UMTRI-97-51 OCTOBER 1997

ERD File Format for Storage and Analysis of Road Profiles

Michael W. Sayers
Steven M. Karamihas

UMTRI
University of Michigan
Transportation Research Institute

Technical Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
UMTRI-97-51		

4. Title and Subtitle	5. Report Date
ERD File Format for Storage and Analysis of Road Profiles	October, 1997
	6. Performing Organization Code

7. Author(s)	8. Performing Organization Report No.
Michael W. Sayers and Steven M. Karamihas	

9. Performing Organization Name and Address	10. Work Unit No. (TRAIS)
The University of Michigan Transportation Research Institute 2901 Baxter Road Ann Arbor, Michigan 48109	11. Contract or Grant No.

12. Sponsoring Agency Name and Address	13. Type of Report and Period Covered
	14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

This document briefly describes the ERD file format used for storing, plotting, and analyzing engineering data. The ERD file format was developed within the Engineering Research Division (ERD) of the University of Michigan Transportation Research Institute (UMTRI). ERD files were used internally by the ERD at UMTRI for storage and analysis of vehicle test data, vehicle simulation output, laboratory measurement of vehicle component behavior, and road profiles.

Since ERD files were used as the standard data file format by the road profile analysis software "RoadRuf," many practitioners of road profile measurement and analysis adopted it as a standard. As such, the need for public documentation of the ERD file format was identified. This document provides the documentation of the ERD file format that appeared in Appendix A of the RoadRuf Reference Manual, circa 1997.

17. Key Word	18. Distribution Statement
data storage, ERD files, road profiles	

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	6	

ERD File Format

The ERD file format was developed within the Engineering Research Division (ERD) of the University of Michigan Transportation Research Institute (UMTRI) to facilitate automated plotting of simulation data, experimentally measured data, and data from various analysis programs.

A freely available plotter called EP (Engineering Plotter) has been developed for viewing data in ERD files. Versions of EP exist for Mac, Windows, and some UNIX platforms. The Windows version is called WinEP.

An ERD file contains two independent sections, the header and data. The header contains only text, and the data section contains only numbers. The numbers can be written in either text or binary form. The text form is convenient for viewing and editing data with a word processor, whereas the binary form provides more efficient access for automatic processing. If the data section is in text format, then both the header and the data are kept in a single file. However, if the data are in binary format, two files are used. The header is in an ordinary text file with the extension `ERD`, and the data are in a file with the name of the header file and the extension `BIN`. For example, if the header file is named `Out.erd` the name `Out.bin` must be used for the data file. Both files must lie in the same folder.

The Header

The ERD file header consists of a series of conventional text lines that are human readable. These lines contain the information used by post-processing tools to read the numerical data.

Required Lines

As a minimum, the header contains three lines of text. The first line identifies the file as following the ERD format. The second line describes the way that the numerical data are stored in the data section of the file. The third required line is an END statement that indicates the end of the header portion. Any number of optional lines can be included between line #2 and the END line. Table 1 summarizes the lines in an ERD file, and describes the parameters used in line #2 to describe the numerical data.

The second line of the file shown in Listing 1 shows that the file contains data for 2 channels, with 529 samples/channel, stored as 1 binary record consisting of 4232 bytes, that the data storage format is type 1 (4-byte binary), that the interval between samples is 1.00, and that the status of the auxiliary numbers is -1.

Table 1. Summary of records in an ERD file header.

Line No.	Description
1	`ERDFILEV2.00` — identifies file as having ERD format
2	NCHAN, NSAMP, NRECS, NBYTES, KEYNUM, STEP, KEYOPT — use commas to separate numbers NCHAN [integer] = Number of data channels NSAMP [integer] = Number of samples for each channel. The total number of sampled values in the data portion of the file is NCHAN × NSAMP. (If unknown, use –1.) NRECS [integer] = Number of records of data. (record ≈ line) Ignored for text data (KEYNUM = 5.) (If unknown, use –1.) NBYTES [integer] *binary data:* Number of bytes per record. If KEYNUM=0,1, or 5, this should be chosen such that each record begins with channel 1: that is, NBYTES = K × NCHAN × B, where K is an integer and B is the number of bytes/number (B=2 for integer, B=4 for floating-point). If KEYNUM=10,11,or 15, this should be NSAMP × B. *text data:* Number of samples per record. Thus each record contains NBYTES × NCHAN numbers (for KEYNUM=5). KEYNUM[integer] Indicates how the data are stored. 0, 10 = 2-byte integer (binary), 1, 11 = 4-byte floating point (binary), 5, 15 = Formatted floating-point (text). The format must be specified using the FORMAT keyword. For KEYNUM=0,1, and 5, the data are stored with all channels for the first sample together, then all channels for the second sample, etc. For KEYNUM=10,11, and 15, the data are stored with all samples for the first channel together, then all samples for the second channel, etc. STEP [real] = sample interval (e.g., time step) KEYOPT [integer] = auxiliary number used by some programs
•	**Optional records**. Each record begins with an 8-character keyword, followed by information associated with that keyword. Table 2 lists keywords that have been used to date.
last line	`END` — indicates the end of the header

Listing 1. Short Header for an ERD File with Binary Data.

```
ERDFILEV2.00
2, 529, 1,  4232,  1,  1.00000,  -1,
TITLE   1993 RPUG Study, Dipstick, Section 1, Measurement 1
SHORTNAMLElev.  RElev.
UNITSNAMft      ft
XLABEL  Distance
XUNITS  ft
END
```

Listing 2 shows a longer header for a file with its data in text form. Note that the data begin immediately after the END line of the header.

Listing 2. Typical Header for an ERD File with Text Data.

```
ERDFILEV2.00
2, 529, 1,  4232,  1,  1.00000,  -1,
TITLE   1993 RPUG Study, Dipstick, Section 1, Measurement 1
SHORTNAMLElev.  RElev.
LONGNAMELeft Elevation                  Right Elevation
UNITSNAMft      ft
GENNAME Profile Elevation               Profile Elevation
XLABEL  Distance
XUNITS  ft
FORMAT  (2G14.6)
PROFINSTDipstick
HISTORY Converted to ERD format at 23:46, Oct. 23, 1994
END
0.000000      0.000000
 0.416667E-03 -0.141667E-02
0.416667E-03  0.583333E-03
0.666667E-03  0.916667E-03
0.133333E-02  0.133333E-02
0.750000E-03 -0.166667E-02
-0.300000E-02 -0.458333E-02
-0.558333E-02 -0.500000E-02
-0.625000E-02 -0.658333E-02
-0.775000E-02 -0.825000E-02
```

Optional Lines with Keywords

Optional lines in the header begin with an eight-character keyword that defines a particular type of data contained in the remainder of the line. Keywords are associated with one of five general data types: integers, floating point (real) numbers, 8-character names, 32-character names, and 80-character names. The number of data items is either one per file (e.g., TITLE of data set in the file), one per channel (e.g., a short name for each channel), an arbitrary number N (e.g., static axle loads for N axles), or repeatable (e.g., comments stored using the HISTORY keyword).

Table 2 lists common keywords recognized by most post-processing tools. The use of some of these keywords is demonstrated in Listing 1 and Listing 2.

Table 2. Keywords in ERD file.

keyword	Description	No. of Values	Variable Type
VERSION	Line 1 in header file.	1	char*32
LINE 2	Line 2 in header file. See Table 2 for details.	7	int, real
&n	Continuation keyword, indicates that the previous line ended in column *n* and is continued in this line in column 9. Used to break long lines into multiple short lines.		
(The following 6 lines are used by EP and are recommended for inclusion in all ERD files)			
GENNAME	Generic names for variables, used for labeling Y axis when several variables are plotted on the same axis (e.g., Force).	NCHAN	char*32
LONGNAME	Long names for channels.	NCHAN	char*32
SHORTNAM	Short names for channels.	NCHAN	char*8
TITLE	Title used for file.	1	char*80
UNITSNAM	Units of channels.	NCHAN	char*8
XUNITS	Units of independent variable (e.g., sec).	1	char*8
(The following line is required for EP to create Channel 0*, e.g., time)*			
XLABEL	Name of ind. variable in ERD file (e.g., time).	1	char*32
FORMAT	FORMAT statement for text data. Ex: (4F10.4)	1	char*32
GAIN	Gains for channels. (Default = 1.) Usually required for integer*2 data.	NCHAN	real
OFFSET	Offsets for channels. (default = 0.) Usually required for integer*2 data.	NCHAN	real
PROFINST	Instrument or model associated with data	1	char*32
RIGIBODY	Body or part associated with each channel	NCHAN	char*32
SPEEDMPH	Speed associated with data, in mile/hr.	1	real
TESTID	Number used to identify a test.	1	real
XSTART	Starting value of ind. variable. At each sample i, the X value is: X = (i-1) * STEP + XSTART	1	real

Usually, names associated with a keyword are shorter than the space allowed. When several names are on the same line, the names are padded with blanks as needed so that following names begin at the correct column positions. For example, the header shown in Listing 1 includes names of the units for each channel, as identified with the keyword UNITSNAM. The name of units for the first channel, ft, has only two characters. Thus, it

is followed by six spaces so that the name for the second channel, ft, begins in the correct column position.

The Data Section

The data section of the ERD file contains nothing but numbers, organized into columns and rows. The form in which the numbers are stored depends on the value of the KEYNUM parameter from line 2 of the header (see Table 1). The total number of values that will appear in the data section is NCHAN x NSAMP. All of the numbers in the data portion are stored in the same format, and there can be no missing values.

Text Data

The text format can be used for transporting data in ERD files between different computers, and sometimes even for reading the same file with different programs on the same computer. It is also convenient when numbers are typed in manually, or when numbers are to be edited using a text editor. There are penalties for using text representations of numbers, however. First and foremost, the computer must work hard to translate the text numbers into binary form. It takes about 10 times longer to read a text file than a binary equivalent. A second penalty is that text files take up much more disk storage than binary files.

When data are stored in text form, the numbers are kept in the same file as the header, with the numbers beginning immediately after the header. The ERD file in Listing 2 shows an example of numerical data in text form.

Another option is available when the numbers are always separated by delimiters such as spaces or commas. This occurs when the numbers are obtained by a commercial analysis program or when they are "captured" from another computer. The file of numbers can be made into an ERD file by inserting a 3-line header at the beginning of the file.

If the header of the ERD file does not contain a line with the FORMAT keyword, it is the same as if the FORMAT is a blank. When this occurs for a text file, the file is assumed to contain numbers in free form. The only restriction on free format numbers is that adjacent numbers must be separated. For example, the following line is valid for representing 5 numbers `1.2, 3, 4, -.0201, 14.3: 1.2000 3 4 -2.01E-01 14.3`

The following line is not, because the third and fourth numbers touch.

```
1.2000 3.0000 4.0000-2.01E-01 14.3000
```

Numbers may be separated by one or more spaces, the tab character, or a comma.

Binary Data

Reading and writing binary data is very efficient, because the computer does not need to perform any conversions or transformations as the data values are moved between the file and the computer memory. When a binary format is used, the data portion of an ERD file

is a direct copy of a portion of the computer memory, corresponding to a two-dimensional array having dimensions sized to the number of channels and the number of samples. As indicated in Table 1, two forms of binary data are presently supported: 2-byte integer and 4-byte floating point. 2-byte integer data are typically obtained by data-acquisition systems. Each integer value is a sampled reading obtained from a digitizer during a test. For most engineering applications, data are stored (in the computer memory) in 4-byte floating point format, also known as single-precision floating point. The 4-byte floating point format is commonly used for data generated by computer. The maximum efficiency for data processing is usually obtained when the 4-byte floating point format is used.

The ERD file format is used on a variety of computer systems and for a variety of mass storage media. On some systems, binary data are stored in discrete records. A computer program reading such a file needs to know how many bytes each record contains, and how many records are in the file. Thus, the header contains these two parameters.

Disk files on workstations and desktop computers are not structured: a binary file is simply a continuous stream of bytes that continues to the end of the file. Thus, technically correct parameter values for the header could be one record, containing all of the bytes for the file. Also, there is a certain amount of overhead associated with reading a record. The time needed to read the data for a file is minimized if a single read operation is performed for the entire file.

On the other hand, if the file is large, the memory needed to read the entire file in one chunk may not be available with some programs. A second problem can occur if the true number of bytes in the file is less than the number as inferred by the parameters nbytes and nrecs (i.e., the total size of the file should be NBYTES x NRECS). The last "record" is not read, resulting in a loss of data. If the records are large, this loss could be significant. These problems are reduced if a value of NBYTES is specified such that it divides the data into NRECS records of smaller chunks of data.