated with a Plot Definition	C	80	N
POWNER	Owner of a Plot Aggregate	C	9	N
PROGRAM	Program Designation for Test Item	C	10	Y
PT_TYPE	Data Point Type	C	2	Y

Table A-1. SPEEDI Indexable Fields (Page 3 of 4)

Field Name	Field Contents	Field Type	Field Width	Permitted Vocabulary
PTITLE1	First Title Associated with a Plot Definition	C	80	N
PTITLE2	Second Title Associated with a Plot Definition	C	80	N
PTITLE3	Third Title Associated with a Plot Definition	C	80	N
PTITLE4	Fourth Title Associated with a Plot Definition	C	80	N
PTITLE5	Fifth Title Associated with a Plot Definition	C	80	N
QA_AUTH	Username of Individual Authorizing Quality Assurance	C	9	N
QA_DATE	Date Quality Assurance Level Authorized	D	25	N
QA_LEVEL	Quality Assurance Level	C	2	Y
R_ID	Run Identifier (Test Run Recorded)	C	6	N
R_NUMBER	The R-Number for the Test as Assigned by the Test Record System for SNL Div. 7500	C	10	N
REF_CHANNEL	Reference A/D Channel Number Used for Modal and Vibration Testing	C	4	N
REFERENCE	Reference Text Associated with a Data Set	C	80	N
RELEASEID	Release Information Associated with the Data Set (from the DOE/DOD Environmental Data Bank)	C	20	N
ROLE	Role of Data in the Test	C	1	Y
RTYPE	RIDDL type	C	2	Y
SAMPLE_RATE	Data Sample Rate	F	13	N
SUBCARRIER	Carrier Subgroup	C	2	Y
SUBEVENT	Stage of Event Tested	C	10	N
SUBORIGIN	Test Facility & Location for Test Data; Computer Code for Analysis Data	C	20	N
SUMMARY	Data Summary Prefix for DB RIDDLs	C	1	N

Table A-1. SPEEDI Indexable Fields (Page 4 of 4)

Field Name	Field Contents	Field Type	Field Width	Permitted Vocabulary
T_AXIS	Transducer Axis	C	10	Y
T_ID	Transducer Identifier	C	10	N
T_LOC	Transducer Location	C	40	N
T_MODEL	Transducer Model	C	10	N
T_SERIAL	Transducer Serial Number	C	10	N
T_TYPE	Transducer Type	C	10	N
TAPE_TRACK	Analog Tape Track Number which is the Source for the Data	C	2	N
TEST_DATE	Date that the Test Run Took Place Specified as DD-MMM-YYYY:HH:MM:SS	D	25	N
TITLE	Title Associated with the Data Set	C	80	N
UNIT	Test Unit, Simulation Model, or Event Site (for Natural Phenomena)	C	10	Y
VERSION	Version Number of the Data Set	I	4	N
XUNITS	X Data Engineering Units	C	20	N
YCALIBRATION	Y Data Calibration Sensitivity (Units/Volt)	F	13	N
YUNITS	Y Data Engineering Units	C	20	N
ZONEEND	Zone End Time in Seconds	F	13	N
ZONESTART	Zone Start Time in Seconds	F	13	N
1DFCUTOFF	Digital Filter Cutoff Frequency Number 1	F	13	N
2DFCUTOFF	Digital Filter Cutoff Frequency Number 2	F	13	N

Distribution:

1510 J. W. Nunziato
1520 L. W. Davison
1521 R. D. Krieg & Staff (12)
1522 R. C. Reuter, Jr. & Staff (15)
1522 C. R. Adams (10)
1523 J. H. Biffle & Staff (13)
1524 D. R. Martinez & Staff (11)
1530 D. B. Hayes
1550 C. W. Peterson
3141 S. A. Landenberger (5)
3141-1 C. L. Ward for DOE/OSTI (8)
3151 W. I. Klein (3)
7541 T. J. Baca (4)
8524 J. A. Wackerly

RE/SPEC Inc.
R. L. Padgett (5)
E. M. Kephart (10)
K. H. Haskell
G. M. von Laven
P. O. Box 14984
Albuquerque, NM 87191