



Recommended standard for seismic (/radar) data files in the personal computer environment

Subcommittee of the SEG Engineering and Groundwater Geophysics Committee,
S. E. Pullan, Chairman*

This paper is the result of the work of a subcommittee of SEG's Engineering and Groundwater Geophysics Committee. It recommends a data file format for raw or processed shallow seismic or digital radar data in the small computer environment. It is recommended that this format be known as the SEG-2 format.

INTRODUCTION

The application of seismic reflection and ground-penetrating radar methods to engineering, groundwater, and environmental problems has increased dramatically over the last 10 years, spurred by the rapid development of new instrumentation and personal computers. This "coming of age" of shallow reflection techniques has been accompanied by the recent development of several different software packages for processing and displaying data. However, this expansion in the user's processing potential has been somewhat hindered by the lack of a standard data format. All software packages have been based on their own, or a manufacturer's, specific data format.

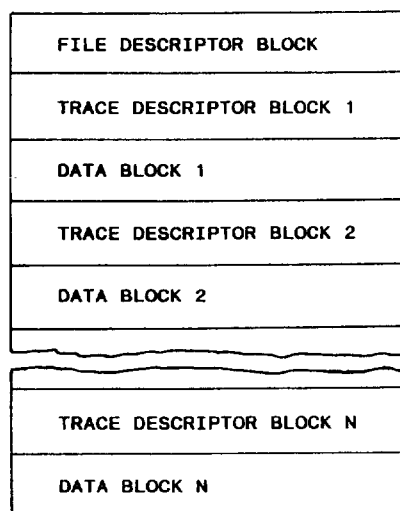
At the October 1987 meeting of SEG's Engineering and Groundwater Geophysics Committee, it was agreed in principle that a standard format for engineering seismic data in the personal computer environment was needed, and a subcommittee was created to set up such a standard. This paper is the result of the work of that subcommittee. The standard format recommended in this paper is a file format for raw or processed shallow seismic or digital radar data in the small computer environment. It is not expected that data will necessarily be stored or recorded by recording instruments in this format. However, instrument manufacturers are asked to provide the means of transferring data acquired by their instruments into this format. It is hoped that software developers may then assume that data from any source will be input to their programs in this format.

STANDARD SPECIFICATION

I. Scope

This is a recommended standard for the construction of seismic or digital radar data files in personal computer environments. The seismic (/radar) data may be either raw data or

BEGINNING OF FILE



END OF FILE

FIG. 1. File format.

processed data organized in one or more blocks of trace data. The computer environment includes but is not limited to IBM and compatible computers running DOS or OS/2, Macintoshes, and UNIX work stations. This standard specifies a format for placing raw or processed data samples with associated overhead information on to one file for access by application programs. This standard does not involve itself with file management concerns such as how data are put on disk or tape media.

II. Requirements for standard

This standard shall be a free-form standard capable of allowing as much or as little overhead data as the data-generating application software or recording seismograph can provide. It shall not depend on the nuances of one programming language, and shall provide for the sharing of data by any of the personal computers or work stations commonly used for processing

*Geological Survey of Canada, 601 Booth St., Ottawa, Ont. Canada K1A 0E8.

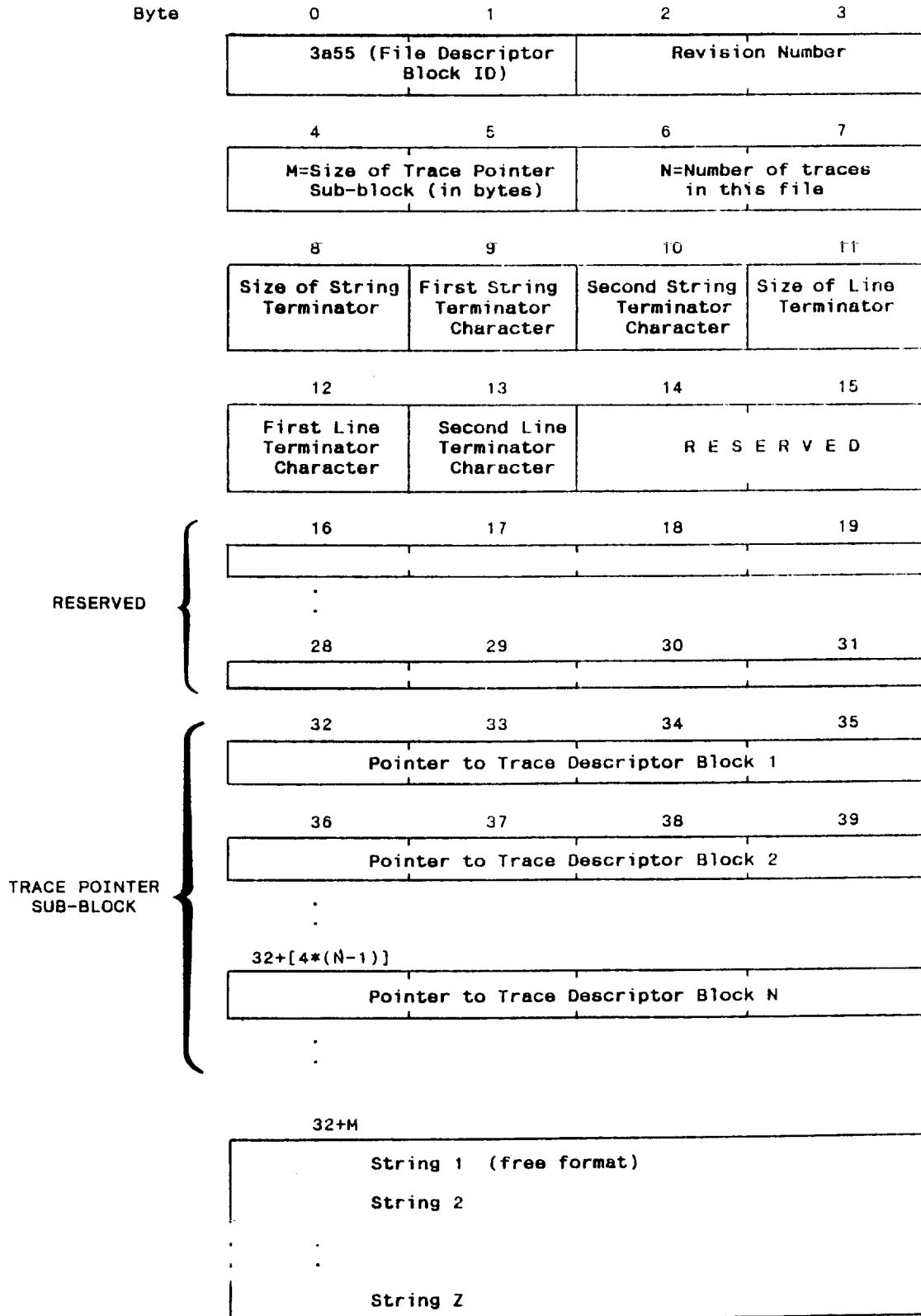


FIG. 2. File descriptor block.

seismic data today and into the foreseeable future. This standard shall be easy to maintain and require a minimum of revision.

The position of overhead data within the file must not be critical. It must not be necessary to put in blank lines for missing data. Instead, each data item should be in a string with an identifying keyword. In this way, included data items may be chosen from a large list of possible items without creating extremely large and cumbersome data files.

Care must be taken not to assume token and line delimiters such as NULL characters, CR, CR/LF, etc. Instead this information and other information dependent upon the data-generating computer, language, and operating system should be included in the file in known locations to allow other computers and software the ability to parse the overhead data and properly interpret the data.

III. Proposed format

A. File structure

The file consists of a File Descriptor Block, one or more Trace Descriptor Blocks, and one or more Data Blocks. The File Descriptor Block provides information required to parse the rest of the overhead data and other information common to all traces in the file. Each Trace Descriptor Block corresponds to one Data Block and provides location, format, and other information pertinent to that Data Block. The Data Blocks immediately follow their corresponding Trace Descriptor Blocks and consist of fixed point or floating point numbers as specified in the Trace Descriptor Block. There is a one to one correspondence between Data Blocks and Trace Descriptor Blocks. The blocks must be arranged in the order shown in Figure 1.

Pointers are used to indicate locations of blocks with respect to the beginning of the file. Pointers are always unsigned long integers (32 bits) rather than the segment: offset format common to some processors. All addressing is to byte boundaries. All blocks must start on double word (32 bit) boundaries.

B. Block structures

B1. File Descriptor Block. The first block in the file is the File Descriptor Block. The construction of the File Descriptor Block is shown in Figure 2. This block is required and holds information pertaining to the structure and interpretation of the file and data common to all traces in the file. This block consists of (i) 32 bytes providing the block identifier, file standard revision number, size of the Trace Pointer Subblock (see below), number of traces in this File, string and line terminator information; (ii) a Trace Pointer Subblock giving pointers to the start of each Trace Descriptor Block in the file; and (iii) a series of optional strings which contain information such as the type of data to follow, acquisition parameters, etc., and any notes that may incorporate operator observations and/or environmental descriptions which apply to the entire file.

The first two bytes (bytes 0 and 1) of this block (and of the file) contain the integer 3a55h. This integer identifies the file as a seismic (/radar) data file following this standard, identifies this block as the File Descriptor Block, and provides the means to determine whether the low byte (least-significant bits) of multibyte entities is written first or last. If the first byte of the file is 55h, the file is recorded low byte first as on an IBM PC,

but if the first byte is 3ah, the file is recorded low byte last as on some 68000 based UNIX machines.

The next two bytes (bytes 2 and 3) contain an integer stating the revision number of this standard used by the data-generating device or program. The version defined by this paper shall be referred to as version 1. Bytes 4 and 5 contain an unsigned integer giving the size (M) of the Trace Pointer Subblock in bytes. The integer byte order of this and all following integers is interpreted according to the first two bytes of this file. This number, as all block sizes, must be divisible by 4 since all blocks must start on double word boundaries. This number must be between 4 and 65,532.

Within these restrictions, the size of this subblock is completely up to the user. For example, it could be just large enough to hold the pointers to the 12 Trace Descriptor Blocks corresponding to a single record from a 12-channel seismograph (i.e., 48 bytes); or it could be large enough to hold the pointers to all processed traces in an entire seismic (/radar) line. The subblock does not have to be full; the number of traces actually contained in the file (N) is specified in bytes 6 and 7. N is an unsigned integer with an allowable range of 1 to 16,383. N must be less than or equal to M/4.

The String Terminator is one or two nonprintable ASCII characters (decimal ASCII codes 0 through 31) used to separate the strings that hold the information in character string form in this block (the File Descriptor Block) and the Trace Descriptor Blocks. Byte 8 is 01h or 02h depending on the number of characters in the String Terminator and the two following bytes contain the String Terminator characters. If only one character is used, that character is in byte 9.

The Line Terminator is one or two nonprintable ASCII characters (normally CR or CR and LF) used to separate the lines of text in a NOTE string. Byte 11 is 01h or 02h depending on the number of characters in the Line Terminator and the two following bytes contain the Line Terminator characters. If only one character is used that character is in byte 12.

The restriction of allowable String and Line Terminators to nonprintable ASCII characters is to avoid any potential conflicts with the characters allowed in the definition of such strings as JOB_ID, etc.

Bytes 14 through 31 are reserved.

The Trace Pointer Subblock starts at byte 32, and contains pointers (unsigned long integers) to the start of each Trace Descriptor Block contained in the file. The length of this subblock in bytes (M) is specified in bytes 4 and 5, and the number of pointers contained in the subblock (N) is specified in bytes 6 and 7 (see above). This fixed format allows a file to be built up one trace at a time by upgrading the value N and adding a pointer to the new trace in the Trace Pointer Subblock, but without otherwise changing the file format.

Following the Trace Pointer Subblock is a free format section containing strings to provide optional information pertinent to all traces in the record. Each string starts with an offset to the next string, is followed by a keyword and related parameters, and is terminated by the string terminator (see section C on String Formats below).

B2. Trace Descriptor Block. There is a one to one correspondence between Data Blocks and the Trace Descriptor Block that describes that Data Block. The file must contain at least one Trace Descriptor Block and matching Data Block. The construc-

tion of the Trace Descriptor Block is shown in Figure 3. The first thirty-two bytes must be included to provide the block identifier, the size of the block, the size of the Data Block, and the data format code. The strings follow with information pertinent to that block such as receiver and shot locations, sample rate, delay, stack count, etc.

The first two bytes contain the unsigned integer 4422h to identify this block as a Trace Descriptor Block. Bytes 2 and 3

contain an unsigned integer giving the size of this block in bytes. This number, as all block sizes, must be divisible by 4 since all blocks must start on double word boundaries. This number must be between 32 and 65,532.

Bytes 4 through 7 contain an unsigned long (32 bit) integer giving the size of the following Data Block in bytes. This number must be divisible by 4. Bytes 8 through 11 contain an unsigned long integer giving the number of samples in the Data Block.

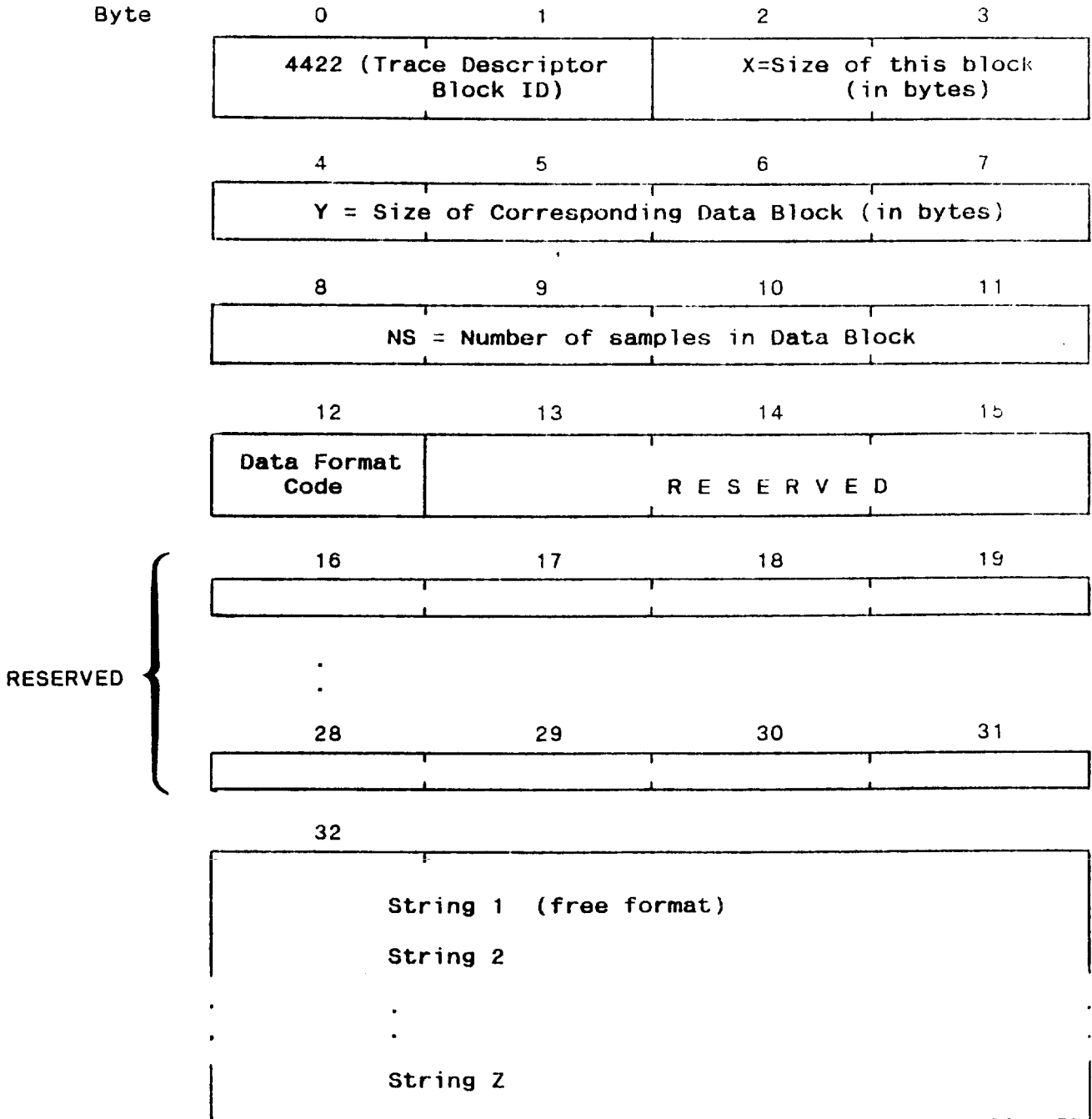


FIG. 3. Trace descriptor block.

Byte 12 specifies the format of the data in the following Data Block according to the table below:

Byte Value	Data Format
01h	16-bit fixed point
02h	32-bit fixed point
03h	20-bit floating point (SEG-D)
04h	32-bit floating point (IEEE standard)
05h	64-bit floating point (IEEE standard)

The fixed point formats imply the 2's complement convention. All formats are defined in Appendix B.

Bytes 13 through 31 are reserved.

A series of free format strings providing information on the acquisition or processing history of the data are found starting at byte 32. Any number of these strings may be included depending on the nature of the data or the processing applied. Some strings will obviously be required for most processing or display sequences that may be applied to the data, and every effort should be made to include all pertinent information in this section of the Trace Descriptor Blocks, especially in raw data files. The string format is discussed below in section C.

B3. Data Block. A Data Block follows each Trace Descriptor Block. Each Data Block is a collection of numbers of the size and in the format specified in the preceding Trace Descriptor Block.

C. String format

The File and Trace Descriptor Blocks contain strings that provide either required or optional information. All strings start with an offset (2 bytes) to the next string and a keyword that identifies the nature of the string, contain a numeric and/or alphanumeric value(s), and end with the string terminator indicated in the File Descriptor Block. An offset of 0 (2 bytes), after the final string, marks the end of the string list. Keywords cannot have embedded spaces. The keyword and the associated data are separated by spaces or tabs. All alpha characters in the keywords themselves must be uppercase. Where the argument associated with a keyword is chosen from a list of specified words, the arguments themselves cannot have embedded spaces and must be in uppercase characters. Not all strings mentioned need be included in the descriptor blocks. Unrecognized keywords will not terminate the process. To assist application program string searches, strings must be ordered alphabetically according to keyword. The only exception to this rule is the NOTE string, which is always at the end of the string list, if it exists at all.

Numeric values may be decimal integers or decimal floating point numbers. Negative decimal numbers are preceded by a minus sign (-). Decimal floating point numbers may use an "E" to express the number in scientific notation. Decimal points must be followed by a numeric character. The numbers in the following list are allowable numeric expressions. Unless stated otherwise, integers must have magnitude less than 32,000 (16 bits).

12, -3, 12.657, -34.6, 1.345E24, -2.3E6, 5.6E-11, -2.0E-9

Some values like time and date may be expressed in the recommended format indicated below. Examples of the string sub-blocks that may be found in a File Descriptor Block and Trace Descriptor Block are shown in Appendix C.

C1. Strings that may be included in the File Descriptor Block.

The File Descriptor Block may contain any or all of the following strings. The keywords themselves must be in uppercase characters and cannot have embedded spaces. Where keywords are two or more words, the words are separated by an underscore (decimal ASCII code 95). Where the argument associated with a keyword is chosen from a list of specified words, the arguments themselves cannot have embedded spaces and must be in uppercase characters. The keywords must be arranged in alphabetical order except for the NOTE string, which will be at the end of the string list, if it exists at all. The strings that are likely to be *required* by application software routines are denoted with an asterisk. It is strongly recommended that these strings exist in all File Descriptor Blocks.

ACQUISITION_DATE <value>

It is recommended that the date be specified in dd/mm/yyyy format. Under this convention, April 1, 1988, would be stored as 01/04/88. Alternatively, in order to avoid any possible confusion in the ordering of the day and month, the date could be in dd/mmm/yyyy format, with mmm being a three-letter abbreviation of the month. For example, April 1, 1988 would be stored as 01/APR/1988.

ACQUISITION_TIME <value>

It is recommended that the time be stored in the 24-hour hh:mm:ss format. For example, 3:30 PM would be stored as 15:30:00.

CLIENT <name>

Name of company or organization sponsoring data acquisition and processing.

COMPANY <name>

This is the name of the company responsible for acquiring and/or processing the data.

GENERAL_CONSTANT <value>

This value must be a positive decimal number of 12 or fewer digits.

INSTRUMENT

<manufacturer name> <model number/name>
<serial number>

This identifies the instrument used to acquire the data in the file.

JOB_ID <character string>

The JOB_ID can be any string of printable ASCII characters (decimal ASCII codes 32 through 126).

OBSERVER <name>

The name of the individual responsible for data acquisition.

PROCESSING_DATE <value>

The date a processed file was created. The date is specified as described in ACQUISITION_DATE.

PROCESSING_TIME <value>

The time a processed file was created. The time is specified as described in ACQUISITION_TIME.

*TRACE_SORT <sort method>

The sort method can be AS_ACQUIRED, CDP_GATHER, CDP_STACK, COMMON_OFFSET, COMMON_RECEIVER, or COMMON_SOURCE.

*UNITS <linear unit system>

This defines the linear units used throughout this file. Allowable systems are FEET, METERS, INCHES, CENTI-METERS, or NONE. If NONE is selected, all measurement interpretation must be made by the user.

NOTE <text>

This string appears as the last string in the list if it exists at all. The format of this string differs from all other strings in that it may contain several lines of text. Each line of text is terminated by the Line Terminator indicated in the File Descriptor Block. The end of the string is indicated by the String Terminator.

C2. Strings that may be included in the Trace Descriptor Block. The Trace Descriptor Block may contain any or all of the following strings. The keywords themselves must be in uppercase characters and cannot have embedded spaces. Where keywords are two or more words, the words are separated by an underscore (decimal ASCII code 95). Where the argument associated with a keyword is chosen from a list of specified words, the arguments themselves cannot have embedded spaces and must be in uppercase characters. The keywords must be arranged in alphabetical order except for the NOTE string, which will be at the end of the string list, if it exists at all. The strings that are likely to be *required* by application software routines are denoted with an asterisk. It is strongly recommended that these strings exist in all Trace Descriptor Blocks.

ALIAS_FILTER <frequency> <slope>

The ALIAS_FILTER values are decimal integer or floating point numbers (can be mixed) expressing the anti-aliasing filter 3 dB frequency in Hz and slope in dB per octave. A value of 0 for the frequency indicates and antialiasing filter was not implemented.

AMPLITUDE_RECOVERY <method> <parameter list>

The method can be NONE, AGC, SPHERICAL_DIVERGENCE, or any appropriate description of the method used. If AMPLITUDE_RECOVERY is included in the string list, some method must be specified immediately after the keyword. The parameter list is the collection of parameters related to method, such as AGC window and AGC constant. The list may be in any format deemed appropriate by the user.

BAND_REJECT_FILTER

<lp freq> <lp slope> <hp freq> <hp slope>

The BAND_REJECT_FILTER values are decimal integer or floating point numbers (can be mixed) expressing the low-pass 3 dB frequency in Hz and slope in dB per octave, followed by the high-pass 3 dB frequency in Hz and slope in dB per octave. These parameters specify a band-reject filter applied by the seismograph during data acquisition. Values of 0 for the frequency parameters indicate the filter was not implemented.

CDP_NUMBER <value>

The CDP_NUMBER is a decimal integer assigned to a particular CDP gather or stacked trace.

CDP_TRACE <value>

The CDP_TRACE is a positive decimal integer giving the trace number within the CDP gather. Each gather starts with trace number one.

CHANNEL_NUMBER <value>

The channel number is a positive integer less than 32,000

identifying the receiver group channel of the recording instrument associated with the data corresponding to the Data Block. If the CHANNEL_NUMBER string is used in one Trace Descriptor Block, it must be used in all Trace Descriptor Blocks in the File. More than one Data Block may be associated with one CHANNEL_NUMBER if this is necessary to accommodate such special circumstances as dynamically switched sample rates, time gaps, or other aberrations. All Data Blocks that are associated with one channel must be arranged such that earlier data precede later data, and data blocks cannot overlap in time.

DATUM <value>

The DATUM is a decimal integer or floating point number giving the elevation (understood to be with respect to mean sea level) about which the source and receiver locations (z-values) are specified. The value is given in the units specified in the "UNITS" string of the File Descriptor Block.

***DELAY** <value>

The value is a floating point number expressing the time (in seconds) elapsed from the start pulse to recording the first sample in the Data Block.

DESCALING_FACTOR <value>

The DESCALING_FACTOR is a floating point number used to get the voltage in millivolts from a sample value according to the formula:

$$\text{voltage due to one shot} = \text{data point} \cdot \text{DESCALING_FACTOR}/\text{STACK}$$

DIGITAL_BAND_REJECT_FILTER

<lp freq> <lp slope> <hp freq> <hp slope>

The DIGITAL_BAND_REJECT_FILTER values are decimal integer or floating point numbers (can be mixed) expressing the low-pass 3 dB frequency in Hz and slope in dB per octave, followed by the high-pass 3 dB frequency in Hz and slope in dB per octave. These parameters specify a digital band reject filter applied after data acquisition or during processing. Values of 0 for the frequency parameters indicate the filter was not implemented.

DIGITAL_HIGH_CUT_FILTER

<frequency> <slope>

The DIGITAL_HIGH_CUT_FILTER values are decimal integer or floating point numbers (can be mixed) expressing the high-cut filter 3 dB frequency in Hz and slope in dB per octave of a digital filter applied after data acquisition or during processing. A value of 0 for the frequency indicates the filter was not implemented.

DIGITAL_LOW_CUT_FILTER <frequency> <slope>

The DIGITAL_LOW_CUT_FILTER values are decimal integer or floating point numbers (can be mixed) expressing the low-cut filter 3 dB frequency in Hz and slope in dB per octave of a digital filter applied after data acquisition or during processing. A value of 0 for the frequency indicates the filter was not implemented.

END_OF_GROUP <value>

The END_OF_GROUP is used to flag a trace (or traces) within the file as the last trace (s) in a user-defined group or sort. A value of 1 indicates that this trace is the last trace in the group; the only other allowable value is zero.

FIXED_GAIN <value>

If a fixed-gain recording instrument was used, this integer value gives the gain in dB (including preamp) applied by the recording instrument to this trace during recording.

HIGH_CUT_FILTER <frequency> <slope>

The **HIGH_CUT_FILTER** values are decimal integer or floating point numbers (can be mixed) expressing the high-cut 3 dB frequency in Hz and slope in dB per octave applied by the seismograph during data acquisition. A value of 0 for the frequency indicates the filter was not implemented.

LINE_ID <character string>

The **LINE_ID** can be any string of printable ASCII characters (decimal ASCII codes 32 through 126).

LOW_CUT_FILTER <frequency> <slope>

The **LOW_CUT_FILTER** values are decimal integer or floating point numbers (can be mixed) expressing the low-cut filter 3 dB frequency in Hz and slope in dB per octave applied by the seismograph during data acquisition. A value of 0 for the frequency indicates the filter was not implemented.

NOTCH_FREQUENCY <value>

The **NOTCH_FREQUENCY** value is a positive decimal integer or floating point number expressing the notch-filter frequency in Hz. A value of 0 indicates a notch filter was not implemented.

POLARITY <value>

The only allowed values are 1 and -1, depending upon whether an upward ground movement at the phone causes a positive or negative data sample.

RAW_RECORD <file name>

The **RAW_RECORD** gives the file name (and extension) of the raw data file from which this trace was derived.

RECEIVER

<type of receiver used> <number of receivers in group>

Receivers could be, for example,

VERTICAL_GEOPHONE,

SH_HORIZONTAL_GEOPHONE,

SV_HORIZONTAL_GEOPHONE,

ACCELEROMETER, **ANTENNA**,

or any appropriate description of the type of receiver used.

If a second parameter is given, it specifies the number of receivers per receiver group.

RECEIVER_GEOMETRY

[n] <x value> [<y value> <z value>]

This is the location of the *n*th receiver of a receiver group with respect to the specified **RECEIVER_LOCATION** (see below). The *x*, *y*, and *z* values may be decimal integers or decimal floating point numbers, and are to be given in the units specified in the "UNITS" string of the File Descriptor Block. If only one value is stated, it is understood to be the dimension along the line (i.e., *x*). Otherwise, all three values must be given, in order to avoid ambiguity with regard to the meaning of a second value. The *z* value is understood to define the elevation or depth (e.g., in a borehole), with positive values implying the upward direction, and negative values implying the downward direction, about the **DATUM** defined above.

***RECEIVER_LOCATION**

<x value> [<y value> <z value>]

This is the location of a receiver or center of the receiver group with respect to the start of line, or any fixed point defined by the user. The *x*, *y*, and *z* values may be decimal integers and/or decimal floating point numbers, and are to be given in the units specified in the "UNITS" string of the File Descriptor Block. If only one value is stated, it is understood to be the dimension along the line (i.e., *x*). Otherwise, all three values must be given, in order to avoid ambiguity with regard to the meaning of a second value. The *z* value is understood to define the elevation or depth (e.g., in a borehole), with positive values implying the upward direction and negative values implying the downward direction. This value is understood to be given with respect to the **DATUM** defined above.

RECEIVER_SPECS

<manufacturer name> <model number or frequency>

This is the name of the manufacturer and model number and/or frequency of the receivers used to acquire the data.

RECEIVER_STATION_NUMBER <value>

The **RECEIVER_STATION_NUMBER** is a decimal integer assigned to a particular receiver location.

***SAMPLE_INTERVAL** <value>

The value is a floating point number expressing the period between samples in seconds.

SHOT_SEQUENCE_NUMBER <value>

The **SHOT_SEQUENCE_NUMBER** is a positive decimal integer that the user assigns to the original field file to allow a user-defined sequential ordering of files (or traces) during subsequent processing runs.

SKEW <value>

The **SKEW** value of a channel is a floating point number expressing the time in seconds between the time the recording instrument starts a sample scan along all the channels, and the time data were actually acquired at this particular channel.

SOURCE <source> <parameter list>

Source can be **HAMMER**, **WEIGHT_DROP**, **GUN**, **DYNAMITE**, **VIBRATOR**, or any appropriate description of the source used. The parameter list may specify any further information about the source; such as the size of the hammer or weight drop, the type of shells fired, size and type of charge detonated, or the sweep frequency, etc., of the vibrator.

SOURCE_GEOMETRY

[n/N] <x value> [<y value> <z value>]

This is the location of the *n*th element of a source array (consisting of *N* elements) with respect to the specified **SOURCE_LOCATION** (see below). The *x*, *y*, and *z* values may be decimal integers or decimal floating point numbers, and are to be given in the units specified in the "UNITS" string of the File Descriptor Block. If only one value is stated, it is understood to be the dimension along the line (i.e., *x*). Otherwise, all three values must be given, in order to avoid ambiguity with regard to the meaning of a second value. The *z* value is understood to define the elevation or depth (e.g., in a shothole), with positive values implying the upward direction,

and negative values implying the downward direction, about the DATUM defined above.

SOURCE_LOCATION <x value> [<y value> <z value>

This is the location of the source (or center of a source array) with respect to the start of line, or any fixed point defined by the user. The x , y , and z values may be decimal integers and/or decimal floating point numbers, and are to be given in the units specified in the "UNITS" string of the File Descriptor Block. If only one value is stated, it is understood to be the dimension along the line (i.e., x). Otherwise, all three values must be given, in order to avoid ambiguity with regard to the meaning of a second value. The z value is understood to define the elevation or depth (e.g., in a shothole), with positive values implying the upward direction and negative values implying the downward direction. This value is understood to be defined with respect to the DATUM defined above.

SOURCE_STATION_NUMBER <value>

The SOURCE_STATION_NUMBER is a decimal integer assigned to a particular source location.

STACK <value>

This stack-count is a positive integer indicating the number of shots which were summed to obtain this trace.

STATIC_CORRECTIONS

<source static> <receiver static>
<total correction applied>

The 3 values specified after STATIC_CORRECTIONS are floating point numbers expressing the time shifts (in seconds) that are attributed to the source and receiver, and the total static correction applied to the data in the following Data Block. If no static correction has been applied to the data, the third value is zero.

TRACE_TYPE <trace type>

The trace type is SEISMIC_DATA, DEAD, TEST_DATA, UPHOLE, or RADAR_DATA.

NOTE <text>

This string appears as the last string in the list if it exists at all. The format of this string is as described under the NOTE string in section C1 above.

Acknowledgments

This standard is the result of input from a large number of people representing instrument manufacturers, software developers, and a number of interested users. Although they are too numerous to name here, their comments and suggestions have been much appreciated and have greatly contributed to the final result.

The contributions of Badru Hyatt from EG&G Geometrics, Sunnyvale, California, and Ted Norminton from Carleton University, Ottawa, Ontario, must be especially recognized. Badru was responsible for writing the first draft of the standard, and both Badru and Ted have been involved in discussions of all the changes and modifications that have been made along the way.

References

- Barry, K. M., Cavers, D. A., and Kneale, C. W., 1975, Recommended standards for digital tape formats: *Geophysics*, **40**, 344-352.
Digital field tape format standard — SEG-D: *in* Digital Tape Standards, 1980, Soc. Explor. Geophys. 31-65.
IEEE Standard for binary floating - point arithmetic: ANSI/IEEE Standard 754-1985, 18 p.

Appendix A DEFINITIONS

Byte

an 8 bit value
if signed this value can range from -128 . . . 127
if unsigned this value can range from 0 . . . 255

Integer

a 16 bit value
if signed this value can range from -32768 . . . 32767
if unsigned this value can range from 0 . . . 65535

Long integer

a 32 bit value
if signed this value can range from
-2147483648 . . . 2147483647
if unsigned this value can range from 0 . . . 4294967295

Decimal integer

a base ten integer
e.g., 12, -3, 10, 352, -511, etc.

Decimal floating point number

a base ten floating point number which may use an "E" to express the number in scientific notation
e.g., 12657, -34.6, 1.345E24, -2.3E6, 5.61E-11, -2.0E-9

Appendix B SPECIFICATION OF ALLOWABLE DATA FORMATS

Fixed point formats

The two fixed point formats allowed by this standard are specified as described in Barry et al. (1975). Both formats imply the 2's complement convention.

1. 16-bit fixed point format (data format code = 01h)

The 16-bit fixed point format uses two successive bytes to record each data sample. Each sample consists of a sign-bit S (1 implies negative) and 15 data bits Q_D , with a radix point at the right of the least significant bit. The format is shown schematically in Figure B-1.

16-bit fixed point format numbers are of the 2's complement type [i.e., if (a) is a 16-bit fixed point number, then (a) + (-a) = 2^{16}]. For example,

$$\begin{array}{r} 1 \text{ (dec)} = 0000000000000001 \\ -1 \text{ (dec)} = 1111111111111111 \\ \hline (1) + (-1) = 1000000000000000 \end{array}$$

The largest available 16-bit fixed point number is 32767 (0111111111111111). The smallest available 16-bit fixed point number is -32768 (1000000000000000).

Byte \ Bit	HB	LB
7	S	Q _D
6	Q _D (msb)	Q _D
5	Q _D	Q _D
4	Q _D	Q _D
3	Q _D	Q _D
2	Q _D	Q _D
1	Q _D	Q _D
0	Q _D	Q _D (lsb)

Byte \ Bit	HB			LB
7	S	Q _D	Q _D	Q _D
6	Q _D (msb)	Q _D	Q _D	Q _D
5	Q _D	Q _D	Q _D	Q _D
4	Q _D	Q _D	Q _D	Q _D
3	Q _D	Q _D	Q _D	Q _D
2	Q _D	Q _D	Q _D	Q _D
1	Q _D	Q _D	Q _D	Q _D
0	Q _D	Q _D	Q _D	Q _D (lsb)

FIG. B-1. Description of one data sample in 16-bit fixed point format (S = sign bit, Q_D = data bit, HB = high byte, LB = low byte (least significant bits), msb = most significant bit, lsb = least-significant bit). In this case, the msb is the value corresponding to 2¹⁴, while the lsb is the value corresponding to 2⁰. Note that the order of the two bytes (i.e., LB before or after HB) is dependent on the specification of the first two bytes of the File Descriptor Block.

FIG. B-2. Description of one data sample in 32-bit fixed point format (S = sign bit, Q_D = data bit, HB = high byte, LB = low byte (least significant bits), msb = most significant bit, lsb = least significant bit). In this case, the msb is the value corresponding to 2³⁰, while the lsb is the value corresponding to 2⁰. Note that, the order of the four bytes (i.e., LB first or HB first) is dependent on the specification of the first two bytes of the File Descriptor Block.

2. 32-bit fixed point format (data format code = 02h).

The 32-bit fixed point format uses four successive bytes to record each data sample. Each sample consists of a sign bit S (1 implies negative) and 31 data bits Q_D, with a radix point at the right of the least significant bit. The format is shown schematically in Figure B-2.

32-bit fixed point format numbers are of the 2 complement type [i.e., if (a) is a 32-bit fixed point number then (a) + (-a) = 2³²]. For example,

$$\begin{array}{r}
 1 \text{ (dec)} = 00000000000000000000000000000001 \\
 -1 \text{ (dec)} = 11111111111111111111111111111111 \\
 \hline
 (1) + (-1) = 10000000000000000000000000000000
 \end{array}$$

Floating point formats

The 20-bit floating point format allowed by this standard is based on the Digital field tape format standards -SEG-D (2½-byte binary exponent data recording method— demulti-plexed). However, in order to be able to express the most significant bits of a 32-bit fixed point number, the data sample is written as a scaled signed integer rather than a signed fraction.

The 32-bit and 64-bit floating point formats allowed by this standard are specified as described in IEEE standard for Binary Floating — Point Arithmetic (ANSI/IEEE Std 754-1985).

3. 20-bit floating point format (data format code = 03h)

The 20-bit floating point format uses 10 successive bytes to record four data samples. The first two bytes contain the exponents for the following four samples. Each exponent is a four bit positive binary exponent of 2 written as 2^{cccc} where

The largest available 32-bit fixed point number is 2147483647.
 The smallest available 32-bit fixed point number is -2147483648.

Byte \ Bit	HB	LB	HB	LB	HB	LB	HB	LB	HB	LB	
7	4n+3 { C(msb) C C C(lsb)	4n+1 { C(msb) C C C(lsb)	S	Q _D	S	Q _D	S	Q _D	S	Q _D	
6			Q _D (msb)	Q _D	Q _D (msb)	Q _D	Q _D (msb)	Q _D	Q _D (msb)	Q _D	
5			Q _D	Q _D	Q _D	Q _D	Q _D	Q _D	Q _D	Q _D	
4			Q _D	Q _D	Q _D	Q _D	Q _D	Q _D	Q _D	Q _D	
3	4n+2 { C(msb) C C C(lsb)	4n { C(msb) C C C(lsb)	Q _D	Q _D	Q _D	Q _D	Q _D	Q _D	Q _D	Q _D	
2			Q _D	Q _D	Q _D	Q _D	Q _D	Q _D	Q _D		
1			Q _D	Q _D	Q _D	Q _D	Q _D	Q _D	Q _D		
0			Q _D	Q _D (lsb)	Q _D	Q _D (lsb)	Q _D	Q _D (lsb)	Q _D	Q _D (lsb)	
		Exponents for data samples 4n through 4n+3		Data sample 4n		Data sample 4n+1		Data sample 4n+2		Data sample 4n+3	

FIG. B-3. Description of four successive data samples in 20-bit floating point format (S = sign bit, C = binary exponent bit, Q_D = data bit, HB = high byte, LB = low byte (least significant bits), msb = most significant bit, lsb = least significant bit). Note that the order of each group of two bytes (i.e., LB first or HB first) is dependent on the specification of the first two bytes of the File Descriptor Block.

CCCC can assume values of 0 through 15. The four exponents are in sample order for the four samples starting with the first sample in bits 0-3 of the low byte (see Figure B-3).

Each data sample consists of a sign bit S (1 implies negative), and a 15-bit two complement binary integer. The radix point is to the right of the least significant bit with the lsb being defined as 2^0 . The sign and mantissa can assume values between -32768 and 32767 . Negative zero is invalid and must be converted to positive zero.

Note that in utilizing this data recording method, the number of samples per trace must be exactly divisible by 4 in order to preserve the data grouping of this method.

4. 32-bit floating point format (data format code = 04h)

The 32-bit floating point format uses four successive bytes to record each data sample. Each sample consists of a sign bit S (1 implies negative), an 8-bit biased binary exponent of 2, and a 23-bit positive binary fraction. The format is shown schematically in Figure B-4.

The exponent has been biased by 127, such that the value to be used in determining the represented number is $2^{(CCCCCCCC-127)}$, where CCCCCCCC can assume values from 0 to 255. Thus, $2^{(CCCCCCCC-127)}$ ranges from 2^{-126} to 2^{127} (or 1.18×10^{-38} to 3.40×10^{38}). Note that CCCCCCCC = 0 or 255 are special cases (see below).

The fraction represents a normalized (most significant bit always 1) positive value. Since the most significant bit is always 1 it is not present in the 23 bits specified in the format (the actual fraction is composed of the specified 23 bits preceded by the implied 1). The radix point is after the implied 1 which is defined as 2^0 . For example:

$$\begin{aligned}
 1 \text{ (dec)} &= \begin{array}{ccc} 0 & 01111111 & 0000000000000000000000 \\ \text{sign} & \text{exponent} & \text{fraction} \\ & \downarrow & \downarrow \\ & 2^{(127-127)} & \{1000000000000000000000\} \end{array} \\
 &= (+1) \times 2^0 \\
 -2 \text{ (dec)} &= \begin{array}{ccc} 1 & 10000000 & 0000000000000000000000 \\ \text{sign} & \text{exponent} & \text{fraction} \\ & \downarrow & \downarrow \\ & 2^{(128-127)} & \{1000000000000000000000\} \end{array} \\
 &= (-1) \times 2^1
 \end{aligned}$$

The sign and fraction (with implied digit) of the above format can represent values from $-2 + 2^{-23}$ to $2 - 2^{-23}$. Special cases:

Exponent bits (C)	Mantissa bits (Q _D)	Meaning
All 0	All 0	Floating point 0
All 0	Nonzero	Underflow
All 1	All 0	Infinity
All 1	Nonzero	Not a number

Byte \ Bit	HB			LB
7	S	C (lsb)	Q _D	Q _D
6	C(msb)	Q _D (msb)	Q _D	Q _D
5	C	Q _D	Q _D	Q _D
4	C	Q _D	Q _D	Q _D
3	C	Q _D	Q _D	Q _D
2	C	Q _D	Q _D	Q _D
1	C	Q _D	Q _D	Q _D
0	C	Q _D	Q _D	Q _D (lsb)

FIG. B-4. Description of one data sample in 32-bit floating point format (S = sign bit, C = binary exponent bit, Q_D = data bit, HB = high byte, LB = low byte (least significant bits), msb = most significant bit, lsb = least significant bit). Note that the order of the 4 bytes (i.e., LB first or HB first) is dependent on the specification of the first two bytes of the File Descriptor Block.

5. 64-bit floating point format (data format code = 05h)

The 64-bit floating point format uses 8 successive bytes to record each data sample. Each sample consists of a sign bit S (1 implies negative), an 11-bit biased binary exponent of 2, and a 52-bit positive binary fraction. The format is shown schematically in Figure B-5.

The exponent has been biased by 1023, such that the value to be used in determining the represented number is $2^{(CCCCCCCCCC-1023)}$, where CCCCCCCCCC can assume values from 0 to 2047. Thus, $2^{(CCCCCCCCCC-1023)}$ ranges from 2^{-1022} to 2^{1023} (or 2.23×10^{-308} to 1.80×10^{308}). Note that CCCCCCCCCC = 0 or 1023 are special cases (see below).

The fraction represents a normalized (most significant bit always 1) positive value. Since the most significant bit is always 1 it is not present in the 52 bits specified in the format (the actual

Byte \ Bit	HB						LB
7	S	C	Q _D	Q _D	Q _D	Q _D	Q _D
6	C(msb)	C	Q _D	Q _D	Q _D	Q _D	Q _D
5	C	C	Q _D	Q _D	Q _D	Q _D	Q _D
4	C	C(lsb)	Q _D	Q _D	Q _D	Q _D	Q _D
3	C	Q _D (msb)	Q _D	Q _D	Q _D	Q _D	Q _D
2	C	Q _D	Q _D	Q _D	Q _D	Q _D	Q _D
1	C	Q _D	Q _D	Q _D	Q _D	Q _D	Q _D
0	C	Q _D	Q _D	Q _D	Q _D	Q _D	Q _D (lsb)

FIG. B-5. Description of one data sample in 64-bit floating point format (S = sign bit, C = binary exponent bit, Q_D = data bit, HB = high byte, LB = low byte (least significant bits), msb = most significant bit, lsb = least significant bit). Note that the order of the 8 bytes (i.e., LB first or HB first) is dependent on the specification of the first two bytes of the File Descriptor Block.

fraction is composed of the 52 specified bits preceded by the implied 1). The radix point is after the implied 1 which is defined as 2^0 . The sign and fraction (with implied digit) of the above

format can represent values from $-2 + 2^{-52}$ to $2-2^{-52}$. The special cases listed under the 32-bit floating point mat also apply in this case.

Appendix C EXAMPLES OF STRING FORMAT SUBBLOCKS

In the examples shown in Figures C-1 and C-2, it is assumed that the String Terminator has been defined in bytes 8 – 10 of the File Descriptor Block as the NULL character (00), and that the Line Terminator has been defined in bytes 11 – 13 as line feed (0A) followed by carriage return (0D).

```

(30) |A,C,Q,U,I,S,I,T,I,O,N,_,D,A,T,E,_,1,6,/,0,8,/,1,9,8,9|00|
(28) |A,C,Q,U,I,S,I,T,I,O,N,_,T,I,M,E,_,1,4,:,2,3,:,0,6|00|
(31) |C,O,M,P,A,N,Y,_,X,Y,Z,_,G,e,o,p,h,y,s,i,c,a,l,_,I,n,c.,_|00|
(42) |I,N,S,T,R,U,M,E,N,T,_,A,c,m,e,_,I,n,s,t,r,u,m,e,n,t,s,_,
    |E,S,-1,2,_,#,0,0,0,1|00|
(33) |J,O,B,_,I,D,_,C,a,s,s,e,l,m,a,n,_,O,n,t,a,r,i,o,_,
    |1,9,8,9|00|
(20) |O,B,S,E,R,V,E,R,_,J,o,e,_,B,l,o,w|00|
(25) |T,R,A,C,E,_,S,O,R,T,_,A,S,_,A,C,Q,U,I,R,E,D|00|
(15) |U,N,I,T,S,_,M,E,T,E,R,S|00|
(54) |N,O,T,E,_,T,h,i,s,_,e,x,a,m,p,l,e,_,i,s,_,f,o,r,_,
    |i,l,l,u,s,t,r,a,t,i,v,e,_,p,u,r,p,o,s,e,s,_,o,n,l,y|00|
(0) |

```

FIG. C-1. Example of a string sub-block that may be part of a File Descriptor Block. The two bytes immediately preceding a keyword give the offset to the next string. The two bytes are indicated by the symbol |_|_|, but the offset itself is shown in small bracketed decimal characters in this example.

```

(26) |A,M,P,L,I,T,U,D,E,_,R,E,C,O,V,E,R,Y,_,N,O,N,E|00|
(14) |D,E,L,A,Y,_,0,_,0,2,0|00|
(16) |F,I,X,E,D,_,G,A,I,N,_,8,4|00|
(27) |H,I,G,H,_,C,U,T,_,F,I,L,T,E,R,_,1,0,0,0,_,-,1,2|00|
(14) |L,I,N,E,_,I,D,_,1,0,0|00|
(24) |L,O,W,_,C,U,T,_,F,I,L,T,E,R,_,2,5,0,_,1,2|00|
(21) |N,O,T,C,H,_,F,R,E,Q,U,E,N,C,Y,_,6,0|00|
(13) |P,O,L,A,R,I,T,Y,_,1|00|
(31) |R,E,C,E,I,V,E,R,_,V,E,R,T,I,C,A,L,_,G,E,O,P,H,O,N,E|00|
(33) |R,E,C,E,I,V,E,R,_,L,O,C,A,T,I,O,N,_,1,2,0,_,5,_,0,_,
    |0,_,2,5|00|
(42) |R,E,C,E,I,V,E,R,_,S,P,E,C,S,_,M,a,r,k,_,P,r,o,d,u,c,t,s,
    |L,2,8,E,_,4,0,_,H,z|00|
(25) |S,A,M,P,L,E,_,I,N,T,E,R,V,A,L,_,0,_,0,0,0,2|00|
(45) |S,O,U,R,C,E,_,1,2,-gauge,_,W,I,N,_,V,W,1,2,B,L,_,
    |b,l,a,c,k,_,p,o,w,d,e,r|00|
(24) |S,O,U,R,C,E,_,L,O,C,A,T,I,O,N,_,1,0,0,_,0|00|
(10) |S,T,A,C,K,_,1|00|
(26) |T,R,A,C,E,_,T,Y,P,E,_,S,E,I,S,M,I,C,_,D,A,T,A|00|
(176) |N,O,T,E,_,T,h,i,s,_,e,x,a,m,p,l,e,_,d,e,s,c,r,i,b,e,s,_,
    |t,r,a,c,e,_,t,h,a,t,_,w,a,s,_,r,e,c,o,r,d,e,d,_,w,i,t,h,
    |_,4,0,_,H,z,_,v,e,r,t,i,c,a,l,_,g,e,o,p,h,o,n,e,_,o,f,f,s,
    |_,2,0,_,5,_,m,_,f,r,o,m,_,t,h,e,_,1,2,-gauge,_,s,h,
    |g,u,n,_,s,o,u,r,c,e,_,|0A|0D|
    |T,h,e,_,s,a,m,p,l,e,_,i,n,t,e,r,v,a,l,_,w,a,s,_,2,_,m,i,c,r,
    |s,e,c,o,n,d,s,_,|00|
(9) |

```

FIG. C-2. Example of a string sub-block that may be part of a Trace Descriptor Block. The two bytes immediately preceding a keyword give the offset to the next string. The two bytes are indicated by the symbol |_|_|, but the offset itself is shown in small bracketed decimal characters in this example.