SEED

Standard for the Exchange of Earthquake Data

Reference Manual
SEED Format Version 2.4

August 2004

Federation of Digital Seismographic Networks
Incorporated Research Institutions for Seismology
United States Geological Survey

This reference manual is published by the Incorporated Research Institutions for Seismology (IRIS).

SEED was jointly developed by members of the Federation of Digital Seismographic Networks (FDSN). We welcome your questions and suggestions. For more information concerning this reference manual or the SEED format, contact:

Dr. Timothy K. Ahern
IRIS
1408 NE 45th Street
Second Floor
Seattle, WA 98105
(206) 547-0393

Dr. Ray Buland
USGS
Box 25046
M/S 967
Denver, CO 80225
(303) 273-8414

Printing History: First Edition - March 1990
Second Edition - February 1993
Third Edition - January 2005

Credits:

Initial design:	Ray Buland, United States Geological Survey
Initial Implementation, and Documentation:	Scott Halbert of the USGS performed the initial documentation.
Documentation:	Tim Ahern, Incorporated Research Institutions for Seismology Ray Styles, Science Information Associates, Inc.
Second Edition	Tim Ahern, Incorporated Research Institutions for Seismology Kris Skjellerup, Incorporated Research Institutions for Seismology
Third Edition	Tim Ahern, Incorporated Research Institutions for Seismology Deborah Barnes, Incorporated Research Institutions for Seismology

Chapter 1

Introducing SEED

What is SEED?

The Standard for the Exchange of Earthquake Data (SEED) is an international standard format for the exchange of digital seismological data. SEED was designed for use by the earthquake research community, primarily for the exchange between institutions of unprocessed earth motion data. It is a format for digital data measured at one point in space and at equal intervals of time.

SEED helps seismologists who record, share, and use seismological data. By providing a standard, SEED makes transmitting, receiving, and processing earthquake data easier and more accurate. Before the introduction of SEED, ease and accuracy had been goals, but not really attained.

SEED's Background

Before the 1970's, analog media (usually paper or film) were the sole means of recording seismic data. When researchers wanted to exchange this recorded data, they had to deliver the original seismogram recordings or copies. When they needed to describe important associated information about a recording, they would write the information — such as the station code, starting and ending date and time, time corrections, instrument orientation, dominant frequency pass band, and magnification — by hand on a seismogram. To extract information from such data, researchers had to measure a seismogram's parameters by hand. Interpretation relied on additional information (including station coordinates, instrument types, and operating organization) derived from the station operator or from standard sources such as the National Earthquake Information Service (NEIS) *Seismograph Station Codes and Coordinates* (see Presgrave (1985)).

Seismologists have used digital recording methods since about 1970. While these methods have increased data quality, they have also meant new challenges. Data exchange has been complicated by different data logger formats, by different computer systems, and by incompatible exchange media. One cannot use, or even visually examine digital data without extensive processing with additional computer hardware and software.

Many researchers collect digital earthquake data to perform computationally intensive waveform analyses. These analyses require much more auxiliary information — information such as detailed instrument responses. This auxiliary information is needed in computer-readable form and is sometimes distributed on the same electronic media as the raw seismic data. Some institutions collect and distribute data in various formats, and whoever receives that data may have to do substantial work to read it. Sometimes the required auxiliary information is incomplete or not even available in computer-readable form.

Seismologists around the world are aware of these problems, and have recognized the need for a seismic data exchange standard to solve them. Many have created exchange formats, but none has succeeded in creating a de facto standard. In 1976, the International Deployment of Accelerometers (IDA) format was introduced. Perhaps because of its wide distribution, the most successful format before SEED may be the United States Geological Survey (USGS), Global Digital Seismograph Network (GDSN), Network-Day Tape format. This format became the only significantly used format within the United States until 1987, when the Federation of Digital Seismographic Networks (FDSN) formally adopted SEED. While the GDSN format has been adopted or emulated for some networks, it has not worked as an exchange standard because: 1) the cost to reformat data for it is often too high, 2) it is too limited for other seismic applications because of its orientation toward continuously-recorded global observatory data, and 3) it is not flexible enough to encompass changes in seismic instrumentation and computer media technologies.

In 1985, the International Association for Seismology and Physics of the Earth's Interior (IASPEI), Commission on Practice, formed a Working Group on Digital Data Exchange to propose a standard for international digital seismic data exchange. Soon afterward, the FDSN formed and assumed the responsibility for developing an exchange format. At their first meeting in August 1987, the Federation working group reviewed a number of existing formats — including SEED, a new format proposed as a starting point for discussion by the USGS for the working group. At a follow-up meeting in December 1987, intensive discussion and numerous clarifications and modifications led to the new format's adoption as a Federation draft standard. Other groups are also considering adopting SEED as their standard. We have prepared this manual to help you discover SEED's benefits.

A Description of SEED Format Versions

The SEED format is presently quite stable. During the first several years of its existence several changes were incorporated into the format. This section briefly summarizes the various versions of SEED and what the major changes and additions are between the format versions.

Items that are new since the last SEED manual was published in March 1990 are labeled either v2.2, v2.3 or v2.4. New blockettes in versions 2.2, 2.3 or 2.4 are indicated in the blockette descriptions. New fields in blockettes in versions 2.2, 2.3 and 2.4 are identified to the left of the box summarizing the field names for each blockette.

Version 2.0, February 25, 1988

The first officially released version of SEED was version 2.0. This version is documented by Halbert, Buland and Hutt in a publication of the United States Geological Survey on February 25, 1988. The basic structure and philosophy of the SEED format have not changed significantly from this first documentation of the format. The existence of the Volume, Abbreviation, Station, and Time Span Control Headers were all defined at this time. The following paragraphs attempt to identify the major changes in the SEED format in the later versions.

Version 2.1, March, 1990

Version 2.1 was documented in an Incorporated Research Institutions for Seismology (IRIS) publication in March of 1990. The manual clarified many items in the SEED format as well as correcting mistakes in the previous manual. Many of the new blockettes that were added were in the area of Response Abbreviation Dictionaries. The following blockettes were added.

Blockette 8 - Telemetry Volume Control Header. This blockette was added to provide a mechanism whereby the Volume Control Headers did not have to be transmitted unless they had changed.

Blockette 43 - Response (Poles and Zeros) Dictionary Blockette. This blockette was added so that poles and zeros could be represented once and then referred to by the Response Reference Blockette. This blockette contains similar information as blockette 53.

Blockette 44 - Response (Coefficients) Dictionary Blockette. Similar to Blockette 43 only for Blockette 54 information.

Blockette 45 - Response List Dictionary Blockette. Similar to Blockette 43 only for Blockette55 information.

Blockette 46 - Generic Response Dictionary Blockette. Similar to Blockette 43 only for Blockette 56 information.

Blockette 47 - Decimation Dictionary Blockette. Similar to Blockette 43 only for Blockette 57 information.

Blockette 48 - Channel Sensitivity/Gain Dictionary Blockette. Similar to Blockette 43 only for Blockette 58 information.

Blockette 57 - Decimation Blockette. This blockette was added to be able to completely specify instrument responses of some of the newer data loggers. This blockette is now routinely used by several Data Centers of the FDSN.

Blockette 60 - Response Reference Blockette. This blockette was added so instrument responses in the response abbreviation dictionaries could be referenced. This mechanism can greatly decrease the amount of overhead within a SEED volume.

Blockette 74 - Time Series Index Blockette. This blockette essentially replaced blockette 73 in version 2.0. It provides an index to all of the continuous time series on the SEED volume and is placed immediately before any data is placed on the volume. This allows one to quickly gain information about the time series in the volume by only referring to the first part of the SEED volume. In version 2.0 this information was intermingled with the data.

Version 2.2, August, 1991

The FDSN Working Group on Data Formats met again in conjunction with the IUGG meeting in Vienna, Austria. The modifications to the SEED format were fairly minor at this meeting. A SEED reference manual was not published to document changes in the SEED format but the minutes of the FDSN meeting in Vienna do provide documentation. The following identifies the most significant changes that were new with SEED version 2.2.

Two new blockettes were adopted.

Blockette 61 - FIR Response Blockette. Blockettes (54) and (44) have traditionally been used to represent FIR filter coefficients in the SEED format. These blockettes require that all coefficients are specified and that an error for each coefficient be given. In practice most FIR filters possess some symmetry properties and the error for the coefficients is not used. For this reason blockette 61 was introduced so that the FIR filter specification would require less space.

Blockette 41 - FIR Dictionary Blockette. Blockette 41 is the abbreviation dictionary blockette that corresponds to blockette 61.

Dataless SEED Volumes. The FDSN also adopted the practice of generating dataless SEED volumes with SEED version 2.2. A dataless SEED volume contains the normal Volume, Abbreviation, and Station Control Headers but omits the Time Span Control Headers and the data records. The purpose of these volumes is to provide an alternate method for making sure that various Data Centers have current and correct information for seismic stations.

Version 2.3, December, 1992

The FDSN met in Seattle, Washington, USA in December of 1992. At that time SEED version 2.3 was adopted. Some new blockettes were added and several additional fields were added to existing blockettes.

The following blockettes have been added with SEED version 2.3:

Blockette 42 – Response Polynomial Dictionary. Use this blockette to characterize the response of a non-linear sensor.

Blockette 62 – Response (Polynomial) Blockette. Use this blockette to characterize the response of a non-linear sensor. The polynomial response blockette describes the output of an Earth sensor in fundamentally a different manner than the other response blockettes. The functional describing the sensor for the polynomial response blockette will have Earth units while the independent variable of the function will be in volts. This is precisely opposite to the other response blockettes. While it is a simple matter to convert a linear response to either form, the non-linear response (which we can describe in the polynomial blockette) would require extensive curve fitting or polynomial inversion to convert from one function to the other. Most data users are interested in knowing the sensor output in Earth units, and the polynomial response blockette facilitates the access to Earth units for sensors with non-linear responses.

Blockette 100 - Sample Rate Blockette. At times the sample rates in blockette 52 and the fixed section of the data header have proven to be inadequate to represent the sample rate to the desired precision. An optional blockette 100 was added to allow a floating point sample rate to be entered. If present, it overrides the sample rate in blockette 52 and the fixed section of the data header.

Blockette 1000 - Data Only (MiniSEED) Blockette. Data records by themselves do not contain enough information by themselves to allow time series data to be plotted or even simply analyzed. With the addition of a small amount of additional information these limitations are removed. Blockette 1000 is the Data Only (MiniSEED) Blockette that will allow SEED data records to be self-sufficient in terms of defining a time series.

Blockette 1001 – Data Extension Blockette.

Blockette 2000 – Variable Length Opaque Data Blockette.

SEED allows for the modification of existing blockettes by appending new fields to the end of existing blockettes. This is always done in a fashion that maintains compatibility with older format versions. SEED readers that do not support the newer version simply ignore the added fields. With SEED version 2.3, two of the existing blockettes were modified.

The following blockettes have been modified with SEED version 2.3.

Blockette 10 - Volume Identifier Blockette. Two fields were adopted at the end of Blockette10. Field 7 is the volume time and provides the actual date and time that the volume was written. Field 8 provides a place to document the name of the organization that wrote the SEED volume. An optional ninth field was added to this blockette that can be used to label a particular SEED volume.

Blockette 50 - Station Identifier Blockette. Field 16 was added to the Station Blockette to designate the Network Code. This two letter code is assigned by the FDSN and must be present.

Blockette 71 - Hypocenter Information Blockette. Three fields were added to blockette 71. Field 12 now contains the Flinn-Engdahl seismic region number. Field 13 was added to provide the Flinn-Engdahl seismic location number. Field 14 was added to provide the Flinn-Engdahl standard geographic name.

Blockette 72 - Event Phases Blockette. Field 11 was added to the Event phases blockette to identify the source of the phase pick. It refers to abbreviation blockette 32. Field 12 was added to the Event Phases Blockette to designate the Network Code of the station for this pick.

Blockette 74 - Time Series Index Blockette. Field 16 was added to designate the Network Code of the designated Time Series.

Fixed Section of Data Header. Field 7 was previously defined as two reserved bytes. With SEED version 2.3 these bytes have now been assigned as the Network Code.

Version 2.4, April, 2004

The following change was introduced in version v2.4.

Fixed Section of Data Header. Field 2 is now defined as the data quality indicator, indicating level of data quality control that has been applied.

What This Manual Covers

SEED's design goals, implementation strategy, and recommended usage are the subjects of the remainder of this first-chapter. Chapter 2 provides an overview of SEED's format structure, and introduces blockettes. Chapter 3 presents some conventions — important information that you will need to use SEED effectively. Chapters 4 through 7 describe the format standard for control headers. Chapter 8 details the data records. A series of appendices follow; each contains information that may help you.

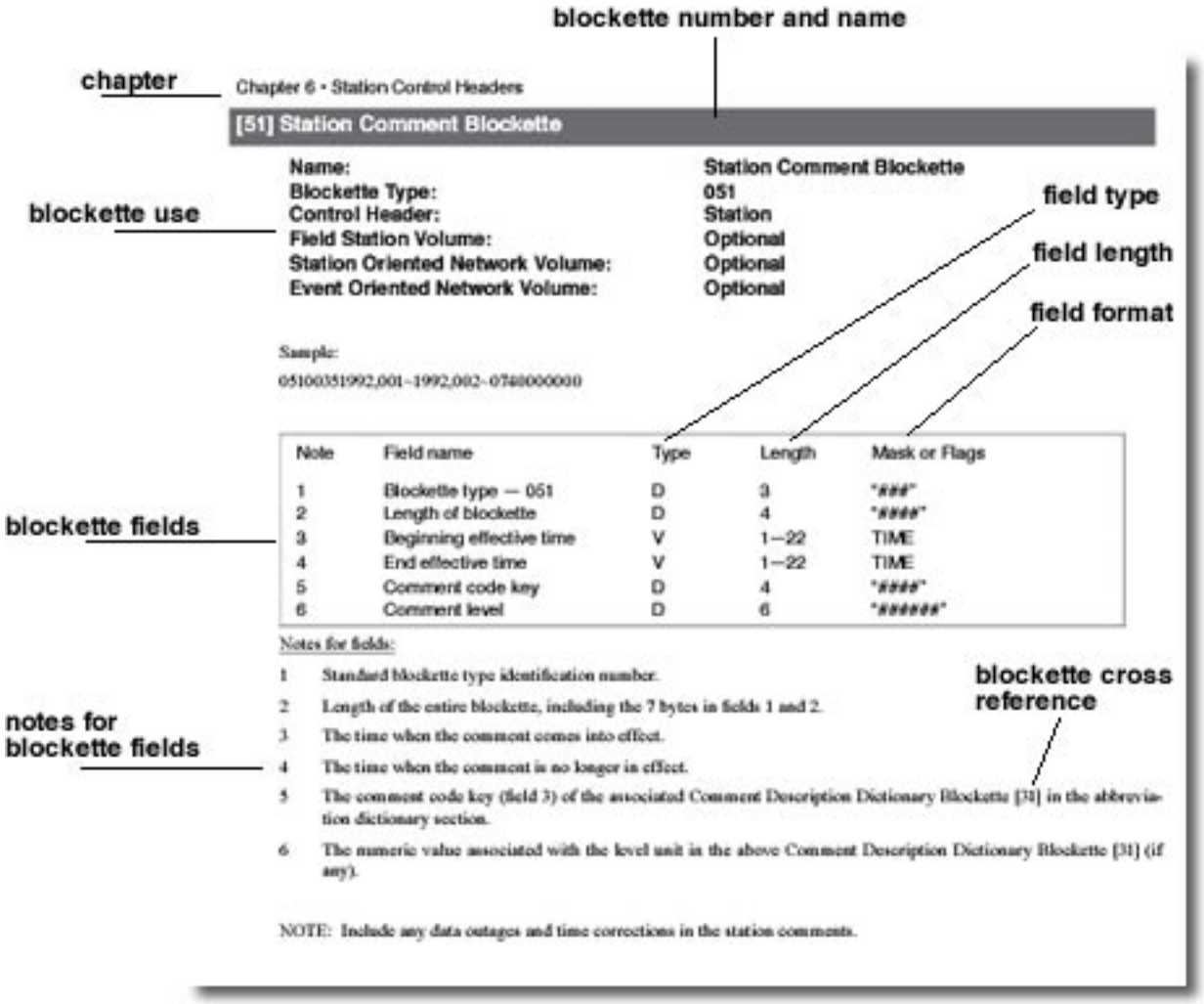


Figure 1: Sample Page

In preparing this manual, we wanted to specify the structure and organization of the format, as well as the placement, value range, and coding of every defined parameter. While many readers may be interested in which parameters are included in the specification, how they are encoded, and how they are organized, we recognize that only a few readers will find the details of the format immediately interesting. However, we wanted to document the format as completely as possible so that you can learn about SEED's features, use it, benefit from its usefulness — and evaluate it thoroughly.

At a Federation meeting held in Blanes, Spain, on June 19, 1988, the Federation working group adopted the SEED format as an international standard for the exchange of Federation seismic data. Beginning on January 1, 1988, GDSN data has been available in SEED format. The Data Management Center (DMC) at the Incorporated Research Institutions for Seismology (IRIS) has adopted SEED, and uses it as the principal format for its IRIS/DMC datasets. IRIS has also developed a SEED reading program to help researchers convert these datasets into trace formats for which analysis tools already exist. By distributing this document now, we want to make the entire seismological community aware of the format, and we want to solicit your comments.

Acknowledgments

This manual represents the combined effort of many individuals within the Federation of Digital Seismographic Networks (FDSN) and could not have been produced without the cooperation of its members. The SEED format was reviewed and modified at a number of meetings from 1987 to 1992; the first was hosted by the USGS in Albuquerque, New Mexico,

December 3-4, 1987. The next meeting was the SEED Programmers' Interest Group in Golden, Colorado, December 14, 1988. May 10-11 of 1989 the Federation working group met in Baltimore, Maryland, where the SEED format was further developed and significant support for the format was shown by both the United States and international attendees. The result of these efforts is evident in the first edition of the SEED Manual version 2.1.

After the publication of the first edition of the SEED Manual there was a meeting in August 1990 in Golden, Colorado at which the SEED format underwent some modification. The next meeting was held in Vienna in August of 1991 at which time more revisions were discussed to improve the SEED format. This resulted in Version 2.2. The last meeting was held in Seattle, Washington, at which enhancements were made resulting in the present version of SEED 2.3. The following is a list of the institutions with representatives in attendance at one or more of the above mentioned meetings.

Australian Geological Survey (AGS)	Australia
Geological Survey of Canada (GSC)	Canada
Geoscope	France
GERESS	Germany
German Regional Network	Germany
Harvard University	U.S.
Grafenberg	Germany
Incorporated Research Institutions for Seismology (IRIS)	U.S.
MEDNET	Italy
NORSAR	Norway
ORFEUS	The Netherlands
Poseidon	Japan
Quanterra, Inc.	U.S.
University of California, San Diego	U.S.
University of Texas	U.S.
University of Washington	U.S.
USGS/Albuquerque Seismic Lab	U.S.
USNSN	U.S.

The SEED Format is maintained by a working group from within the Federation. Its present members are listed on http://www.fdsn.org.

Introduction to the Format

The SEED format may be used in successive steps. For example, the format might be used to transfer data from a station processor to a data collection center, then to a data management center and, finally, to an end user. (The SEED databases may be augmented or modified at stages along the way.) Additionally, data collection centers and data management centers are using features of the format for archival storage and data retrieval.

Although SEED has evolved primarily for institutions that exchange data, some seismologists have already proposed other uses for the format. Some IRIS seismic station processors will generate SEED data in the field. This will facilitate the collection and distribution efforts required, and the possibility of error introduction should diminish. Seismologists are also using the format to transmit seismic data by electronic means, such as a packet switching network. The Yellowknife array of the Geological Survey of Canada is transmitting SEED data via satellite. Finally, researchers working with other ground based geophysical observations unrelated to earthquakes (e.g., strain, tilt, or magnetic field) may find SEED suitable.

The SEED format was not designed for use with non-time series data, nor with time series data sampled at unequal intervals in time (nevertheless, mechanisms for including console logs were easily accommodated). These types of data are rare enough that complicating the format to include them does not seem worthwhile. We also did not design SEED for the exchange of processed (e.g., filtered) data or synthetic data (i.e., created by computer modeling). While such use is possible, we will not support it.

Design Goals and Strategies

The SEED format results from the design contributions of many seismologists and computer professionals. Their experience includes constructing other special seismic data distribution formats; in coordinating computer, operating system, and mass storage device compatibilities, behaviors, and peculiarities across a wide range of manufacturers; and in working with the International Standards Organization (ISO), American National Standards Institute (ANSI), and other industry standards. The result: a format that can meet the needs of many individuals and institutions that collect, record, transmit, and read seismological data.

The SEED format relies on a few assumptions that are common to all digital seismic data exchange formats currently in use for network data. First, computer architectures commonly use the 8-bit byte, so this has become a de facto basis for the format. Second, data from several sources — many field stations, each containing different channels, recorded over a few discrete time spans — might make up a typical logical SEED volume. (Note that these assumptions do not prohibit less demanding uses, such as the recording of data from a single geophysical observatory.) Several logical volumes together could fit on a single physical volume. Each logical volume should begin with auxiliary, or parametric, information organized into one or more control headers, followed by a stream of raw, time series data.

Based on those assumptions, the SEED designers applied their experience to create a format with certain goals in mind. As they worked, they realized that they needed to implement certain strategies in order to reach those goals. They created SEED to meet these criteria:

General. SEED works for continuous, station oriented and event oriented waveform data from single field stations, observatories, networks, or arrays. (SEED can also work with other geophysical time series data of potential seismological interest.) However, generality sufficient to support both station and event-oriented data requires considerable format complexity.

Generality in defining important auxiliary information adds to this complexity. Because computers read SEED data, this complexity requires a sophisticated data-reading program. Such reader software is available from IRIS and other sources.

Self-defining. The data from each channel includes all the needed information. This self-defining feature makes automatic processing easier. A series of control headers defines the auxiliary information as global to the entire volume, as well as specific to each channel. At a minimum, information about the volume, about station-channel characteristics, and about the data's time span appears in separate control headers that precede the time series data. Station control headers include station location and channel response (transfer function) information. The time span control headers allow for hypocenter and phase reading information. Additional auxiliary information, specific to a particular station-channel at a particular time, is embedded in the data stream.

Robust. SEED data include enough embedded information to allow recovery from recording and transmission errors. Logical records divide each volume. Each logical record begins with information on the record type and its absolute sequence within the volume. Logical control header records are encoded entirely in printable American Standard Code for Information Interchange (ASCII) characters conforming to the ANSI standard. They are therefore directly readable by people, so control header records that have been damaged or incorrectly written can be interpreted. In addition, each logical data record has header information embedded between the record identification information and the data. This embedded information permits unambiguous identification of the data, should all the control headers be missing or destroyed.

Efficient. SEED minimizes wasted space for storage and distribution. As digital seismic data become available in increasing quantities, wasted space becomes too costly. SEED's storage efficiency creates an additional benefit: you can access the data with fewer input requests. SEED uses variable length and optional fields within the control headers to conserve space. You can abbreviate lengthy, repeated ASCII data items: an abbreviation dictionary control header defines those abbreviations. After the control headers, all subsequent data records (with embedded information) are coded as binary, and can be variable in their total length. A system of indices (cross references) to logical records make for efficient

access: the volume header refers to the station-channel and time span headers; the time span headers refer to the data records.

Portable. The SEED format is suitable for easy and effective reading by any commercially available computer, using any commercially available operating system and any commercially available dismountable mass storage media. At the time of this manual's preparation, several SEED reading and writing programs exist for platforms by Digital Equipment Corporation, Sun Microsystems, and some personal computers. Operating systems that work with SEED now are SOLARIS, LINUX, Windows, OS X and other forms of UNIX are supported. Also, the lengths (in 8-bit bytes) of SEED logical records are always a fixed power of two. This means efficient data storage and rapid random access on the widest possible variety of computer equipment. Exchange volumes can have logical record sizes of 256 8-bit bytes or larger. Formatted data structures and unformatted data types are appropriately aligned, conforming to high-level computer language standards.

Computer readable. Humans do not usually read comment information associated with data, despite the apparent advantages of including such comments. Nevertheless, a computer can extract a summary of relevant comment information for the user from SEED data. For example, the IRIS SEED reader software currently provides summary listings of comments for station and channel data. The SEED control headers are coded in a computer readable hierarchy of blockettes. These blockettes contain fixed and variable length data fields. The blockettes are self-identifying sequences of data fields that assist computer readability, storage efficiency, and flexibility. Comment information is contained in a brief, computer readable form; the actual comment is defined in the abbreviation dictionary control header.

Referencing fields within blockettes relate important information. For example, fields 8 and 9 of blockette [52] refer to blockette [34]. For more information about cross-referencing fields, see Appendix F.

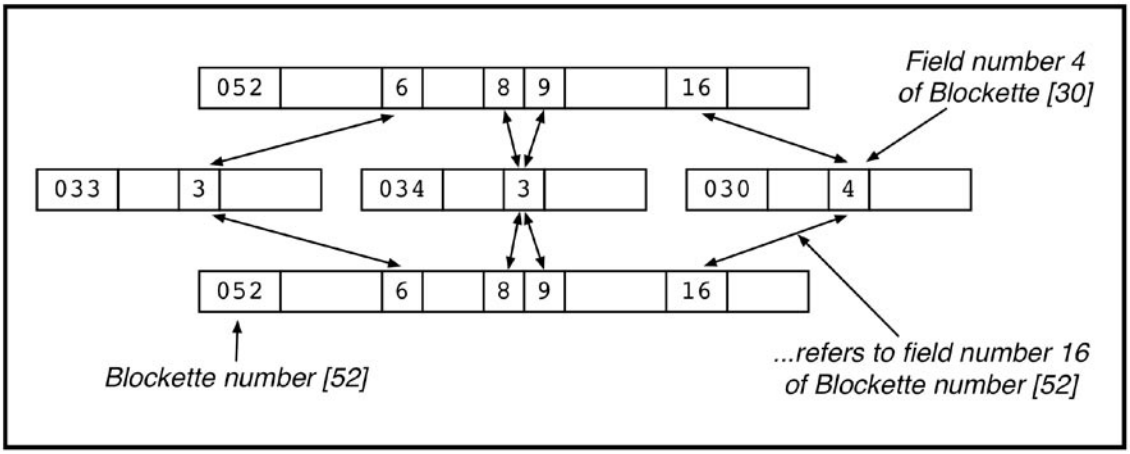


Figure 2: Blockettes with Cross-Referencing Fields

Computer usable. By not having to transcribe large amounts of information to prepare SEED data — an awkward activity at best — fewer human errors are introduced. You do not have to use text-editing software because SEED supports the automatic writing and reading of auxiliary data. You can write all parametric information using any computer language for any computer. Since auxiliary information is binary and is embedded in the data records, you cannot easily read or write it without a computer. This keeps data safe from accidental editing.

Flexible. SEED supports currently unforeseen future usage by including extensible auxiliary information with the time series data. If desired, you can define new field digitization formats without modifying the format standard. Each blockette and its trailing data fields are optional. Blockettes that are represented always identify themselves and measure their lengths. This allows you to append new data fields to blockettes that are already defined (or new blockettes to be defined), all without altering the existing control header structures in any way. This overcomes a significant failing of other, older formats — a failing that quickly made them obsolete.

Self-correcting. Even though errors will find their way into distributed station-channel descriptive information and other data, SEED provides a means of distributing corrections for preceding volumes within a subsequent logical volume. You can distribute the corrected station control header, flagging it to supersede the previously distributed information that needs updating for a given effective time period. You can specify that the correction information should apply to the new data on the same distribution volume, or not. If you do not want the correction information to apply to that new data, just create more than one station control header for the same station on the SEED volume — one for the older data, and one for the new data. You will also need more than one station control header for the same station if station or channel characteristics were changed during the time span of the volume — for example, after a maintenance site visit. (In this case, do not set the update flag.)

Can store most field digitization formats. SEED efficiently uses field digitization and recording data word formats. This is a benefit because separate conversions to other formats can be costly and may sacrifice precision or waste storage space. Also, data logger problems may only be traceable in the original format — which is what SEED uses. SEED's data description language permits specifying any reasonable field digitization or recording data word format. SEED reading programs convert it into binary data suitable for the reading computer. The abbreviation dictionary control header abbreviates the ASCII data description string.

Can access any sequential or random access media. Although seismologists will continue to use sequential media such as magnetic tapes for the foreseeable future, SEED also supports more efficient data access on randomly accessible media, such as magnetic and optical disks. SEED uses fixed length logical records. These allow efficient random access on any media available to any computer. SEED provides indices to allow efficient random access of logical records when available. And, for sequential media, these indices facilitate the use of skip rather than read operations.

Usable as a field recording format. SEED allows you to use a modified subset of the format for data recording at a field station, but several constraints are imposed by the real-time nature of station processors. The entire time span control header and some fields in the other control headers cannot be created because the relevant information will not be available when the headers are written. Also, station processor memory constraints may require smaller logical records and block multiplexing of data channels (each logical record contains data from one channel, but successive logical records are interspersed with logical records from other channels). In addition, state-of-health, console log, and calibration information — of direct interest only to the institution collecting the data — can be included. Summaries of such data may appear in appropriate station control header fields when the exchange volume is created. Finally, a special telemetry volume format is available.

Efficient at merging data. Data collection and data management centers can merge data from several standard format volumes onto a new volume, with minimal changes to the auxiliary information and with no changes to the logical block structure for the data and for most of the control headers. SEED supports different logical record sizes. However, to simplify the merging of volumes, the lengths (in bytes) of logical records will always be a power of two. This means that any logical record size is a multiple or sub-multiple of any other, ensuring compact storage and rapid access on most computer devices. You can efficiently string together many smaller logical records into one larger logical record, without changing any data item except the logical record identification information.

Recommended Uses

Although the SEED format is flexible, we recommend limiting its use in some cases. As you use this format, keep the following in mind:

Multiplexing. Whenever possible, de-multiplex all data. We discourage the multiplexing of data channels with a common sample rate, and from one station, even though SEED supports it. The format can become very complicated and convoluted, and data compression can become difficult when using multiplexing. We support block multiplexing of channels from one station for field tapes, but not for data exchange. SEED permits the multiplexing of array data if absolutely necessary.

Special Considerations for Multiplexed Data in Data Records. The following blockettes must be properly specified. Blockette 30: Read Appendix D carefully, paying attention to the section on Integer Format - Family 0. Key 1 should specify the number of channels being multiplexed. Blockette 52: A blockette 52 should be written for each multiplexed channel. Field 5 will specify which subchannel is being described. Sub channel numbering starts with one.

Word order. SEED utilizes the Big Endian word order as its standard.

Logical record size. Each volume establishes a logical record size for itself. We strongly recommend a logical record size of 4096 bytes for volumes created by data collection and data management centers. At the time of this writing (2004), logical record sizes in excess of 4096 bytes have not been written. Logical record sizes as small as 256 bytes are supported. While all logical records on a SEED volume are usually the same size, the format can make exceptions for data records written in the field. We recommend using data records as large as 4096, and using the variable size feature only when absolutely necessary — and then only after coordination with the appropriate data collection center to ensure readable data. In practice, smaller record lengths are better for real-time data, as this reduces latencies.

Channel response. You can define each station-channel's transfer function in a variety of ways. These include a concatenation, or "cascade" of formal mathematical descriptions of analogue and digital filter sections, tables of amplitude and phase, or simple approximations. At a minimum, give the most complete and mathematically correct description of each station- channel transfer function available. In addition, you can give alternate descriptions that may have more intuitive appeal.

Hypocenters and phase data. For event oriented data, you can assume that the phase arrival times or readings given in a time span control header are associated with the hypocenter(s) given in that same control header. For station oriented data, the hypocenters are optional and may be taken from a catalog. The readings are also optional, and they may be unassociated estimates made automatically in the field by the station processors. In either case, the readings refer to positions in the time series data that immediately follow.

Dataless SEED Volumes. It is legal to produce a SEED volume that contains only header information and no data. This may prove to be an expedient way for various data centers to insure that they have current information from various networks. In this case one would include volume, abbreviation and station control header but omit time span control headers and data records.

Dataonly SEED Volumes. Just as the SEED format encompasses Dataless SEED volumes, it also defines the use of Dataonly or MiniSEED volumes. Dataonly SEED volumes contain only SEED data records and blockettes in data records as defined in this manual (see Appendix G). For individuals that feel sure that they know station characteristics and do not need to repeatedly receive SEED volume control header information, Dataonly SEED volumes may be an appropriate method of data transfer. Other than fixed section of data header information, these volumes contain nothing but time series. In fact this subset of SEED looks very similar to other trace analysis formats. If possible use the MiniSEED blockette (blockette 1000) to insure the information is totally self-defining.

Console logs. Data written at a field station may include console logs. The log information appears as a separate data channel with the appropriate data family code. Printable ASCII, standard forms control characters, and other special characters such as the ASCII bell (BEL) are acceptable. Alternate character sets (such as Kanji) are acceptable for languages that do not use the U.S. ASCII character set. We suggest formatting console log information in a way that permits automatic parsing at a data collection center — reducing manual labor and human error. These same comments apply to nominal data such as telemetry flag status words, or door opened/closed flags, except that this information should appear as if it were an equally sampled time series.

Conventions

In addition to the recommended uses described above, SEED supports the following features:

End-of-file marks. Occasionally, some types of media need end-of-file marks added to the data stream (or they may be convenient for some purpose beyond the scope of normal SEED usage). SEED makes no use of single end-of-file marks and will ignore any present. The logical record sequence numbers must continue to increment after the embedded end-of-file mark. SEED interprets multiple end-of-file marks in sequence as the end-of-information for the physical volume. We require four end-of-file marks in sequence at the end of physical magnetic tape volumes to ensure that the reading program unambiguously interprets that point as the end of the volume.

Noise records. SEED writing programs can write blank or noise records at any time; SEED readers ignore such records. (These records are typically used to avoid a bad place on magnetic tape.) Use a correct logical record sequence number, ensure that the noise record has the correct logical record length, and set the remainder of the record (particularly the record type code) to spaces (ASCII 32). Noise records are particularly useful for field recording and for the starting points of 9-track magnetic tapes that have been used before.

Blank fields. Verify that all auxiliary information is as complete and correct as possible. If the current value of a particular field is not available, leave the field blank or set it to zero, as appropriate.

Field recording termination. Flush all data buffers before terminating a local field station recording. This ensures that data records for all channels will begin at about the same time on the following physical volume.

Header flushing. Sometimes it is useful to flush all data buffers and repeat all control header information periodically (e.g., each day at midnight). When flushing buffers, continue data recording after the repeated control headers. This strategy guarantees that low data rate channels are saved periodically and synchronizes the start times of data records for all channels. Control headers should contain current information if it has changed since the beginning of the volume. SEED supports, but does not require, header flushing as needed. Flush headers that have changed as a result of a site maintenance visit as soon as possible after making the change. SEED requires this, but only for the control headers that have changed.

Calibration. When performing calibration, testing, or repairs, field stations should always flag that maintenance operations are in progress. This notifies data collection personnel and end users of potential difficulties with the data from those field stations at those times.

Standards. Wherever practical, the SEED format follows existing internationally agreed-upon standards. SEED codes all character data according to the ANSI standard ASCII format (we recommend using uppercase alphabetic characters whenever possible). Where possible, we try to give physical units for transfer functions in uppercase SI units (Le Système International d'Unités, the international standard metric system). SEED uses other *de facto* standards such as 8-bit bytes and the two's complement binary integer data word format.

Chapter 2

An Overview of SEED

Format Organization

Because the SEED format was designed to achieve many goals, it may appear complex or formidable at first. Actually, its structure is straightforward. This section demonstrates that by outlining its organization. In this outline, we use a number of specialized terms. These terms have specific meanings for the SEED format; they are defined in the glossary at the end of this manual. Also, Appendix E shows some sample data in logical volumes; refer to it for typical examples. Keep in mind that only a subset of SEED would be used for any given application.

A complete and internally consistent implementation of the SEED format results in one logical volume. Depending on the media type, you can distribute one or more logical volumes on one physical volume. You may not, however, create a logical volume that spans more than one physical volume.

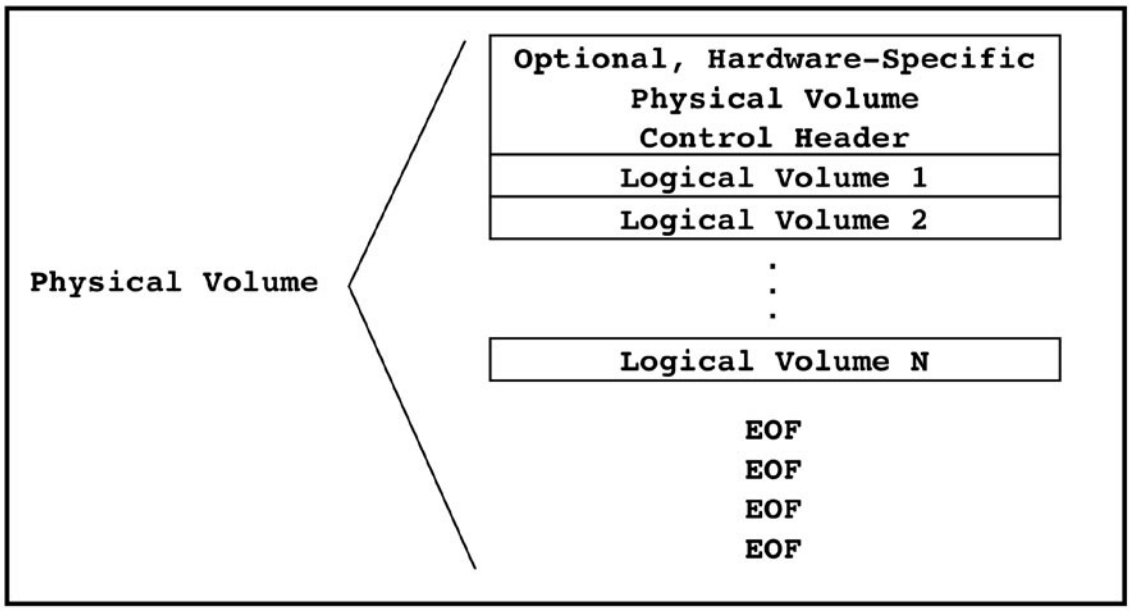


Figure 3: Logical Volume Organization Within a Physical Volume

Physical and Logical Volumes

At the highest level of organization (a physical volume), the SEED format consists of one or more logical volumes, one after another. In addition, some randomly accessible media require placing a device-dependent control header at the start of each physical volume to access the associated logical volumes. (This physical header is external to SEED. Reading and writing software does not have to manage these headers — the computer's media-accessing software must take care of that.)

Three types of logical volumes are possible: field station-, station network-, and event network- oriented. The structure of each logical volume is the same, but the interpretation of some data fields, particularly for hypocenters and phase arrival times, will be different.

Format Objects

Both station network and event network volumes contain sequences of format objects — complete and internally consistent collections of related information that describe some aspect of the data on the volume. Two format objects are used:

- *control headers* (formatted in ASCII) that contain auxiliary information about the volume, the station-channels, and the data
- *time series* (binary, unformatted) that contain raw data and embedded auxiliary information that is channel specific and time dependent

Control Headers

Four control headers are defined:

• *volume index control headers*: These contain information about the time of the data, logical record length, and format version of the logical volume, as well as indices to the station and time span control headers. The indices help retrieve data quickly.

• *abbreviation dictionary control headers*: These contain the definitions of abbreviations used in other control headers. Abbreviation dictionaries are referred to by 1) other abbreviation dictionary entries, 2) blockette [400] in the time series format objects, 3) station identifier blockettes [50] and [51], and 4) channel identifier blockettes [52] through [59] (see Appendix F).

Space is saved by making abbreviations for lengthy comments and field data format definitions. Other abbreviations are used to facilitate automatic processing and to guarantee coding consistency, particularly for units fields. (Note: The abbreviations, if used, are not optional; the fields that need them defined have no room themselves for their definitions.)

• *station control headers*: These provide relevant operating characteristics for a station and all of its channels — including station location, instrument types, and channel transfer functions. At least one station control header must exist for each station. The station control headers permitted on a volume are those that pertain to the stations on that volume.

• *time span control headers*: These identify the time span within which the time series that follow were recorded. They also provide indices to each time series, as well as information about the seismic events that may have occurred during the time interval. One volume may contain many time spans.

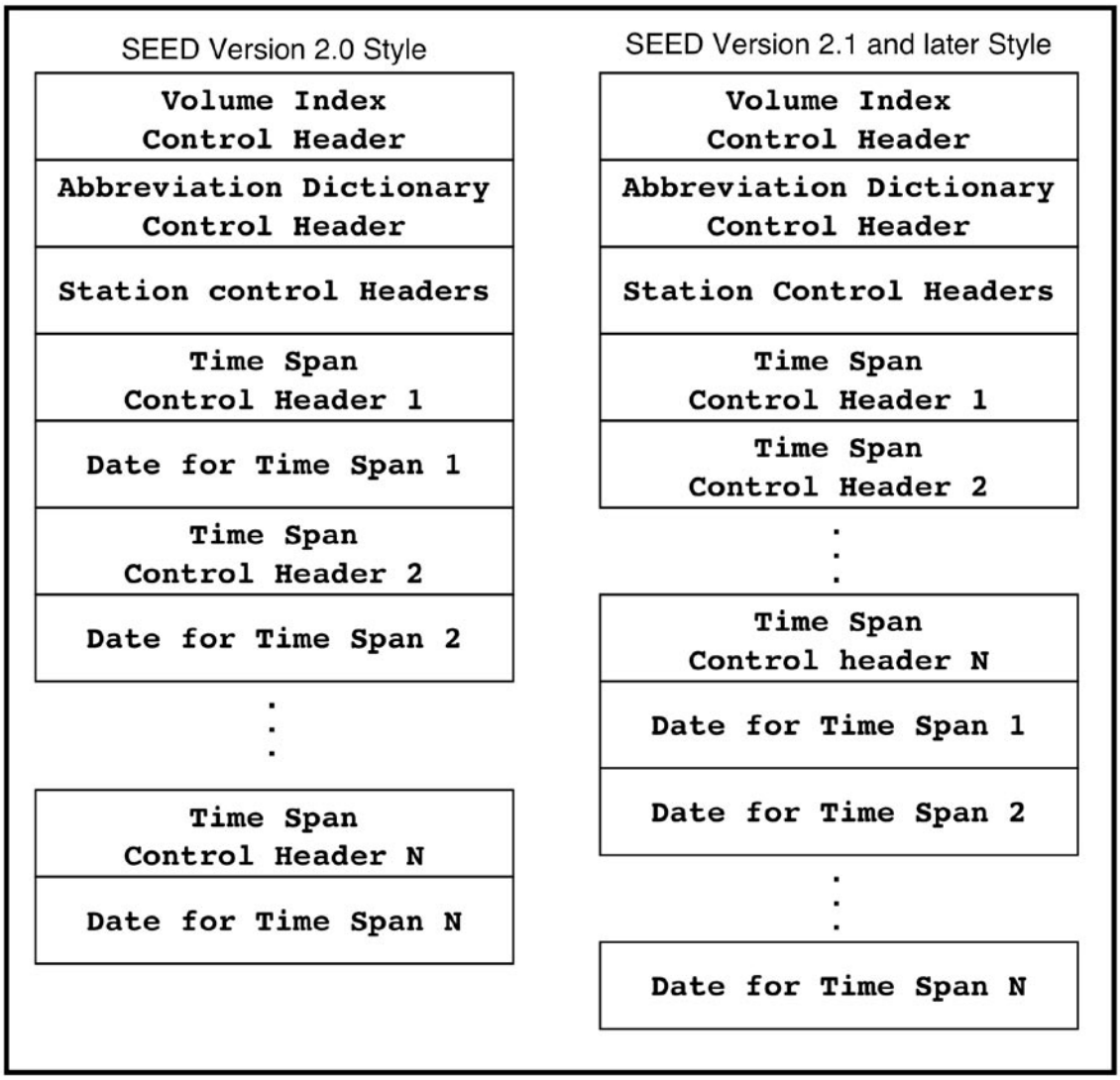


Figure 4: Format Object Organization Within a Logical Volume

SEED divides each format object into one or more fixed length logical records. Each logical record begins with a logical record identification block, and usually contains one or more fixed length physical records. However, it is possible to pack several logical records into one physical record — for example, the Albuquerque Seismological Laboratory (ASL) packs eight 4096- byte logical records into each 32768-byte physical record to minimize record mark overhead

(we do not recommend this practice for random access media). Different storage devices use physical records of different lengths, but this does not affect SEED users.

Version 2.1 requires that SEED writing programs move the time span headers together to a position immediately following the station control headers. Version 2.0 required that data for a time span be written after its associated time span header, resulting in time span headers being interspersed with their time data. SEED reading programs should support both format object organizations, but SEED writing programs should only write according to the version 2.1 organization.

Blockettes

Each control header is made up of a sequence of blockettes — data structures that contain a type identifier, length, and sequence of data fields specific to the blockette type. Blockettes may be either ASCII formatted (in control headers) or unformatted binary (in data records). Each data field contains auxiliary information on one topic, and may be either fixed or variable in length. Most blockettes are optional.

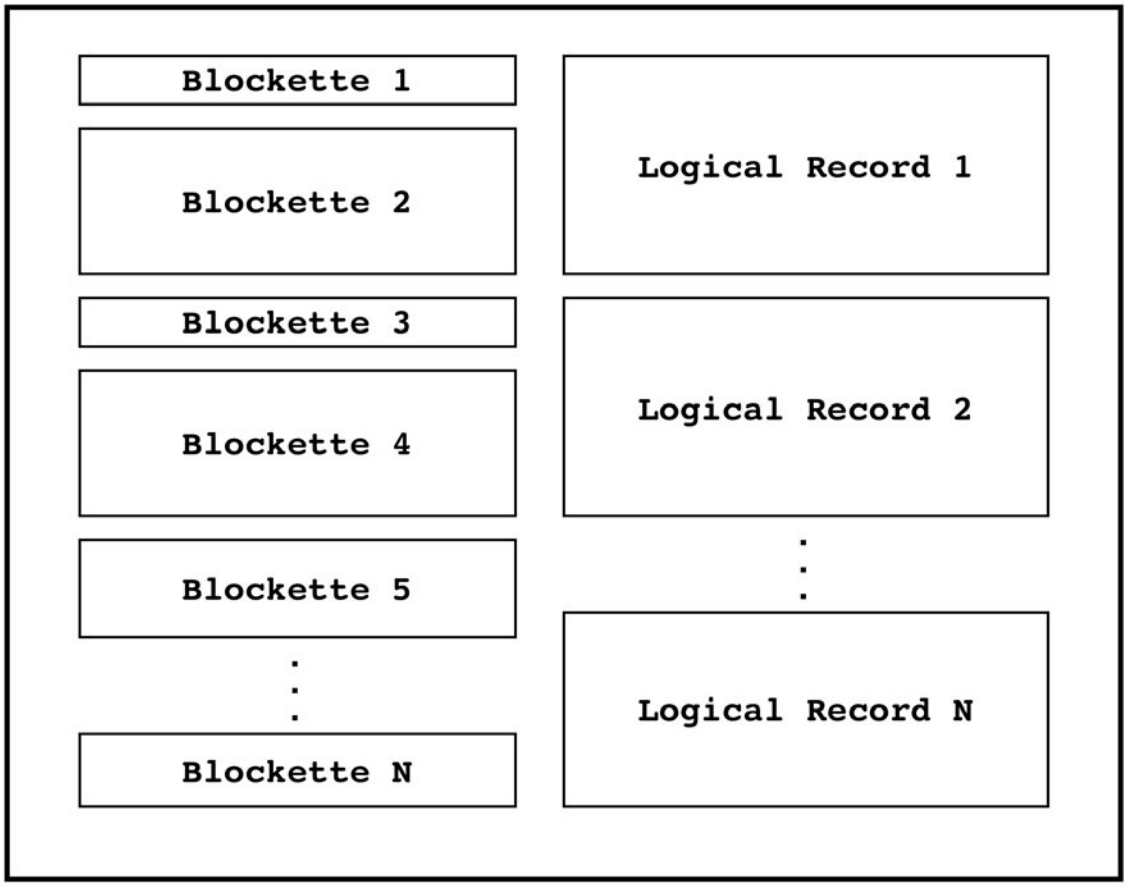


Figure 5: Organization of Control Headers

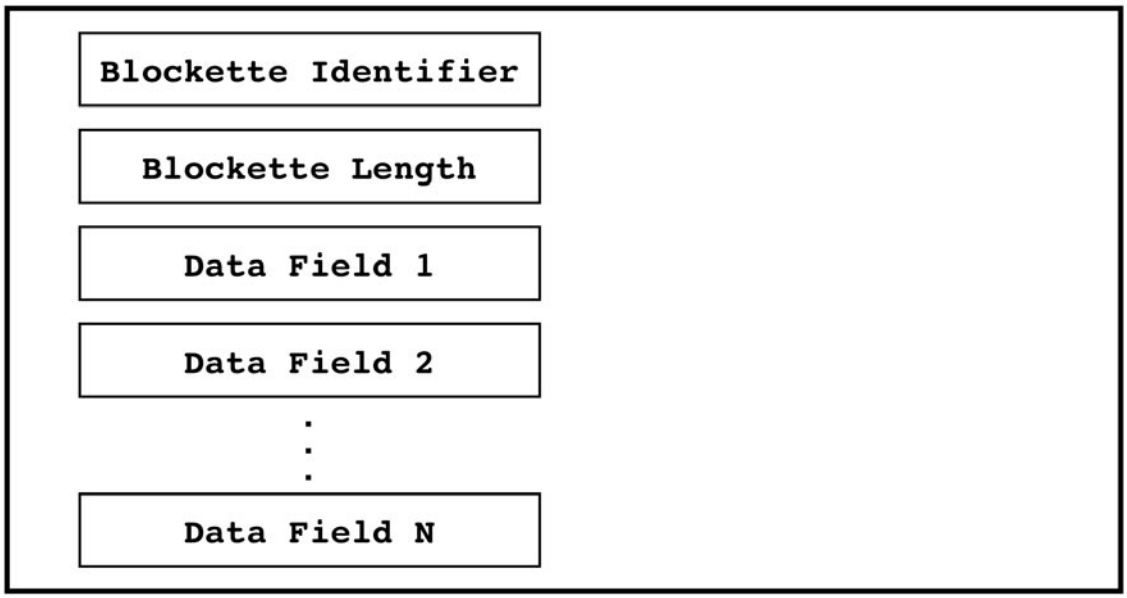


Figure 6: Organization of a Blockette

Control headers are coded entirely in printable ASCII characters, and they contain no data records. In contrast, time series objects are binary (not ASCII-formatted), and are subdivided into data records, each with a data record identification block.

Data Records

A physical record may contain one or more logical records, which in turn may contain one or more data records. The SEED structure allows conversion from one logical record size to another without having to reformat data records — provided that the data record's size is smaller than the new logical record. Each data record contains a fixed header section, a variable header section, and a data section.

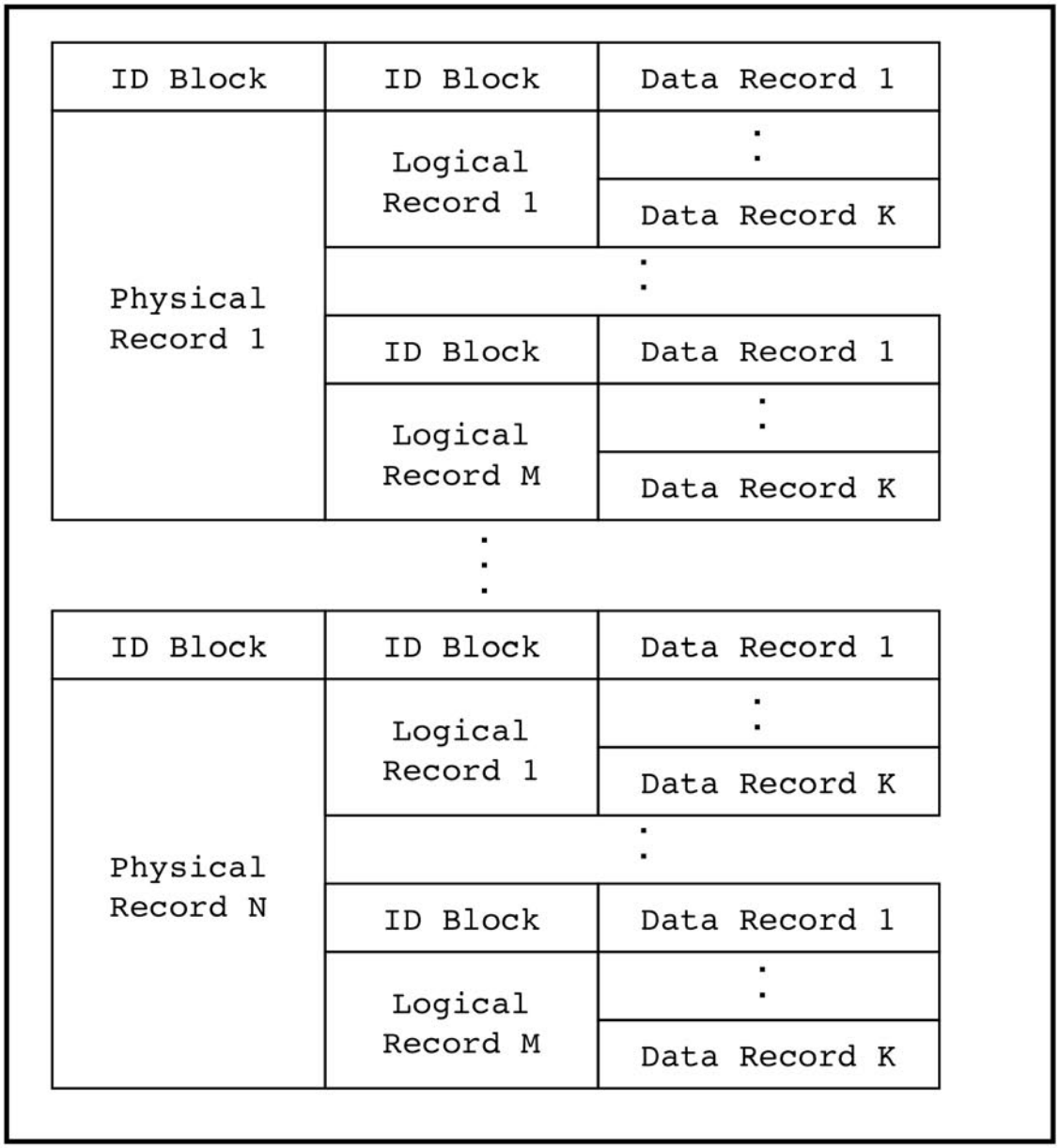


Figure 7: Record Organization Within a Time Series Format Object

The fixed header section provides the minimal self definition needed to use any portion of the data without any other auxiliary information. A sequence of unformatted blockettes, each of which is optional, make up the variable header section. These blockettes provide information about channel specific, time dependent events such as automatically determined phase arrival information or a calibration in progress. The data section contains the actual time series data.

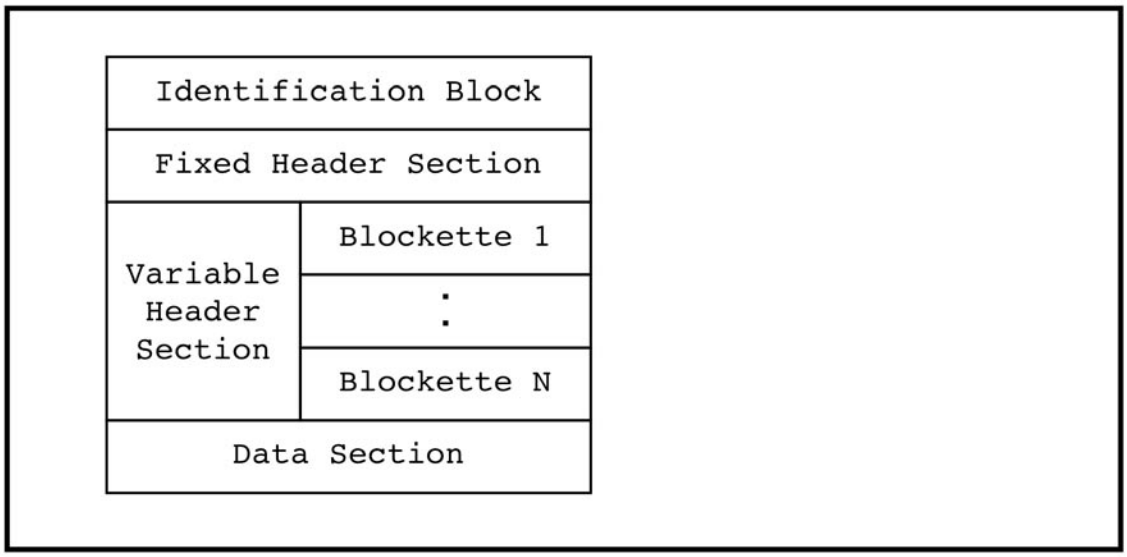


Figure 8: Organization of Data Records

How to Write SEED Data

The following two algorithms are written in Pascal-like pseudo code. They show the order in which data must be written to a SEED volume. The first algorithm describes a field station volume; the second outlines the station and event oriented network volumes. We also include a section describing electronic data transmission and telemetry volumes.

Procedure to Write Field Station Volumes

Procedure	Blockette
begin	
Start_Volume_Header_Records();	
Write_Volume_ID_Blockette 5();	[5]
Flush_Any_Remaining_Volume_Header_Records();	
Start_Abbreviation_Dictionary_Header_Records( );	
for each_data_format_type **do**	
Write_Data_Format_Dictionary_Blockette 30();	[30]
Start_Generic_Abbreviation_Header_Records ();	
for each_abbreviation **do**	
Write_Generic_Abbreviation_Blockette 33();	[33]
Start_Units_Header_Records ();	
for each_unit **do**	
Write_Unit_Blockette 34();	[34]
Flush_Any_Remaining_Dictionary_Header_Records ();	
Start_FIR_Dictionary_Records ();	
for each_dictionary **do**	
Write_Dictionary_Blockette 41();	[41]
Start_Poles & Zeros_Dictionary_Records ();	
for each_poles & zeros **do**	
Write_Response_(Poles & Zeros)_Dictionry_Blockette 43();	[43]
Start_Coefficients_Dictionary_Records ();	
for each_coefficient **do**	
Write_Coefficient_Dictionary_Blockette 44();	[44]
Start_List _Dictionary_Records ();	
for each_list **do**	
Write_Response_List_Dictionary_Blockette 45();	[45]
Start_Generic_Response_Dictionary_Records ();	
for each_generic_response **do**	
Write_Generic_Response_Dictionary_Blockette 46();	[46]
Start_Decimation_Records();	
for each_decimation **do**	
Write_Decimation_Dictionary_Blockette 47();	[47]
Start_Channel_Sensitivity/Gain_Dictionary_Records();	
for each_channel_sensitivity **do**	
Write_Channel_Sensitivity/Gain_Dictionary_Blockette 48();	[48]
for each_station **do begin**	
Start_Station_Header_Records ();	
Write_Station_ID_Blockette 50();	[50]
for each_station_comment **do**	

```
                Write_Station_Comment_Blockette 51();                    [51]
            for each_channel do begin
                Write_Channel_ID_Blockette 52();                         [52]
            for each_stage do begin
                if poles_and_zeros then
                    Write_Response_Blockette 53();                       [53]
                    or Write_Response_Blockette 60();                    [60]
                if coefficients then
                    Write_Response_Blockette 54();                       [54]
                    or Write_Response_Blockette 61();                    [61]
                    or Write_Response_Blockette 60();                    [60]
                if decimation then
                    Write_Decimation_Blockette 57();                     [57]
                    or Write_Response_Blockette 60();                    [60]
                if gain then
                    Write_Gain_Blockette 58();                           [58]
                    or Write_Response_Blockette 60();                    [60]
            end
            if final_sensitivity then
                Write_Sensitivity_Blockette 58();                        [58]
                or Write_Response_Blockette 60();                        [60]
            for each_channel_comment do
                Write_Channel_Comment_Blockette 59();                    [59]
        end
        Flush_Any_Remaining_Station_Header_Records ();
end

Start
    until end_of_media or operator_abort do begin
        if station_change or channel_change do begin
            Start_Station_Header_Records();
            Write_Station_ID_Blockette 50();                             [50]
            for each_changed_channel do begin
                Write_Channel_ID_Blockette 52();                         [52]
                for each_stage do begin
                    if poles_and_zeros_changed then
                        Write_Response_Blockette 53();                   [53]
                        or Write_Response_Blockette 60();                [60]
                    if coefficients_changed then
                        Write_Response_Blockette 54();                   [54]
                        or Write_Response_Blockette 60();                [60]
                    if decimation then
                        Write_Decimation_Blockette 57();                 [57]
                        or Write_Response_Blockette 60();                [60]
                    if individual_Gain_changed then
                        Write_Gain_Blockette 58();                       [58]
                        or Write_Response_Blockette 60();                [60]
                end
```

```
            if FIR_Response_changed then
                Write_FIR Response_Blockette 61();                    [61]
                or Write_Response_Blockette 60();                     [60]
        end
            Flush_Any_Remaining_Station_Header_Records();
        end

        if data_record_ready_to_write  do
            Write_Data_Records();
    end
end
```

Procedure to Write Station and Event Oriented Network Volumes

Procedure	Blockette
Begin	
Start_Volume_Header_Records();	
Write_Volume_ID_Blockette 10();	[10]
Write_Station_Index_Blockette 11();	[11]
Write_Time_Span_Index_Blockette 12();	[12]
Flush_Any_Remaining_Volume_Header_Records();	
Start_Abbreviation_Dictionary_Header_Records();	
for each_data_format_type **do**	
Write_Data_Format_Dictionary_Blockette 30();	[30]
Start_Comment_Dictionary_Header_Records();	
for each_comment_type **do**	
Write_Comment_Dictionary_Blockette 31();	[31]
if event_volume **then begin**	
Start_Cited_Source_Dictionary_Header_Records();	
for each_cited_source **do**	
Write_Cited_Source_Dictionary_Blockette 32();	[32]
end	
Start_Generic_Abbreviation_Header_Records();	
for each_abbreviation **do**	
Write_Generic_Abbreviation_Blockette 33();	[33]
Start_Units_Header_Records();	
for each_unit **do**	
Write_Unit_Blockette 34();	[34]
Flush_Any_Remaining_Dictionary_Header_Records();	
Start_FIR_Dictionary_Records();	
for each_dictionary **do**	
Write_Dictionary_Blockette 41();	[41]
Start_Poles & Zeros_Dictionary_Records();	
for each_poles & zeros **do**	

```
    Write_Response_(Poles & Zeros)_Dictionry_Blockette 43();                [43]

Start_Coefficients_Dictionary_Records();
for each_coefficient do
    Write_Coefficient_Dictionary_Blockette 44();                            [44]

Start_List _Dictionary_Records();
for each_list do
    Write_Response_List_Dictionary_Blockette 45();                          [45]

Start_Generic_Response_Dictionary_Records();
for each_generic_response do
    Write_Generic_Response_Dictionary_Blockette 46();                       [46]

Start_Decimation_Records();
for each_decimation do
    Write_Decimation_Dictionary_Blockette 47();                             [47]

Start_Channel_Sensitivity/Gain_Dictionary_Records();
for each_channel_sensitivity do
    Write_Channel_Sensitivity/Gain_Dictionary_Blockette 48();               [48]
for each_station do begin
    Start_Station_Header_Records();
    for orignal_and_any_updates do
        Write_Station_ID_Blockette 50();                                    [50]
    for each_station_comment do
        Write_Station_Comment_Blockette 51();                               [51]
    for each_channel do begin
        for original_channel_and_any_updates do begin
            Write_Channel_ID_Blockette 52();                                [52]
            for each_stage do begin
                if poles_and_zeros then
                    Write_Response_Blockette 53();                          [53]
                    or Write_Response_Blockette 60();                       [60]
                if coefficients then
                    Write_Response_Blockette 54();]                         [54
                    or Write_Response_Blockette 61();                       [61]
                    or Write_Response_Blockette 60();                       [60]
                if response_list then
                    Write_Response_List_Blockette 55();                     [55]
                    or Write_Response_Blockette 60();                       [60]
                if generic_response then
                    Write_Generic_Response_Blockette 56();                  [56]
                    or Write_Response_Blockette 60();                       [60]
                if decimation then
                    Write_Decimation_Blockette 57();                        [57]
                    or Write_Response_Blockette 60();                       [60]
                if individual_sensitivity then
                    Write_Sensitivity_Blockette 58();                       [58]
                    or Write_Response_Blockette 60();                       [60]
            end
            if final_sensitivity then
                Write_Sensitivity_Blockette 58();                           [58]
                or Write_Response_Blockette 60();                           [60]
        end
```

```
            for each_channel_comment do
                Write_Channel_Comment_Blockette 59();                    [59]
        end
        Flush_Any_Remaining_Station_Header_Records();
    end

    for each_time_span do begin
        Start_Time_Span_Header_Records();
        Write_Time_Span_ID_Blockette 70();                               [70]
        if event_network_volume then begin
            Write_Hypocenter_Info_Blockette 71();                        [71]
            for each_Hypocenter do
                for each_station do
                    for each_channel do
                        for each_phase do
                            Write_Event_Phases_Blockette 72();           [72]
    end
    for each_station do
        for each_channel do
            Write_Time_Series_Index_Blockette 74();                      [74]
        Flush_Remaining_Time_Span_Header_Records();
    end
    for each_station do
        for each_channel do
            for data_record_for_channel do
                Write_Data_Records();
    end
end
```

Field Station Volumes

Field station recordings use only a small portion of the SEED format: only a few brief control headers near the beginning of the volume, and no indices. Data are usually written to a field tape, and for only one station. In other cases — arrays of several stations, for example — headers written at the beginning of the volume describe all the stations and data format types. The software should write data as buffers are filled, and complete volumes as they approach the physical end of the media, or as operations personnel terminate them.

The Field Volume Identifier Blockette [5] should always appear in the Volume Index Control Header at the beginning of each volume. The Abbreviation Dictionary Control Header follows. Include a Data Format Dictionary Blockette [30] for each data format used (usually only one or two). Write Generic Abbreviation Blockettes [33] for each abbreviation used in the various station and channel blockettes, and Units Abbreviation Blockettes [34] for units used.

When writing data from multiple stations, flush out the previous logical record before starting to write the station control header, so that the station control header will start on a new record. Start the station record with the Station Identifier Blockette [50]. (Station comments may be included in Station Comment Blockettes [51].) Follow this with the information for each channel. Each channel should appear as a Channel Identifier Blockette [52], followed by the channel response. Use any of the following blockettes that apply to a particular response configuration to describe it exactly as it would appear on a network volume:

- Response (Poles & Zeros) Blockettes [53]
- Response (Coefficients) Blockettes [54]

- Decimation Blockettes [57]

- Channel Sensitivity/Gain Blockettes [58]

Write a channel blockette and a set of response blockettes for each channel. (See Appendix C for more information.) To ensure that valid and accurate station identification information is available, write it every few days, or when restarting the station processor. Write new station identification information if the station configuration changes in the middle of a volume — for example, after operator action or maintenance activity (which can also happen via remote dialup). The Volume Index Control Header may appear multiple times on the volume, each time delineating a new sub-volume.

After writing the station control header(s), start recording station data. Mix — i.e., block multiplex — channels in any way (there are no restrictions on timing or spacing between channels), but keep the data for each channel in time sequence.

Write several tape marks (at least four) to the volume when the station operator wants to terminate the volume, or when the end of the volume nears. Doing this tells the data collection center that any subsequent data does not belong with this volume, but may be from an older, recycled volume. If the reporting station "crashes" and does not successfully write EOFs (end- of-file marks) to the volume, the data collection center may have to examine the times, and possibly the station identification information, to determine the end of the current data.

When terminating a volume, flush all data buffers — prematurely, if necessary — to the volume. This will keep data (especially very long period data) from accumulating in a buffer for several days, then showing up later on subsequent volumes. While you can place single EOFs anywhere within the SEED format for any reason, multiple EOFs must only appear at the final end of the data.

The field station volume control headers are similar to those of a station oriented network volume, with these exceptions:

- some fields in the volume index control header (the volume ending date and time, and the indices to other control headers) are not known when writing the header

- the station control header information may be incomplete

- no time span control headers are present because the necessary information is not available

- small data record lengths may be used and different channels may use different data record lengths

- the data records are block multiplexed for all of the channels

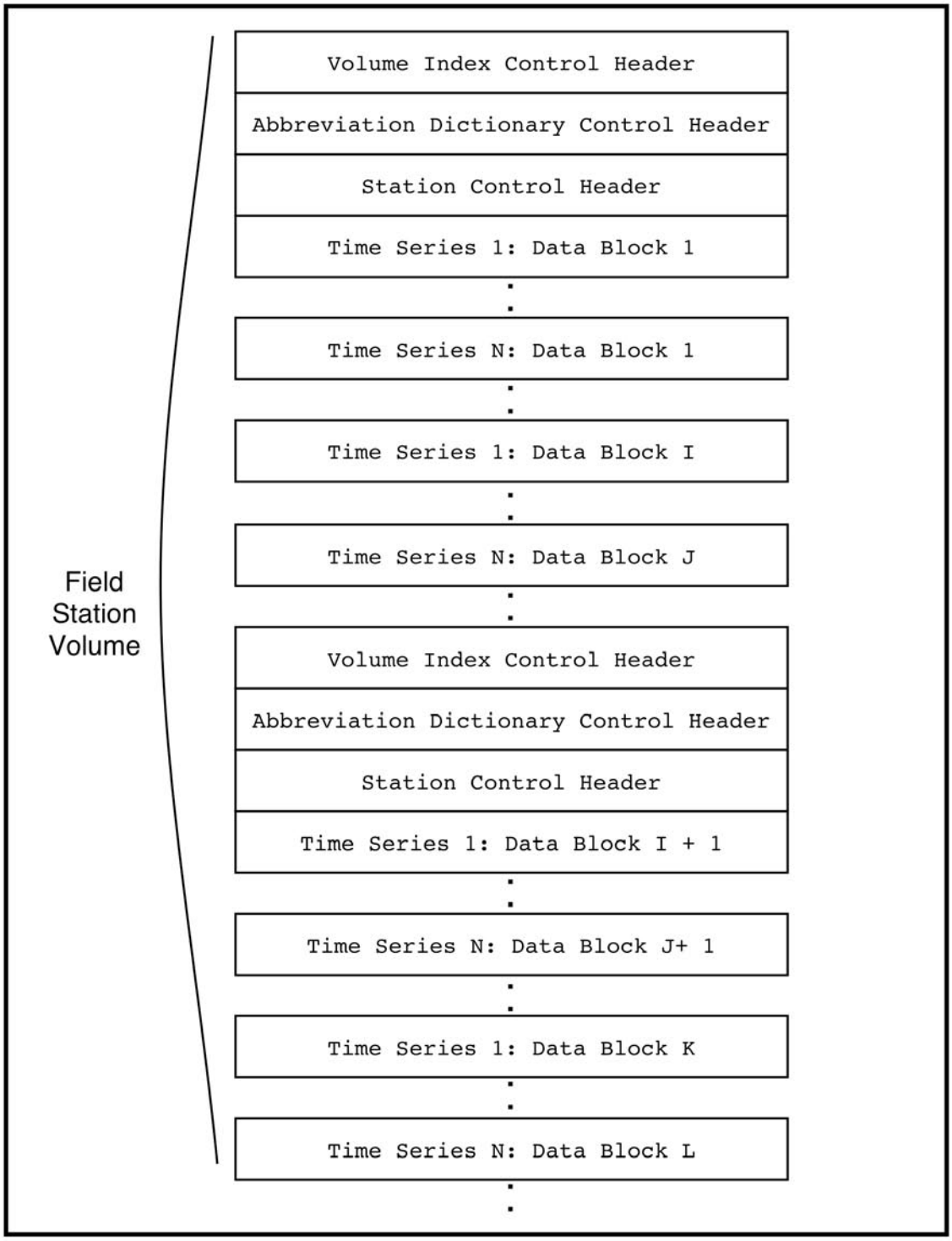


Figure 9: Organization of a Field Station Volume

If desired, control headers may be flushed periodically to form a sequence of logical sub- volumes within the field station volume (see the Glossary for definitions). For a station network volume, there is only one data record per logical record. When field station volumes are combined into a network, data records must be collected to satisfy the standard. SEED allows you to concatenate several data records into one logical record to minimize computer resources required to solve this problem.

Merging Field Station Volumes into Network Volumes

Field station volumes differ from station network volumes. In particular, time span and indexing information are unavailable to the station processor, as may some station channel information. Logical records may be different sizes for different channels, but only one size in a given volume for each channel. Data records for different channels will be block multiplexed.

Merge station volumes into network volumes by:

- adding missing auxiliary information
- block de-multiplexing
- concatenating data records to make logical records of the required fixed size
- calculating time span information
- calculating indexing information
- creating time span control headers

If the missing auxiliary information is available on-line at the data collection center, complete this entire process automatically. Just compile a table of data record sizes and locations for each station channel while transcribing the station volume from the field recording media onto temporary disk storage. Then, in a second pass through the data, reformat the station volume as a network volume. With enough available random access storage, you can apply this process to a number of station volumes simultaneously, and produce a merged network volume. Although this procedure requires substantial amounts of random access storage (which is generally quite inexpensive), it minimizes processor and input/output time.

Also, remember that station oriented network volumes use time span control headers differently than **do** event oriented network volumes: for station oriented volumes there will be one time span until the sub-volume ends, or the control header changes; for event oriented volumes, there will be one time span per event (unless the events overlap).

Telemetry Volumes and Electronic Data Transmission

A special volume format — called the telemetry volume format — lets a data transmitter assume that the data receiver has the most up-to-date control header information, unless otherwise requested. This means that, in many cases, only the data and minimal control header information need to be transmitted; if the receiver needs more control headers, the transmitter can send them, too. This procedure is not mandatory, but it can significantly reduce overhead on the available communications bandwidth. The pseudo-code and complete protocol for telemetry volumes have not yet been designed.

Electronically transmitting digital seismic data is becoming increasingly important as the need for near-real-time seismic monitoring grows. Electronic computer links can emulate sequential storage media to provide transparent physical blocking of data as well as error detection, correction, and retransmission. Such links are exceptional only in their relatively low data rate and, in some cases, support for ASCII data only. The SEED data record format, as defined for storage media, is already compact, minimally self defining, and reasonably robust. Also, SEED provides a special Telemetry Blockette [8], allowing data transmission of only the newest data— as long as the receiver does not specifically request a retransmission of previously transmitted control header information. Therefore, we recommend

Chapter 3

SEED Conventions

ASCII Header Field Conventions

SEED uses four types of control headers: volume identifier headers, abbreviation dictionary headers, station headers, and time span headers. (SEED also uses data record headers; these are described later in this manual.) Each header can use several blockettes — individual "portions" of information that are header-specific, and that conform to the organization rules of their volume type. Some blockettes vary in length, and can be longer than the logical record length.

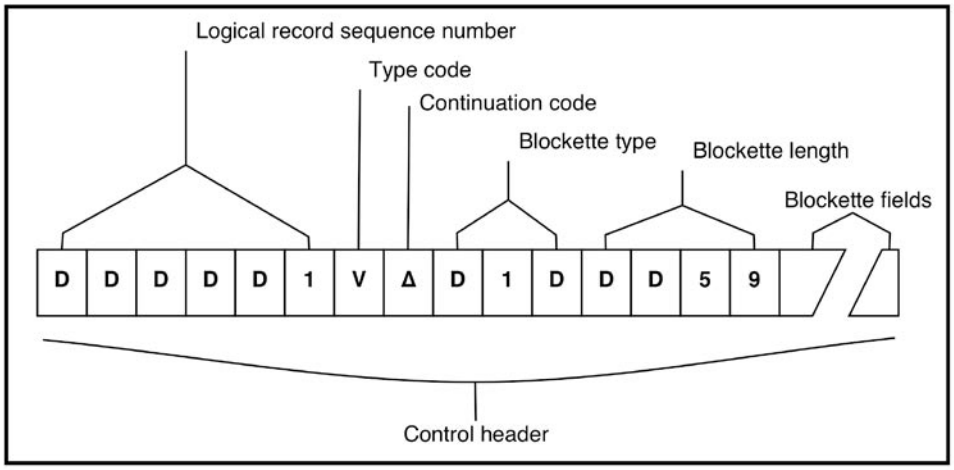


Figure 10: Beginning Fields of a Control Header

Unlike data records with primarily binary data fields, data fields in control headers are not stored as binary values, but are formatted in ASCII.

We provide four categories of information for the fields within each control header blockette listed in this reference manual:

- field name
- field type
- field length
- masks and flags

A field's contents are usually described by a field name — for example, "Station call letters"

is the third field of the Station Identifier Blockette [50].

A field's type describes how the field formats its data:

A — Alphanumeric ASCII string (fixed length)

D — Decimal integer or fixed point decimal number

F — Floating point number with an exponent

V — Variable length ASCII string, ending with a tilde: ~ (ASCII 126)

The length equals the exact number of characters in the field. Variable field length is described as a range (a—b), where a is the minimum number of characters, and b is the maximum. Some variable length fields have no fixed maximum length. Character counts for variable length fields do not include the tilde terminator. The next field starts immediately after the end of the current field, or after the tilde for variable length fields. Always use the tilde to terminate variable length fields, even if they are zero length.

Masks show how to place data in the space provided. Most computer languages provide some method for creating these fields. Here are some examples of acceptable data for each mask, where the "Δ" denotes a single space (ASCII 32):

Mask	Data Type	Example
"####"	Unsigned integer	"0023" or "ΔΔ23"
"-####"	Signed integer	"00023" or "Δ0023" or "ΔΔΔ23" or "+0023" or "-0023" or "-ΔΔ23" or "ΔΔ-23" or "ΔΔ+23"
"####.####"	Unsigned fixed point	"0003.1416" or "ΔΔΔ3.1416" or "ΔΔ23.0000" or "ΔΔΔΔ.0200"
"-###.####"	Signed fixed point	"-003.1416" or "ΔΔ-3.1416" or "ΔΔ23.0000" or "-ΔΔΔ.0200"

Leading spaces or leading zeros are allowed before the number, to the left of the decimal point. All unused places to the right of the decimal point must contain zeros. Signs can be minus or plus, and can float to the beginning of the first digit. A zero or space can fill the sign position. No sign specified implies a positive number.

The floating point mask behaves as described above, except that it contains the "E" exponential notation, and has another sign for the exponent.

Mask	Data Type	Example
"-#.####E-##"	Signed Exponential	"Δ3.1416E000" or "Δ3.1416EΔ00" or "03.1416e+00" or "-1.0000e-02"

Use a special mask, shown as TIME, for the ASCII date and time. The time field works like a variable length field, described above. Truncate the time at the most significant valid time; leave off unneeded or unavailable time precision. A few situations use an optional time field; in such cases the field appears empty, with just the tilde terminator. Arrange the data inside as "YYYY,DDD,HH:MM:SS.FFFF" and use these subfields:

Mask subfield	What it means
YYYY	The year with the century (e.g., 1987)
DDD	The julian day of the year (January 1 is 001)
HH	The hour of the day UTC (00—23)
MM	The minute of the day (00—59)
SS	The seconds (00—60; use 60 only to note leap seconds)
FFFF	The fraction of a second (to .0001 seconds resolution)

All positions of a time field must be zero padded to the left, but they do not need padding to the right if the time will be truncated. “1987,023,04:23:05.1” and “1987,023” are correct.

Flags determine what ASCII characters can be placed in an alphanumeric or a variable length field:

Flag value	Permitted characters
U	Upper case A—Z
L	Lower case a—z
N	Digits 0—9
P	Any punctuation characters (including “_”)
S	Spaces between words
_	Underline symbol

Variable length fields cannot have leading or trailing spaces. Leave fixed length alphanumeric fields left justified (no leading spaces), and pad them with spaces (after the field’s contents).

How to Assemble Control Headers

When assembling a header, first write an identifier block — the incrementing sequence number in the record, then the record type code, followed by the continuation code — for a total of eight bytes. (New control headers use blank (Δ, or ASCII 32) continuation codes; continuing records use an asterisk (*, or ASCII 42).)

Field name	Type	Length	Mask or Flags
Sequence number (first record is 1)	D	6	“######”
Control header type code V — Volume header A — Dictionary header S — Station header T — Time Span header	A	1	
Record continuation code * — if continued from last record Δ — if not continued	A	1	

Next, write out the blockettes needed in the control header (you can write several blockettes, one after another, all under one identifier block). For each blockette, write the blockette type in the record and then the total blockette length — including the seven bytes for the blockette type and length. Finally, write the entire blockette (or as much of it as will fit in the logical record).

Field name	Type	Length	Mask or Flags
Blockette type	D	3	“###”
Blockette length	D	4	“####”
Blockette data	(see subsequent chapters)		

If the blockette fits, then start the next blockette at the byte immediately following the end of the last. If the blockette does not fit, assemble a new record, increment the sequence number, and set the continuation code to an asterisk. The record then resumes on the byte after the continuation code. If you must write a record when it is not full (to begin a record of another type, for example), fill the remainder of the record after the last blockette with spaces. Then flush (write out)

the record. If there are less than seven bytes remaining, the record *must* be flushed. Never split a blockette's "length/ blockette type" section across records.

How Binary Data Fields are Described in This Manual

Throughout this manual, we use some conventions to describe the sizes of fields in the SEED format. Here are the binary data types used in fixed headers and in data blockettes:

Field type	Number of bits	Field description
UBYTE	8	Unsigned quantity
IBYTE	8	Two's complement signed quantity
UWORD	16	Unsigned quantity
WORD	16	Two's complement signed quantity
ULONG	32	Unsigned quantity
LONG	32	Two's complement signed quantity
CHAR * n	n * 8	n characters, each 8 bits and each with a 7-bit ASCII character (high bit always 0)
FLOAT	32	IEEE Floating point number

The IEEE floating point format consists of three stored components: a sign (+ or -), an exponent, and a fraction. In the following description of the storage format these notations will be used.

s = sign e = biased exponent f = fraction

The sign is the sign of the fraction. Rather than storing the sign of the exponent a bias is added to the exponent, and the biased exponent is stored.

	s	e	f
bit positions	31	30:23	22:0

IEEE single precision values occupy one 32 bit word as shown above in 68000 byte order. Bits 0:22 store the 23 bit fraction, bits 23:30 store the 8 bit exponent, and the high order bit 31 stores the sign bit. The 23 bit fraction combined with the implicit leading bit provide 24 bits of precision in normalized numbers. The value of an IEEE single precision floating point number is calculated as

$$-1^{s} \times 2^{(e-127)} \times 1.f$$

The byte order of a FLOAT is specified in the station identifier blockette [50].

Binary data types are used in the BTIME structure:

Field type	Number of bits	Field description
UWORD	16	Year (e.g., 1987)
UWORD	16	Day of Year (Jan 1 is 1)
UBYTE	8	Hours of day (0—23)
UBYTE	8	Minutes of day (0—59)
UBYTE	8	Seconds of day (0—59, 60 for leap seconds)
UBYTE	8	Unused for data (required for alignment)
UWORD	16	.0001 seconds (0—9999)

NOTE: The BTIME structure differs from the ASCII variable length TIME structure used in the control headers.

All binary 32 bit words begin on long-word boundaries, 16 bit words begin on word boundaries, and all bytes on byte boundaries. The fixed portion of the header always ends at the end of a long-word boundary, and each blockette is an integer number of long-words in length. Pack data in either VAX or 68000 swapping order. This swapping key is in the Station Identifier Blockette [50], and may differ for each station. Decoding programs will automatically adapt to specified swapping orders. Negative numbers utilize standard two's complement representation. The data description language in the data dictionary, referred to by the Channel Identifier Blockette [52] for the represented channel, governs byte swapping within the data. Mantissas and exponents for floating point numbers are expressed as binary two's complement integers. The most significant bit of the number (bit 15, or the left most bit) is always set to zero for a positive number, and the most significant bit of the mantissa is in bit 14. For negative numbers, the most significant bit is always set to one, and the integer is in two's complement format.

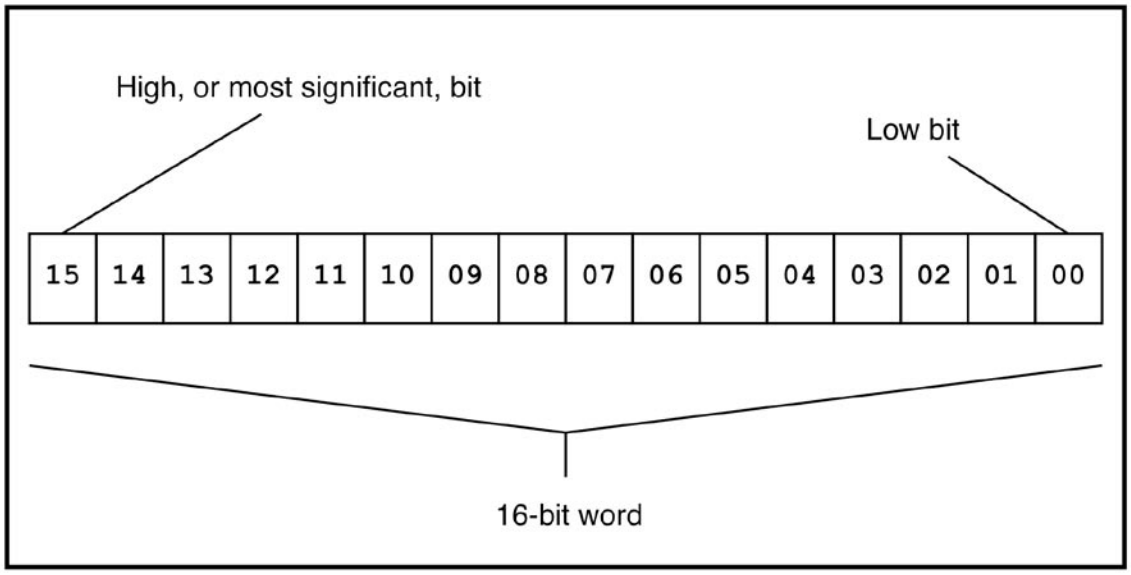


Figure 11: High and Low Bits in a 16-Bit Word shown in 68000 word order

Chapter 4

Volume Index Control Headers

Volume index control headers precede all data. Their primary purpose is to provide a directory to differentiate parts of the volume for network and event distributions. Only field station volumes use Field Volume Identifier Blockette [5].

[5] Field Volume Identifier Blockette

Name: **Field Volume Identifier Blockette**
Blockette Type: **005**
Control Header: **Volume Index**
Field Station Volume: **Required**
Station Oriented Network Volume: **Not Applicable**
Event Oriented Network Volume: **Not Applicable**

Field stations use the Field Volume Identifier Blockette [5], and usually produce only one volume. They should include this blockette once at the beginning of each logical volume or sub- volume.

Note	Field name	Type	Length	Mask or Flags
1	Blockette type 005	D	3	"###"
2	Length of blockette	D	4	"####"
3	Version of format	D	4	"##.#"
4	Logical record length	D	2	"##"
5	Beginning of volume	V	1—22	TIME

Notes for fields:

1 Standard blockette type identification number.

2 Length of the entire blockette, including the 7 bytes in fields 1 and 2.

3 Version number of the format, currently "V2.4."

4 Volume logical record length expressed as a power of 2. A 4096 byte record would be 12. Logical record lengths can be from 256 bytes to 32,768 bytes. 4096 bytes is preferred.

5 Nominal starting time of the volume. Since all data are normally flushed at tape termination time, all data should start at nearly the same time. Record that time here. If, however, the data are not flushed, the time here should be the time of the earliest — not necessarily the first — record of the logical volume. The calculations to arrive at this time do not have to include the longer period data, as that data may have been buffered for a long time prior to its recording on the current volume.

[8] Telemetry Volume Identifier Blockette

Name: **Telemetry Volume Identifier Blockette**
Blockette Type: **008**
Control Header: **Volume Index**
Field Station Volume: **Optional**
Station Oriented Network Volume: **Optional**
Event Oriented Network Volume: **Optional**

Field stations or networks can use the Telemetry Volume Identifier Blockette [8] when electronically transmitting SEED data. Without this blockette, header records can take up an unnecessarily large amount of the transmission — especially if the receiver already has them from a prior transmission. We suggest using the Telemetry Volume Identifier Blockette [8] as follows: 1) The transmitter should send this blockette with information on the effective start and end times of the header information associated with the data to be transmitted. 2) The transmitter should follow this with the data. 3) The receiver should then respond as to whether or not it needs additional header information (dictionaries, station information, or channel and response information) for the received data. 4) If the receiver needs those headers, the transmitter can send them. (NOTE: This blockette had not yet been used at the time this manual went to press.)

Note	Field name	Type	Length	Mask or Flags
1	Blockette type 008	D	3	"###"
2	Length of blockette	D	4	"####"
3	Version of format	D	4	"##.#"
4	Logical record length	D	2	"##"
5	Station identifier	A	5	[UN]
6	Location identifier	A	2	[UN]
7	Channel identifier	A	3	[UN]
8	Beginning of volume	V	1—22	TIME
9	End of volume	V	1—22	TIME
10	Station information effective date	V	1—22	TIME
11	Channel information effective date	V	1—22	TIME
12	Network Code	A	2	[ULN]

Notes for fields:

1 Standard blockette type identification number.

2 Length of the entire blockette, including the 7 bytes in fields 1 and 2.

3 Version number of the format, currently "V2.4."

4 Volume logical record length expressed as a power of 2. A 4096 byte record would be 12.

5 This component's station name.

6 This component's location code. (This is the array subcode to the station.)

7 Standard channel identifier (see Appendix A).

8 Nominal starting time of the transmitted volume.

9 Ending time of the transmitted volume.

10 Time of associated station header information.

11 Time of associated channel information.

[10] Volume Identifier Blockette

Name:	**Volume Identifier Blockette**
Blockette Type:	**010**
Control Header:	**Volume Index**
Field Station Volume:	**Not Applicable**
Station Oriented Network Volume:	**Required**
Event Oriented Network Volume:	**Required**

This is the normal header blockette for station or event oriented network volumes. Include it once at the beginning of each logical volume or sub-volume.

Sample:

```
010009502.1121992,001,00:00:00.0000~1992,002,00:00:00.0
000~1993,029~IRIS_DMC~Data for 1992,001~
```

	Note	Field name	Type	Length	Mask or Flags
	1	Blockette type 010	D	3	“###”
	2	Length of blockette	D	4	“####”
	3	Version of format	D	4	“##.#”
	4	Logical record length	D	2	“##”
	5	Beginning time	V	1—22	TIME
	6	End time	V	1—22	TIME
V2.3 -	7	Volume Time	V	1—22	TIME
V2.3 -	8	Originating Organization	V	1—80	
V2.3 -	9	Label	V	1-- 80	

Notes for fields:

1 Standard blockette type identification number.

2 Length of the entire blockette, including the 7 bytes in fields 1 and 2.

3 Version number of the format, currently “V2.4.”

4 Volume logical record length, expressed as a power of 2. A 4096 byte logical record would have “12” in this field. Logical record lengths can be from 256 bytes to 32,768 bytes. 4096 bytes is preferred.

5 The earliest time seen in the time span list for this logical volume.

6 The latest time on the logical volume.

7 The actual date and time that the volume was written.

8 The organization writing the SEED volume.

9 An optional label that can be used to identify this SEED volume. For instance a label such as “Loma Prieta Earthquake” could be designated. If there is no label a ~ must be inserted.

[11] Volume Station Header Index Blockette

Name:	**Volume Station Header Index Blockette**
Blockette Type:	**011**
Control Header:	**Volume**
Index Field Station Volume:	**Not Applicable**
Station Oriented Network Volume:	**Required**
Event Oriented Network Volume:	**Required**

This is the index to the Station Identifier Blockettes [50] that appear later in the volume. Do not use this blockette for station volumes; station volumes contain no indices. This blockette refers to each station described in the station header section.

Sample:

```
V0110054004AAK__000003ANMO_000007ANTO_000010BJI__000012
```

Note	Field name	Type	Length	Mask or Flags
1	Blockette type 011	D	3	"###"
2	Length of blockette	D	4	"####"
3	Number of stations	D	3	"###"
	REPEAT fields 4 — 5 for the Number of stations:			
4	Station identifier code	A	5	
5	Sequence number of station header	D	6	"######"

Notes for fields:

1 Standard blockette type identification number.

2 Length of the entire blockette, including the 7 bytes in fields 1 and 2. It is possible (but not very likely) that this blockette could exceed 9999 bytes. In this case, the writer should cease writing stations into the index before it exceeds 9999 bytes; close the blockette; write it out; and continue with a new blockette [11]. The count in field 3 should reflect the count in each blockette. The byte count in field 2 should represent the size of the individual blockettes as they are written.

3 The number of stations that will be represented later by Station identifier Blockettes [50] in the station header section. The next two fields each repeat, once per station, for the total number of stations.

4 The official station code assignment of the recording station, as assigned by the NEIC.

5 The sequence number of the logical record on the current logical volume that contains the Station Identifier Blockette [50] for the station named in field 4. Since records must be flushed when the stations are written, this indexing value will always be unique and will never refer to more than one station. Note that station identifier blockette update records are not included here, as they combine with a primary blockette to form the station header.

[12] Volume Time Span Index Blockette

Name:	**Volume Time Span Index Blockette**
Blockette Type:	**012**
Control Header:	**Volume Index**
Field Station Volume:	**Not Applicable**
Station Oriented Network Volume:	**Required**
Event Oriented Network Volume:	**Required**

This blockette forms an index to the time spans that encompass the actual data. One index entry exists for each time span recorded later in the volume. Time spans are not used for field station type volumes. There should be one entry in this index for each time span control header. (For more information, see the notes for blockettes [70], [73], and [74].)

Sample:

```
012006300011992,001,00:00:00.0000~1992,002,00:00:00.0000~000014
```

Note	Field name	Type	Length	Mask or Flags
1	Blockette type 012	D	3	"###"
2	Length of blockette	D	4	"####"
3	Number of spans in table	D	4	"####"
	REPEAT fields 4 — 6 for the Number of spans in table:			
4	Beginning of span	V	1—22	TIME
5	End of span	V	1—22	TIME
6	Sequence number of time span header	D	6	"######"

Notes for fields:

1 Standard blockette type identification number.

2 Length of the entire blockette, including the 7 bytes in fields 1 and 2. It is possible (but not very likely) that this blockette could exceed 9999 bytes. In this case, the writer should cease writing time spans to the index before it exceeds 9999 bytes; close the blockette; write it out; and continue a new blockette. The count in field 3 should reflect the count in each blockette. The byte count in field 2 should represent the size of the individual blockettes as they are written.

3 The number of time spans present in this blockette. The next three fields each repeat, once per time span, for the total number of time spans.

4 The beginning time of the time span. This should be the same time as is in the Time Span Identifier Blockette [70] to which it refers.

5 The time span ending time.

6 The sequence number of the record on which the Time Span Identifier Blockette [70] referred to starts.

Chapter 5

Abbreviation Dictionary Control Headers

Dictionary records let you use abbreviations to refer to lengthy descriptions without having to create any external tables. Blockettes [43] through [48] help reduce the amount of space used to specify intricate channel responses; they are almost identical to blockettes [53] through [58], but differ in that they are set up for use as response dictionary entries. Use them with the Response Reference Blockette [60].

[30] Data Format Dictionary Blockette

Name: **Data Format Dictionary Blockette**
Blockette Type: **030**
Control Header: **Abbreviation Dictionaries**
Field Station Volume: **Required**
Station Oriented Network Volume: **Required**
Event Oriented Network Volume: **Required**

All volumes, no matter what type, must have a Data Format Dictionary Blockette [30]. Each Channel Identifier Blockette [52] has a reference (field 16) back to a Data Format Dictionary Blockette [30], so that SEED reading programs will know how to decode data for the channels. Because every kind of data format requires an entry in the Data Format Dictionary Blockette [30], each recording network will have at least one, maybe more, entries.

Sample:

```
0300087CDSNΔGain-RangedΔFormat~000200104M0~W2ΔD0-13ΔA-8191~D1415~P0:#0,1:#2,2:#4,3:#7~
```

Note	Field name	Type	Length	Mask or Flags
1	Blockette type — 030	D	3	“###”
2	Length of blockette	D	4	“####”
3	Short descriptive name	V	1—50	[UNLPS]
4	Data format identifier code	D	4	“####”
5	Data family type	D	3	“###”
6	Number of decoder keys	D	2	“##”
	REPEAT field 7 for the Number of decoder keys:			
7	Decoder keys	V	any	[UNLPS]

Notes for fields:

1 Standard blockette type identification number.

2 Length of the entire blockette, including the 7 bytes of fields 1 and 2.

3 A short name describing the data type. See Appendix D.

4 A cross reference number, used in later blockettes, to indicate this particular dictionary entry. The writer program for each volume creates this code, and it pertains to one particular volume only. The code is never guaranteed to have meaning outside of that volume, and it may be different for any two volumes. Writers usually assign 1 for the first code, then increment for each additional code. Because a unique code is assigned for each data type, it will be unique in each blockette.

5 A field used by the data decoder to describe the data family type. This field tells a potential decoder program what general algorithm to use to decode the associated data. Each algorithm requires some number of decoder keys that contain special additional information, enabling the algorithm to decode the data. See Appendix D for more information about decoders and some examples of their use. As of this manual’s printing, the currently defined family types are:

- 0 Integer format fixed interval data
- 1 Gain ranged fixed interval data
- 50 Integer differences compression
- 80 ASCII text with line control (for console logs)
- 81 Non-ASCII text (for other language character sets)
- 90 Opaque data
- 91 Blockette-only information

6 The number of decoder keys used by the data family type (see Appendix D).

7 The decoder keys used by the data family type. Place a tilde after each key in the sequence to separate them. See Appendix D.

[31] Comment Description Blockette

Name: Comment Description Blockette
Blockette Type: 031
Control Header: Abbreviation Dictionaries
Field Station Volume: Optional
Station Oriented Network Volume: Required if referred to
Event Oriented Network Volume: Required if referred to

Station operators, data collection centers, and data management centers can add descriptive comments to data to indicate problems encountered or special situations.

Sample:

```
03100720750StimeΔcorrectionΔdoesΔnotΔincludeΔleapΔsecond,Δ(-1000ms).~000
```

Note	Field name	Type	Length	Mask or Flags
1	Blockette type — 031	D	3	“###”
2	Length of blockette	D	4	“####”
3	Comment code key	D	4	“####”
4	Comment class code	A	1	[U]
5	Description of comment	V	1—70	[UNLPS]
6	Units of comment level	D	3	“###”

Notes for fields:

1 Standard blockette type identification number.

2 Length of the entire blockette, including the 7 bytes in fields 1 and 2.

3 Code key used to uniquely identify comments. The code keys are assigned by any user’s convention, and may not be consistent between volumes. (We hope that these codes might be standardized, so that reader programs might automatically determine data quality from them.) See Appendix E for a sample set of comment codes. Field 5 of Station Comment Blockette [51] and field 5 of Channel Comment Blockette [59] refer to this code key.

4 A single letter code, assigned by the user, which determines to what the code refers.

5 The comment’s text. A brief sentence should describe the condition. Use upper and lower case alphanumeric characters, with punctuation. (Comments may optionally contain a numeric value to denote magnitude, frequency, or some other value, giving numeric weight to the comment. For example, such a numeric value is often used for time corrections, where it represents the number of milliseconds of the correction.)

6 If a value is associated with the comment, place the unit lookup code from field 3 of the Units Abbreviation Blockette [34] abbreviation dictionary here; otherwise this value would be zero.

[32] Cited Source Dictionary Blockette

Name: **Cited Source Dictionary Blockette**
Blockette Type: **032**
Control Header: **Abbreviation Dictionaries**
Field Station Volume: **Not Applicable**
Station Oriented Network Volume: **Optional**
Event Oriented Network Volume: **Required**

This blockette identifies the contributing institution that provides the hypocenter and magnitude information. This blockette is usually used only in event oriented network volumes.

Note	Field name	Type	Length	Mask or Flags
1	Blockette type — 032	D	3	"###"
2	Length of blockette	D	4	"####"
3	Source lookup code	D	2	"##"
4	Name of publication/author	V	1—70	[UNLPS]
5	Date published/catalog	V	1—70	[UNLPS]
6	Publisher name	V	1—50	[UNLPS]

Notes for fields:

1 Standard blockette type identification number.

2 Length of the entire blockette, including the 7 bytes of fields 1 and 2.

3 A cross reference number, used in subsequent blockettes to indicate this particular dictionary entry. The writer program for each volume creates this number, and it pertains only to that particular volume. This number is never guaranteed to have meaning outside of the volume, and it may be different for any given two volumes. Writing programs usually assign 1 for the first code, and increment it for each subsequent code. Because a unique code is assigned for each data type, it will be unique in each of these blockettes.

4 A standard name for the publication from which the epicenter/ hypocenter information was obtained.

5 Date published and catalog information for this citation from the publication.

6 The name of the publisher.

[33] Generic Abbreviation Blockette

Name: **Generic Abbreviation Blockette**
Blockette Type: **033**
Control Header: **Abbreviation Dictionaries**
Field Station Volume: **Required**
Station Oriented Network Volume: **Required**
Event Oriented Network Volume: **Required**

Sample:

```
0330055001(GSN)ΔGlobalΔSeismographΔNetworkΔ(IRIS/USGS)~
```

Note	Field name	Type	Length	Mask or Flags
1	Blockette type — 033	D	3	"###"
2	Length of blockette	D	4	"####"
3	Abbreviation lookup code	D	3	"###"
4	Abbreviation description	V	1—50	[UNLPS]

Notes for fields:

1 Standard blockette type identification number.

2 Length of the entire blockette, including the 7 bytes of fields 1 and 2.

3 A cross reference code, used in later blockettes (field 10 of the Station Identifier Blockette [50], and field 6 of the Channel Identifier Blockette [52]) to indicate this particular dictionary entry (see Appendix F). The writer program for each volume creates this code, and it pertains only to that particular volume. This number is never guaranteed to have meaning outside of the volume, and it may be different for any two volumes. Writing programs usually assign 1 for the first code, and increment it for each succeeding code. Because a unique code is assigned for each data type, it will be unique in each of these blockettes.

4 The descriptive text for the abbreviation, as a brief sentence. Use upper and lower case alphanumeric text.

[34] Units Abbreviations Blockette

Name:	**Units Abbreviations Blockette**
Blockette Type:	**034**
Control Header:	**Abbreviation Dictionaries**
Field Station Volume:	**Required**
Station Oriented Network Volume:	**Required**
Event Oriented Network Volume:	**Required**

This blockette defines the units of measurement in a standard, repeatable way. Mention each unit of measurement only once.

Sample:

```
0340044001M/S~VelocityΔinΔMetersΔPerΔSecond~
```

Note	Field name	Type	Length	Mask or Flags
1	Blockette type — 034	D	3	“###”
2	Length of blockette	D	4	“####”
3	Unit lookup code	D	3	“###”
4	Unit name	V	1—20	[UNP]
5	Unit description	V	0—50	[UNLPS]

Notes for fields:

1 Standard blockette type identification number.

2 Length of the entire blockette, including the 7 bytes of fields 1 and 2.

3 A unit lookup code, used in later blockettes to indicate this particular dictionary entry. As of this manual’s publication, the following fields and blockettes refer to this code:

- field 6 of the Comment Description Dictionary Blockette [31]
- field 6 of the Response (Poles & Zeros) Dictionary Blockette [43]
- field 7 of the Response (Poles & Zeros) Dictionary Blockette [43]
- field 6 of the Response (Coefficients) Blockette [44]
- field 7 of the Response (Coefficients) Blockette [44]
- field 5 of the Response List Blockette [45]
- field 6 of the Response List Blockette [45]
- field 5 of the Generic Response Blockette [46]
- field 6 of the Generic Response Blockette [46]
- field 8 of the Channel Identifier Blockette [52]
- field 9 of the Channel Identifier Blockette [52]
- field 5 of the Response (Poles & Zeros) Blockette [53]
- field 6 of the Response (Poles & Zeros) Blockette [53]
- field 5 of the Response (Coefficients) Blockette [54]
- field 6 of the Response (Coefficients) Blockette [54]
- field 4 of the Response List Blockette [55]
- field 5 of the Response List Blockette [55]

- field 4 of the Generic Response Blockette [56]
- field 5 of the Generic Response Blockette [56]

The writing program for each volume creates this code, and it pertains only to that particular volume. This number is never guaranteed to have meaning outside of the volume, and it may be different for any two volumes. Writing programs usually assign 1 for the first code, and increment it for each succeeding code. Because a unique code is assigned for each data type, it will be unique in each of these blockettes.

4 The basic unit name, formatted as FORTRAN-like equations with all alphabetic characters in upper case. Specify exponents by the "**" format, and use parentheses sparingly — only when normal FORTRAN precedence would not be correct. Use the standard exponential notation (e.g., "1E-9" not "1*10**-9") for powers of 10. Use SI units and their standard abbreviations whenever possible; spell out and do not abbreviate non-SI units. For reasons of convertibility, abbreviations are represented in uppercase although this is not SI convention.

Units of ground motion are typically defined as:

Displacement	M
Velocity	M/S
Acceleration	M/S**2

5 A description of the unit.

[35] Beam Configuration Blockette

Name: **Beam Configuration Blockette**
Blockette Type: **035**
Control Header: **Abbreviation Dictionaries**
Field Station Volume: **Optional**
Station Oriented Network Volume: **Optional**
Event Oriented Network Volume: **Optional**

Use this blockette to describe the configuration of an instrument array for the synthetic output of a beam forming algorithm. The beam blockette [400] refers to this dictionary in the data headers of the data section. This is the only dictionary that is used directly by the data section; only the control header section uses all other dictionaries.

Note	Field name	Type	Length	Mask or Flags
1	Blockette type — 035	D	3	"###"
2	Length of blockette	D	4	"####"
3	Beam lookup code	D	3	"###"
4	Number of components	D	4	"####"
	REPEAT fields 5 — 9 for the Number of components:			
5	Station identifier	A	5	[UN]
6	Location identifier	A	2	[UN]
7	Channel identifier	A	3	[UN]
8	Sub-channel identifier	D	4	"####"
9	Component weight	D	5	"#.###"

Notes for fields:

1 Standard blockette type identification number.

2 Length of the entire blockette, including the 7 bytes of fields 1 and 2. Huge beam formings could cause the 9999 byte total limit or the 99 station limit to overflow. If this happens, write additional blockettes, each with the same beam lookup code.

3 A cross reference code, used in later blockettes (as of this manual's publication, only field 5 of the Beam Blockette [400]) to indicate this particular dictionary entry. The blockette is referred to by the beam blockette in the data section. The writing program for each volume creates this code, and it pertains only to that particular volume. This number is never guaranteed to have meaning outside of the volume, and it may be different for any two volumes. Writing programs usually assign 1 for the first code, and increment it for each succeeding code. Because a unique code is assigned for each data type, it will be unique in each of these blockettes.

4 The number of components included in the repeat section that follows.

5 This component's station name. (See Appendix G for some station names and codes.)

6 This component's location code. (This is the array subcode to the station.)

7 Standard channel identifier (see Appendix A).

8 The sub-channel identifier of the component; for use when the input channel is multiplexed.

9 The weight that was given to this component in the calculations of this beam

[41] FIR Dictionary Blockette

	Name:	**FIR Dictionary Blockette**
	Blockette Type:	**041**
	Control Header:	**Abbreviation Dictionaries**
	Field Station Volume:	**Some Response Required**
	Station Oriented Network Volume:	**Some Response Required**
	Event Oriented Network Volume:	**Some Response Required**
V2.2-	**Introduced in SEED version**	**2.2**

The FIR blockette is used to specify FIR (Finite Impulse Response) digital filter coefficients. It is an alternative to blockette [44] when specifying FIR filters. The blockette recognizes the various forms of filter symmetry and can exploit them to reduce the number of factors specified in the blockette.

Note	Field name	Type	Length	Mask or Flags
1	Blockette type — 041	D	3	"###"
2	Length of blockette	D	4	"####"
3	Response Lookup Key	D	4	"####"
4	Response Name	V	1— 25	[UN_]
5	Symmetry Code	A	1	[U]
6	Signal In Units	D	3	"###"
7	Signal Out Units	D	3	"###"
8	Number of Factors	D	4	"####"
	REPEAT field 9 for Number of Coefficients			
9	FIR Coefficient	F	14	"-#.#######E-##"

Notes for fields:

1 Standard blockette type identification number.

2 Length of the entire blockette, inclusive of the 7 bytes in fields 1 and 2. This blockette could exceed the maximum number of 9,999 characters. If so, continue on the next record. Field 4 should be set the same, but fields 5 — 7 in subsequent blockettes should be ignored.

3 The numeric key that is used by blockette [60], field 6, for referring to the response dictionaries. These numbers are arbitrary, and are assigned only in context to a given volume. Zero is not a legal key.

4 A descriptive name for the response.

5 The symmetry code. Designates how the factors will be specified. See the tables that follow to see examples of these different types of symmetry.

A — No Symmetry - all Coefficients are specified.

Example:	Coeff	Factor	Value
	1	1	-1.1396359E+02
	2	2	6.5405190E+01
	3	3	2.9333237E+02
	4	4	6.8279054E+02
	5	5	1.1961222E+03
	6	6	1.8402642E+03
	7	7	2.6360273E+03

B — Odd number Coefficients with symmetry.

Example:	Coeff	Factor	Value
	1 & 25	1	-1.1396359E+02
	2 & 24	2	6.5405190E+01
	3 & 23	3	2.9333237E+02
	4 & 22	4	6.8279054E+02
	5 & 21	5	1.1961222E+03
	6 & 20	6	1.8402642E+03
	7 & 19	7	2.6360273E+03
	8 & 18	8	3.4843128E+03
	9 & 17	9	4.8191733E+03
	10& 16	10	5.4920540E+03
	11& 15	11	6.0588989E+03
	12& 14	12	6.3135828E+03
	13	13	2.3400203E+02

C — Even number Coefficients with symmetry.

Example:	Coeff	Factor	Value
	1 & 24	1	-1.1396359E+02
	2 & 23	2	6.5405190E+01
	3 & 22	3	2.9333237E+02
	4 & 21	4	6.8279054E+02
	5 & 20	5	1.1961222E+03
	6 & 19	6	1.8402642E+03
	7 & 18	7	2.6360273E+03
	8 & 17	8	3.4843128E+03
	9 & 16	9	4.8191733E+03
	10& 15	10	5.4920540E+03
	11& 14	11	6.0588989E+03
	12& 13	12	6.3135828E+03

6 A Unit Lookup Key that refers to the Units Abbreviation Blockette [34], field 3, for the units for the incoming signal to this stage of the filter. It will usually be ground motion, volts, or counts, depending on where in the filter system it is.

7 Like field 6, but for the stages output signal. Analog filters usually output volts, digital filters output counts.

8 The number of factors that follow.

A No Symmetry — All Coefficients specified

$f = c$. "f" denotes number of factors, "c" is number of coefficients

B Odd — First half of all coefficients and center coefficient specified

$$f = \frac{c+1}{2}$$

C Even — First half of all coefficients specified

$$f = \frac{c}{2}$$

9 FIR Filter Coefficients.

[42] Response (Polynomial) Dictionary Blockette

Name: **Response (Polynomial) Dictionary Blockette**
Blockette Type: **042**
Control Header: **Abbreviation Dictionaries**
Field Station Volume: **Some Response Required**
Station Oriented Network Volume: **Some Response Required**
Event Oriented Network Volume: **Some Response Required**
Introduced in SEED version **2.3**

Use this blockette to characterize the response of a non-linear sensor.

Note	Field name	Type	Length	Mask or Flags
1	Blockette type — 042	D	3	"###"
2	Length of blockette	D	4	"####"
3	Response Lookup Key	D	4	"####"
4	Response Name	V	1-25	"[UN_]"
5	Transfer Function Type	A	1	[U]
6	Stage Signal Input Units	D	3	"###"
7	Stage Signal Output Units	D	3	"###"
8	Polynomial Approximation Type	A	1	[U]
9	Valid Frequency Units	A	1	[U]
10	Lower Valid Frequency Bound	F	12	"-#.#####E-##"
11	Upper Valid Frequency Bound	F	12	"-#.#####E-##"
12	Lower Bound of Approximation	F	12	"-#.#####E-##"
13	Upper Bound of Approximation	F	12	"-#.#####E-##"
14	Maximum Absolute Error	F	12	"-#.#####E-##"
15	Number of Polynomial Coefficients	D	3	"###"
	(Repeat fields 16 and 17 for each polynomial coefficient)			
16	Polynomial Coefficient	F	12	"-#.#####E-##"
17	Polynomial Coefficient Error	F	12	"-#.#####E-##"

Notes for Fields:

1 Standard blockette type identification number.

2 Length of entire blockette, including the 7 bytes in fields 1 and 2.

3 A unique cross reference number, used in later blockettes to indicate this particular entry.

4 The identifying name of this response. This field gives a unique name to each dictionary entry.

5 A single letter "P" describing this type of stage.

6 A unit lookup key that refers to field 3 of the Units Abbreviation Blockette [34] for the units of the incoming signal to this stage of the filter.

7 A unit lookup key that refers to field 3 of the Units Abbreviation Blockette [34] for the stages output signal.

8 A single character describing the type of polynomial approximation (this field is mandatory) [Note: The input units (x) into the polynomial will most always be in Volts. The output units (pn(x)) will be in the units of field 5.]:

M — MacLaurin

$$pn(x) = a0 + a1*x + a2*x^2 + ... + an*x^n$$

Note: The following three fields play no part in the calculation to recover Earth units (i.e. field 5) for this response. If these fields are available from the instrumentation literature, they can be used in post-processing to assess the frequency domain validity.

9 A single character describing valid frequency units:
"A" — rad/sec
"B" — Hz

10 If available, the low frequency corner for which the sensor is valid. 0.0 if unknown or zero.

11 If available, the high frequency corner for which the sensor is valid. Nyquist if unknown.

12 Lower bound of approximation. This should be in units of 5.

13 Upper bound of approximation. This should be in units of 5.

14 The maximum absolute error of the polynomial approximation. Put 0.0 if the value is unknown or actually zero.

15 The number of coefficients that follow in the polynomial approximation. The 49 polynomial coefficients are given lowest order first and the number of coefficients is one more than the degree of the polynomial.

16 The value of the polynomial coefficient.

17 The error for field 12. Put 0.0 here if the value is unknown or actually zero. This error should be listed as a positive value, but represents a +/- error (i.e. 2 standard deviations).

[43] Response (Poles & Zeros) Dictionary Blockette

Name: **Response (Poles & Zeros) Dictionary Blockette**
Blockette Type: **043**
Control Header: **Abbreviation Dictionaries**
Field Station Volume: **Some Response Required**
Station Oriented Network Volume: **Some Response Required**
Event Oriented Network Volume: **Some Response Required**

Note	Field name	Type	Length	Mask or Flags
1	Blockette type — 043	D	3	"###"
2	Length of blockette	D	4	"####"
3	Response Lookup Key	D	4	"####"
4	Response Name	V	1-25	"[UN_]"
5	Response type	A	1	[U]
6	Stage signal input units	D	3	"###"
7	Stage signal output units	D	3	"###"
8	AO normalization factor (1.0 if none)	F	12	"-#.#####E-##"
9	Normalization frequency (Hz)	F	12	"-#.#####E-##"
10	Number of complex zeros	D	3	"###"
	REPEAT fields 11 — 14 for the Number of complex zeros:			
11	Real zero	F	12	"-#.#####E-##"
12	Imaginary zero	F	12	"-#.#####E-##"
13	Real zero error	F	12	"-#.#####E-##"
14	Imaginary zero error	F	12	"-#.#####E-##"
15	Number of complex poles	D	3	"###"
	REPEAT fields 16 — 19 for the Number of complex poles:			
16	Real pole	F	12	"-#.#####E-##"
17	Imaginary pole	F	12	"-#.#####E-##"
18	Real pole error	F	12	"-#.#####E-##"
19	Imaginary pole error	F	12	"-#.#####E-##"

NOTE: See Response (Poles & Zeros) Blockette [53] for more information.

Notes for fields:

1 Standard blockette type identification number.

2 Length of the entire blockette, including the 7 bytes in fields 1 and 2.

3 A unique cross reference number, used in later blockettes to indicate this particular dictionary entry.

4 The identifying name of this response. This field gives a unique name to each dictionary entry.

5 A single character describing the type of stage:

A — Laplace transform analog response, in rad/sec

B — Analog response, in Hz

C — Composite (currently undefined)

D — Digital (Z - transform)

6 A unit lookup key that refers to the Units Abbreviation Blockette [34] for the units of the incoming signal to this stage of the filter. This signal will usually be ground motion, volts, or counts, depending on where it is in the filter system.

7 Like field 6, but for the stage's output signal. Analog filters usually emit volts, and digital filters usually emit counts.

8 An optional field, used as a multiplicative factor to normalize the filter. Otherwise, put 1.0 here.

9 The frequency, f_n, in Hertz, at which the value in field 8 is normalized (if any).

10 The number of complex zeros that follow.

11 The real portion of the complex zero value.

12 The imaginary portion of the complex zero value.

13 The error for field 11. For example, if the value of real zero (field 11) were 200.0 and the error was 2 per cent, use 4.0 for the error value in field 12. Put 0.0 here if the value is unknown.

14 As in field 13, this is the error for field 12.

15 The number of poles that follow.

16 The real portion of the complex pole.

17 The imaginary portion of the complex pole.

18 The error value for field 16.

19 The error value for field 17.

[44] Response (Coefficients) Dictionary Blockette

Name: **Response (Coefficients) Dictionary Blockette**
Blockette Type: **044**
Control Header: **Abbreviation Dictionaries**
Field Station Volume: **Some Response Required**
Station Oriented Network Volume: **Some Response Required**
Event Oriented Network Volume: **Some Response Required**

Note	Field name	Type	Length	Mask or Flags
1	Blockette type — 044	D	3	"###"
2	Length of blockette	D	4	"####"
3	Response Lookup Key	D	4	"####"
4	Response Name	V	1-25	"[UN_]"
5	Response type	A	1	[U]
6	Signal input units	D	3	"###"
7	Signal output units	D	3	"###"
8	Number of numerators	D	4	"####"
	REPEAT fields 9 — 10 for the Number of numerators:			
9	Numerator coefficient	F	12	"-#.#####E-##"
10	Numerator error	F	12	"-#.#####E-##"
11	Number of denominators	D	4	"####"
	REPEAT fields 12 — 13 for the Number of denominators:			
12	Denominator coefficient	F	12	"-#.#####E-##"
13	Denominator error	F	12	"-#.#####E-##"

NOTE: See Response (Coefficients) Blockette [54] for more information.

Notes for fields:

1 Standard blockette type identification number.

2 Length of the entire blockette, including the 7 bytes in fields 1 and 2.

3 A unique cross reference number, used in later blockettes to indicate this particular dictionary entry. For continuation records of this type, use the same Response lookup key and append part 1, part 2 etc. to the response name in field 4.

4 The identifying name of this response. This field gives a unique name to each dictionary entry.

5 A single character describing the type of stage:
A — Laplace transform analog response, in rad/sec
B — Analog response, in Hz
C — Composite (currently undefined)
D — Digital (Z - transform)

6 A unit lookup key that refers to the Units Abbreviation Blockette [34] for the units of the incoming signal to this stage of the filter. This signal will usually be ground motion, volts, or counts, depending on where it is in the filter system.

7 Like field 6, but for the stage's output signal. Analog filters usually emit volts, and digital filters usually emit counts.

8 The number of numerator values that follow.

9 The numerator coefficient value.

10 The error of field 9.

11 The number of denominator values that follow. Denominators are only used for IIR filters. FIR type filters use only the numerator. If there are no denominators, place a zero here and stop the blockette.

12 The denominator coefficient value.

13 The error of field 12.

[45] Response List Dictionary Blockette

Name: **Response List Dictionary Blockette**
Blockette Type: **045**
Control Header: **Abbreviation Dictionaries**
Field Station Volume: **Some Response Required**
Station Oriented Network Volume: **Some Response Required**
Event Oriented Network Volume: **Some Response Required**

Note	Field name	Type	Length	Mask or Flags
1	Blockette type — 045	D	3	"###"
2	Length of blockette	D	4	"####"
3	Response Lookup Key	D	4	"####"
4	Response Name	V	1-25	"[UN_]"
5	Signal input units	D	3	"###"
6	Signal output units	D	3	"###"
7	Number of responses listed	D	4	"####"
	REPEAT fields 8 — 12 for the Number of responses listed:			
8	Frequency (Hz)	F	12	"-#.#####E-##"
9	Amplitude	F	12	"-#.#####E-##"
10	Amplitude error	F	12	"-#.#####E-##"
11	Phase angle (degrees)	F	12	"-#.#####E-##"
12	Phase error (degrees)	F	12	"-#.#####E-##"

NOTE: See Response List Blockette [55] for more information.

Notes for fields:

1 Standard blockette type identification number.

2 Length of the entire blockette, including the 7 bytes in fields 1 and 2.

3 A unique cross reference number, used in later blockettes to indicate this particular dictionary entry.

4 The identifying name of this response. This field gives a unique name to each dictionary entry.

5 A unit lookup key that refers to the Units Abbreviation Blockette [34] for the units of the incoming signal to this stage of the filter. This signal will usually be ground motion, volts, or counts, depending on where it is in the filter system.

6 Like field 5, but for the stage's output signal. Analog filters usually emit volts, and digital filters usually emit counts.

7 The number of responses in the repeat block that follows.

8 The frequency of this response.

9 The amplitude of this response.

10 The error of the amplitude.

1 The phase angle at this frequency.

12 The error of the phase angle.

[46] Generic Response Dictionary Blockette

Name: **Generic Response Dictionary Blockette**
Blockette Type: **046**
Control Header: **Abbreviation Dictionaries**
Field Station Volume: **Some Response Required**
Station Oriented Network Volume: **Some Response Required**
Event Oriented Network Volume: **Some Response Required**

Note	Field name	Type	Length	Mask or Flags
1	Blockette type — 046	D	3	"###"
2	Length of blockette	D	4	"####"
3	Response Lookup Key	D	4	"####"
4	Response Name	V	1-25	"[UN_]"
5	Signal input units	D	3	"###"
6	Signal output units	D	3	"###"
7	Number of corners listed	D	4	"####"
	REPEAT fields 8 — 9 for the Number of corners listed:			
8	Corner frequency (Hz)	F	12	"-#.#####E-##"
9	Corner slope (db/decade)	F	12	"-#.#####E-##"

NOTE: See Generic Response Blockette [56] for more information.

Notes for fields:

1 Standard blockette type identification number.

2 Length of the entire blockette, including the 7 bytes in fields 1 and 2.

3 A unique cross reference number, used in later blockettes to indicate this particular dictionary entry.

4 The identifying name of this response. This field gives a unique name to each dictionary entry.

5 A unit lookup key that refers to the Units Abbreviation Blockette [34] for the units of the incoming signal to this stage of the filter. The signal will usually be ground motion, volts, or counts, depending on where it is in the filter system.

6 Like field 5, but for the stage's output signal. Analog filters usually emit volts, digital filters usually emit counts.

7 The number of response corner frequencies specified in the repeat block that follows.

8 The corner frequency.

9 The slope of the line after the corner, measured in db/decade.

[47] Decimation Dictionary Blockette

Name: **Decimation Dictionary Blockette**
Blockette Type: **047**
Control Header: **Abbreviation Dictionaries**
Field Station Volume: **Required for Digital Stage**
Station Oriented Network Volume: **Required for Digital Stage**
Event Oriented Network Volume: **Required for Digital Stage**

Note	Field name	Type	Length	Mask or Flags
1	Blockette type — 046	D	3	"###"
2	Length of blockette	D	4	"####"
3	Response Lookup Key	D	4	"####"
4	Response Name	V	1-25	"[UN_]"
5	Input sample rate	F	10	"#.####E-##"
6	Decimation factor	D	5	"#####"
7	Decimation offset	D	5	"#####"
8	Estimated delay (seconds)	F	11	"-#.####E-##"
9	Correction applied (seconds)	F	11	"-#.####E-##"

NOTE: See Decimation Blockette [57] for more information.

Notes for fields:

1 Standard blockette type identification number.

2 Length of the entire blockette, including the 7 bytes in fields 1 and 2.

3 A unique cross reference number, used in later blockettes to indicate this particular dictionary entry.

4 The identifying name of this response. This field gives a unique name to each dictionary entry.

5 The incoming sample rate, in samples per second.

6 The decimation factor. When this number of samples are read in, one final sample comes out. Calculate the output sample rate by dividing field 5 by the decimation factor.

7 This field determines which sample is chosen for use. Make the value of this field greater than or equal to zero, but less than the decimation factor. If you pick the first sample, set this field to zero. If you pick the second sample, set it to 1, and so forth.

8 The estimated pure delay for the stage; it may or may not be also corrected in field 7. This field's value is nominal, and may be unreliable.

9 The time shift applied to the time tag due to delay at this stage of the filter; a negative number indicating the 9 of time added to the former time tag. The actual delay is difficult to estimate, and the correction applied neglects dispersion. This field allows the user to know how much correction was used, in case a more accurate correction is to be applied later. A zero here implies no correction was done.

[48] Channel Sensitivity/Gain Dictionary Blockette

Name: **Channel Sensitivity/Gain Dictionary Blockette**
Blockette Type: **048**
Control Header: **Abbreviation Dictionaries**
Field Station Volume: **Required**
Station Oriented Network Volume: **Required**
Event Oriented Network Volume: **Required**

Note	Field name	Type	Length	Mask or Flags
1	Blockette type — 048	D	3	"###"
2	Length of blockette	D	4	"####"
3	Response Lookup Key	D	4	"####"
4	Response Name	V	1-25	"[UN_]"
5	Sensitivity/gain	F	12	"-#.#####E-##"
6	Frequency (Hz)	F	12	"-#.#####E-##"
7	Number of history values	D	2	"##"
	REPEAT fields 8 — 10 for the Number of history values:			
8	Sensitivity for calibration	F	12	"-#.#####E-##"
9	Frequency of calibration sensitivity	F	12	"-#.#####E-##"
10	Time of above calibration	V	1—22	TIME

NOTE: See Channel Sensitivity/Gain Blockette [58] for more information.

Notes for fields:

1 Standard blockette type identification number.

2 Length of the entire blockette, including the 7 bytes in fields 1 and 2.

3 A unique cross reference number, used in later blockettes to indicate this particular dictionary entry.

4 The identifying name of this response. This field gives a unique name to each dictionary entry.

5 The gain at this stage, or the sensitivity for the channel.

6 The frequency, f_n, at which the value in field 5 is correct.

7 You may record any number of standard calibration values for a history of the calculation of the sensitivity value (calibration methods usually only give information about the final channel response, not the individual stages). This field represents the number of calibration history entries that follow. If there is no history, or this is a gain value, put zero here and stop the blockette.

8 The recorded amplitude value of this history entry.

9 The frequency for this calibration; you can use a zero for a step calibration.

10 The time when the calibration was done.

[49] Response (Polynomial) Dictionary Blockette

Name: **Response (Polynomial) Dictionary Blockette**
Blockette Type: **049**
Control Header: **Abbreviation Dictionaries**
Field Station Volume: **Some Response Required**
Station Oriented Network Volume: **Some Response Required**
Event Oriented Network Volume: **Some Response Required**
Introduced in SEED version **2.3**

Use this blockette to characterize the response of a non-linear sensor.

Note	Field name	Type	Length	Mask or Flags
1	Blockette Type — 049	D	3	"###"
2	Length of Blockette	D	4	"####"
3	Response Lookup Key	D	4	"####"
4	Response Name	V	1-25	"[UN_]"
5	Transfer Function Type	A	1	[U]
6	Stage Signal Input Units	D	3	"###"
7	Stage Signal Output Units	D	3	"###"
8	Polynomial Approximation Type	A	1	[U]
9	Valid Frequency Units	A	1	[U]
10	Lower Valid Frequency Bound	F	12	"-#.#####E-##"
11	Upper Valid Frequency Bound	F	12	"-#.#####E-##"
12	Lower Bound of Approximation	F	12	"-#.#####E-##"
13	Upper Bound of Approximation	F	12	"-#.#####E-##"
14	Maximum Absolute Error	F	12	"-#.#####E-##"
15	Number of Polynomial Coefficients	D	3	"###"
	(Repeat fields 16 and 17 for each polynomial coefficient)			
16	Polynomial Coefficient	F	12	"-#.#####E-##"
17	Polynomial Coefficient Error	F	12	"-#.#####E-##"

Notes for Fields:

1 Standard blockette type identification number.

2 Length of entire blockette, including the 7 bytes in fields 1 and 2.

3 A unique cross reference number, used in later blockettes to indicate this particular entry.

4 The identifying name of this response. This field gives a unique name to each dictionary entry.

5 A single letter "P" describing this type of stage.

6 A unit lookup key that refers to field 3 of the Units Abbreviation Blockette [34] for the units of the incoming signal to this stage of the filter.

7 A unit lookup key that refers to field 3 of the Units Abbreviation Blockette [34] for the stages output signal.

8 A single character describing the type of polynomial approximation (this field is mandatory): (Note: The input units (x) into the polynomial will most always be in Volts. The output units (pn(x)) will be in the units of field 5.)

M -- MacLaurin

$$pn(x) = a0 + a1*x + a2*x^2 + ... + an*x^n$$

(Note: The following three fields play no part in the calculation to recover Earth units (ie field 5) for this response. If these fields are available from the instrumentation literature, they can be used in post-processing to assess the frequency domain validity.)

9 A single character describing valid frequency units:

"A" -- rad/sec

"B" -- Hz

10 If available, the low frequency corner for which the sensor is valid. 0.0 if unknown or zero.

11 If available, the high frequency corner for which the sensor is valid. Nyquist if unknown.

12 Lower bound of approximation. This should be in units of 5.

13 Upper bound of approximation. This should be in units of 5.

14 The maximum absolute error of the polynomial approximation. Put 0.0 if the value is unknown or actually zero.

15 The number of coefficients that follow in the polynomial approximation. The polynomial coefficients are given lowest order first and the number of coefficients is one more than the degree of the polynomial.

16 The value of the polynomial coefficient.

17 The error for field 12. Put 0.0 here is the value is unknown or actually zero. This error should be listed as a positive value, but represent a +/- error (ie 2 standard deviations).

Chapter 6

Station Control Headers

The station header records contain all the configuration and identification information for the station and all its instruments. The SEED format provides a great deal of flexibility for associating recording channels to the station, including the ability to support different data formats dynamically. For each new station, start a new logical record, set the remainder of any previous header records to blanks, and write it out.

For analog cascading, use the Response (Poles & Zeros) Blockette [53], and the Channel Sensitivity/Gain Blockette [58] if needed. For digital cascading, use the Response (Coefficients) Blockette [54], and the Decimation Blockette [57] or Channel Sensitivity/Gain Blockette [58] if needed. For additional documentation, you may also use the Response List Blockette [55] or the Generic Response Blockette [56].

[50] Station Identifier Blockette

Name: **Station Identifier Blockette**
Blockette Type: **050**
Control Header: **Station**
Field Station Volume: **Required**
Station Oriented Network Volume: **Required**
Event Oriented Network Volume: **Required**

Sample:

0500098ANMOΔΔ+34.946200-106.456700+1740.00006001Albuquerque,ΔNewMexico,ΔUSA~0013210101989,241~~NIU

	Note	Field name	Type	Length	Mask or Flags
	1	Blockette type — 050	D	3	"###"
	2	Length of blockette	D	4	"####"
	3	Station call letters	A	5	[UN]
	4	Latitude (degrees)	D	10	"-##.######"
	5	Longitude (degrees)	D	11	"-###.######"
	6	Elevation (m)	D	7	"-####.#"
	7	Number of channels	D	4	"####"
	8	Number of station comments	D	3	"###"
	9	Site name	V	1—60	[UNLPS]
	10	Network identifier code	D	3	"###"
	11	32 bit word order	D	4	"####"
	12	16 bit word order	D	2	"##"
	13	Start effective date	V	1—22	TIME
	14	End effective date	V	0—22	TIME
	15	Update flag	A	1	
V2.3-	16	Network Code	A	2	[ULN]

Notes for fields:

1 Standard blockette type identification number.

2 Length of the entire blockette, including the 7 bytes in fields 1 and 2.

3 Station call letters.

4 Earth latitude in degrees from the equator. Use a negative number for a southern latitude. The Station Identifier Blockette [50] contains the latitude of the station address associated with the call letters in field 3 and is often the same as the instruments' coordinates in the Channel Identifier Blockettes [52]. Seismologists should use the coordinates in the Channel Identifier Blockette [52] and site preparation teams should ensure that these numbers are calculated as accurately as possible.

5 Earth longitude from Greenwich. A negative number denotes a western longitude.

6 Elevation of local ground level in meters.

7 The number of channels that follow, not including any channel update blockettes (optional; we recommend not using this field and leaving it set to blanks).

8 The number of Station Comment Blockettes [51] that follow (optional; we recommend not using this field and leaving it set to blanks).

9 The station site, usually as "Local town/city, major political subdivision (state/province), country/territory".

10 The abbreviation lookup code (field 3) from the Generic Abbreviation Blockette [33] abbreviation dictionary, that refers to the network to which the station belongs. If you are coding from an experimental system, or from a special site or group collaboration, you can use an abbreviation to specify the contributors.

11 The swap order in which 32-bit quantities are specified in the data headers. The swap order of the data itself is specified for the channel in the data format dictionary. This dictionary entry describes the exact format of the data with the data description language — see Appendix D. Some swap orders for computers are:

Little Endian (Intel, etc.) — "0123"

Big Endian (Motorola, etc.) — "3210"

This order also applies to the FLOAT Binary Data Field.

12 16-bit quantity byte swapping order (as for longword above), for the data headers only. Some swap orders for computers are:

Little Endian (Intel, etc.) — "01"

Big Endian (Motorola, etc.) — "10"

13 The earliest known date that information in this header record is correct (used with update records). Use the date when the database was last changed; if you do not know this date, use the start date of the volume.

14 The latest date when this information is correct. The minimum length of this field can be zero, implying that the information is still correct.

15 The update flag indicates to what the data update records refer. Use update records to either describe changes to the condition of a station during this volume or to refer to previous volumes (as errata distributions). Use one of these flags:

N Effective dates pertain to these data

U Control header updates information previously sent

See Appendix H for more information.

16 A two character alphanumeric identifier that uniquely identifies the network operator responsible for the data logger. This identifier is assigned by the IRIS Data Management Center in consultation with the FDSN working group on the SEED format. This code is repeated in field 7 of the fixed section of data headers. See Appendix J for the current list of Network Codes.

NOTE: If information in the Station Identifier Blockette [50] were to be changed during the time interval of the volume, additional blockettes with the new information and new effective dates would immediately follow the first blockette (if there are several changes, there should be additional blockettes).

The 16 bit and 32 bit swap orders pertain only to the fields in the fixed headers and blockettes of the data records. The swap order in the data section itself is described with the data description language (see Appendix D). All of these fields must be present and accurate for any decoder to operate correctly. These headers also mean that network volumes can contain data with different swap orders. We recommend that network volumes convert data headers to a unified swapping arrangement, but the SEED format does not require this.

To eliminate a potential problem, all data records and blockettes for a given station must use the same byte ordering within a SEED volume.

[51] Station Comment Blockette

Name: **Station Comment Blockette**
Blockette Type: **051**
Control Header: **Station**
Field Station Volume: **Optional**
Station Oriented Network Volume: **Optional**
Event Oriented Network Volume: **Optional**

Sample:

05100351992,001~1992,002~0740000000

Note	Field name	Type	Length	Mask or Flags
1	Blockette type — 051	D	3	"###"
2	Length of blockette	D	4	"####"
3	Beginning effective time	V	1—22	TIME
4	End effective time	V	1—22	TIME
5	Comment code key	D	4	"####"
6	Comment level	D	6	"######"

Notes for fields:

1 Standard blockette type identification number.

2 Length of the entire blockette, including the 7 bytes in fields 1 and 2.

3 The time when the comment comes into effect.

4 The time when the comment is no longer in effect.

5 The comment code key (field 3) of the associated Comment Description Dictionary Blockette [31] in the abbreviation dictionary section.

6 The numeric value associated with the level unit in the above Comment Description Dictionary Blockette [31] (if any).

NOTE: Include any data outages and time corrections in the station comments.

[52] Channel Identifier Blockette

Name: **Channel Identifier Blockette**
Blockette Type: **052**
Control Header: **Station**
Field Station Volume: **Required**
Station Oriented Network Volume: **Required**
Event Oriented Network Volume: **Required**

Sample:

```
0520119BHE0000004~001002+34.946200-106.456700+1740.0100.0090.0+00.0000112Δ2.000E+01Δ2.00
0E-030000CG~1991,042,20:48~~N
```

Note	Field name	Type	Length	Mask or Flags
1	Blockette type — 052	D	3	"###"
2	Length of blockette	D	4	"####"
3	Location identifier	A	2	[UN]
4	Channel identifier	A	3	[UN]
5	Subchannel identifier	D	4	"####"
6	Instrument identifier	D	3	"###"
7	Optional comment	V	0—30	[UNLPS]
8	Units of signal response	D	3	"###"
9	Units of calibration input	D	3	"###"
10	Latitude (degrees)	D	10	"-##.######"
11	Longitude (degrees)	D	11	"-###.######"
12	Elevation (m)	D	7	"-####.#"
13	Local depth (m)	D	5	"###.#"
14	Azimuth (degrees)	D	5	"###.#"
15	Dip (degrees)	D	5	"-##.#"
16	Data format identifier code	D	4	"####"
17	Data record length	D	2	"##"
18	Sample rate (Hz)	F	10	"#.####E-##"
19	Max clock drift (seconds)	F	10	"#.####E-##"
20	Number of comments	D	4	"####"
21	Channel flags	V	0—26	[U]
22	Start date	V	1—22	TIME
23	End date	V	0—22	TIME
24	Update flag	A	1	

Notes for fields:

1 Standard blockette type identification number.

2 Length of the entire blockette, including the 7 bytes in fields 1 and 2.

3 Describes the individual sites on an array station, operated by the same network operator. Do not use this field to distinguish multiple data loggers operated by different networks at the same station. Field 16 of blockette 50 is used for that purpose.

4 Standard channel identifier (See Appendix A).

5 Used for a multiplexed data channel. (Normally, data are not multiplexed, but you can do this if necessary.) The Data Format Dictionary Blockette [30] for this channel must correctly describe the multiplexing being used. Create a Channel Identifier Blockette [52] for each multiplexed subchannel.

6 An abbreviation lookup code (field 3) from the Generic Abbreviation Blockette [33] abbreviation dictionary that contains a name for this instrument.

7 An optional comment given to the instrument. It can be anything, including a serial number or a notation of special modifications.

8 A unit lookup key that refers to field 3 of the Units Abbreviation Blockette [34] for the signal response of the instrument. This is usually the ground motion response, such as M/S for velocity sensitive seismometers. These units should be the same as the signal input units in the first filter stage (often a blockette 53).

9 A unit lookup key that refers to field 3 of the Units Abbreviation Blockette [34] for the units of calibration input, usually volts or amps.

10 Latitude for the instrument. The Station Identifier Blockette [50] might contain different coordinates.

11 Longitude for the instrument.

12 Elevation of the instrument. To find the local ground depth, add the depth field to the instrument elevation. If negative elevations are less than -9999 m, the decimal value should not be used. This allows depths up to -99999 m to be accommodated.

13 The local depth or overburden of the instrument's location. For downhole instruments, the depth of the instrument under the surface ground level. For underground vaults, the distance from the instrument to the local ground level above. Surface instruments can use zero. If negative elevations are less than -999 m, the decimal value should not be used. This allows depths up to -9999 m to be accommodated.

14 The azimuth of the instrument in degrees from north, clockwise.

15 The dip of the instrument in degrees, down from horizontal.
The azimuth and the dip describe the direction of the sensitive axis of the instrument (if applicable). Motion in the same direction as this axis is positive. SEED provides this field for non-traditional instruments or for traditional instruments that have been oriented in some non-traditional way. Here are traditional orientations:

Z — Dip -90, Azimuth 0 (Reversed: Dip 90, Azimuth 0)

N — Dip 0, Azimuth 0 (Reversed: Dip 0, Azimuth 180)

E — Dip 0, Azimuth 90 (Reversed: Dip 0, Azimuth 270)

Traditionally, the mass (boom) on vertical seismometers is oriented to the north, but sometimes this is not possible. If you know the vertical orientation, place it in the azimuth field. If the orientation is 0 degrees, use an azimuth of 360.0. If you do not know the orientation, set the azimuth to 0.0.

If instruments are reversed in the field, reverse the dip/azimuth fields. Data collection centers and data management centers should never actually modify the data, but report on its quality. User reading programs can automatically reverse the data if they report that they are doing so. Here are dip and azimuth examples of some tri-axial instruments:

A — Dip -60, Azimuth 0 (Reversed: Dip 60, Azimuth 180)

B — Dip -60, Azimuth 120 (Reversed: Dip 60, Azimuth 300)

C — Dip -60, Azimuth 240 (Reversed: Dip 60, Azimuth 60)

16 A data format lookup key that refers to field 4 of a Data Format Dictionary Blockette [30] that describes the format of the data in the data section for this channel.

17 The exponent (as a power of two) of the record length for these data. The data record can be as small as 256 bytes but never greater than 4096 bytes. Place a "12" in this field for a 4096 byte record. 4096 is preferred.

18 Sample rate in samples per second. This field contains the nominal sampling interval of the digitizer. No considerations for drift or time correction go here. Set this rate to zero for channels not sampled at regular intervals (such as console logs or alarms).

19 A tolerance value, measured in seconds per sample, used as a threshold for time error detection in the data. The number of samples in a record are multiplied by this value to calculate a maximum drift allowed for the time in the next record. If the difference in times is less than this drift, consider them in the same time series.

20 The number of Channel Comment Blockettes [59] that follow (optional; we recommend not using this field and leaving it set to blanks).

21 Channel type flags:

T — Channel is triggered

C — Channel is recorded continuously

H — State of health data

G — Geophysical data

W — Weather or environmental data

F — Flag information (nominal, not ordinal)

S — Synthesized data

I — Channel is a calibration input

E — Channel is experimental or temporary

M — Maintenance tests are underway on channel; possible abnormal data

B — Data are a beam synthesis

Here are some uses for the channel flags field:

G — Geophysical data:

- Seismic (seismometer, geophone)
- Earth electric field
- Magnetic (magnetometer)
- Gravity (gravimeter)
- Tilt (tilt meter)
- Strain (strain meter)

W — Weather or environmental data (readings inside vault/downhole or in equipment boxes may also be State-of-Health [H]):

- Wind speed or direction
- Pressure
- Temperature
- Humidity
- Precipitation

H — State of Health:

- Power supply voltages
- Status of system peripherals
- Door open or closed
- Room temperature
- System temperature

F — Flag Information. Includes all on/off or go/no-go conditions, such as power supply okay, line power okay, door open, or system too hot

B — Beam synthesis channel (Do not set the 'S' (synthetic) flag for beams)

S — Synthetic Data. Oddly oriented devices that have been rotated mathematically into a traditional orientation, or the output of a synthetic seismogram program should have the "S" flag set.

22 The earliest known date that information in this blockette is correct, or the response blockettes that follow this blockette. Used with update records. If possible, list the time when the database was last changed; but if this date is not known, use the start date of the volume.

23 The latest date when this information is correct. A zero length implies that the information was still valid when the volume was generated. Use this field for current volumes, when the final date is in the future. You can place the date of the end of the volume here.

24 Indicate what data the update records refer to. Use update records to either denote changes to the condition of a station during this volume, or to refer to previous volumes in an errata distribution. Here are the possible values for the flag:

N — Effective dates pertain to these data

U — Control header updates information previously sent

See Appendix H for more information.

NOTE: If any of the channel data in this blockette, or in the response blockettes that follow, changes during the time interval of the volume, repeat the Channel Identifier blockette after the last channel blockette with the new effective date (even if data in the Channel Identifier blockette did not change). Then place any channel blockettes, and their associated response blockettes, that changed.

[53] Response (Poles & Zeros) Blockette

Name: **Response (Poles & Zeros) Blockette**
Blockette Type: **053**
Control Header: **Station**
Field Station Volume: **Some Response Required**
Station Oriented Network Volume: **Some Response Required**
Event Oriented Network Volume: **Some Response Required**

Use this blockette for the analog stages of filter systems and for infinite impulse response (IIR) digital filters. Digital filters usually have a Decimation Blockette [57] following, and most stages have a Sensitivity/Gain Blockette [58] following. The stage sequence takes into account the fact that newer seismic systems will contain combinations of analog and digital filtering, allowing different deconvolution algorithms to be run sequentially (in cascade). SEED reserves the composite function to describe analog instruments with digital feedback circuitry. Stage order is the same as the original convolution order. Use the original earth units for the input units of stage 1. Use digital counts for the output units on the last stage. (See Appendix C for more information.)

Sample:

```
0530382BΔ1007008Δ7.87395E+00Δ5.00000E-02ΔΔ3
Δ0.00000E+00Δ0.00000E+00Δ0.00000E+00Δ0.00000E+00
Δ0.00000E+00Δ0.00000E+00Δ0.00000E+00Δ0.00000E+00
-1.27000E+01Δ0.00000E+00Δ0.00000E+00Δ0.00000E+00ΔΔ4
-1.96418E-03Δ1.96418E-03Δ0.00000E+00Δ0.00000E+00
S-1.96418E-03-1.96418E-03Δ0.00000E+00Δ0.00000E+00
53-6.23500E+00Δ7.81823E+00Δ0.00000E+00Δ0.00000E+00
-6.23500E+00-7.81823E+00Δ0.00000E+00Δ0.00000E+00
```

Note	Field name	Type	Length	Mask or Flags
1	Blockette type — 053	D	3	"###"
2	Length of blockette	D	4	"####"
3	Transfer function type	A	1	[U]
4	Stage sequence number	D	2	"##"
5	Stage signal input units	D	3	"###"
6	Stage signal output units	D	3	"###"
7	AO normalization factor (1.0 if none)	F	12	"-#.#####E-##"
8	Normalization frequency fn(Hz)	F	12	"-#.#####E-##"
9	Number of complex zeros	D	3	"###"
	REPEAT fields 10 — 13 for the Number of complex zeros:			
10	Real zero	F	12	"-#.#####E-##"
11	Imaginary zero	F	12	"-#.#####E-##"
12	Real zero error	F	12	"-#.#####E-##"
13	Imaginary zero error	F	12	"-#.#####E-##"
14	Number of complex poles	D	3	"###"
	REPEAT fields 15 — 18 for the Number of complex poles:			
15	Real pole	F	12	"-#.#####E-##"
16	Imaginary pole	F	12	"-#.#####E-##"
17	Real pole error	F	12	"-#.#####E-##"
18	Imaginary pole error	F	12	"-#.#####E-##"

Notes for fields:

1 Standard blockette type identification number.

2 Length of the entire blockette, including the 7 bytes in fields 1 and 2. This blockette could exceed the maximum of 9999 characters. If so, continue on the next record. Set field 4 the same, but ignore fields 5—8 in subsequent blockettes (neither write nor read them).

3 A single character describing the type of stage:

A — Laplace transform analog response, in rad/sec

B — Analog response, in Hz

C — Composite (currently undefined)

D — Digital (Z - transform)

4 The identifying number of this stage. Stages are numbered starting at 1, with one response per stage (blockettes [53], [54], [55], or [56]), optionally followed by a Decimation Blockette [57] or a Sensitivity/ Gain Blockette [58].

5 A unit lookup key that refers to field 3 of the Units Abbreviation Blockette [34] for the units of the incoming signal to this stage of the filter. This signal will usually be ground motion, volts, or counts, depending on where it is in the filter system.

6 A unit lookup key that refers to field 3 of the Units Abbreviation Blockette [34] for the stage's output signal. Analog filters usually emit volts, and digital filters usually emit counts.

7 This field is mandatory and must be set such that when you evaluate the polynomial at the reference frequency the result will be 1.

8 The frequency f_{n},, in Hertz, at which the value in field 7 is normalized (if any).

9 The number of complex zeros that follow.

10 The real portion of the complex zero value.

11 The imaginary portion of the complex zero value.

12 The error for field 10. For example, if the value of real zero (field 10) were 200.0 and the error was 2 per cent, use 4.0 for the error value in field 12. Put 0.0 here if the value is unknown or is actually zero. This error should be listed as a positive value, but represents a +/- error.

13 As in field 12, this is the error for field 11.

14 The number of poles that follow.

15 The real portion of the complex pole.

16 The imaginary portion of the complex pole.

17 The error value for field 15.

18 The error value for field 16.

[54] Response (Coefficients) Blockette

Name: **Response (Poles & Zeros) Blockette**
Blockette Type: **054**
Control Header: **Station**
Field Station Volume: **Some Response Required**
Station Oriented Network Volume: **Some Response Required**
Event Oriented Network Volume: **Some Response Required**

This blockette is usually used only for finite impulse response (FIR) filter stages. You can express Laplace transforms this way, but you should use the Response (Poles & Zeros) Blockettes [53] for this. You can express IIR filters this way, but you should use the Response (Poles & Zeros) Blockette [53] here, too, to avoid numerical stability problems. Usually, you will follow this blockette with a Decimation Blockette [57] and a Sensitivity/Gain Blockette [58] to complete the definition of the filter stage.

This blockette is the only blockette that might overflow the maximum allowed value of 9,999 characters. If there are more coefficients than fit in one record, list as many as will fit in the first occurrence of this blockette (the counts of Number of numerators and Number of denominators would then be set to the number included, not the total number). In the next record, put the remaining number. Be sure to write and read these blockettes in sequence, and be sure that the first few fields of both records are identical. If desired, reading (and writing) programs can work with both blockettes as one after reading (or before writing).

Sample:

```
S0542400DΔ4011010ΔΔ99Δ6.67466E-06Δ0.00000E+00Δ1.09015E-05Δ0.00000E+00Δ1.49367E-
05Δ0.00000E+00Δ1.36129E-05Δ0.00000E+00Δ1.68905E-06Δ0.00000E+00-2.55129E-
5405Δ0.00000E+00-6.96400E-05Δ0.00000E+00-1.26610E-04Δ0.00000E+00-1.84580E-
04Δ0.00000E+00-2.23689E-04Δ0.00000E+00-2.18583E-04Δ0.00000E+00-1.44157E-
04Δ0.00000E+00Δ1.60165E-05Δ0.00000E+00Δ2.60152E-04Δ0.00000E+00Δ5.60233E-
04Δ0.00000E+00Δ8.58722E-04Δ0.00000E+00Δ1.07275E-03Δ0.00000E+00Δ1.10758E-
03Δ0.00000E+00Δ8.78519E-04Δ0.00000E+00Δ3.38276E-04Δ0.00000E+00-4.96462E-
04Δ0.00000E+00-1.52660E-03Δ0.00000E+00-2.56818E-03Δ0.00000E+00-3.37140E-03Δ0.00000E+00-
3.66272E-03Δ0.00000E+00-3.20570E-03Δ0.00000E+00-1.87049E-03Δ0.00000E+00Δ3.03131E-
04Δ0.00000E+00Δ3.06640E-03Δ0.00000E+00Δ5.96158E-03Δ0.00000E+00Δ8.37105E-
03Δ0.00000E+00Δ9.61594E-03Δ0.00000E+00Δ9.09321E-03Δ0.00000E+00Δ6.42899E-
03Δ0.00000E+00Δ1.61822E-03Δ0.00000E+00-4.88235E-03Δ0.00000E+00-1.21360E-
02Δ0.00000E+00-1.87996E-02Δ0.00000E+00-2.32904E-02Δ0.00000E+00-2.40261E-02Δ0.00000E+00-
1.97035E-02Δ0.00000E+00-9.56741E-
```

Note	Field name	Type	Length	Mask or Flags
1	Blockette type — 054	D	3	"###"
2	Length of blockette	D	4	"####"
3	Response type	A	1	[U]
4	Stage sequence number	D	2	"##"
5	Signal input units	D	3	"###"
6	Signal output units	D	3	"###"
7	Number of numerators	D	4	"####"
	REPEAT fields 8 — 9 for the Number of numerators:			
8	Numerator coefficient	F	12	"-#.#####E-##"
9	Numerator error	F	12	"-#.#####E-##"
10	Number of denominators	D	4	"####"
	REPEAT fields 11 — 12 for the Number of denominators:			
11	Denominator coefficient	F	12	"-#.#####E-##"
12	Denominator error	F	12	"-#.#####E-##"

Notes for fields:

1 Standard blockette type identification number.

2 Length of the entire blockette, including the 7 bytes in fields 1 and 2. This blockette could exceed the maximum of 9999 characters. If so, continue with another blockette [55] on the next record. Set field 4 the same, but ignore fields 5—6 in subsequent blockettes.

3 A single character describing the type of stage:

A — Analog (rad/sec)

B — Analog (Hz)

C — Composite

D — Digital

4 The identifying number of the stage. Stages are numbered starting at 1. Use one response per stage (blockettes [53], [54], [55], or [56]); you may follow it with a Decimation Blockette [57] or a Sensitivity/Gain Blockette [58].

5 A unit lookup key that refers to field 3 of the Units Abbreviation Blockette [34] for the units of the incoming signal to this stage of the filter. This signal will usually be ground motion, volts, or counts, depending on where it is in the filter system.

6 A unit lookup key that refers to field 3 of the Units Abbreviation Blockette [34] for the stage's output signal. Analog filters usually emit volts, and digital filters usually emit counts.

7 The number of numerator values that follow.

8 The numerator coefficient value.

9 The error of field 8.

10 The number of denominator values that follow. Denominators are only used for IIR filters. FIR type filters use only the numerator. If there are no denominators, place a zero 55 here and stop the blockette.

11 The denominator coefficient value.

12 The error of field 11.

[55] Response List Blockette

Name: **Response List Blockette**
Blockette Type: **055**
Control Header: **Station**
Field Station Volume: **Some Response Required**
Station Oriented Network Volume: **Some Response Required**
Event Oriented Network Volume: **Some Response Required**

This blockette alone is not an acceptable response description; always use this blockette along with the standard response blockettes ([53], [54], [57], or [58]). If this is the only response available, we strongly recommend that you derive the appropriate poles and zeros and include blockette 53 and blockette 58.

Note	Field name	Type	Length	Mask or Flags
1	Blockette type — 055	D	3	"###"
2	Length of blockette	D	4	"####"
3	Stage sequence number	D	2	"##"
4	Signal input units	D	3	"###"
5	Signal output units	D	3	"###"
6	Number of responses listed	D	4	"####"
	REPEAT fields 7 — 11 for the Number of responses listed:			
7	Frequency (Hz)	F	12	"-#.#####E-##"
8	Amplitude	F	12	"-#.#####E-##"
9	Amplitude error	F	12	"-#.#####E-##"
10	Phase angle (degrees)	F	12	"-#.#####E-##"
11	Phase error (degrees)	F	12	"-#.#####E-##"

Notes for fields:

1 Standard blockette type identification number.

2 Length of the entire blockette, including the 7 bytes in fields 1 and 2. This blockette could exceed the maximum of 9999 characters. If so, continue with another blockette [55] on the next record. Set field 3 the same, but ignore fields 4 and 5 in subsequent blockettes.

3 The identifying number of the stage. Stages are numbered starting at 1.

4 A unit lookup key that refers to field 3 of the Units Abbreviation Blockette [34] for the units of the incoming signal to this stage of the filter. This signal will usually be ground motion, volts, or counts, depending on where it is in the filter system.

5 A unit lookup key that refers to field 3 of the Units Abbreviation Blockette [34] for the stage's output signal. Analog filters usually emit volts, and digital filters usually emit counts.

6 The number of responses in the repeat block that follows.

7 The frequency of this response.

8 The amplitude of this response.

9 The error of the amplitude.

10 The phase angle at this frequency.

11 The absolute error of the phase angle.

[56] Generic Response Blockette

Name: **Generic Response Blockette**
Blockette Type: **056**
Control Header: **Station**
Field Station Volume: **Some Response Required**
Station Oriented Network Volume: **Some Response Required**
Event Oriented Network Volume: **Some Response Required**

This blockette alone is not an acceptable response description; always use this blockette along with the standard response blockettes ([53], [54], [57], or [58]). You can, however, use this blockette alone to provide additional documentation.

Note	Field name	Type	Length	Mask or Flags
1	Blockette type — 056	D	3	"###"
2	Length of blockette	D	4	"####"
3	Stage sequence number	D	2	"##"
4	Signal input units	D	3	"###"
5	Signal output units	D	3	"###"
6	Number of corners listed	D	4	"####"
	REPEAT fields 7 — 8 for the Number of corners listed:			
7	Corner frequency (Hz)	F	12	"-#.#####E-##"
8	Corner slope (db/decade)	F	12	"-#.#####E-##"

Notes for fields:

1 Standard blockette type identification number.

2 Length of the entire blockette, including the 7 bytes in fields 1 and 2. This blockette could exceed the maximum of 9999 characters. If so, continue on the next record. Set field 3 the same, but ignore fields 4 and 5 in subsequent blockettes.

3 The identifying number of the stage. Stages are numbered starting at 1.

4 A unit lookup key that refers to field 3 of the Units Abbreviation Blockette [34] for the units of the incoming signal to this stage of the filter. The signal will usually be ground motion, volts, or counts, depending on where it is in the filter system.

5 A unit lookup key that refers to field 3 of the Units Abbreviation Blockette [34] for the stage's output signal. Analog filters usually emit volts, digital filters usually emit counts.

6 The number of response corner frequencies specified in the repeat block that follows.

7 The corner frequency.

8 The slope of the line to the right of the corner frequency, measured in db/decade.

[57] Decimation Blockette

Name: **Decimation Blockette**
Blockette Type: **057**
Control Header: **Station**
Field Station Volume: **Required for Digital Stage**
Station Oriented Network Volume: **Required for Digital Stage**
Event Oriented Network Volume: **Required for Digital Stage**

Many digital filtration schemes process a high sample rate data stream; filter; then decimate, to produce the desired output. Use this blockette to describe the decimation phase of the stage. You would usually place it between a Response (Coefficients) Blockette [54] and the Sensitivity/Gain Blockette [58] phases of the filtration stage of the channel. Include this blockette with non-decimated stages because you must still specify the time delay. (In this case, the decimation factor is 1 and the offset value is 0.)

Sample:

```
057005132Δ.0000E+02ΔΔΔΔ1ΔΔΔΔ0Δ0.0000E+00Δ0.0000E+00
```

Note	Field name	Type	Length	Mask or Flags
1	Blockette type — 057	D	3	"###"
2	Length of blockette	D	4	"####"
3	Stage sequence number	D	2	"##"
4	Input sample rate (Hz)	F	10	"#.####E-##"
5	Decimation factor	D	5	"#####"
6	Decimation offset	D	5	"#####"
7	Estimated delay (seconds)	F	11	"-#.####E-##"
8	Correction applied (seconds)	F	11	"-#.####E-##"

Notes for fields:

1 Standard blockette type identification number.

2 Length of the entire blockette, including the 7 bytes in fields 1 and 2.

3 The stage that is being decimated.

4 The incoming sample rate, in samples per second.

5 The decimation factor. When this number of samples are read in, one final sample comes out. Calculate the output sample rate by dividing field 4 by the decimation factor.

6 This field determines which sample is chosen for use. Make the value of this field greater than or equal to zero, but less than the decimation factor. If you pick the first sample, set this field to zero. If you pick the second sample, set it to 1, and so forth.

7 The estimated pure delay for the stage, it may or may not be also corrected in field 8. This field's value is nominal, and may be unreliable.

8 The time shift applied to the time tag due to delay at this stage of the filter; a negative number indicating the amount of time added to the former time tag. The actual delay is difficult to estimate, and the correction applied neglects dispersion. This field allows the user to know how much correction was used, in case a more accurate correction is to be applied later. A zero here implies no correction was done.

[58] Channel Sensitivity/Gain Blockette

Name: **Channel Sensitivity/Gain Blockette**
Blockette Type: **058**
Control Header: **Station**
Field Station Volume: **Required**
Station Oriented Network Volume: **Required**
Event Oriented Network Volume: **Required**

When used as a gain (stage ≠ 0), this blockette is the gain for this stage at the given frequency. Different stages may be at different frequencies. However, it is strongly recommended that the same frequency be used in all stages of a cascade, if possible. When used as a sensitivity(stage=0), this blockette is the sensitivity (in counts per ground motion) for the entire channel at a given frequency. The frequency here may be different from the frequencies in the gain specifications, but should be the same if possible. If you use cascading (more than one filter stage), then SEED requires a gain for each stage. A final sensitivity is optional. If you do not use cascading (only one stage), then SEED must see a gain, a sensitivity, or both.

Sample:

```
0580035Δ3Δ3.27680E+03Δ0.00000E+00Δ0
```

Note	Field name	Type	Length	Mask or Flags
1	Blockette type — 058	D	3	"###"
2	Length of blockette	D	4	"####"
3	Stage sequence number	D	2	"##"
4	Sensitivity/gain (Sd)	F	12	"-#.#####E-##"
5	Frequency (Hz) (fs)	F	12	"-#.#####E-##"
6	Number of history values	D	2	"##"
	REPEAT fields 7 — 9 for the Number of history values:			
7	Sensitivity for calibration	F	12	"-#.#####E-##"
8	Frequency of calibration (Hz)	F	12	"-#.#####E-##"
9	Time of above calibration	V	1—22	TIME

Notes for fields:

1 Standard blockette type identification number.

2 Length of the entire blockette, including the 7 bytes in fields 1 and 2. This blockette could exceed the maximum of 9999 characters, but it should never be necessary to record that much calibration history; usually a few values will suffice.

3 The stage for which this gain applies. If you set this number to zero, SEED will consider it a channel sensitivity value.

4 The gain (S_d) at this stage, or the sensitivity (S_d) for the channel (depending on the value in field 3).

5 The frequency (f_s) at which the value in 4 is correct.

6 You may record any number of standard calibration values for a history of the calculation of the sensitivity value (calibration methods usually only give information about the final channel response, not the individual stages). This field represents the number of calibration history entries that follow. If there is no history, or this is a gain value, put zero here and stop the blockette.

7 The recorded amplitude value of this history entry.

8 The frequency for this calibration; you can use a zero for a step calibration.

9 The time when the calibration was done.

[59] Channel Comment Blockette

Name: **Channel Comment Blockette**
Blockette Type: **059**
Control Header: **Station**
Field Station Volume: **Optional**
Station Oriented Network Volume: **Optional**
Event Oriented Network Volume: **Optional**

Sample:

```
05900351989,001~1989,004~4410000000
```

Note	Field name	Type	Length	Mask or Flags
1	Blockette type — 059	D	3	"###"
2	Length of blockette	D	4	"####"
3	Beginning effective time	V	1—22	TIME
4	End effective time	V	0—22	TIME
5	Comment code key	D	4	"####"
6	Comment level	D	6	"######"

Notes for fields:

1 Standard blockette type identification number.

2 Length of the entire blockette, including the 7 bytes in fields 1 and 2.

3 The time when the comment comes into effect.

4 The time when the comment is no longer in effect.

5 The comment code key (field 3) of the associated Comment Description Dictionary Blockette [31] in the abbreviation dictionary section.

6 The numeric value (if any) associated with the Units of comment level (field 6) in the same Comment Description Dictionary Blockette [31]. Together, this numeric value, its units, and the Description of comment (field 5) of the associated Comment Description Blockette [31], all describe a comment for the channel.

[60] Response Reference Blockette

Name: **Response Reference Blockette**
Blockette Type: **060**
Control Header: **Station**
Field Station Volume: **Optional**
Station Oriented Network Volume: **Optional**
Event Oriented Network Volume: **Optional**

Use this blockette whenever you want to replace blockettes [53] through [58] and [61] with their dictionary counterparts, blockettes [43] through [48] and [41]. We recommend placing responses in stage order, even if this means using more than one Response Reference Blockette [60]. Here is an example:

Stage 1: Response (Poles & Zeros) Blockette [53]
Channel Sensitivity/Gain Blockette [58]
First response reference blockette:
Response Reference Blockette [60]
Stage 2: [44] [47] [48]
Stage 3: [44] [47] [48]
Stage 4: [44] [47]
Channel Sensitivity/Gain Blockette [58]
Stage 5: Response (Coefficients) Blockette [54]
(End of first response reference blockette)
Second response reference blockette:
Response Reference Blockette [60]
Stage 5 (continued): [47] [48]
Stage 6: [44] [47] [48]
(End of second response reference blockette)

Substitute Response Reference Blockette [60] anywhere the original blockette would go, but be sure to place it in the same position as the original would have gone. (Note that this blockette uses a repeating field (response reference) *within* another repeating field (stage value). This is the only blockette in the current version (2.1) that has this "two dimensional" structure.)

Note	Field name	Type	Length	Mask or Flags
1	Blockette type — 060	D	3	"###"
2	Length of blockette	D	4	"####"
3	Number of stages	D	2	"##"
	REPEAT field 4, with appropriate fields 5 and 6, for each filter stage			
4	Stage sequence number	D	2	"##"
5	Number of responses	D	2	"##"
	REPEAT field 6, one for each response within each stage:			
6	Response lookup key	D	4	"####"

Notes for fields:

1 Standard blockette type identification number.

2 Length of the entire blockette, including the 7 bytes in fields 1 and 2.

3 The number of stages.

4 The stage sequence numbers — create one number (each number with a set of responses that follow) for each of the total number of stages.

5 The number of responses within a stage.

6 The unique response lookup key — note one key for each of the total number of responses.

[61] FIR Response Blockette

	Name:	**FIR Response Blockette**
	Blockette Type:	**061**
	Control Header:	**Channel Response**
	Field Station Volume:	**Some Response Required**
	Station Oriented Network Volume:	**Some Response Required**
	Event Oriented Network Volume:	**Some Response Required**
V2.2-	**Introduced in SEED Version**	**2.2**

The FIR blockette is used to specify FIR (Finite Impulse Response) digital filter coefficients. It is an alternative to blockette [54] when specifying FIR filters. The blockette recognizes the various forms of filter symmetry and can exploit them to reduce the number of factors specified to the blockette.

Note	Field name	Type	Length	Mask or Flags
1	Blockette type — 061	D	3	"###"
2	Length of blockette	D	4	"####"
3	Stage sequence number	D	2	"##"
4	Response Name	V	1 — 25	[UN_]
5	Symmetry Code	A	1	[U]
6	Signal In Units	D	3	"###"
7	Signal Out Units	D	3	"###"
8	Number of Coefficients	D	4	"####"
	REPEAT field 9 for the Number of Coefficients			
9	FIR Coefficient	F	14	"-#.#######E-##"

Notes for Field

1 Standard blockette type identification number.

2 Length of the entire blockette, inclusive of the 7 bytes in fields 1 and 2. This blockette could exceed the maximum of 9,999 characters. If so, continue on the next record. Field 4 should be set the same, but fields 5 — 7 in subsequent blockettes should be ignored.

3 The identifying number of the stage.

4 A descriptive name for the response.

5 The symmetry code. Designates how the factors will be specified. See the tables that follow to see examples of these different types of symmetry.

A — No Symmetry - all Coefficients are specified.

Example:	Coeff	Factor	Value
	1	1	-.113963588000E+03
	2	2	.654051896200E+02
	3	3	.293332365600E+03
	4	4	.682790540100E+03
	5	5	.119612218000E+04
	6	6	.184026419900E+04
	7	7	.263602733900E+04

B — Odd number Coefficients with symmetry

Example:	Coeff	Factor	Value
	1 & 25	1	- .113963588000E+03

2 & 24	2	.654051896200E+02
3 & 23	3	.293332365600E+03
4 & 22	4	.682790540100E+03
5 & 21	5	.119612218000E+04
6 & 20	6	.184026419900E+04
7 & 19	7	.263602733900E+04
8 & 18	8	.348431282900E+04
9 & 17	9	.481917329000E+04
10& 16	10	.549205395300E+04
11& 15	11	.605889892900E+04
12& 14	12	.631358277400E+04
13	13	.234002034020E+03

C — Even number Coefficients with symmetry.

Example:	Coeff	Factor	Value
	1 & 24	1	-.113963588000E+03
	2 & 23	2	.654051896200E+02
	3 & 22	3	.293332365600E+03
	4 & 21	4	.682790540100E+03
	5 & 20	5	.119612218000E+04
	6 & 19	6	.184026419900E+04
	7 & 18	7	.263602733900E+04
	8 & 17	8	.348431282900E+04
	9 & 16	9	.481917329000E+04
	10& 15	10	.549205395300E+04
	11& 14	11	.605889892900E+04
	12& 13	12	.631358277400E+04

6 A Unit Lookup Key that refers to the Units Abbreviation Blockette [34], field 3, for the units for the incoming signal to this stage of the filter. Will usually be ground motion, volts, or counts, depending on where in the filter system it is.

7 Like field 6, but for the stages output signal. Analog filters usually output volts, digital filters output counts.

8 The number of factors that follow.

A No Symmetry — All Coefficients specified

$f = c$. "f" denotes number of factors, "c" is number of coefficients

B Odd — First half of all coefficients and center coefficient specified

$$f = \frac{c + 1}{2}$$

C Even — First half of all coefficients specified

$$f = \frac{c}{2}$$

9 FIR Filter Coefficients.

[62] Response [Polynomial] Blockette

	Name:	**Response (Polynomial) Blockette**
	Blockette Type:	**062**
	Control Header:	**Channel Response**
	Field Station Volume:	**Some Response Required**
	Station Oriented Network Volume:	**Some Response Required**
	Event Oriented Network Volume:	**Some Response Required**
V2.3-	**Introduced in SEED Version**	**2.3**

Use this blockette to characterize the response of a non-linear sensor. The polynomial response blockette describes the output of an Earth sensor in fundamentally a different manner than the other response blockettes. The functional describing the sensor for the polynomial response blockette will have Earth units while the independent variable of the function will be in volts. This is precisely opposite to the other response blockettes. While it is a simple matter to convert a linear response to either form, the non-linear response (which we can describe in the polynomial blockette) would require extensive curve fitting or polynomial inversion to convert from one function to the other. Most data users are interested in knowing the sensor output in Earth units, and the polynomial response blockette facilitates the access to Earth units for sensors with non-linear responses.

Note	Field name	Type	Length	Mask or Flags
1	Blockette type — 062	D	3	“###”
2	Length of blockette	D	4	“####”
3	Transfer Function Type	A	1	[U]
4	Stage Sequence Number	D	2	“##”
5	Stage Signal Input Units	D	3	“###”
6	Stage Signal Output Units	D	3	“###”
7	Polynomial Approximation Type	A	1	[U]
8	Valid Frequency Units	A	1	[U]
9	Lower Valid Frequency Bound	F	12	“-#.#####E-##”
10	Upper Valid Frequency Bound	F	12	“-#.#####E-##”
11	Lower Bound of Approximation	F	12	“-#.#####E-##”
12	Upper Bound of Approximation	F	12	“-#.#####E-##”
13	Maximum Absolute Error	F	12	“-#.#####E-##”
14	Number of Polynomial Coefficients	D	3	“###”
	(REPEAT fields 15 and 16 for each polynomial coefficient)			
15	Polynomial Coefficient	F	12	“-#.#####E-##”
16	Polynomial Coefficient Error	F	12	“-#.#####E-##”

Notes for Fields.

1 Standard blockette type identification number.

2 Length of the entire blockette, including the 7 bytes in fields 1 and 2.

3 A single letter “P” describing the type of stage.

4 The identifying number of this stage.

5 A unit lookup key that refers to field 3 of the Units Abbreviation Blockette [34] for the units of the incoming signal to this stage of the filter.

6 A unit lookup key that refers to field 3 of the Units Abbreviation Blockette [34] for the stages output signal.

7 A single character describing the type of polynomial approximation (this field is mandatory): (Note: The input units (x) into the polynomial will most always be in Volts. The output units [pn(x)] will be in the units of field 5.)

M - MacLaurin

pn(x)=a0+a1*x+a2*x^2+...+an*x^n

(Note: The following three fields play no part in the calculation to recover Earth units [i.e., field 5] for this response. If these fields are available from the instrumentation literature, they can be used in post-processing to assess the frequency domain validity.

8 A single character describing valid frequency units:

"A" — rad/sec

"B" — Hz

9 If available, the low frequency corner for which the sensor is valid. 0.0 if unknown or zero.

10 If available, the high frequency corner for which the sensor is valid. Nyquist if unknown.

11 Lower bound of approximation. This should be in units of 5.

12 Upper bound of approximation. This should be in units of 5.

13 The maximum absolute error of the polynomial approximation. Put 0.0 if the value is unknown or actually zero.

14 The number of coefficients that follow in the polynomial approximation. The polynomial coefficients are given lowest order first and the number of coefficients is one more than the degree of the polynomial.

15 The value of the polynomial coefficient.

16 The error for field 12. Put 0.0 here if the value is unknown or actually zero. This error should be listed as a positive value, but represent a +/- error (i.e. 2 standard deviations).

Examples:

1. Polynomial representation of the pressure response of a Setra Model 270 Pressure Transducer.

The Setra Model 270 Pressure Transducer is listed as valid between 600 mbar and 1100 mbar with a nominal output of 0-5 volts and it is presumed to be linear with respect to pressure. I haven't found any error representation. No frequency bounds are given for the transducer.

$$pn(x) = a0 + a1*x$$

where x = voltage, and pn(x) = pressure

Using 0 volts and then 5 volts input with the pressure range, we get:

a0 = 600 a0 error = 0.0

a1 = 100 a1 error = 0.0

Sample voltage to pressure conversion:

Volts(x)	Pressure (mbar) pn(x)
0.0	600
1.0	700
2.0	800
3.0	900
4.0	1000
5.0	1100

Bound Values for polynomial:

Lower	600 mbar
Upper	1100 mbar

Assume we use an 8 bit digitizer where 0 counts = 0 volts and 255 counts = 5 volts. This translates to a digitizer gain of 51 Counts/volt.

This provides the following conversion from counts to pressure:

Counts	Volts (x) gain*counts	Pressure (mbar) pn(x)
0	0.0	600
51	1.0	700
102	2.0	800
153	3.0	900
204	4.0	1000
255	5.0	1100

The Polynomial Blockette representation for this sensor:

Field name	Type	Length	Mask or Flags
Blockette type	D	3	“062”
Length of blockette	D	4	“129”
Transfer Function Type	A	1	“P”
Stage Sequence Number	D	2	“??”
Stage Signal Input Units	D	3	“???”
Stage Signal Output Units	D	3	“???”
Polynomial Approximation Type	A	1	“M”
Valid Frequency Units	A	1	“B”
Lower Valid Frequency Bound	F	12	“0.00000E+00”
Upper Valid Frequency Bound	F	12	“0.00000E+00”
Lower Bound of Approximation	F	12	“ 6.00000E+02”
Upper Bound of Approximation	F	12	“ 1.10000E+03”
Maximum Absolute Error	F	12	“0.00000E+00”
Number of Polynomial Coefficients	D	3	“ 2”
a0 Polynomial Coefficient	F	12	“ 6.00000E+02”
a0 Polynomial Coefficient Error	F	12	“ 0.00000E+00”
a1 Polynomial Coefficient			“1.00000E+02
a1 Polynomial Coefficient Error			“ 0.00000E+00”

2. Polynomial representation of the temperature response of a thermistor.

In order to sense the temperature of the Berkeley Digital Seismic Network (BDSN) seismometers, we use a Yellow Springs Instrument Co. (YSI) 44031 thermistor. To convert the thermistor resistance to a usable voltage signal, we have installed the thermistor into a bridge amplification circuit. The calibrated response is:

Voltage	Temperature
-2.00	-5.02
-1.90	-4.43
-1.80	-3.81
-1.70	-3.18
-1.60	-2.53
-1.50	-1.85
-1.40	-1.15
-1.30	-0.43
-1.20	0.32
-1.10	1.10
-1.00	1.92
-0.90	2.76
-0.80	3.64
-0.70	4.56
-0.60	5.52
-0.50	6.53
-0.40	7.60
-0.30	8.72
-0.20	9.90
-0.10	11.15
0.00	12.49
0.10	13.91
0.20	15.43
0.30	17.07
0.40	18.85
0.50	20.78
0.60	22.91
0.70	25.25
0.80	27.88
0.90	30.85
1.00	34.26
1.10	38.26
1.20	43.07
1.30	49.09
1.40	57.04
1.50	68.59

The resistance of the thermistor is a non-linear function of the temperature and its response can be described by a polynomial. YSI claims that all 44031 thermistors fall within 0.2 degrees C of the nominal response so we want to model the response to at least an accuracy of 0.2 degrees C. This temperature response can be adequately represented by a McLaurin polynomial of the form:

$$pn(x) = a0 + a1*x + a2*x^2 + ... + an*x^n$$

where x is the voltage and pn(x) is the temperature. The coefficients required to approximate the above temperature-voltage data to the desired accuracy are:

Coefficient	Value	Error
a0	0.12505E+02	0.14223E-03
a1	0.13824E+02	0.22350E-02
a2	0.41039E+01	0.86810E-02
a3	0.12932E+01	0.44581E-01
a4	0.18741E+01	0.34469E-01
a5	0.17250E+01	0.91194E-01
a6	-0.61021E+00	0.22029E-01
a7	-0.10540E+01	0.28643E-01
a8	0.13974E+00	0.39675E-02
a9	0.39061E+00	0.10566E-02
a10	0.95345E-01	0.15096E-03

The maximum error of the polynomial representation is -0.072 degrees C at a temperature of 57 degrees C and the temperature bounds are:

Bound	Value
Lower	-5.02
Upper	68.59

The parameter that is the most difficult to quantify is the frequency response of the thermistor. YSI states that the thermistor time constant varies from second in well-stirred oil to 10 seconds in still air. We have encapsulated the thermistor and its leads in heat-shrink tubing for protection from mechanical damage and this has the effect of lengthening the thermistor time constant. We estimate the thermal time constant of the thermistor to be of order 20 seconds. However, the thermistor assembly is placed inside a heavily insulated BDSN pier and seismometer enclosure that has a thermal time constant of order several hours to a few days. Thus the temperature sensed by the thermistor varies so slowly that the thermistor time constant is not significant.

The Polynomial Blockette representation for this sensor:

Field name	Type	Length	Mask or Flags
Blockette type	D	3	“062”
Length of blockette	D	4	“345”
Transfer Function Type	A	1	“P”
Stage Sequence Number	D	2	“??”
Stage Signal Input Units	D	3	“???”
Stage Signal Output Units	D	3	“???”
Polynomial Approximation Type	A	1	“M”
Valid Frequency Units	A	1	“B”
Lower Valid Frequency Bound	F	12	“ 0.00000E+00”
Upper Valid Frequency Bound	F	12	“ 0.10000E-01”
Lower Bound of Approximation	F	12	“-5.0200E-00”
Upper Bound of Approximation	F	12	“ 6.85900E+01”
Maximum Absolute Error	F	12	“ 0.72000E-01”
Number of Polynomial Coefficients	D	3	“ 11”
a0 Polynomial Coefficient	F	12	“ 0.12505E+02”
a0 Polynomial Coefficient Error	F	12	“ 0.14223E-03”
a1 Polynomial Coefficient	F	12	“ 0.13824E+02”
a1 Polynomial Coefficient Error	F	12	“ 0.22350E-02”
a2 Polynomial Coefficient	F	12	“ 0.41039E+01”
a2 Polynomial Coefficient Error	F	12	“ 0.86810E-02”
a3 Polynomial Coefficient	F	12	“ 0.12932E+01”
a3 Polynomial Coefficient Error	F	12	“ 0.44581E-01”
a4 Polynomial Coefficient	F	12	“ 0.18741E+01”
a4 Polynomial Coefficient Error	F	12	“ 0.34469E-01”
a5 Polynomial Coefficient	F	12	“ 0.17250E+01”
a5 Polynomial Coefficient Error	F	12	“ 0.91194E-01”
a6 Polynomial Coefficient	F	12	“-0.61021E+00”
a6 Polynomial Coefficient Error	F	12	“ 0.22029E-01”
a7 Polynomial Coefficient	F	12	“-0.10540E+01”
a7 Polynomial Coefficient Error	F	12	“ 0.28643E-01”
a8 Polynomial Coefficient	F	12	“ 0.13974E+00”
a8 Polynomial Coefficient Error	F	12	“ 0.39675E-02”
a9 Polynomial Coefficient	F	12	“ 0.39061E+00”
a9 Polynomial Coefficient Error	F	12	“ 0.10566E-02”
a10 Polynomial Coefficient	F	12	“ 0.95345E-01”
a10 Polynomial Coefficient Error	F	12	“ 0.15096E-03”

Chapter 7

Time Span Control Headers

Station oriented and event oriented network volumes use time span headers to index the actual data records. Stations are recorded sequentially, and within each station the channels are recorded sequentially. Field station recordings do not use these headers.

For event oriented network volumes, the time span refers to the recording time that begins before and ends after a geophysical event. These events may use blockettes to describe the hypocenter and phase arrivals measured at each station.

For station oriented network volumes, the time span refers to a standard interval for the whole volume or for each day on the volume. Either place a set of time span identification records and index records before the data for each time span, or, preferably, place all the time span control headers at the beginning of the volume to increase efficiency at extracting subsets of data. This latter method, implemented in version 2.1 of SEED, allows a reading program to easily uncover what data is on the volume and exactly where it is, without having to read through all the data. Because either method can be used, reading programs should never assume the order, but instead search for the time span control headers at the locations described by the Volume Time Span Index Blockette [12], and search for the data at the locations described by the Time Span Data Start Index Blockette [73] or the Time Series Index Blockette [74].

Versions 2.1 and later place all time span control headers together, after station control headers, and before data. Version 2.0 placed time spans together, with each time span consisting first of a time span control header, followed by its data. To allow for upward compatibility, SEED reading programs should be able to read both versions, but SEED writing programs should always write using the latest version of the format.

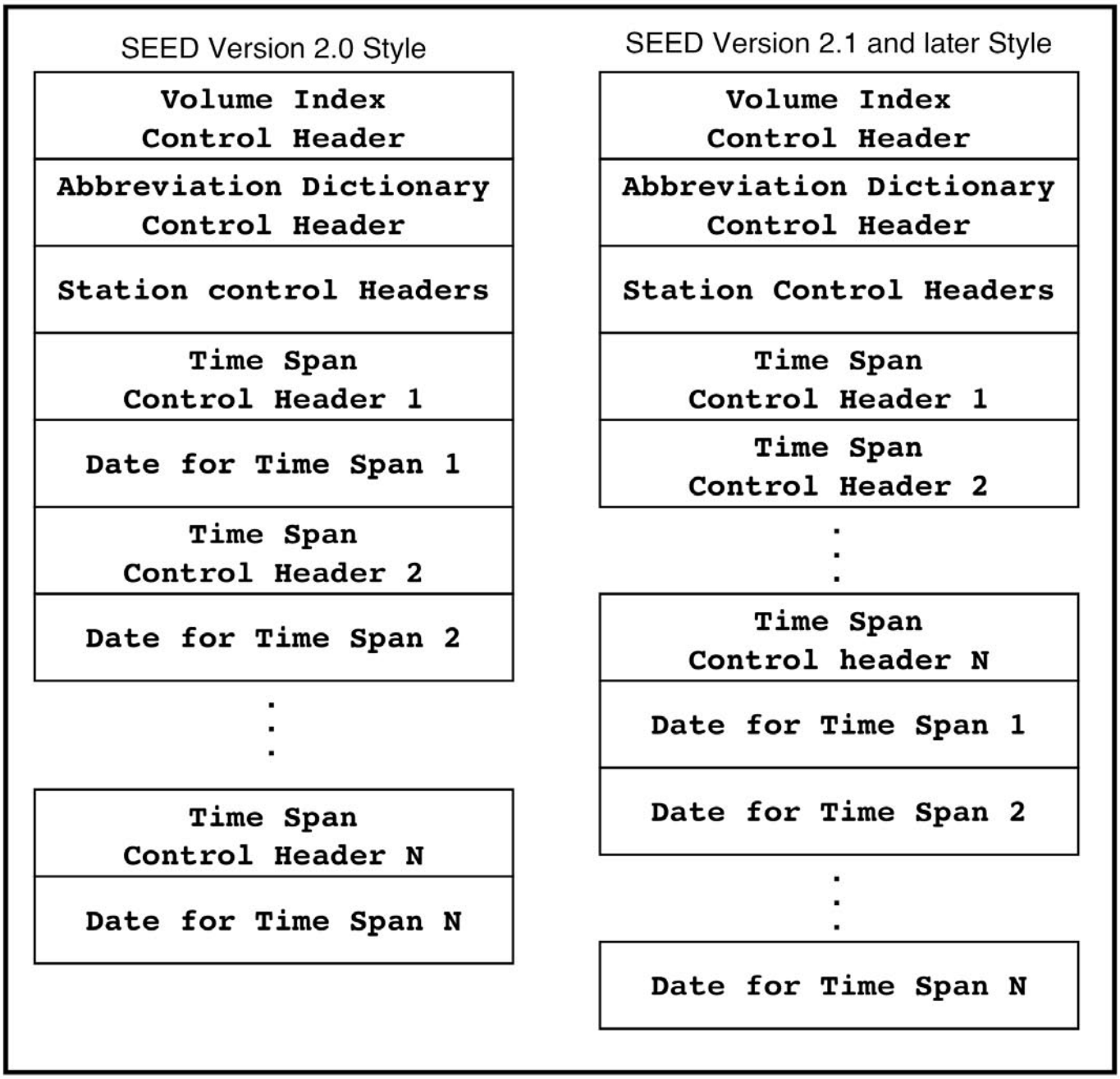


Figure12: Alternate time Span Header Placements

[70] Time Span Identifier Blockette

Name: **Time Span Identifier Blockette**
Blockette Type: **070**
Control Header: **Time Span**
Field Station Volume: **Not Applicable**
Station Oriented Network Volume: **Required**
Event Oriented Network Volume: **Required**

Use this blockette to describe the time series—when it starts, when it ends, and whether its data are event oriented or not.

Sample:

```
0700054P1989,003,00:00:00.0000~1989,004,00:00:00.0000~
```

Note	Field name	Type	Length	Mask or Flags
1	Blockette type — 070	D	3	"###"
2	Length of blockette	D	4	"####"
3	Time span flag	A	1	[U]
4	Beginning time of data span	V	1—22	TIME
5	End time of data span	V	1—22	TIME

Notes for fields:

1 Standard blockette type identification number.

2 Length of the entire blockette, including the 7 bytes in fields 1 and 2.

3 Indicates whether the data are for a station network volume's accounting period or for an event network volume.

E — Data are event oriented

P — Data are for a given period

4 The time when this time span begins.

5 The time when this time span ends.

[71] Hypocenter Information Blockette

Name: **Hypocenter Information Blockette**
Blockette Type: **071**
Control Header: **Time Span**
Field Station Volume: **Not Applicable**
Station Oriented Network Volume: **Optional**
Event Oriented Network Volume: **Desirable**

Use this blockette to include hypocenter subsidiary information.

	Note	Field name	Type	Length	Mask or Flags
	1	Blockette type — 071	D	3	“###”
	2	Length of blockette	D	4	“####”
	3	Origin time of event	V	1—22	TIME
	4	Hypocenter source identifier	D	2	“##”
	5	Latitude of event (degrees)	D	10	“-##.######”
	6	Longitude of event (degrees)	D	11	“-###.######”
	7	Depth (Km)	D	7	“####.##”
	8	Number of magnitudes	D	2	“##”
		REPEAT fields 9 — 11 for the Number of magnitudes:			
	9	Magnitude	D	5	“##.##”
	10	Magnitude type	V	1—10	[UNLPS]
	11	Magnitude source	D	2	“##”
V2.3 -	12	Seismic region	D	3	“###”
V2.3 -	13	Seismic Location	D	4	“####”
V2.3 -	14	Region Name	V	1— 40	[UNLPS]

Notes for fields:

1 Standard blockette type identification number.

2 Length of the entire blockette, including the 7 bytes in fields 1 and 2.

3 The event’s origin time.

4 The source quoted for the hypocenter information. Use a source lookup code key that refers to field 3 of the Cited Source Dictionary Blockette [32], where the source reference is located.

5 The event’s latitude (negative is south).

6 The event’s longitude (negative is west).

7 Depth in kilometers.

8 The number of magnitude listings that follow.

9 The magnitude value.

10 The magnitude type (MB, Msz, etc.)

11 Reference for the magnitude. This field may be set to zero if it is the same as the value in field 4, above. Otherwise, use a source lookup code key that refers to field 3 of the Cited Source Dictionary Blockette [32], where the source reference is located.

12 The Flinn-Engdahl seismic geographic region number. This is from a table of numeric codings for general geographic Earth regions. This is a number from (currently) 1 to 50. See Appendix K.

13 The Flinn-Engdahl seismic location number. Refers to Earth place names. This is a number from (currently) 1 to 729. See Appendix K.

14 The Flinn-Engdahl standard place name. See Appendix K.

NOTE: For station oriented network volumes, you may include a list of event hypocenters for the time interval, or a list of manually or automatically determined arrival times.

[72] Event Phases Blockette

Name: **Event Phases Blockette**
Blockette Type: **072**
Control Header: **Time Span**
Field Station Volume: **Not Applicable**
Station Oriented Network Volume: **Optional**
Event Oriented Network Volume: **Desirable**

This blockette lists the phase arrivals at the different stations. A large number of these are frequently placed within a SEED volume.

	Note	Field name	Type	Length	Mask or Flags
	1	Blockette type — 072	D	3	"###"
	2	Length of blockette	D	4	"####"
	3	Station identifier	A	5	[UN]
	4	Location identifier	A	2	[UN]
	5	Channel identifier	A	3	[UN]
	6	Arrival time of phase	V	1—22	TIME
	7	Amplitude of signal	F	10	"#.####E-##"
	8	Period of signal (seconds)	F	10	"#.####E-##"
	9	Signal-to-noise ratio	F	10	"#.####E-##"
	10	Name of phase	V	1—20	[UNLP]
V2.3 -	11	Source	D	2	"##"
V2.3 -	12	Network Code	A	2	[ULN]

Notes for fields:

1 Standard blockette type identification number.

2 Length of the entire blockette, including the 7 bytes in fields 1 and 2.

3 Standard station identifier.

4 Standard location name.

5 Standard channel identifier (see Appendix A).

6 Phase arrival time at the station.

7 The amplitude of the pick, usually measured in the same ground motion units as the channel. See field 8 in the Channel Identifier Blockette [52] for the units of the channel's ground motion.

8 Signal period.

9 The signal's signal-to-noise ratio, if known; otherwise, set to 0.0.

10 The standard name of the phase. Station event pickers can use the "P" phase.

11 Reference for source of the phase pick. Refers to blockette [32], field 3.

12 The two character identifier that identifies the network operator. See Appendix J for the current list of Network Codes.

[73] Time Span Data Start Index Blockette

Name: **Time Span Data Start Index Blockette**
Blockette Type: **073**
Control Header: **Time Span**
Field Station Volume: **Not Applicable**
Station Oriented Network Volume: **Required in V 2.0; superseded by 074 in V 2.1**
Event Oriented Network Volume: **Required in V 2.0; superseded by 074 in V 2.1**

This blockette stores information about the different time series in the time span. There is usually one index entry here for each time span encountered. (NOTE: This blockette was used by SEED 2.0. Current SEED writing programs should not use this blockette, but reading programs should allow for it.)

Sample:

```
07300710002BJIΔΔΔΔVPE1989,003,00:00:09.4400~00029701ΔΔΔΔΔΔΔΔΔΔ~00030101
```

Note	Field name	Type	Length	Mask or Flags
1	Blockette type — 073	D	3	“###”
2	Length of blockette	D	4	“####”
3	Number of data pieces	D	4	“####”
	REPEAT fields 4 — 9 for the Number of data pieces:			
4	Station identifier of data piece	A	5	[UN]
5	Location identifier	A	2	[UN]
6	Channel identifier	A	3	[UN]
7	Time of record	V	1—22	TIME
8	Sequence number of first record	D	6	“######”
9	Sub-sequence number	D	2	“##”

Notes for fields:

1 Standard blockette type identification number.

2 Length of the entire blockette, including the 7 bytes in fields 1 and 2. This blockette may exceed the maximum length of 9999 bytes when there are time problems at the station or when event pickers are too sensitive. SEED writing programs should stop the blockette before it reaches 9999 bytes and start on a new one; SEED reading programs should expect to see multiple blockettes.

3 The number of index entries in the following repeat block.

4 Standard station identifier (see Appendix G).

5 Standard location identifier.

6 Standard channel identifier (see Appendix A).

7 The time when this time series starts (same as the start time for the record).

8 The data sequence number of the data record.

9 The sub-sequence number, used when the data record size is less than the logical record size. Number the first data record in the logical record “1.”

NOTE: Add a trailing record with a blank station and channel, to indicate the record number of the last data record plus one. SEED needs this done so it can compute the length of the last indexed data segment.

[74] Time Series Index Blockette

Name: **Time Series Index Blockette**
Blockette Type: **074**
Control Header: **Time Span**
Field Station Volume: **Not Applicable**
Station Oriented Network Volume: **Required**
Event Oriented Network Volume: **Required**

This blockette replaces the Time Span Data Start Index Blockette [73], and allows version 2.1 and later of SEED to correctly document time tears, events, and time indexes. There should be one Time Series Index Blockette [74] for each continuous time series and/or each station/ channel combination in the time span. This blockette provides indices and times of both the beginning and end of the time series described. Writing programs can also provide indices and times of intervening records to speed direct access — particularly useful for compressed data.

Sample:

```
0740084BJIΔΔΔΔBHZ1992,001,20:18:54.5700~003217011992,001,20:29:36.7200~00322301000CD
```

	Note	Field name	Type	Length	Mask or Flags
	1	Blockette type — 074	D	3	"###"
	2	Length of blockette	D	4	"####"
	3	Station identifier	A	5	[UN]
	4	Location identifier	A	2	[UN]
	5	Channel identifier	A	3	[UN]
	6	Series start time	V	1—22	TIME
	7	Sequence number of first data	D	6	"######"
	8	Sub-sequence number	D	2	"##"
	9	Series end time	V	1—22	TIME
	10	Sequence number of last record	D	6	"######"
	11	Sub-sequence number	D	2	"##"
	12	Number of accelerator repeats	D	3	"###"
	REPEAT fields 13 — 15 for the Number of accelerator repeats:				
	13	Record start time	V	1—22	TIME
	14	Sequence number of record	D	6	"######"
	15	Sub-sequence number	D	2	"##"
V2.3 -	16	Network Code	A	2	[ULN]

Notes for fields:

1 Standard blockette type identification number.

2 Length of the entire blockette, including the 7 bytes in fields 1 and 2. This blockette may exceed the maximum length of 9999 bytes when the number of accelerator indices becomes large. SEED writing programs should stop the blockette before it reaches 9999 bytes and start on a new one; SEED reading programs should expect to see multiple blockettes.

3 Standard station identifier (see Appendix G).

4 Standard location identifier.

5 Standard channel identifier (see Appendix A).

6 The time when this time series starts (same as the start time for the first record).

7 The sequence number index to the start of the data.

8 The sub-sequence number, used when the data record size is less than the logical record size. Number the first data record in the logical record "1."

9 The time when this time series ends (same as the end time for the last record).

10 The sequence number index to the last record of the data.

11 The sub-sequence number that refers to the last of the data.

12 The number of accelerator indices in the time series. An accelerator index is an indexed intervening record, placed somewhere between the start and end of a time series. SEED reading programs use accelerator indices to quickly access any record in the series: first, these programs use an accelerator index to access the data record; then, they perform a sequential search of the data records to find the final, target data record. We suggest placing an accelerator index record every 32 records.

13 The start time of the record to which the accelerator index refers.

14 The data record's sequence number to which the accelerator index refers.

15 The data record's sub-sequence number to which the accelerator index refers.

16 The two character identifier that identifies the network operator. See Appendix J for the current list of Network Codes.

NOTE: Accelerator indices are redundant for uncompressed data, because the appropriate record number can be easily computed.

Chapter 8

Data Records

Each data record in a SEED volume consists of a fixed header, followed by optional data blockettes in the variable header. These data blockettes are structurally different from control header blockettes: they are binary, and they are found only in data records, just after the fixed header. (The control header blockettes listed in the previous chapters of this reference manual — describing volume, dictionary abbreviation, station, and time span information — are ASCII formatted, and are found only in control headers. Do not confuse these two types of blockettes.)

Fixed Section of Data Header (48 bytes)

The data record header starts at the first byte. The next eight bytes follow the same structure as the control headers. Byte seven contains an ASCII "D," indicating it is a data record. (The eighth byte, or third field, is always an ASCII space — shown here as a "Δ"). The next ten bytes contain the station, location, and channel identity of the record. The rest of the header section is binary.

	Note	Field name	Type	Length	Mask or Flags
	1	Sequence number	A	6	"######"
V2.4 -	2	Data header/quality indicator ("D"|"R"|"Q")	A	1	
	3	Reserved byte ("Δ")	A	1	
	4	Station identifier code	A	5	[UN]
	5	Location identifier	A	2	[UN]
	6	Channel identifier	A	3	[UN]
V2.3 -	7	Network Code	A	2	[ULN]
	8	Record start time	B	10	
	9	Number of samples	B	2	
	10	Sample rate factor	B	2	
	11	Sample rate multiplier	B	2	
	12	Activity flags	B	1	
	13	I/O and clock flags	B	1	
	14	Data quality flags	B	1	
	15	Number of blockettes that follow	B	1	
	16	Time correction	B	4	
	17	Beginning of data	B	2	
	18	First blockette	B	2	

Notes for fields: * indicates mandatory information

1 * Data record sequence number (Format "######").

2 * "D" or "R" or "Q" — Data header/quality indicator. Previously, this field was only allowed to be "D" and was only used to indicate that this is a data header. As of SEED version 2.4 the meaning of this field has been extended to also indicate the level of quality control that has been applied to the record.

D —The state of quality control of the data is indeterminate.

R — Raw Waveform Data with no Quality Control

Q — Quality Controlled Data, some processes have been applied to the data.

3 Space (ASCII 32) — Reserved; do not use this byte.

4 * Station identifier designation (see Appendix G). Left justify and pad with spaces.

5 * Location identifier designation. Left justify and pad with spaces.

6 * Channel identifier designation (see Appendix A). Left justify and pad with spaces.

7 * A two character alphanumeric identifier that uniquely identifies the network operator responsible for the data logger. This identifier is assigned by the IRIS Data Management Center in consultation with the FDSN working group on the SEED format.

8 * BTIME: Start time of record.

9 * UWORD: Number of samples in record.

10 * WORD: Sample rate factor:

> 0 — Samples/second

< 0 — Seconds/sample

= 0 — Seconds/sample Use this for ASCII/OPAQUE DATA records

11 * WORD: Sample rate multiplier:

>0 — Multiplication factor

<0 — Division factor

12 UBYTE: Activity flags:

[Bit 0] — Calibration signals present

* [Bit 1] — Time correction applied. Set this bit to 1 if the time correction in field 16 has been applied to field 8. Set this bit to 0 if the time correction in field 16 has not been applied to field 8.

[Bit 2] — Beginning of an event, station trigger

[Bit 3] — End of the event, station detriggers

[Bit 4] — A positive leap second happened during this record (A 61 second minute).

[Bit 5] — A negative leap second happened during this record (A 59 second minute). A negative leap second clock correction has not yet been used, but the U.S. National Bureau of Standards has said that it might be necessary someday.

[Bit 6] — Event in progress

13 UBYTE: I/O flags and clock flags:

[Bit 0] — Station volume parity error possibly present

[Bit 1] — Long record read (possibly no problem)

[Bit 2] — Short record read (record padded)

[Bit 3] — Start of time series

[Bit 4] — End of time series

[Bit 5] — Clock locked

14 UBYTE: Data quality flags

[Bit 0] — Amplifier saturation detected (station dependent)

[Bit 1] — Digitizer clipping detected

[Bit 2] — Spikes detected

[Bit 3] — Glitches detected

[Bit 4] — Missing/padded data present

[Bit 5] — Telemetry synchronization error

[Bit 6] — A digital filter may be charging

[Bit 7] — Time tag is questionable

15 * UBYTE: Total number of blockettes that follow.

16 * LONG: Time correction. This field contains a value that may modify the field 8 record start time. Depending on the setting of bit 1 in field 12, the record start time may have already been adjusted. The units are in 0.0001 seconds.

17 * UWORD: Offset in bytes to the beginning of data. The first byte of the data records is byte 0.

18 * UWORD: Offset in bytes to the first data blockette in this data record. Enter 0 if there are no data blockettes. The first byte in the data record is byte offset 0.

NOTE: All unused bits in the flag bytes are reserved and must be set to zero.

The last word defines the length of the fixed header. The next-to-last word fixes the length reserved for the entire header.

If glitches (missing samples) are detected, set bit 3 of the data quality flags, and code missing data values as the largest possible value. Do this for any data format, even if you are using the Steim Compression algorithm.

Here is an algorithm and some sample rate combinations that describe how the sample rate factors and multipliers work:

If Sample rate factor > 0 and Sample rate Multiplier > 0,

Then nominal Sample rate = Sample rate factor X Sample rate multiplier

If Sample rate factor > 0 and Sample rate Multiplier < 0,

Then nominal Sample rate = -1 X Sample rate factor / Sample rate multiplier

If Sample rate factor < 0 and Sample rate Multiplier > 0,

Then nominal Sample rate = -1 X Sample rate multiplier / Sample rate factor

If Sample rate factor < 0 and Sample rate Multiplier < 0,

Then nominal Sample rate = 1/ (Sample rate factor X Sample rate multiplier)

Sample rate	Sample rate factor	Sample rate multiplier
330 SPS	33	10
	330	1
330.6 SPS	3306	-10
1 SP Min	-60	1
0.1 SPS	1	-10
	-10	1
	-1	-10

[100] Sample Rate Blockette (12 bytes)

V 2.3 – Introduced in SEED Version 2.3

Note	Field name	Type	Length	Mask or Flags
1	Blockette type — 100	B	2	
2	Next blockette's byte number	B	2	
3	Actual Sample Rate	B	4	
4	Flags (to be defined)	B	1	
5	Reserved byte	B	3	

Notes for fields:

1 UWORD: Blockette type (100): sample rate.

2 UWORD: Byte number of next blockette. (Calculate this as the byte offset from the beginning of the logical record — including the fixed section of the data header; use 0 if no more blockettes will follow.)

3 FLOAT: Actual sample rate of this data block.

4 BYTE: Flags (to be defined)

5 UBYTE: Reserved; do not use.

[200] Generic Event Detection Blockette (52 bytes)

Note	Field name	Type	Length	Mask or Flags
1	Blockette type — 200	B	2	
2	Next blockette's byte number	B	2	
3	Signal amplitude	B	4	
4	Signal period	B	4	
5	Background estimate	B	4	
6	Event detection flags	B	1	
7	Reserved byte	B	1	
8	Signal onset time	B	10	
9	Detector Name	A	24	

Notes for fields:

1 UWORD: Blockette type [200]: event detection information.

2 UWORD: Byte number of next blockette. (Calculate this as the byte offset from the beginning of the logical record — including the fixed section of the data header; use 0 if no more blockettes will follow.)

3 FLOAT: Amplitude of signal (for units, see event detection flags, below; 0 if unknown).

4 FLOAT: Period of signal, in seconds (0 if unknown).

5 FLOAT: Background estimate (for units, see event detection flags, below; 0 if unknown).

6 UBYTE: Event detection flags:

[Bit 0] — If set: dilatation wave; if unset: compression

[Bit 1] — If set: units above are after deconvolution

(see Channel Identifier Blockette [52], field 8); if unset: digital counts

[Bit 2] — When set, bit 0 is undetermined

[Other bits reserved and must be zero.]

7 UBYTE: Reserved; do not use.

8 BTIME: Time of the onset of the signal.

9 CHAR*24: The name of the event detector.

[201] Murdock Event Detection Blockette (60 bytes)

Note	Field name	Type	Length	Mask or Flags
1	Blockette type — 201	B	2	
2	Next blockette's byte number	B	2	
3	Signal amplitude	B	4	
4	Signal period	B	4	
5	Background estimate	B	4	
6	Event detection flags	B	1	
7	Reserved byte	B	1	
8	Signal onset time	B	10	
9	Signal-to-noise ratio values	B	6	
10	Lookback value	B	1	
11	Pick algorithm	B	1	
12	Detector name	A	24	

Notes for fields:

1 UWORD: Blockette type [201]: event detection information.

2 UWORD: Byte number of next blockette (0 if no more).

3 FLOAT: Amplitude of signal (in counts).

4 FLOAT: Period of signal (in seconds).

5 FLOAT: Background estimate (in counts).

6 UBYTE: Event detection flags:

[Bit 0] — If set: dilatation wave; if unset: compression

[Other bits reserved and must be zero.]

7 UBYTE: Reserved; do not use.

8 BTIME: Onset time of the signal.

9 UBYTE*6: Signal-to-noise ratio values.

10 UBYTE: Lookback value (0,1,2).

11 UBYTE: Pick algorithm (0,1).

12 CHAR*24: The name of the event detector.

NOTE: See Murdock (1983) and Murdock (1987) for more information on this type of event detector, and on what the fields listed above should contain.

[202] Log-Z Event Detection Blockette (reserved)

See Blandford (1981) for more information on this type of event detector. As of this printing, no blockette layout for this detector has been created.

[300] Step Calibration Blockette (60 bytes)

Note	Field name	Type	Length	Mask or Flags
1	Blockette type — 300	B	2	
2	Next blockette's byte number	B	2	
3	Beginning of calibration time	B	10	
4	Number of step calibrations	B	1	
5	Calibration flags	B	1	
6	Step duration	B	4	
7	Interval duration	B	4	
8	Calibration signal amplitude	B	4	
9	Channel with calibration input	A	3	
10	Reserved byte	B	1	
11	Reference amplitude	B	4	
12	Coupling	A	12	
13	Rolloff	A	12	

Notes for fields:

1 UWORD: Blockette type [300]: step calibration.

2 UWORD: Byte number of next blockette (0 if no more).

3 BTIME: Beginning time of calibration.

4 UBYTE: Number of step calibrations in sequence.

5 UBYTE : Calibration flags:

[Bit 0] — If set: first pulse is positive

[Bit 1] — If set: calibration's alternate sign

[Bit 2] — If set: calibration was automatic; if unset: manual

[Bit 3] — If set: calibration continued from previous record(s)

[Other bits reserved and must be zero.]

6 ULONG: Number of .0001 second ticks for the duration of the step.

7 ULONG: Number of .0001 second ticks for the interval between times the calibration step is "on."

8 FLOAT: Amplitude of calibration signal in units (see Channel Identifier Blockette [52], field 9).

9 CHAR*3: Channel containing calibration input (blank means none). SEED assumes that the calibration output is on the current channel, identified in the fixed header.

10 UBYTE: Reserved; do not use.

11 ULONG: Reference amplitude. This is a user defined value that indicates either the voltage or amperage of the calibration signal when the calibrator is set to 0dB. If this value is zero, then no units are specified, and the amplitude (Note 4) will be reported in "binary decibels" from to -96.

12 CHAR*12: Coupling of calibration signal, such as "Resistive " or "Capacitive".

13 CHAR*12: Rolloff characteristics for any filters used on the calibrator, such as "3dB@10Hz".

[310] Sine Calibration Blockette (60 bytes)

Note	Field name	Type	Length	Mask or Flags
1	Blockette type — 310	B	2	
2	Next blockette's byte number	B	2	
3	Beginning of calibration time	B	10	
4	Reserved byte	B	1	
5	Calibration flags	B	1	
6	Calibration duration	B	4	
7	Period of signal (seconds)	B	4	
8	Amplitude of signal	B	4	
9	Channel with calibration input	A	3	
10	Reserved byte	B	1	
11	Reference amplitude	B	4	
12	Coupling	A	12	
13	Rolloff	A	12	

Notes for fields:

1 UWORD: Blockette type [310]: sine calibration.

2 UWORD: Byte number of next blockette (0 if no more).

3 BTIME: Beginning time of calibration.

4 UBYTE: Reserved; do not use.

5 UBYTE : Calibration flags:

[Bit 2] — If set: calibration was automatic; otherwise: manual

[Bit 3] — If set: calibration continued from previous record(s)

[Bit 4] — If set: peak-to-peak amplitude

[Bit 5] — If set: zero-to-peak amplitude

[Bit 6] — If set: RMS amplitude

[Other bits reserved and must be zero.]

6 ULONG: Number of .0001 second ticks for the duration of calibration.

7 FLOAT: Period of signal in seconds.

8 FLOAT: Amplitude of signal in units (see Channel Identifier Blockette [52], field 9).

9 CHAR*3: Channel containing calibration input (blank means none).

10 UBYTE: Reserved; do not use.

11 ULONG: Reference amplitude. This is a user defined value that indicates either the voltage or amperage of the calibration signal when the calibrator is set to 0dB. If this value is zero, then no units are specified, and the amplitude (Note 4) will be reported in "binary decibels" from 0 to -96.

12 CHAR*12: Coupling of calibration signal such as "Resistive or "Capacitive".

13 CHAR*12: Rolloff characteristics for any filters used on the calibration, such as "3dB@10Hz".

NOTE: Only one of flag bits 4, 5, and 6 can be set at one time, but one of them must be set.

[320] Pseudo-random Calibration Blockette (64 bytes)

Note	Field name	Type	Length	Mask or Flags
1	Blockette type — 320	B	2	
2	Next blockette's byte number	B	2	
3	Beginning of calibration time	B	10	
4	Reserved byte	B	1	
5	Calibration flags	B	1	
6	Calibration duration	B	4	
7	Peak-to-peak amplitude of steps	B	4	
8	Channel with calibration input	A	3	
9	Reserved byte	B	1	
10	Reference amplitude	B	4	
11	Coupling	A	12	
12	Rolloff	A	12	
13	Noise type	A	8	

Notes for fields:

1 UWORD: Blockette type [320]: pseudo-random binary sequence.

2 UWORD: Byte number of next blockette (0 if no more).

3 BTIME: Beginning time of calibration.

4 UBYTE: Reserved; do not use.

5 UBYTE : Calibration flags:

[Bit 2] — If set: calibration was automatic; otherwise: manual

[Bit 3] — If set: calibration continued from previous record(s)

[Bit 4] — If set: random amplitudes

(must have a calibration in channel)

[Other bits reserved and must be zero.]

6 ULONG: Number of .0001 second ticks for the duration of calibration.

7 FLOAT: Peak-to-peak amplitude of steps in units (see Channel Identifier Blockette [52], field 9).

8 CHAR*3: Channel containing calibration input (blank if none).

9 UBYTE: Reserved; do not use.

10 ULONG: Reference amplitude. This is a user defined value that indicates either the voltage or amperage of the calibration signal when the calibrator is set to 0dB. If this value is zero, then no units are specified, and the amplitude (Note 4) will be reported in "binary decibels" from 0 to -96.

11 CHAR*12: Coupling of calibration signal such as "Resistive or "Capacitive".

12 CHAR*12: Rolloff characteristics for any filters used on the calibration, such as "3dB@10Hz".

13 CHAR*8: Noise characteristics, such as "White" or "Red".

NOTE: When you set calibration flag bit 4, the amplitude value contains the maximum peak- to-peak amplitude.

[390] Generic Calibration Blockette (28 bytes)

Note	Field name	Type	Length	Mask or Flags
1	Blockette type — 390	B	2	
2	Next blockette's byte number	B	2	
3	Beginning of calibration time	B	10	
4	Reserved byte	B	1	
5	Calibration flags	B	1	
6	Calibration duration	B	4	
7	Calibration signal amplitude	B	4	
8	Channel with calibration input	A	3	
9	Reserved byte	B	1	

Notes for fields:

1 UWORD: Blockette type [390]: generic calibration.

2 UWORD: Byte number of next blockette (0 if no more).

3 BTIME: Beginning time of calibration.

4 UBYTE: Reserved; do not use.

5 UBYTE: Calibration flags:

[Bit 2] — If set: calibration was automatic; otherwise: manual

[Bit 3] — If set: calibration continued from previous record(s)

[Other bits reserved and must be zero.]

6 ULONG: Number of .0001 second ticks for the duration of calibration.

7 FLOAT: Amplitude of calibration in units, if known (see Channel Identifier Blockette [52], field 9).

8 CHAR*3: Channel containing calibration input (must be specified).

9 UBYTE: Reserved; do not use.

[395] Calibration Abort Blockette (16 bytes)

Note	Field name	Type	Length	Mask or Flags
1	Blockette type — 395	B	2	
2	Next blockette's byte number	B	2	
3	End of calibration time	B	10	
4	Reserved bytes	B	2	

Notes for fields:

1 UWORD: Blockette type [395]: calibration abort.

2 UWORD: Byte number of next blockette (0 if no more).

3 BTIME: Time calibration ends.

4 UWORD: Reserved; do not use.

[400] Beam Blockette (16 bytes)

Note	Field name	Type	Length	Mask or Flags
1	Blockette type — 400	B	2	
2	Next blockette's byte number	B	2	
3	Beam azimuth (degrees)	B	4	
4	Beam slowness (sec/degree)	B	4	
5	Beam configuration	B	2	
6	Reserved bytes	B	2	

This blockette is used to specify how the beam indicated by the corresponding Beam Configuration Blockette [35] was formed for this data record. For beams formed by non-plane waves, the Beam Delay Blockette [405] should be used to determine the beam delay for each component referred to in the Beam Configuration Blockette [35].

Notes for fields:

1 UWORD: Blockette type [400]: beam forming.

2 UWORD: Byte number of next blockette (0 if no more).

3 FLOAT: Azimuth of beam (degrees clockwise from north).

4 FLOAT: Beam slowness (sec/degree).

5 UWORD: Beam configuration (see field 3 of the Beam Configuration Blockette [35] abbreviation dictionary). NOTE: This field is a binary equivalent of the ASCII formatted dictionary key entry number in the Beam Configuration Blockette [35]. This is the only place in SEED Version 2.1 where this "ASCII-to-binary" conversion needs to be made.

6 UWORD: Reserved; do not use.

[405] Beam Delay Blockette (6 bytes)

Use this blockette to define beams that do not travel as plane waves at constant velocities across arrays. This blockette, if used, will always follow a Beam Blockette [400]. The Beam Delay Blockette [405] describes the delay for each input component in the samples. SEED reading programs must find a corresponding entry in the Beam Delay Blockette [405] for each component in the Beam Configuration Blockette [35] of the abbreviation dictionary control headers, indexed by the Beam Blockette [400].

Note	Field name	Type	Length	Mask or Flags
1	Blockette type — 405	B	2	
2	Next blockette's byte number	B	2	
3	Array of delay values	B	2	

Notes for fields:

1 UWORD: Blockette type [405]: beam delay.

2 UWORD: Byte number of next blockette (0 if no more).

3 UWORD: Array of delay values (one for each entry of the Beam Configuration Blockette [35]. The array values are in .0001 second ticks.

[500] Timing Blockette (200 bytes)

Note	Field name	Type	Length	Mask or Flags
1	Blockette type — 500	B	2	
2	Next blockette offset	B	2	
3	VCO correction	B	4	
4	Time of exception	B	10	
5	μsec	B	1	
6	Reception Quality	B	1	
7	Exception count	B	4	
8	Exception type	A	16	
9	Clock model	A	32	
10	Clock status	A	128	

Notes for fields:

1 UWORD : Blockette type [500]: Timing blockette.

2 UWORD : Byte number of next blockette (0 if no more).

3 FLOAT: VCO correction is a floating point percentage from 0.0 to 100.0% of VCO control value, where 0.0 is slowest , and 100.0% is fastest.

4 BTIME: Time of exception, same format as record start time.

5 ~~U~~BYTE: μsec has the clock time down to the microsecond. The SEED format handles down to 100μsecs. This field is an offset from that value. The recommended value is from -50 to +49 μsecs. At the users option, this value may be from 0 to +99 μsecs.

6 UBYTE: Reception quality is a number from 0 to 100% of maximum clock accuracy based only on information from the clock.

7 ULONG: Exception count is an integer count, with its meaning based on the type of exception, such as 15 missing timemarks.

8 CHAR*16: Exception type describes the type of clock exception, such as “Missing” or “Unexpected”.

9 CHAR*32: Clock model is an optional description of the clock, such as “Quanterra GPS1/QTS”.

10 CHAR*128: Clock status is an optional description of clock specific parameters, such as the station for an Omega clock, or satellite signal to noise ratios for GPS clocks.

[1000] Data Only SEED Blockette (8 bytes)

V 2.3 – Introduced in SEED Version 2.3

Note	Field name	Type	Length	Mask or Flags
1	Blockette type — 1000	B	2	
2	Next blockette's byte number	B	2	
3	Encoding Format	B	1	
4	Word order	B	1	
5	Data Record Length	B	1	
6	Reserved	B	1	

Notes for fields:

1 UWORD : Blockette type (1000): Data Only SEED

2 UWORD : Byte number of next blockette. (Calculate this as the byte offset from the beginning of the logical record - including the fixed section of the data header; use 0 if no more blockettes will follow.)

3 BYTE : A code indicating the encoding format. This number is assigned by the FDSN Data Exchange Working Group. To request that a new format be included contact the FDSN through the FDSN Archive at the IRIS Data Management Center. To be supported in Data Only SEED, the data format must be expressible in SEED DDL. A list of valid codes at the time of publication follows.

CODES 0-9	GENERAL
0	ASCII text, byte order as specified in field 4
1	16 bit integers
2	24 bit integers
3	32 bit integers
4	IEEE floating point
5	IEEE double precision floating point
CODES 10 - 29	FDSN Networks
10	STEIM (1) Compression
11	STEIM (2) Compression
12	GEOSCOPE Multiplexed Format 24 bit integer
13	GEOSCOPE Multiplexed Format 16 bit gain ranged, 3 bit exponent
14	GEOSCOPE Multiplexed Format 16 bit gain ranged, 4 bit exponent
15	US National Network compression
16	CDSN 16 bit gain ranged
17	Graefenberg 16 bit gain ranged
18	IPG - Strasbourg 16 bit gain ranged
19	STEIM (3) Compression
CODES 30 - 39	OLDER NETWORKS
30	SRO Format
31	HGLP Format
32	DWWSSN Gain Ranged Format
33	RSTN 16 bit gain ranged

4 The byte swapping order for 16 bit and 32 bit words. A 0 indicates little-endian order and a 1 indicates big-endian word order. See fields 11 and 12 of blockette 50.

5 The exponent (as a power of two) of the record length for these data. The data record can be as small as 256 bytes and, in Data Only SEED format as large as 2 raised to the 256 power.

[1001] Data Extension Blockette (8 bytes)

V 2.3 – Introduced in SEED Version 2.3

Note	Field name	Type	Length	Mask or Flags
1	Blockette type — 1001	B	2	
2	Next blockette's byte number	B	2	
3	Timing quality	B	1	
4	μsec	B	1	
5	Reserved	B	1	
6	Frame count	B	1	

Notes for fields:

1 UWORD: Blockette type [1001]: Data extension blockette

2 UWORD: Byte number of next blockette.

3 UBYTE: Timing quality is a vendor specific value from 0 to 100% of maximum accuracy, taking into account both clock quality and data flags.

4 ~~U~~BYTE: μsec has the data start time down to the microsecond. The SEED format handles down to 100μsecs. This field is an offset from that value. The recommended value is from -50 to +49μsecs. At the users option, this value may be from 0 to +99μsecs.

5 Reserved byte.

6 UBYTE: Frame count is the number of 64 byte compressed data frames in the 4K record (maximum of 63). Note that the user may specify fewer than the maximum allowable frames in a 4K record to reduce latency

[2000] Variable Length Opaque Data Blockette

V 2.3 – Introduced in SEED Version 2.3

Note	Field name	Type	Length	Offset
1	Blockette type — 2000	B	2	0
2	Next blockette's byte number	B	2	2
3	Total blockette length in bytes	B	2	4
4	Offset to Opaque Data	B	2	6
5	Record number	B	4	8
6	Data Word order	B	1	12
7	Opaque Data flags	B	1	13
8	Number of Opaque Header fields	B	1	14
9	Opaque Data Header fields	V	V	15
	a Record type			
	b Vendor type			
	c Model type			
	d Software			
	e Firmware			
10	Opaque Data	Opaque		

More than one blockette 2000 may be stored in a SEED data record if the SEED data record 41timetag is not required for precise timing of the data in the opaque blockette. Under normal usage, there would be no data in the data portion of the SEED data record. However, it is possible that the blockette 2000 could be used to provide additional information for a normal timeseries data channel.

Notes for Fields:

1 UWORD: Blockette type (2000): Opaque Data blockette.

2 UWORD: Byte number of next blockette (Calculate this as the byte offset from the beginning of the logical record - including the fixed section of the data header; use 0 if no more blockettes will follow.)

3 UWORD: Blockette length. The total number of bytes in this blockette, including the 6 bytes of header. The only restriction is that the blockette must fit within a single SEED data record for the channel. Otherwise, the blockette must be partitioned into multiple blockettes.

4 UWORD: Offset to Opaque Data. Byte offset from beginning of blockette to Opaque Data.

5 ULONG: Record Number. The record number may be used for sequence identification of stream, record, or file oriented data. If a record is partitioned into multiple opaque blockettes, each blockette containing a portion of the record should contain the identical record number. It is strongly recommended that the record number be used to aid in the detection of missing data and in merging data from different telemetry streams. Use 0 if data is not record oriented, or if record number is not required.

6 UBYTE: Word order of binary opaque data. See field 4 of blockette 1000, and fields 11 and 12 of blockette 50.

0 = little-endian.

1 = big-endian.

7 UBYTE: Opaque Data flags.

[bit 0] Opaque blockette orientation.

0 = record oriented.

1 = stream oriented.

[bit 1] Packaging bit.

0 = Blockette 2000s from multiple SEED data records with different timetags may be packaged into a single SEED data record. The exact original timetag in each SEED Fixed Data Header is not required for each blockette 2000.

1= Blockette 2000s from multiple SEED data records with differing timetags may NOT be repackaged into a single SEED data record. Set this bit if the timetag in the SEED Fixed Data Header is required to properly interpret the opaque data.

[bits 2-3] Opaque blockette fragmentation flags.

00 = opaque record identified by record number is completely contained in this opaque blockette.

01 = first opaque blockette for record spanning multiple blockettes.

11 = continuation blockette 2...N-1 of record spanning N blockettes.

10 = final blockette for record spanning N blockettes.

[bits 4-5] File blockette information.

00 = not file oriented.

01 = first blockette of file.

10 = continuation of file.

11 = last blockette of file.

8 UBYTE: Number of Opaque Header fields. Each opaque header field is a variable length ASCII string, terminated by the character "~".

9 VAR: Opaque Data Header string, which contains the ASCII variable length fields. Each field is terminated by a "~". The definition of the fields may be defined by the originator and receiver, but the following are recommended. Any of the fields may be empty.

a Record Type - name of the type of record (e.g. "GPS", "GPS MBEN").

b Vendor Type - name of equipment vendor (e.g. "ASHTECH").

c Model type - model type of equipment (e.g. "Z12").

d Software Version - software version number (e.g. "").

e Firmware Version - firmware version number (e.g. "1G0C").

10 OPAQUE: Opaque Data - bytes of opaque data. Total length of opaque data in bytes is blockette_length - 15 - length (opaque_data_header_string)

The Data Section

The data section of a data record is located at the byte specified in the fixed header, at the beginning of the data. Programs that write data in the SEED format can place that first byte where appropriate. Data begin after the end of the last header blockette. Leave a gap of any size between the end of the header and the beginning of the data, but it should not be large unless you require space to add more blockettes later.

The Channel Identifier Blockette [52] in the control headers will define which format will be present in the record. This provides a coded reference to an entry in the Data Format Dictionary Blockette [30], which describes the internal representation of the data in the Standard Data Description Language.

For the Steim compression algorithm, the data start at the first available frame (64 X n bytes) after the end of the header blockettes. The first and last constant of integration (CI) are placed in the beginning of the data as embedded header information. See Appendix B for a detailed description of the Steim algorithm and its use with the SEED format.

Glossary

Abbreviation Dictionary Control Header. A set of control headers that define volume-wide abbreviations, especially for data format descriptions and station channel comments.

Accelerator Index. A periodic index used to locate a specific time segment within a time series.

Auxiliary Information. Supplementary information about a seismic station or a logical volume that may be needed to process the raw data completely.

Block Multiplexing. Interspersing data records for different channels. Normally, network volumes require that time series data be recorded as a straight sequence of data records for each time span. Some station processors at field stations are not capable of this (especially for multiplexed time series data). This means that these processors have to write, at specific times, data records that contain various time series. Block multiplexed data are the result.

Blockette. A data structure consisting of an identification code, a length specification, and a sequence of related data fields. Formatted blockettes are used in control headers and unformatted blockettes are used in the header portions of data records. Blockettes are strung together in specific sequences to make up much of the SEED format.

Calibration. Also, cal. An operation on equipment such as field station instrumentation which determines the sensitivity of the instrumentation to ground motion.

Calibration Information. Raw data from a calibration signal generator. Some calibration techniques require calibration information to be recorded while the output channels of the instrument is calibrated.

Cascade. The response of a channel may be separated into a series of generalized filter sections. Such a separation assumes that each section behaves independently. The filter sections may be analog or digital, but one section will include the response of the instrument itself, making the series "generalized." To determine the overall response of the cascaded filters, the individual sections are convolved in time.

Channel. Also, station channel. Recorded digitized output that forms a time series from a field station instrument through a particular set of filters.

Concatenate. To string together, one item after another.

Continuous Time Series. Continuously recorded raw data. A continuous time series is arbitrarily divided into a number of time series, each written into a different time span. Continuous time series appear only on station oriented network volumes.

Control Header. Related auxiliary information, formatted according to rules specifying one of four types, and designed to make SEED data self defining.

Corner. The point at which the slope of the logarithm of the frequency response curve changes.

Data Field. One item of auxiliary information. Data fields are formatted (ASCII) or unformatted (binary). Formatted data fields may be of fixed or variable length. Unformatted data fields are always fixed in length.

Data Piece. A time series represented in one or more data records. A data piece contains no time tears, and is therefore continuous. Time span control headers contain entries that indicate where data pieces are located within the SEED format.

Data Record. A SEED data structure that consists of a data record identification block, a fixed header section, a variable header section, and a data section. One or more data records make up a logical record. Time series and multiplexed time series are written as a sequence of one or more data records.

Data Record Identification Block. A fixed length block of bytes containing a sequence number (usually set to zero), a format object type flag, and a blockette continuation flag. The first data record identification block in a logical record is the logical record identification block; its sequence number is never zero.

Data Section. The portion of a data record actually containing time series data.

Dynamic Information. Continuously changing information. All raw data, status, and log information are dynamic.

Event. A part of continuous time series data that is denoted by an algorithm-declared trigger event flag set in the data header.

Event Oriented Network Volume. A logical volume in which event triggered time series from each station channel are organized into separate time spans for each event (or perhaps a small number of nearly concurrent events).

Event Triggered Time Series. A recorded time series of finite length, initiated or triggered by some external event. Only a fraction of the raw data from an event triggered channel is recorded in a number of discrete event triggered time series. Event triggered time series may appear on network (station oriented) volumes or event (event oriented) volumes.

Event Volume. See: Event Network Volume.

Field Recording Format. The original binary representation of recorded raw data. In most cases, the recording format and the format in which raw data are originally acquired are identical.

Field Station. A physical site where instruments and recording equipment are located.

Field Station Volume. A logical volume that contains channel data recorded by or transmitted from one field station. It differs from a station oriented network volume in that its control headers may be incomplete, time series may be block multiplexed, several of its data structures differ, and data block sizes may vary from channel to channel.

FIR filter. A finite impulse response digital filter.

Fixed Header Section. A SEED data structure that contains unformatted identification and status information that appears in a fixed sequence at the beginning of every data record.

Format Object. A control header, a time series, or a multiplexed time series. Format objects appear as a sequence of one or more logical records.

Formatted. Information coded as a character string. A formatted write from a high level computer language will produce formatted data.

Geophysical Information. Raw data from field station instruments.

Header Flushing. When writing a logical volume, you can flush all time series logical records, repeat all control headers, and resume writing time series data periodically. In effect, this creates a number of logical sub-volumes within one logical volume. Header flushing provides redundant control header information on station volumes, and synchronizes the start times of data records for all station channels. SEED preserves channel synchronization on network volumes, although the redundant headers are discarded.

IIR filter. An infinite impulse response digital filter.

Index. An index to a specific logical record. Indices stored in control headers allow a data user to directly use a logical record, skipping over information not of immediate interest. Time series data may have sub-indices. These represent data records of interest within the indexed logical records.

Logical Record. A SEED data structure that can be individually located and that begins with a logical record identification block. A sequence of logical records make up a format object.

Logical Record Identification Block. The identification block of a logical record containing a control header, or the first data record identification block appearing in a logical record; a fixed length block of bytes containing the absolute sequence number of the logical record within the logical volume, a format object type flag, and a blockette continuation flag.

Logical Sub-Volume. One of several complete logical volume structures within a station logical volume. Each logical sub-volume begins with a complete set of control headers.

Logical Volume. One complete data set, usually comprising all raw data and all auxiliary data from a set of stations during one or more time intervals. One or more logical records make up a logical volume. Use a complete, internally consistent implementation of the SEED format to write a logical volume. One or more logical volumes may appear on one physical volume, but a logical volume may not span more than one physical volume. (Control headers can be duplicated from one physical volume to another, allowing very large amounts of data to be stored on more than one physical volume.)

Multiplexed Time Series. More than one time series stored in a sequence of multiplexing frames.

Multiplexing Frame. Raw data samples from more than one channel, taken at nearly the same time and stored one after the other in a fixed sequence.

Network Volume. See: Station Oriented Network Volume and Event Oriented Network Volume.

One's Complement. Also: 1's complement. A means of storing numbers in a computer or on a mass storage device, whereby a negative number is created from its positive counterpart by inverting all the number's bits.

Physical Record. The amount of data accessed on a computer mass storage device in one input or output (I/O) operation. Depending on the device, the length of the physical record can be either fixed or variable. Logical records usually occupy a sequence of one or more physical records, although one physical record may contain several shorter logical records. Logical records may begin and end within a physical record.

Physical Volume Control Header. Physical volume control information that locates separate logical volumes. This information is always device- and often operating system-dependent, and may not exist for some devices (particularly sequential devices). The physical volume control header can be created on some devices under user control, but is not part of the SEED standard.

Physical Volume. One unit of dismountable computer mass storage media (e.g., a magnetic tape reel). One or more physical records make up a physical volume.

Raw Data. Data samples in the original field recording format.

Sequence Number. A unique, sequential identifying number given to a data record when it is equal to or larger than a logical record.

State-of-Health Information. Raw data relevant to the operation of the station equipment or the station environment.

Static Information. Information that does not change for long periods of time. Most control header information is static (comments are not).

Station Channel. See: Channel.

Station Control Header. Static auxiliary information about one station and all of its channels, particularly station location and channel transfer function information.

Station Log Information. A station volume's formatted auxiliary information that describes the station processor's status and/or the interactions between station operators and station processors; often printed on a console.

Station Oriented Network Volume. A logical volume in which station channels are organized into arbitrary time spans. One continuous time series, or a sequence of event triggered time series with time gaps between them (in which no data were recorded), are included for each entire time span for each station channel.

Station Processor. A dedicated computer system at a digital seismic station. This processor controls the station and can be reprogrammed as needed. It typically controls digital data acquisition, data buffering, calibration, event detection, data formatting, local data storage, and data telemetry to a central site.

Station Volume. See: Field Station Volume.

Status Information. Auxiliary information pertaining to a station channel for a particular time interval.

Time Pointer. See: Accelerator index.

Time Series. Raw data from one station channel which have been continuously recorded during a finite time interval.

Time Span Control Header. A SEED data structure that contains information pertaining to a fixed time interval, including hypocenter and phase arrival time information, as well as indices to time series.

Time Tear. A time gap, greater than an allowed tolerance, in a time series.

Transfer Function. The response of a channel, usually in counts per physical units as a function of frequency. This response may appear in several ways, the most precise being a complete description of the Laplace transform of the impulse response of the analog system, cascaded with a complete description of the digitizing and digital filtering performed.

Two's Complement. Also: 2's complement. A means of storing numbers in a computer or on a mass storage device, whereby a negative number is created from its positive counterpart by inverting all the number's bits, and adding 1 to the result.

Unformatted. Information coded as a sequence of binary data types. Character data mixed with binary data are also unformatted. An unformatted write from a high level computer language will produce unformatted data.

Variable Header Section. A sequence of optional unformatted blockettes following the fixed header section and preceding the data section of a data record.

Volume Index Control Header. A control header containing information about a complete logical volume, including indices to station control headers and time span control headers.

Appendix A: Channel Naming

Contributed by Scott Halbert

Seismologists have used many conventions for naming channels. Usually, these conventions are designed to meet the particular needs of one network. But general recording systems — such as the various Global Seismographic Network (GSN) systems that can record many channels at high sample rates — create a need for a standard to handle the variety of instruments that can be recorded. Modern instrumentation and the need for conformity among cooperating networks have greatly complicated the problem. Sensors are available in narrow band and broadband configurations with pass bands in very different parts of the spectrum of interest. Each sensor may have several different outputs with different spectral shaping. In addition, station processors often derive several data streams from one sensor channel by digital filtering. These possibilities require a comprehensive convention. The desire to combine data from cooperating networks and to search for like channels automatically requires standardization.

The SEED format uses three letters to name seismic channels, and three letters to name weather or environmental channels. In the following convention, each letter describes one aspect of the instrumentation and its digitization. SEED does not require this convention, but we recommend it as a usage standard for Federation members to facilitate data exchange.

Band Code

The first letter specifies the general sampling rate and the response band of the instrument. (The "A" code is reserved for administrative functions such as miscellaneous state of health.)

Band code	Band type	Sample rate (Hz)	Corner period (sec)
E	Extremely Short Period	≥ 80	< 10 sec
S	Short Period	≥ 10 to < 80	< 10 sec
H	High Broad Band	≥ 80	≥ 10 sec
B	Broad Band	≥ 10 to < 80	≥ 10 sec
M	Mid Period	> 1 to < 10	
L	Long Period	≈ 1	
V	Very Long Period	≈ 0.1	
U	Ultra Long Period	≈ 0.01	
R	Extremely Long Period	≈ 0.001	
A	Administrative		
W	Weather/Environmental		
X	Experimental		

Instrument Code and Orientation Code

The second letter specifies the family to which the sensor belongs. The third letter specifies the physical configuration of the members of a multiple axis instrument package or other parameters as specified for each instrument.

Seismometer: Measures displacement/velocity/acceleration along a line defined by the dip and azimuth.

Instrument Code

H	High Gain Seismometer
L	Low Gain Seismometer
G	Gravimeter
M	Mass Position Seismometer
N*	Accelerometer
	* historically some channels from accelerometers have used instrumentation codes of L and G. The use of N is the FDSN convention as defined in August 2000.

Orientation Code

Z N E	Traditional (Vertical, North-South, East-West)
A B C	Triaxial (Along the edges of a cube turned up on a corner)
T R	For formed beams (Transverse, Radial)
1 2 3	Orthogonal components but non traditional orientations
U V W	Optional components
Dip/Azimuth:	Ground motion vector (reverse dip/azimuth if signal polarity incorrect)
Signal Units:	M, M/S, M/S**2, (for G & M) M/S**2 (usually)
Channel Flags:	G

Tilt Meter: Measures tilt from the horizontal plane. Azimuth is typically N/S or E/W.

Instrument Code

A

Orientation Code

N E	Traditional
Dip/Azimuth:	Ground motion vector (reverse dip/azimuth if signal polarity incorrect)
Signal Units:	Radians
Channel Flags:	G

Creep Meter: Measures the absolute movement between two sides of a fault by means of fixing a metal beam on one side of the fault and measuring its position on the other side. This is also done with light beams.

The orientation and therefore the dip and azimuth would be perpendicular to the measuring beam (light or metal), which would be along the average travel vector for the fault. Positive/Negative travel would be arbitrary, but would be noted in the dip/azimuth. Another type of Creep Meter involves using a wire that is stretched across the fault. Changes in wire length are triangulated to form movement vector.

Instrument Code
B
Orientation Code
Unknown
Dip/Azimuth: Along the fault or wire vector
Signal Units: M
Channel Flags: G

Calibration Input: Usually only used for seismometers or other magnetic coil instruments. This signal monitors the input signal to the coil to be used in response evaluation. Usually tied to a specific instrument. Sometimes all instruments are calibrated together, sometimes horizontals are done separately from verticals.

Instrument Code
C
Orientation Code
A B C D. for when there are only a few cal sources for many devices.
Blank if there is only one calibrator at a time or, Match Calibrated Channel (is. Z, N or E)

Pressure: A barometer, or microbarometer measures pressure. Used to measure the weather pressure or sometimes for state of health monitoring down hole. This includes infrasonic and hydrophone measurements.

Instrument Code
D
Orientation Code
O Outside
I Inside
D Down Hole
F Infrasound
H Hydrophone
U Underground
Dip/Azimuth: Not applicable — Should be zero.
Signal Units: Pa (Pascals)
Channel Flags: W or H

Electronic Test Point: Used to monitor circuitry inside recording system, local power or seismometer. Usually for power supply voltages, or line voltages.

Instrument Code
E
Orientation code
Designate as desired, make mnemonic as possible, use numbers for test points, etc.
Dip/Azimuth: Not applicable
Signal Units: V, A, Hz, Etc.
Channel Flags: H

Magnetometer: Measures the magnetic field where the instrument is sitting. They measure the part of the field vector that is aligned with the measurement coil. Many magnetometers are three axis. The instrument will typically be oriented to local magnetic north. The dip and azimuth should describe this in terms of the geographic north.

Example: Local magnetic north is 13 degrees east of north in Albuquerque. So if the magnetometer is pointed to magnetic north, the azimuth would be + 103 for the E channel. Some magnetometers do not record any vector quantity associated with the signal, but record the total intensity. So, these would not have any dip/azimuth.

Instrument Code
F
Orientation Code

Z N E	Magnetic
Signal Units:	T — Teslas
Channel Flags:	G

Humidity: Absolute/Relative measurements of the humidity. Temperature recordings may also be essential for meaningful results.

Instrument Code
I
Orientation Code

O	Outside Environment
I	Inside Building
D	Down Hole
1 2 3 4	Cabinet Sources

All other letters available for mnemonic source types.

Dip/Azimuth:	Not applicable — Should be zero.
Signal Units:	%
Channel Flags:	W

Temperature: Measurement of the temperature at some location. Typically used for measuring:

1. Weather - Outside Temperature
2. State of Health - Inside recording building
   - Down hole
   - Inside electronics

Instrument Code
K
Orientation Code

O	Outside Environment
I	Inside Building
D	Down Hole
1 2 3 4	Cabinet sources

All other letters available for mnemonic types.

Dip Azimuth:	Not applicable — Should be zero.
Signal Units:	deg C or deg K
Channel Flags:	W or H

Water Current: This measurement measures the velocity of water in a given direction. The measurement may be at depth, within a borehole, or a variety of other locations.

Instrument Code
O
Orientation Code
Unknown
Dip/Azimuth: Along current direction
Signal Units: M/S
Channel Flags: G

Geophone: Very short period seismometer, with natural frequency 5 - 10 Hz or higher.

Instrument Code
P
Orientation Code
Z N E Traditional
Dip/Azimuth: Ground Motion Vector (Reverse dip/azimuth if signal polarity incorrect)
Signal Units: M, M/S, M/S
Channel Flags: G

Electric Potential: Measures the Electric Potential between two points. This is normally done using a high impedance voltmeter connected to two electrodes driven into the ground. In the case of magnetotelleuric work, this is one parameter that must be measured.

Instrument Code
Q
Orientation Code
Unknown
Signal Units: V — Volts
Channel Flags: G

Rainfall: Measures total rainfall, or an amount per sampling interval.

Instrument Code
R
Orientation Code
Unknown
Dip/Azimuth: Not applicable — Should be zero.
Signal Units: M, M/S
Channel Flags: W

Linear Strain: One typical application is to build a very sensitive displacement measuring device, typically a long quartz rod. One end is affixed to a wall. On the free end, a pylon from the floor reaches up to the rod where something measures the position of the pylon on the rod (like a large LVDT).

There are also some interferometry projects that measure distance with lasers. Dip/Azimuth are the line of the movement being measured. Positive values are obtained when stress/distance increases, negative, when they decrease.

Instrument Code
S
Orientation Code
Z N E Vertical, North-South, East-West
Dip/Azimuth: Along axis of instrument
Signal Units: M/M
Channel Flags: G

Tide : Not to be confused with lunar tidal filters or gravimeter output. Tide instruments measure the depth of the water at the monitoring site.

Instrument Code
T
Orientation Code
Z Always vertical
Dip/Azimuth: Always vertical
Signal Units: M — Relative to sea level or local ocean depth
Channel Flags: G

Bolometer: Infrared instrument used to evaluate average cloud cover. Used in astronomy to determine observability of sky.

Instrument Code
U
Orientation Code
Unknown
Dip/Azimuth: Not applicable — Should be zero.
Signal Units: Unknown
Channel Flags: W

Volumetric Strain: Unknown

Instrument Code
V
Orientation Code
Unknown
Dip/Azimuth: Not Applicable — Should be zero.
Signal Units: M**3/M**3
Channel Flags: G

Wind: Measures the wind vector or velocity. Normal notion of dip and azimuth does not apply.

Instrument Code
W
Orientation Code
S Wind speed
D Wind Direction Vector — Relative to geographic North
Dip/Azimuth: Not Applicable — Should be zero.
Channel Flags: W

Synthesized Beams: This is used when forming beams from individual elements of an array. Refer to blockettes 35, 400, & 405.

Instrument Code
Z
Orientation Code
I Incoherent Beam
C Coherent Beam
F FK Beam
O Origin Beam
Dip/Azimuth: Ground motion vector (reverse dip/azimuth if signal polarity incorrect)
Signal Units: M, M/S, M/S**2, (for G & M) M/S**2 (usually)
Channel Flags: G

Channel Code

We suggest that two sequences be reserved for special channels: the "LOG" channel for the console log, and the "SOH" channel for the main state of health channel. Subsidiary logs and state of health channels should begin with the "A" code; the source and orientation fields can then be used in any way.

Here are some typical channel arrangements used by a GSN system:

Channel	Description
EHZ/EHN/EHE	Short Period 100 sps
BHZ/BHN/BHE	Broad Band 20 sps
LHZ/LHN/LHE	Long Period 1 sps
VHZ/VHN/VHE	Very Long Period 0.1 sps
BCI	Broad Band Calibration Signal
ECI	Short Period Cal
LOG	Console Log

NOTE: Log Records: Log records has a channel identifier code of "LOG" and a sample rate of zero. The number of samples field is the number of characters in the record (including the carriage return and line feed that terminates each line). Log messages are packed into records until a message falls into a new minute. Log records have no blockettes, so the strings start at offset 48.

Appendix B: Compression Algorithms

Steim1 Compression Scheme

Contributed by C.R. Hutt

Within a time series of signed 32-bit integers (2's complement format), each data sample or integer consists of four bytes (1 byte = 8 bits). Such a time series representing seismometer output data under normal seismic background conditions is usually highly correlated — that is, each data sample is highly predictable, given the previous few samples. Also, the normal seismic background tends to be of a fairly low frequency compared to the sample rate, so that differences between consecutive samples are generally quite small compared to a full-scale 32- bit number. In fact, such differences seldom require more than four or five bits to represent them to the same accuracy found in the original 32-bit number. These differences, then, can easily be represented as 1-byte quantities without any information loss. If we do this, we can significantly compress the data and reduce the space required to store it.

Significant seismic activity, however, can create consecutive differences larger than an 8-bit quantity. In such infrequent cases (less than one per cent of the time), a 2-byte or a 4-byte quantity can be used to represent the sample difference. But we would need some kind of code to tell us when a sample difference is fully represented by one byte, two bytes, or four bytes. Given this code and the first data sample (not the difference), we could then reconstruct the original 32-bit series of data samples with a series of differences that are each only eight bits wide more than 99 per cent of the time. If such a scheme is used to store data on magnetic tape or some other storage medium, we can achieve a compression ratio of greater than 3.5 to 1, compared to storing all of the original data samples in 32-bit format.

Let the original time series be the samples x_{-1}, x_0, x_1, …, where each x_i is a 32-bit (or smaller) signed integer. Let d_0, d_1, …, be the first difference time series ,where:

$$d_0 = x_0 - x_{-1},$$
$$d_1 = x_1 - x_0,$$
$$\vdots$$

$$d_i = x_i - x_{i-1},$$
$$d_{i+1} = x_{i+1} - x_i,$$
$$\vdots$$
$$d_n = x_n - x_{n-1}$$

The d_i are all 32 bits wide. Now we look at each d_i to see if we can represent them with eight bits or 16 bits instead of 32 bits. If we can represent four consecutive d_i with eight bits, then we can form a single 32-bit word w_k that represents four 8-bit quantities:

$$(d_i)_1, (d_{i+1})_1, (d_{i+2})_1, (d_{i+3})_1$$

where the subscript "1" means "the 1-byte version of what is in the parentheses." We can keep track of the fact that w_k represents four 8-bit quantities with a 2-bit code, $c_k = 01_2$.

Suppose now that one or both of the first two consecutive d_i will not fit within eight bits, but both will fit within 16 bits. That is, if d_i is positive, greater than 127 and less than or equal to 32,767, or if d_i is negative, greater than or equal to -32,768 and less than -128, then we will construct a w_k that consists of two 16-bit quantities:

$$(d_i)_2, (d_{i+1})_2,$$

where the subscript "2" means "the 2-byte version of what is in the parentheses." Again, we keep track of the fact that w_k represents two 16-bit quantities with a 2-bit code, $c_k = 10_2$.

Now suppose that d_i is greater than 32,767 or less than -32,768. This means that we need more than 16 bits to represent it with no loss of information, so we let wk equal $(d_i)_4$, where the subscript "4" means "the 4-byte version of what is in the parentheses." We keep track of the fact that w_k represents one 32-bit quantity with the 2-bit code, $c_k = 11_2$.

In cases where c_k does not correspond to any of the 1-byte, 2-byte, or 4-byte sample differences, we let $c_k = 00_2$. This is a special two-bit code standing for anything other than data differences, such as header information.

Next, we need a convenient record format for storing c_k, w_k, and header information on the volume. Since the w_k contains only the difference d_i, the header must include a forward integration constant: the 32-bit quantity x_0, so that all subsequent values in the time series (x_1,x_2, …) can be calculated, as follows:

$$x_0 = x_{-1} + d_0,$$
$$x_1 = x_0 + d_1,$$
$$\vdots$$
$$x_i = x_{i-1} + d_i,$$
$$\vdots$$
$$x_n = x_{n-1} + d_n$$

We also include the reverse integration constant x_n , so that the time series could be calculated backwards from the last difference in the record, in case of bit errors in the middle of the record:

The reverse integrating constant also provides for a quick data integrity check when compared with the last computed sample. A discrepancy indicates that the contents of the data are garbled.

$$x_{n-1} = x_n - d_n,$$
$$\vdots$$
$$x_i = x_{i+1} - d_{i+1},$$

$$x_{i-1} = x_i - d_i,$$

$$\vdots$$

$$x_0 = x_1 - d_1$$

The reverse integrating constant also provides for a quick data integrity check when compared with the last computed sample. A discrepancy indicates that the contents of the data are garbled.

Note that any given record will contain n + 1 differences (d_0 - d_n), and that x_n for this record will be equal to x_{-1} for the next record. Note also that, when the recording system is cold-started, x_{-1} is not known and so is set to zero. This means that on a volume's first record, when the system is cold-started or re-initialized, we will have:

$$d_0 = x_0 - x_{-1} = x_0.$$

The following figure shows the volume format used for the compressed data. Place the data in a data record starting at byte 64. The records' size should be 4096 bytes, but if memory constraints disallow such a buffer size, use a smaller size. The first 64 bytes will always contain the fixed data header (bytes 0 through 47 will contain the actual data header, and the following 16 bytes will be set to zeros). 63 data frames will follow.

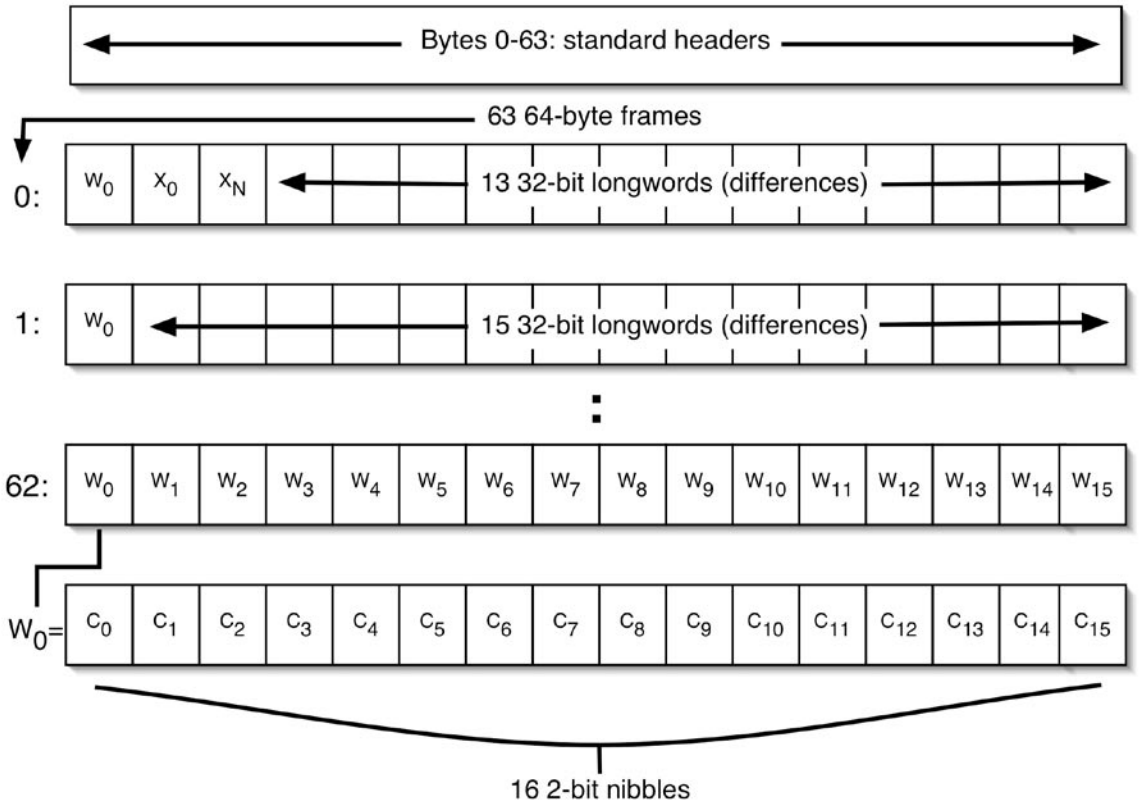


Figure 13: Compressed Data Format

Each c_k is a 2-bit nibble code that corresponds to a 4-byte quantity in the frame:

$c_k = 00_2$ = special: w_k contains non-data information, such as headers or w_0

$c_k = 01_2$ = four 1-byte differences contained in w_k (four 8-bit samples)

$c_k = 10_2$ = two 2-byte differences contained in w_k (two 16-bit samples)

$c_k = 11_2$ = one 4-byte difference contained in w_k (one 32-bit sample)

c_0 corresponds to w_0 (which always contains the code bits), so c_0 always equals 00_2. In frame 0, $c_0 = c_1 = c_2 = 00_2$; c_3 through c_{15} correspond to w_3 through w_{15}. In frame 0, $w_1 = x_0$ (the forward integration constant) and $w_2 = x_n$ (the reverse integration constant).

The figure above has a 64-byte header — followed by 63 data frames, each of which is 64 bytes wide — so that the record is 4096 bytes long. The first four bytes (w_0) of each frame contain 16 2-bit codes c_0 through c_{15} for that frame. The first frame (frame 0) contains the integration constants in the next 8 bytes (w_1 and w_2). $w_1 = x_0$, the forward integration constant, and $w_2 = x_n$, the reverse integration constant. In Frame 0, w_3 through w_{15} contain sample differences d_i as specified by the 2-bit codes c_3 through c_{15}. Subsequent frames (1 through 62) each contain codes c_0 through c_{15} in w_0, and w_1 through w_{15} each contain four 1-byte, two 2-byte, or one 4-byte difference as specified by codes c_1 through c_{15}. Since each frame can contain a maximum of 60 differences (frame 0 has eight less due to the constants stored there), each data record can contain a maximum of (63 X 60) - 8 = 3,772 differences, corresponding to 3,772 original data samples (including x_0 from the header). The very first difference (d_0) of the data actually represents the difference between the last sample of the previous record, and x_0, the first sample of the current record.

On the other extreme, if every w_k corresponds to only one 32-bit difference (no compression at all), then all frames can contain 15 differences (except for frame 0 which has 13). The minimum number of differences is (63 X 15) - 2 = 943 differences. This corresponds to 943 original data samples (including x_0). The data format, however, allows the record to be any length shorter than this. So, when the number of samples (in the fixed data header) have been read, everything in the record after that number is ignored.

If differencing and compression are not used and we record, instead, the original data samples in 32-bit format, using four bytes for every sample and no code, then all frames would contain16 data samples for a total of (63 X 16) = 1,008 data samples per record. If all of the data samples were compressible to 8-bit differences, the same information could have been recorded in 1,008/3,772 of a record, or 0.267 records. This is a compression ratio of about 3.75 to 1; that is, about 3.75 compressed data samples can be put into the same space as one uncompressed data sample. If we assume that the data are compressible to eight bits 99% of the time and must be represented as 32 bits 1% of the time (this is worst case, since part of this 1% would be 16- bit compressed data), the compression works out to be approximately 3.72 to 1 — still well over 3.5 times more data per volume than without compression.

This example assumes a header length of 64 bytes followed by 63 data frames. A situation such as an event detection or a calibration could force the header to be larger then 64 bytes. In this case, create a 128-byte header, followed by 62 data frames.

Several differences exist between compressed and uncompressed tape formats, but these do not seem to be great disadvantages:

- Data must be “decompressed” or converted to original 32-bit data samples before analysis.

- Only one single channel of data should be contained in a record. Several seismic data channels should not be multiplexed into a single record, since the compression ratios for channels will usually differ.

- Although the records are often fixed in length at 4096 bytes, they are not required to be that size, and they are also not fixed in the number of samples per record. With Steim1 compression, each record may contain from 943 to 3,772 samples. The maximum number of samples a 4096 seed record could contain with Steim 2 is 6601. This means that the header time differences between consecutive records of the same data channel will not be fixed, but can be calculated easily by using the sample rate and the number of samples represented in each record. Consequently, the recording system computer must place the number of samples for each record in the header of that record.

Note: This algorithm is copyrighted by Dr. Joseph Steim.

Steim2 Compression Scheme

Contributed by Caryl Peterson

The second Steim compression scheme, called "Steim2", allows for a greater variety in the number of bits used to represent the differences. In all the cases except one (8 bit differences), the high order two bits in the 32-bit word (w_k) of compressed data samples is needed for further decoding of the compression scheme. These high order two bits will be referred to as *dnib*, for "decode nibble". In the following description, c_k is the same as described in Figure 13, and *dnib* refers to the high order two bits of w_k.

c_k = 00_2 = same as Steim1, special: w_k contains non-data information

c_k = 01_2 = same as Steim1, four 1-byte (8 bit) differences contained in w_k

c_k = 10_2 = look in *dnib*, the high order two bits of w_k

dnib = 01_2 = one 30-bit difference in w_k

dnib = 10_2 = two 15-bit differences in w_k

dnib = 11_2 = three 10 bit differences in wk

c_k = 11_2 = look in *dnib*, the high order two bits of w_k

dnib = 00_2 = five 6-bit differences in wk

dnib = 01_2 = six 5-bit differences in w_k

dnib = 10_2 = seven 4-bit differences in wk

If the difference between two samples is between -8 and +7, that difference can be expressed

Ain 4 bits. If seven consecutive differences are in this range, the compression algorithm puts seven of these 4-bit differences in a single 32 bit word using 28 of the available 32 bits. The c_k in w_0 would be 11_2 indicating that either seven 4-bit differences, six 5-bit differences, or five 6-bit differences are in w_k. The high order two bits of w_k would contain 10_2 to indicate that seven 4-bit differences are in the last 28 bits of w_k. The two bits following *dnib* would not be used.

Negative differences between -16 and -8 or positive differences between +7 and +15 would take

5 bits to represent. Six consecutive differences in this range could be placed in a 32-bit word. Again, c_k in w_0 would contain 11_2, but the high order two bits in wk would contain 01_2 meaning that w_k contains six 5-bit differences.

Six bits could represent differences that fall in the range between -32 and -16 or +15 and +31. The c_k bits in w_0 would still be 11_2, but *dnib*, the high order two bits of w_k, would be 00_2, and the remaining bits would contain the five differences.

Differences lying between -128 and -32 or +31 and +127 would be represented by 8 bits. In this case, all 32-bits of the word are used by four of these 8-bit differences. No further decoding of c_k is necessary in the case of 8-bit differences.

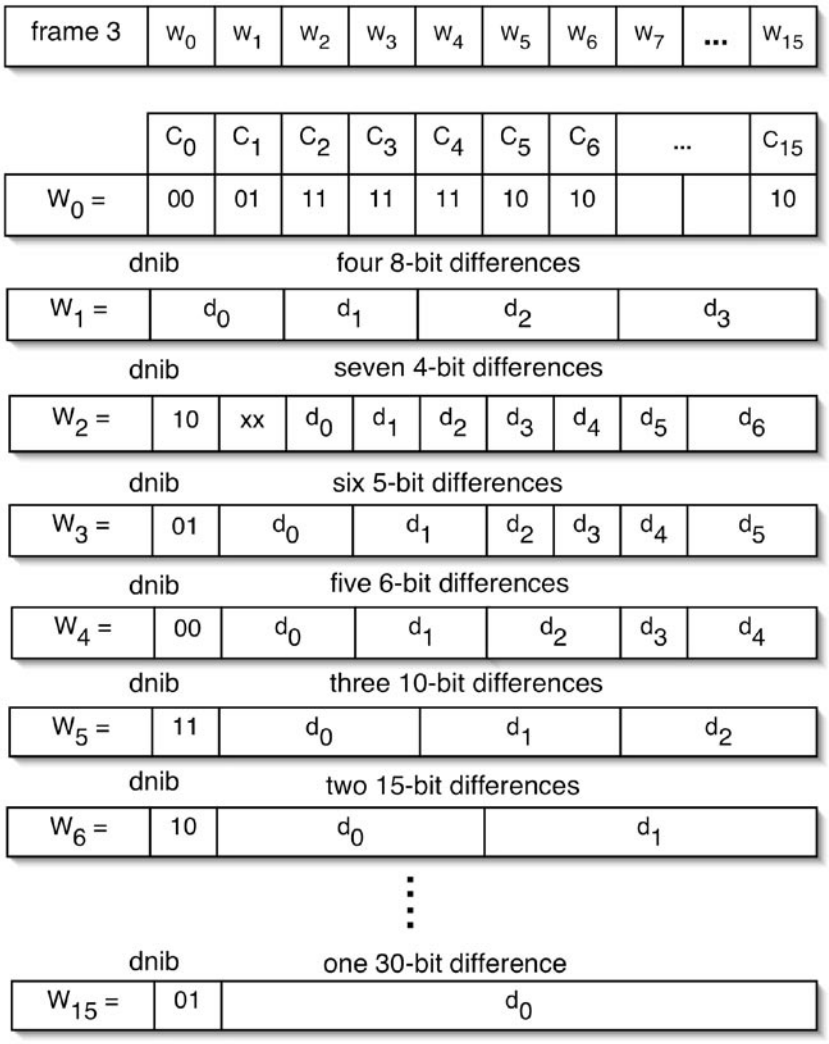


Figure 14: Steim2 compression data format

Ten bits would represent differences that are negative and greater than or equal to -512 but less than -128 or positive and greater than +127 but less than or equal to +511. In this case, c_k would be 10_2 to indicate either one 30-bit difference, two 15-bit differences, or three 10-bit differences; *dnib* would be 11_2 indicating that three 10-bit differences are in the remaining 30 bits.

If the differences are between -16384 and -512 or between +511 and 16383, 15 bits represent those differences. Two of these differences would be placed in w_k. c_k would be 10_2 and *dnib* would be 10_2.

The largest differences that could be represented by Steim2 are those differences that are between -2^{29} and -2^{14} and between $+2^{14}-1$ and $+2^{29}-1$. These differences are represented by 30 bits. If a difference is in this range, one 30 bit difference is placed in w_k; c_k is 10_2 and *dnib* is 01_2.

Figure 14 shows an example of a frame of compressed data where each of the first eight w_k are comprised of unique sized differences.

The best compression ratio is achieved with Steim2 if all the differences can be expressed in 4 bits. In this case, the compression ratio is 6.74 to 1. In a comparative study of data compression schemes, the best compression ratio for Steim2 was 6.07 to 1. This compression ratio was achieved on 20 Hz low noise seismic data. The same data was compressed using

the Steim1 compression scheme. In that case, the compression ratio was 3.67 to 1. In a seismically noisy environment or during large events, the compression ratio for both schemes is reduced.

USNSN Data Compression

Contributed by Ray Buland

The United States National Seismic Network data compression scheme follows the same philosophy as all other data compression methods in common use for seismological data today. Compression depends on the fact that seismological data acquisition systems are designed to record very large values of ground motion that are very rarely attained. Thus, the compression results from eliminating as much of the sign extension portion of each raw data value as possible in such a way that the original value is exactly recoverable. The algorithm is divided into two portions: preprocessing to minimize the fluctuations of the (preprocessed) data about zero as much as possible, and encoding of the data values into variable length data fields and adding information in accompanying key fields to permit decoding. In addition, the algorithm makes provisions for effective use of fixed length (computer storage) records, imbedded header information (channel identification, time, status, etc.) and redundant information to promote fault recovery. The NEIC has programmed this algorithm in FORTRAN for little endian machines and in C for big endian machines. Conversion to other systems is not difficult.

In USNSN compression, preprocessing is a simple first differencing that requires at least one integration constant to be recoverable. Encoding is patterned on a scheme developed at the Geological Survey of Canada by Ken Beverly. Each key field corresponds to a fixed number of fixed length data fields. Taken together, the data fields belonging to one key field always begin and end on an (8-bit) byte boundary in the computer storage, but will, in general occupy different numbers of bytes for different key values. The organization and definitions of the key values has been modified from the Canadian scheme to adapt it for fixed length records and to achieve less key overhead for small data values. A fixed length header section is provided for at the beginning of each record as part of the header. Only the first constant is actually required. In addition, back pointers are imbedded into the encoded data and a reverse integration constant added at the end of each time series. This redundant information has been added to permit the recovery of as much data as possible should any portion of the compressed data be corrupted.

Before describing the algorithm in detail, it is useful to define some specialized terms to avoid ambiguity.

Data Value: The digitally encoded value of a sensor output. Note that the data values acquired

by an analog-to-digital converter and the data values used in a computer must be numerically identical, but may be encoded differently (e.g. fixed versus floating point or different sign representations). The format of a data value encoded into a data field may be different again and will have been preprocessed as well.

Time Series: A time series is a number of data values acquired from one sensor output at contiguous, equally spaced time intervals.

Byte: A number of contiguous bits in computer storage. Standardization in the computer industry has specialized the use of the term byte to mean eight contiguous bits. A byte is generally the smallest individually addressable unit of computer storage.

Record: A fixed length logical record of computer mass storage. Because of the organization of most modern mass storage devices, a moderately large power-of-two number of 8-bit bytes will provide both rapid access and compact storage.

Header: A fixed length section of user defined information located at the beginning of each record. Note that a portion of the header is reserved for information required by the compression algorithm.

Trailer: A fixed length section of compression algorithm defined information located at the end of the last record of a time series.

Nibble: A variable number of contiguous bits of computer storage (a generalized byte). Thus, while all bytes henceforth will be 8-bit bytes, nibbles may be 4-bits, 6-bits, 8-bits, etc. The largest nibble used will be 32-bits (the length of a standard integer (long integer, longword, etc.).

Data Field: A nibble containing one preprocessed, encoded data value.

Data Section: A number of contiguous data fields.

Key Value: An unsigned integer constant specifying the encoding of a data section.

Key Field: A nibble containing one key value.

Key Section: A number of contiguous key fields.

Frame: A key section and all of its corresponding data sections.

Block: A number of frames preceded by block header information and/or followed by block trailer information.

In USNSN compression, each time series is preprocessed and encoded into a number of 2048- byte (2^{11}) records. Note that this implies that all data is demultiplexed. Demultiplexing is required as it is the continuity of data from one source that make the first differencing preprocessing effective. All data values are presumed to be representable as fixed point numbers (integers) for encoding purposes. The length of the header is user definable but must be constant once set. Six bytes of the header are required by the algorithm for a 4-byte, 2's complement forward integration constant and a 2-byte, unsigned number of first differences encoded in the record (subsequently called the record sample count). In the USNSN telemetry scheme, there is the 14-byte general header, the 6-byte data header, the 6-byte compression header (for 26 bytes), and 10-bytes reserved for the X25 protocol. Therefore, there are 2012 bytes left for the compressed data (including keys, block delimiters, and the trailer). The trailer is 5-bytes long and includes a 1-byte, unsigned number of first differences encoded in the last frame (hereafter called the last frame sample count) and a 4-byte, 2's complement reverse integration constant. Note that the trailer appears only at the end of the last record of a time series and is used only for backwards decompression in the case that the last record has been corrupted.

With the exception of user defined information in the header, all information stored by the compression algorithm is in the form of fixed point numbers. All data values (integration constants and first differences) are represented in 2's complement notation and all control information (sample counts, key values, and back pointers) are unsigned.

If the time series is denoted by A(0), A(1), A(2),...A(n) then the preprocessed series can be denoted by A(0) plus B(1)=A(i)-A(i-1), i=1, 2,...,n, where A(0) is the first integration constant and A(n) is the reverse integration constant. Only the B(i) are encoded into the compression frames. If B(i),i=1, 2,...,m first differences will fit into the first record, then the record sample count (in the header) will be m. However, there will actually be m+1 data values in the record as the first integration constant is an independent datum. The forward integration constant in the second record will be A(m) and the first differences in the second record will begin with B(m+1). Note that integration constants after the first are redundant. They are provided to permit maximal data recovery should any portion of the compressed data be corrupted. They are also useful as consistency checks. Because the forward integration constant is redundant, the record sample count in the header of the second record is the actual number of data in the record. Succeeding records are encoded in the same way with the trailer (including the reverse integration constant) added to the end of the last record. USNSN time codes attached to compression blocks reflect the fact that the first record of a compression set will return one more data value

than subsequent packets. If it is necessary to begin decompression with a packet other than the first one, the time code must be corrected by minus one digitizing interval.

The compression algorithm encodes sequences of 4, 8 of 12 successive first difference values into a data section comprised of consecutive data fields of the same nibble size and assigns the data section a 4-bit key value. Nibble sizes of 4-bits, 6-bits, 8-bits, 10-bits, 12-bits, 14-bits, 16- bits, 20-bits, 24-bits, 28-bits, and 32-bits are supported. Note that data sections are an integral, but variable number of bytes long. The key section is made up of two keys and is thus, 1-byte long. A frame consists of one key section followed by the two corresponding data sections. A block consists of seven frames followed by a 1-byte block trailer containing a back pointer. The back pointer is an unsigned integer specifying the number of bytes in the block. The back pointer can be used to decompress backward (from the reverse integration constant) and it provides a consistency check when decompressing in the forward direction. All records except the last consist of a header section and as many blocks as will fit. The last block may contain less than seven frames, but should contain as many frames as will fit in the record. All frames, including the last, must be completely filled with data values. The remainder of the record, following the last block trailer, must be zero filled.

The key values and their meaning is as follows:

key value	number of data fields	nibble length (in bits)	data section length (in bytes)
0	4	4	2
1	8	4	4
2	12	4	6
3	4	6	3
4	8	6	6
5	4	8	4
6	8	8	8
7	4	10	5
8	8	10	10
9	4	12	6
10	4	14	7
11	4	16	8

As in all compression algorithms, encoding the last record poses special problems. For all records except the last, the forward integration constant for the following record provides a reverse integration constant. Also, the sum of the data field length (specified by the keys) and the record sample count (specified in the record header) are redundant. This redundancy is needed for backward decompression. It is possible because of the zero fill at the end of the record and the fact that the back pointers must be non-zero (thus, the last non-zero byte of each record, except the last, is the back pointer for the last block). Because, in general, the amount of data in the time series will not fill the last record, the time series must be padded with zero first differences in order to fill the last frame of the last block. Note that this may require the entire second data section of the last frame to be padding. In forward decompression, the record sample count in the header will specify the true last point in the original time series. For backward decompression, a trailer is added to provide the desired redundancy. The trailer consists of a 1-byte last frame sample count (the actual number of first differences, not padding, encoded in the last frame) and the reverse integration constant. The count is placed immediately after the back pointer for the last block and the reverse integration constant is placed in the last four bytes of the record. The byte between the count and the reverse integration constant must be zero.

NOTE: When data compression packets are sent over the network, the reverse integration constant is stored as the last four bytes of the packet. The receiving program must check for an 'end-of-file' bit in the packet and if it is found, move the last four bytes of the packet to the end of the compression buffer and zero fill any bytes between where the reverse integration constant was stored and where it is moved. Hence, if the compression block length is 2012 and a packet received via the satellite system is marked with end-of-file and indicates the compression block portion of the packet (i.e. the total packet length less non-compression block overhead) is 2006 bytes long, bytes 2003-2006 are copied to 2009-2012 and bytes 2003-2008 are zero filled. This is done so that packets going through the satellite system do not need to send the needless zero filled bytes. This is, of course, more important when the satellite packet is considerably shorter than a full packet.

Compressing data using the USNSN algorithm is straightforward. Each record is constructed in memory and then written out. The data is first differenced as needed and is analyzed in groups of four first differences to determine the nibble size that may be used. A four first difference look ahead is maintained at all times to determine whether a longer run can be encoded (for small nibble sizes) and if the record is full. Each time enough data is available, a frame is encoded. Every seven frames, the back pointer is computed and added. Determining the forward integration constants after the first is a little tricky due to the look ahead. Constructing the trailer is obvious. Problems with determining when a record is full are eliminated by the simplicity of the algorithm and the look ahead.

Forward decompression is also straightforward. Beginning with the first forward integration constant, frames are decoded and the first differences integrated to recover the input data values. At the end of each block, the back pointer should be checked for consistency. The number of first differences in the record can be used to determine when the last frame in the record has been processed. At the end of the record, the number of samples and the forward integration constant at the beginning of the next record should also be checked for consistency. In the last record, the number of first differences in the record should be used to determine both when the last frame has been reached and how many first differences in the last frame are data and not padding. The number of samples in the frame should be checked using the number of first differences in the last frame from the trailer and the reverse integration constant should be checked for consistency.

In fact, the presence of the number of first differences in the last frame count (which must be non-zero) is a flag for the last record (and hence, the end of the time series). Note that decompression may begin from any record in the time series if necessary.

Backward decompression is not quite as easy. In this case, decompression begins from the end of a record. If this record is the last record, the integration constant is available from the trailer. The number of first differences in the last frame count can be found by skipping over the zero fill. The next byte backwards will be the last block back pointer. The back pointer is used to compute the location of the start of the block. From this point all of the key fields in the block may be located and decoded (to give the location of the next key field and the number of first differences in the frame). In this way, the number of first differences in the block can be computed (using the actual number of first differences in the last frame from the count). Note that the number of frames can be determined from the back pointer position. In fact, if the last frame does not end on the byte before the back pointer position, then the block is inconsistent. Once the number of first differences is known, the data can be decoded and backwards integrated to recover the original data values. Successive blocks are decoded in the same way, but are somewhat easier as there will always be 7 frames in each block and the last frame count is not needed. If they are available, the number of first differences in the record and the forward integration constant in the header can be checked for consistency. If backwards decompression begins in a record that is not the last, the procedure is the same except that the forward integration constant in the next record must be used as the reverse integration constant. Also the last frame count will not be available, but will not be needed as the last frame will be full of data.

Note that the consistency checks are available to determine where the data may have been corrupted. If such a place is found, forward decompression cannot continue until the next uncorrupted header. Some of the data between the next header and the corrupted location can be recovered by backwards decompression. Care must be exercised, however, as the backwards decompression uses the back pointer, which may also be corrupted.

Appendix C: Specifying and Using Channel Response Information

Contributed by C.R. Hutt

Introduction

SEED volumes usually use complex-valued functions of frequency in response functions. Usually, these functions will not be single expressions, but rather the products of several expressions. Most seismic systems can be regarded as cascades of stages — for example, a seismometer, followed by an amplifier, followed by an analog filter, followed by an analog/ digital converter, followed by a digital filter. A blockette's stage sequence number shows the order of the stages, as shown in figure 1 below:

Figure 1: Example of a sequence of stages.

Before the age of high speed digital computers and digital signal processing (DSP) chips, all low-pass filtering (for the purpose of preventing aliasing) was performed in the analog stages before digitizing. The digitizer would operate at a fairly low sample rate equal to the sample rate being recorded. Typically, the corner frequency of the low-pass filter would be 1/8 to 1/20 of the sample rate (1/4 to 1/10 of the Nyquist frequency). Therefore, the low-pass anti-alias filter response would typically begin to attenuate at frequencies well within the band of interest.

Modern seismic data acquisition systems make use of over-sampling and decimation (to grossly over-simplify the inner workings of high resolution ADC's) to achieve high resolution. This technique relaxes the analog anti-alias filtering requirement and moves the low-pass filtering job into the digital domain. Decimation (sample rate reduction) must be preceded by sufficient low-pass filtering to prevent aliasing at the new lower sample rate. Many modern high resolu-

tion ADC's include two or more stages of Finite Impulse Response (FIR) filters to accomplish this task. These may be followed by further low-pass filter and decimate stages within the data acquisition computer to derive lower sample rate data streams (such as deriving Long Period data from Broadband data).

FIR filters are simply weighted averages of some number of data samples — the "weights" are the coefficients specified in Blockette (54) (for a "D" type stage). FIR filters are usually designed to approximate a "boxcar" response. That is, they typically have a very flat in-band response and a sharp, steep cut-off at their corner frequency, which may be set at 70% to 90% of the Nyquist frequency. In-band ripple is usually only a few percent. Also, FIR filters are usually designed to have linear phase, and the data acquisition systems usually time-tag the data so that the phase shift appears to be nearly zero.

All of this means that the average data user probably doesn't need to correct for the effect of such FIR filters. Examples of some common FIR filter amplitude responses for 20 sps data are shown in Figures 2 through 5 following. Note that in-band ripple can be several percent, but corner frequencies (-3 db points) are usually very close to the Nyquist frequency (which is 1/2 the sample rate). Also note that stop band gain can vary significantly: from -75 db to -120 db in these three examples.

It was previously stated that modern data acquisition systems using digital FIR filters usually time tag the data so that the filter delay (phase lag) appears to be nearly zero. Figure 2.B. contains a plot of the phase shift that results when the data are time tagged in such a way as to correct precisely for the theoretical filter delay (which is 1.625 seconds in this case). As is stated later in this Appendix, Blockette [57] should always be used when specifying a digital filter to completely describe how the time tag is applied. Some data acquisition systems (those installed through August 1992 by the USGS) correct for the FIR filter delays, resulting in near-zero phase shift, but did not specify this in a Blockette [57]. A data user may take the absence of Blockette [57] to mean that there exists a phase lag in the data that is really not there in these systems. That is, a Fourier transform of the FIR coefficients would indicate a pure delay, when in fact there is really no delay. The absence of Blockette [57] in USGS-supplied data will be remedied as soon as possible after August 1992. Anyone specifying digital filters in SEED format should always include the complete specification, including Blockette [57]

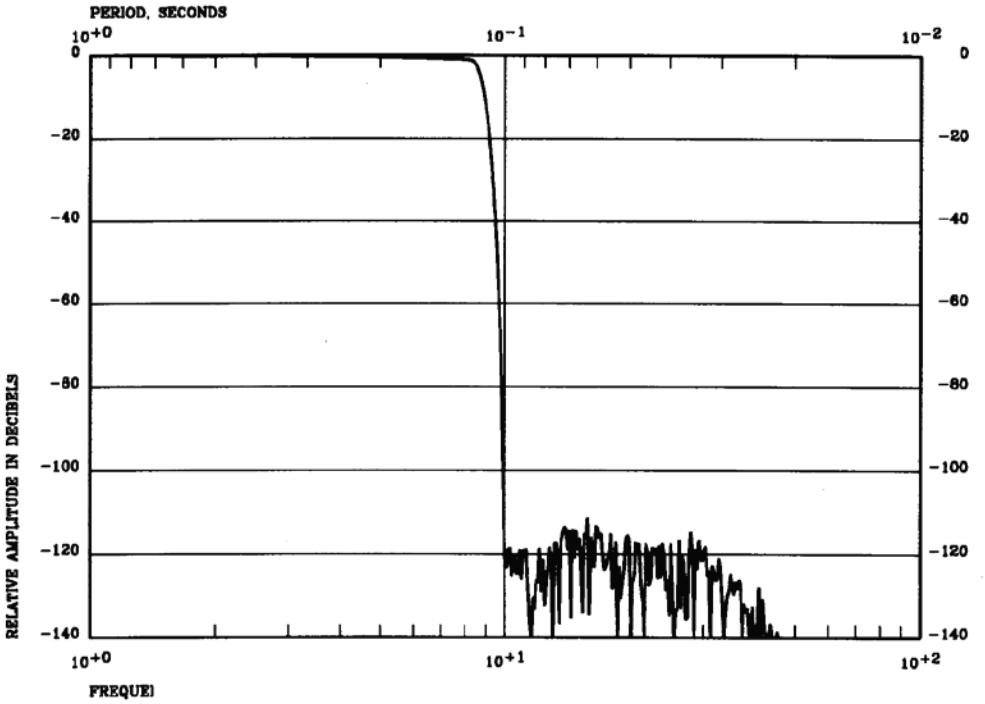


Figure 2A

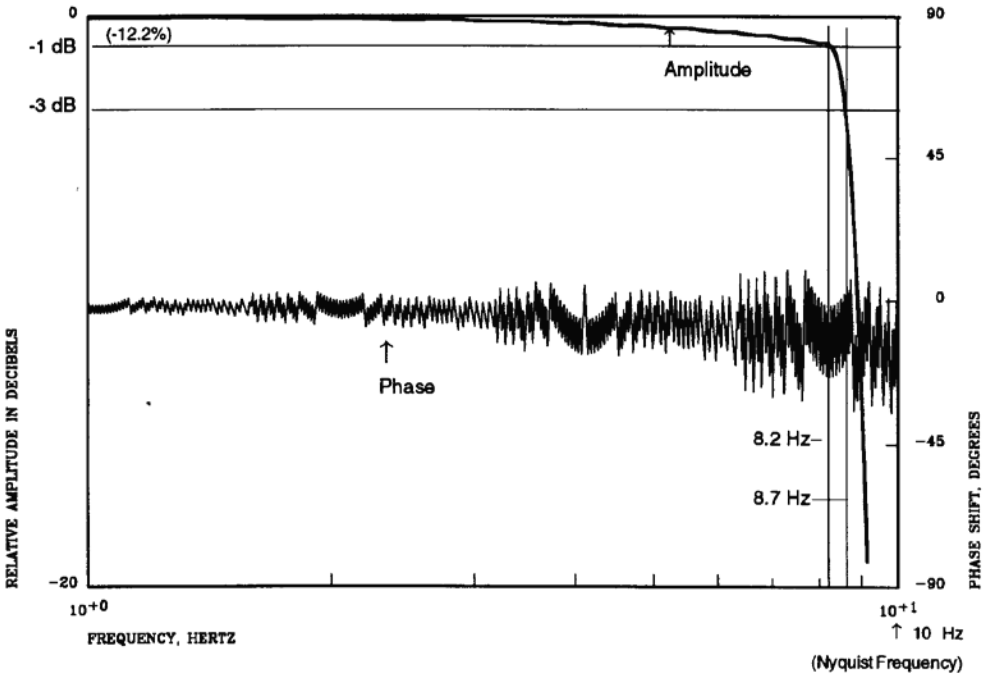


Figure 2b

Figure 2: Amplitude response of combined FIR filters used in Martin-Mariette digitizers of IRIS/USGS IRIS-2 systems, 20 sps (BB) data. Gain has been normalized to OdB at O Hz (DC). Note the different scales in the two figures above.

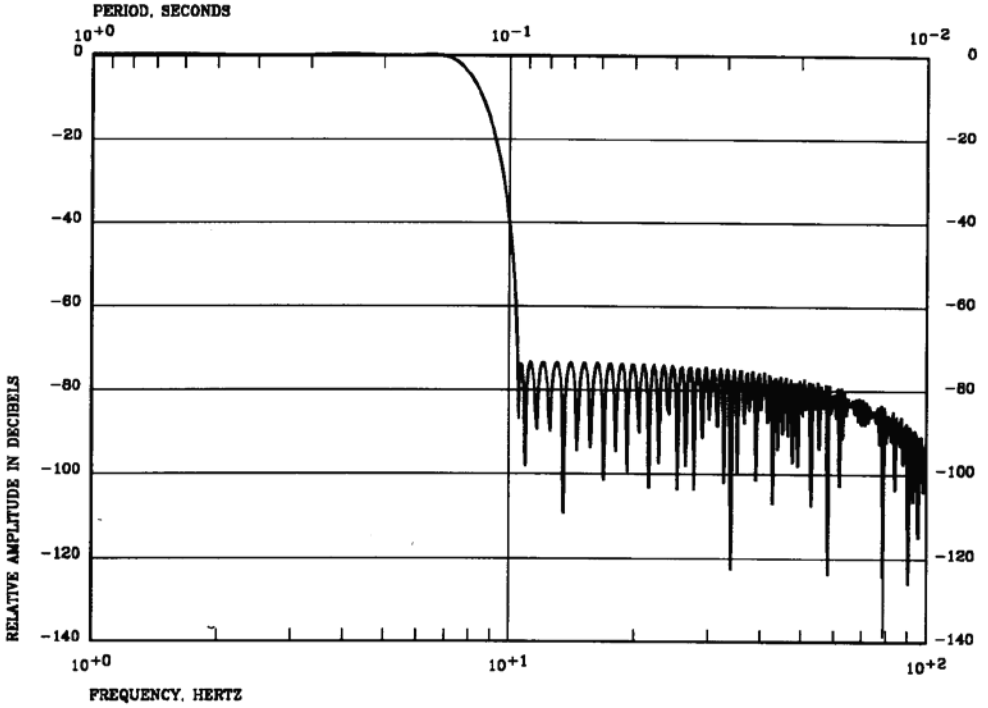


Figure 3A

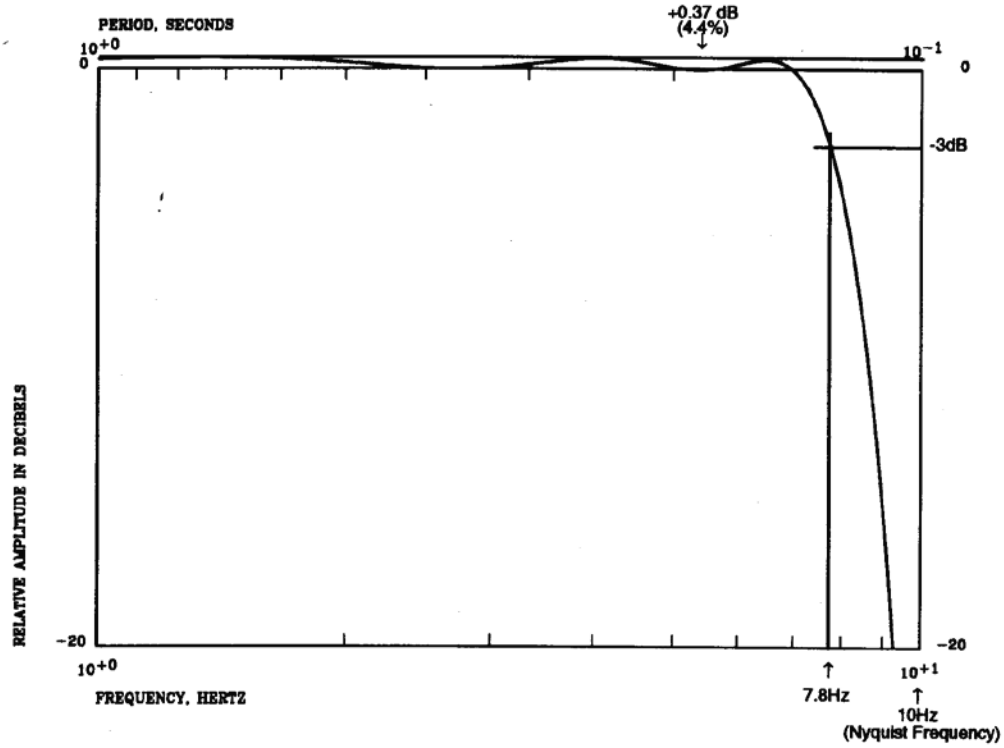


Figure 3B

Figure 3: Amplitude response of combined FIR filters used in "Quantagrator" model digitizers built by Quanterra, Inc. for 20 sps (BB) data. Gain has been normalized to Odb at OHz (DC).

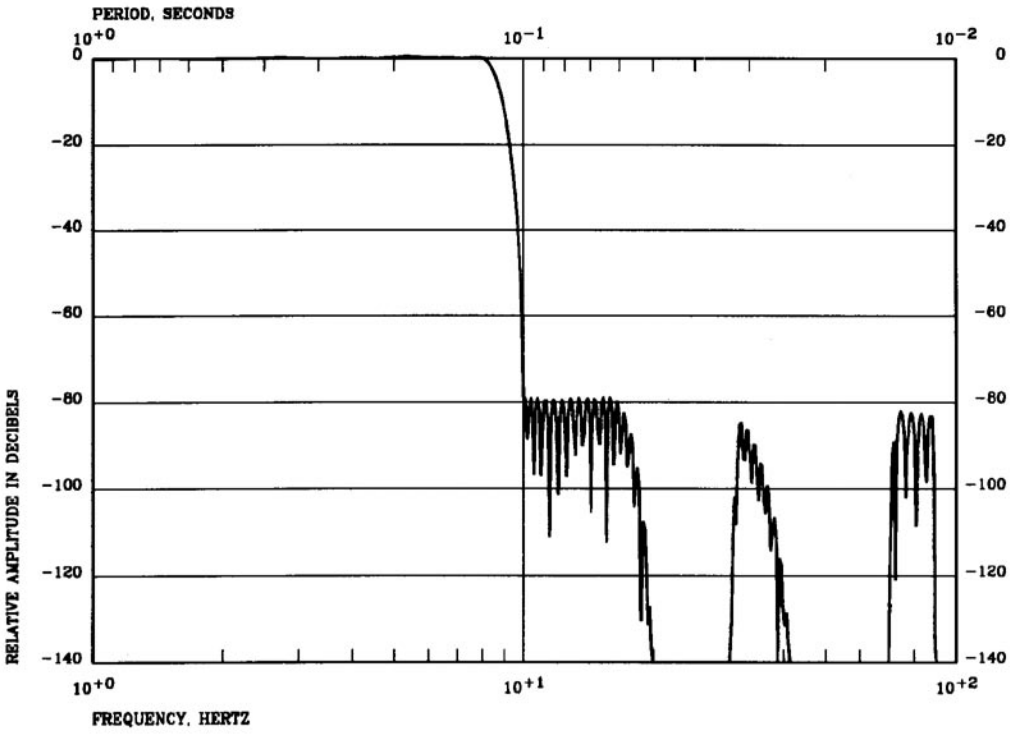


Figure 4A

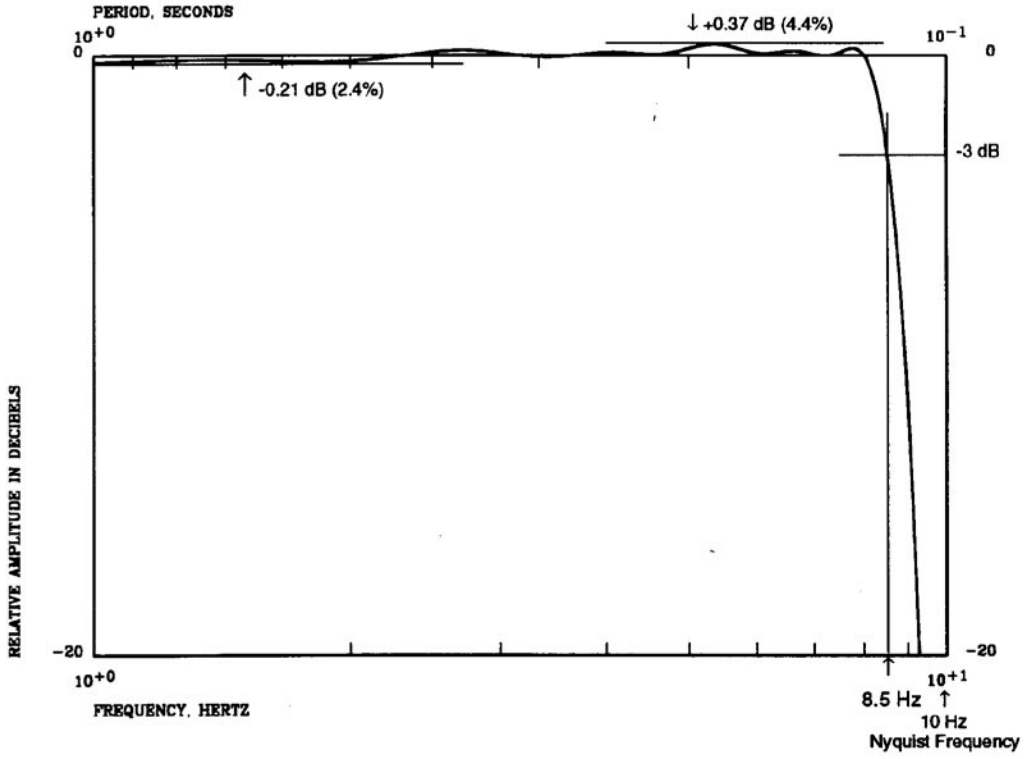


Figure 4B

Figure 4: Amplitude response of combined FIR filters used in Q380 and Q680 model digitizers built by Quanterra, Inc. for 20 sps (BB) data. Gain has been normalized to Odb at O Hz (DC).

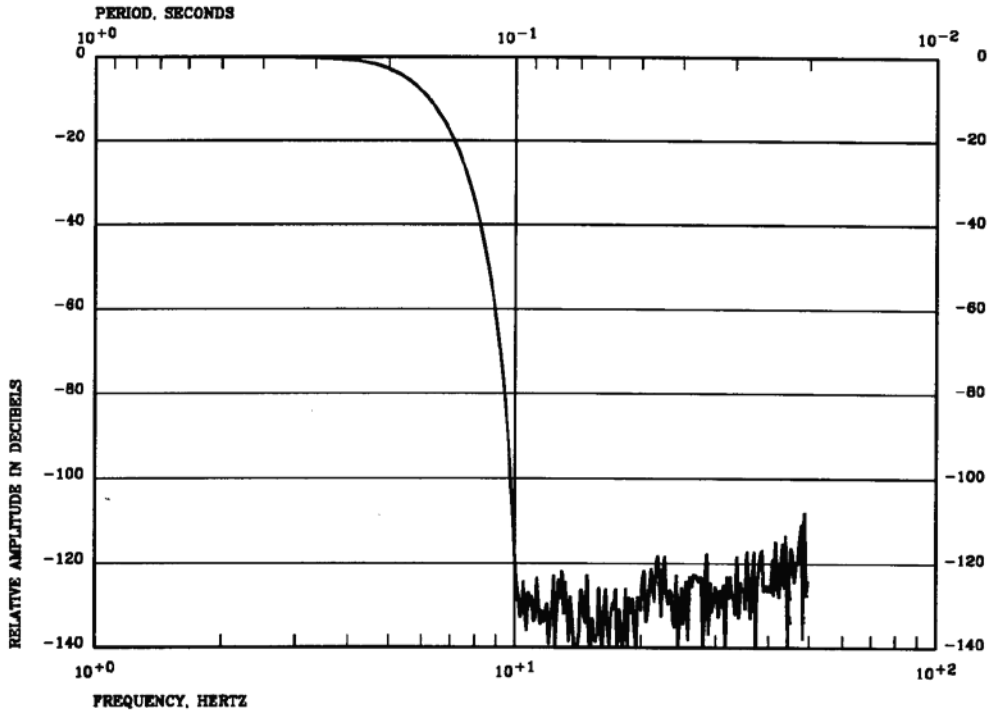


Figure 5A

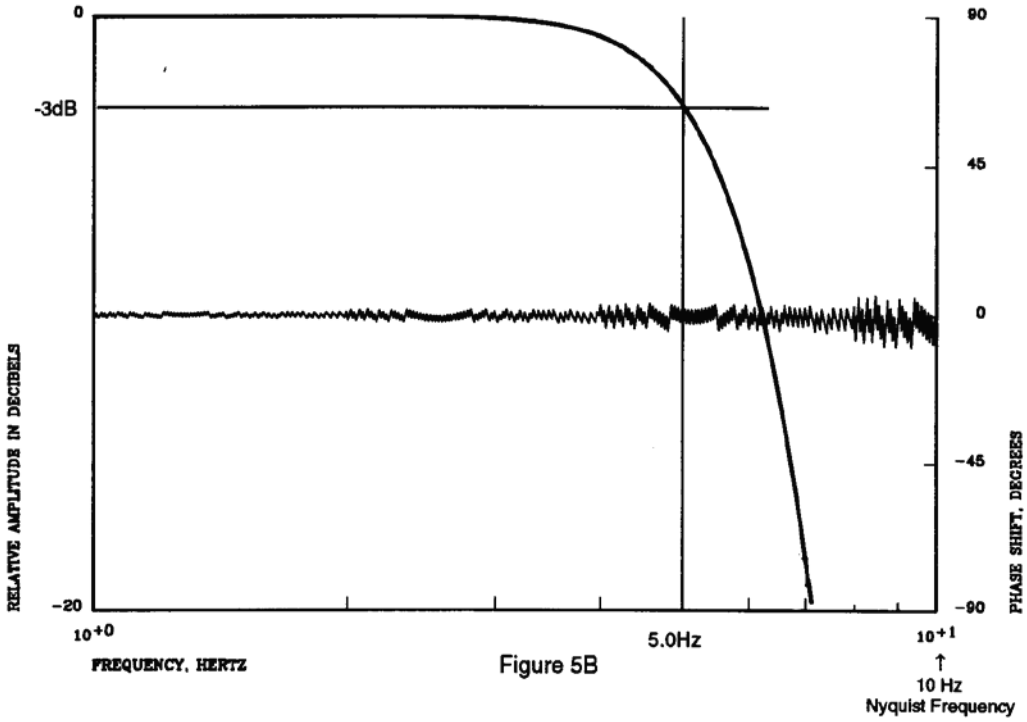


Figure 5B

Figure 5: Amplitude response of Ormsby FIR filter used in the Reftek 24-bit digitizers of the IRIS/IDA IRIS-3 systems. 20 sps (BB) data. Gain is 0 dB at 0 Hz (DC). Note the different scales in the two figures above.

In Figure 1, the seismometer response would have stage sequence number 1 and the digital filter would have stage sequence number 5. If there are K stages and the complex frequency response of the i-th one is G_i (f), the system response is:

$$\prod_{i=1}^{K} Gi\,(f) \qquad (1)$$

This appendix will show how to represent the stages (G_i's) using SEED blockettes. In Figure 1, each stage can be described by one or a combination of blockettes. Analog stages may be partially described by either Blockette [55] (Response List) or by Blockette [56] (Generic Response), but must also be described fully by using either the Poles and Zeros Blockette [53] or the Coefficient Blockette [54] along with [58] Channel Sensitivity/Gain Blockette:

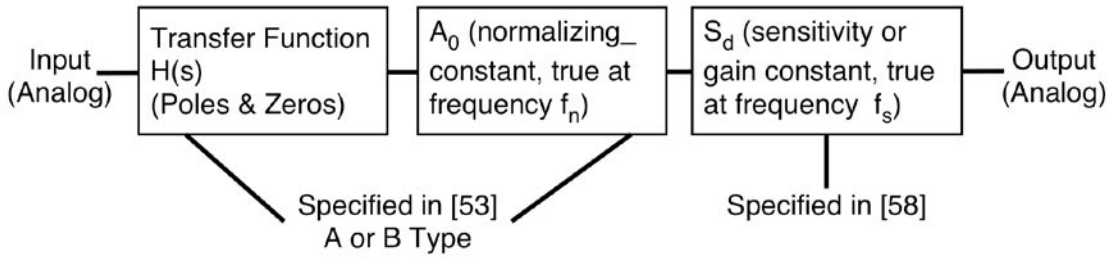


Figure 6: Example Analog Stage Using Poles and Zeros Representation

Note that A_o is chosen so that, at the normalization frequency, f_n, $|H(i2\pi f_n)| A_0 = 1.0$. Also note that it is most convenient, **and strongly recommended, that $f_n = f_s$.**

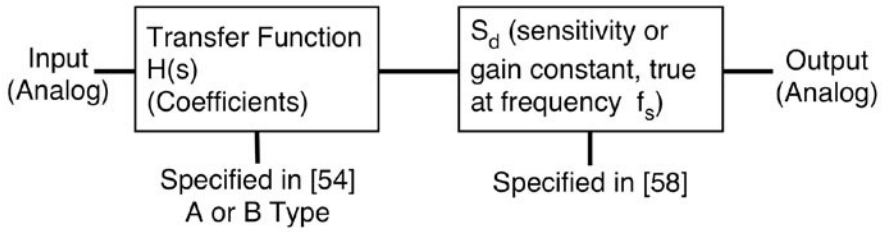


Figure 7: Example Analog Stage Using Coefficients Representation

Note that the coefficients of H(s) are chosen so that at the frequency of sensitivity f_s.$|H(i2\pi f_s)$.=1.0. Here f_s should be equal to f_s and f_n for all previous stages in the sequence, if possible.

Figure 8: Example Digital Stage Using Coefficients Representation

The coefficients are chosen so that at the frequency of sensitivity f_s, $|\boldsymbol{H}\,(e^{2\pi i f s \Delta t})| = 1.0$. Here, f_s should be equal to f_s and f_n for previous analog stages in the sequence, if possible. If the digital stage is a FIR filter, it is also convenient to use f_s = 0 Hz (DC), because the DC gain of a FIR filter is just the sum of the coefficients. However this should only be done if the DC gain is within 1% or 2% of the gain at f_s in previous stages.

Conventions

At any frequency, the modulus (absolute value) of the complex response function is the amplitude response of that stage. The phase of the complex response function is the phase response of that stage, with negative phase (output phase lagging the input) indicating a delay. Analog stages are represented by the Laplace transform of the linear system impulse response:

$$H(s) = \int_0^\infty h(t)\, e^{-st}\, dt \qquad (2)$$

h(t) is called the stage impulse response function, and its transform, H (s), is called the stage transfer function. H(s) may be specified in polynomial form (Blockette [54]) or in factored form (Blockette [53]).

Digital stages are represented by the Z-transform of the sampled time series corresponding to the stage impulse response:

$$\boldsymbol{H}(z) = \sum_{m=-\infty}^{\infty} h_m\, z^{-m} \qquad (3)$$

h_m is called the stage impulse response function, and its transform, $\boldsymbol{H}(Z)$, is called the stage transfer function. $\boldsymbol{H}(z)$ may be specified in polynomial form (Blockette [54], usually used for FIR filters) or in factored form (Blockette [53], usually used for IIR filters).

Normalization

For most stages, the frequency response is given in the form:

$$G(f) = S_d\ R(f) \qquad (4)$$

where R (f) is a function of frequency (usually complex-valued), specified by some combination of Blockettes [53], [54], [55], [56], and [57] (see below for which combinations are preferred for particular systems). R (f) is normalized so that $|R(f_s)| = 1.0$, where f_s is the frequency specified in Blockette [58]. S_d is the stage gain at that frequency. Using frequency response normalization helps by providing a check (you can compute $G(f_s)$ and make sure that it is indeed S_d), and by keeping track of the response functions of analog systems.

In cases where G(f) corresponds to an analog-type stage, a Poles and Zeros type response Blockette [53] is normally used to specify this stage. In this case, R(f) is expressed in this form:

$$R(f) = A_0\, H_p(s) \qquad (5)$$

where $s = i\, 2\, \pi\, f$ or $s = i\, f$ ($i = \sqrt{-1}$) as specified below equation (6) and $H_p(s)$ represents the transfer function ratio of polynomials specified by their roots, as in equation (6). For proper normalization, we chose A_0 such that $|R(f_s)| = 1.0$; that is $A_0 = 1/|H_p(s_s)|$, where $s_s = i 2\pi f_s$ $\frac{\text{rad}}{\text{sec}}$ or $s_s = i\, f_s$ (depending on whether we have represented the poles and zeros of Hp in terms of radians per second or Hz).

In cases where G(f) corresponds to an analog-type stage and the coefficient representation is used, as in equation (7), then the coefficients a_j and b_j of $H_c(s)$ are chosen such that $|H_c(s_s)| = 1.0$, where $s_s = i\, 2\, \pi\, f_s$ or $s_s = i\, f_s$ Hz.

When G(f) corresponds to a digital-type stage and is represented with poles and zeros, as is usually the case with IIR filters (those with feedback), we again chose $A_0 = 1/|\boldsymbol{H}_p(z_s)|$ where $\boldsymbol{H}_p(z)$ is defined as the ratio of polynomials in equation (11), and $z_s = e^{2\, \pi\, i\, f s \Delta t}$, where Δt is the sample interval and f_s is specified in the stage description.

Finally, when G(f) specifies a digital-type filter and is represented with coefficients, as is usually the case with FIR filters (those without feedback), the coefficients b_n of $\boldsymbol{H}_c(z)$ in equation (9) are chosen such that $|\boldsymbol{H}_c(z_s)| = 1.0$, where z_s is defined as in the previous paragraph.

This normalization works for stages 1 through K. If Blockette [58] has a stage number of 0, SEED assumes that the sensitivity S_d given in field 4 of Blockette [58] applies to the system as a whole, at the frequency f_s given in field 5 of Blockette [58]. Note that f_s should, if possible, be equal to the normalization frequency f_n given in any of stages 1 through K. In fact, within any stage, f_s should be equal to f_n. If no other stages are specified, SEED programs should conclude that this is our total knowledge of the system response. If we specify other stages, the stage-zero sensitivity will serve as a check on the sensitivity we can arrive at by multiplying together the responses G_1, …, G_K. In this case, the stage-zero sensitivity is not multiplied together with the gains of the other stages. Rather, the stage-zero sensitivity should be equal to the product of the gains of the other stages at frequency $f_s = f_n$. If we have <u>not</u> used the same frequencies, f_s and f_n, for all stages 1 through K, then we can only say that the product of the sensitivities for each stage may be <u>approximately</u> equal to the stage 0 sensitivity. Note that this idea is much more intuitive and easier to work with if f_s and f_n are the same for all stages.

A possible exception is when the stage is a low pass digital FIR filter. The stage sensitivity for a FIR stage may be stated at f_s = 0 Hz (DC) if the in-band ripple is less than say, 1 or 2%. The DC gain of an FIR filter is the sum of the coefficients and so is easy to calculate.

Analog Stages

The first part of any seismic sensor will be some sort of linear system that operates in continuous time, rather than discrete time. Usually, any such system has a frequency response that is the ratio of two complex polynomials, each with real coefficients. These polynomials can be represented either by their coefficients or by their roots (poles and zeros). The latter is the preferred mode, but either is acceptable.

Pole-Zero Representation for Analog Stages

The polynomials are specified by their roots. The roots of the numerator polynomial are the instrument zeros, and the roots of the denominator polynomial are the instrument poles. Because the polynomials have real coefficients, complex poles and zeros will occur in complex conjugate pairs. By convention, the real parts of the poles and zeros are negative, which leads to the form of function given below.

The fullest possible specification will utilize Blockettes [53] and [58]. Blockette [53] will specify N zeros, r_1, r_2,…, r_N, M poles p_1, p_2,…,p_M, a normalization factor A_0, and a reference frequency. The reference frequency is 1 radian/second if field 3 of Blockette [53] is the character A, and 1 Hz if field 3 of Blockette [53] is the character B. Blockette [58] will specify a scaling factor S_d. Then at any frequency f (in Hz), the response is:

$$G(f) = S_d A_0 \frac{\prod_{n=1}^{N} (s - r_n)}{\prod_{m=1}^{M} (s - p_m)} = Sd\ A0\ Hp\ (s) \qquad (6)$$

where $s = i\ 2\ \pi\ f$ if the reference frequency is 1 radian/second, and $s = i\ f$ if the reference frequency is 1 Hz.

Using two multiplicative coefficients, A_0 and S_d, in the equation above appears to be redundant, but we suggest that you partition the response by choosing A_0 so that the modulus of A_0 times the modulus of the ratio of polynomials equals 1.0 at the normalizing frequency f_n (also specified in Blockette [53]); the S_d specified in Blockette [58] is then the stage gain at that frequency, so $| G (f_n) | = S_d$. This division allows Blockette [53] to remain the same for many systems, with

the small differences between them expressed by the single number S_d in Blockette [58]. This simplifies keeping track of system responses. The "frequency of sensitivity factor" in Blockette [58] (f_s) should be the same as the normalizing frequency f_n in Blockette [53].

If Blockette [53] is omitted, SEED assumes that A_0 will be 1. This would be appropriate for an amplifier with no significant departure from a fixed gain S_d, or for a stage about which nothing was known but its gain at one frequency. SEED allows these combinations of blockettes for a stage of this type: [53], [58] or [58] by itself. Blockette [58] by itself would correspond to an amplifier with a flat response.

Coefficient Representation for Analog Stages

The polynomials are specified by their coefficients. The fullest possible specification will utilize Blockettes [54] and [58]. Blockette [54] will specify N+1 numerator coefficients, a_0, a_1,…, a_N, M +1 denominator coefficients b_0, b_1, …, b_M. Blockette [58] will specify a scaling factor S_d. Then, at any frequency f (in Hz) the response is:

$$G(f) = S_d \frac{\sum_{n=0}^{N} (a_n s^n)}{\sum_{m=0}^{M} (b_m s^m)} = S_d H_c(s) \qquad (7)$$

where $s = i\, 2\, \pi\, f$ if field 3 of [54] = A, and $s = i\, f$ if field 3 of [54] = B.

As in the pole-zero case, the coefficient S_d appears to be redundant, but the response should be partitioned as described above by choosing polynomial coefficients so that the ratio of polynomials have a magnitude of 1 at $f = f_s$, so that $| G(f_s) | = S_d$ at the frequency f_s (in this case specified only in Blockette [58]); the S_d specified in Blockette [58] is then the stage gain at that frequency.

If Blockette [54] is omitted, SEED will assume the ratio of polynomials equals 1.

SEED allows these combinations of blockettes for a stage of this type: [54], [58] or [58] by itself.

Analog-Digital Converter

This stage is the transition between the analog stage (for which the input units are ground behavior and the output some other analog signal, usually volts), and the purely digital stages. This stage has no frequency response (except for a possible delay between the sample-and-hold time and the time-tagging), but it does have a gain (in digital counts per analog unit in). Use Blockettes [54], [57], and [58] to specify the nature of this stage. In Blockette [54], fields 5 and 6 of Blockette [54] give the units involved; fields 7 and 10 of Blockette [54] should both be set to zero. In Blockette [57], field 4 gives the sample rate, with field 5 set to 1 to indicate that this is also the output sample rate. Fields 7 and 8 of Blockette [57] describe any empirically determined delays and applied time shifts respectively. (Use the delay field, field 7 of Blockette [57], only in this case.) In Blockette [58], field 4 gives the digitizer response (in counts/analog unit); and field 8 may be any frequency.

Note that it is acceptable (but discouraged) to combine the digitizer description with the first FIR stage. In this case, the input units would be volts and the output units would be counts.

Digital Stages

These stages operate on sampled data, and thus operate in discrete time rather than continuous time. All operations are digital, done to finite precision; however, SEED does not describe the level of precision actually used, and, for most purposes, all arithmetic is assumed to be done to infinite precision. In general, a digital stage will consist of:

1. A discrete-time filter, either FIR (finite impulse response, also called convolution filter), or IIR (infinite impulse response, also called recursive filter).
2. Resampling of the filter output to a new rate. Usually this rate is lower, in which case this operation is called decimation.
3. Time-shifting of the decimated series by assigning a time-tag to each value that corresponds not to the time at which it was computed, but to some other time. The difference between these times is the time-shift, which is usually non-positive (where the assigned time is earlier than the actual time) to minimize the phase shift introduced by the digital filter.

Coefficient Representation for Digital Stages

This type of stage is usually used to specify Finite Impulse Response (FIR) filters. In this type ($-\infty \le k \le \infty$) is convolved with the L+1 weights or coefficients $b_0, b_1, \ldots, b_L$ to produce the output series $y_{k:}$

$$y_k = \sum_{n=0}^{L} b_n x_{k-n} \tag{8}$$

Filters of this type are specified by Blockettes [54], [57], and [58] (or, in a special case, by using only Blockette [58]). Blockette [54] contains the weights b_n as the numerator coefficients. (There are no denominator coefficients in this case.) Blockette [57] specifies the input sample rate and the decimation factor. (Use a decimation factor of 1 if the output rate equals the input rate). Blockette [58] specifies a scaling factor, S_d. The transfer function for this filter is:

$$G(f) = S_d \sum_{n=0}^{L} b_n z^{-n} = S_d \boldsymbol{H}_c(z) \tag{9}$$

where the z-transform variable is $z = e^{2\pi i f \Delta t}$, with Δt = the input sample interval specified in Blockette [57], and f is the frequency in Hz.

Scale the coefficients b_n so that $|Hc(zs)| = 1.0$ where $z_s = e^{2\pi i fs \Delta t}$, f_s is specified in Blockette [58]. The S_d specified in Blockette [58] is then the stage gain at f_s.

If Blockette [53] is omitted, SEED will assume that the polynomial is 1.0; this would be appropriate for a pure multiplication.

Pole - Zero Representation for Digital Stages

This type of stage is usually used to specify Infinite Impulse Response (IIR) filters (those with feedback). In this type of digital filter, the input series x_k ($-\infty \le k \le \infty$) is convolved with the LB + 1 weights $b_0, b_1, \ldots, b_{LB}$; and past values of the output series y_k are convolved with the LA weights $a_1, a_2, \ldots, a_{LA}$, to produce the output value y_k:

$$y_k = \sum_{n=0}^{LB} b_n x_{k-n} - \sum_{n=1}^{LA} a_n y_{k-n} \tag{10}$$

The transfer function of this filter is:

$$H(z) = \frac{B(z)}{A(z)} = \frac{b_0 + b_1 z^{-1} + \ldots + b_{LB} z^{-LB}}{1 + a_1 z^{-1} + \ldots + a_{LA} z^{-LA}} \tag{11}$$

where the z-transform variable is $z = e^{2\pi i f \Delta t}$, with Δt = the input sample interval specified in Blockette [57].

Specify filters of this type with Blockettes [53], [57], and [58]. (Blockette [54] could be used to provide the coefficients, but because of the loss of precision possible in this case, we recommend not using it.) Blockette [53] will specify LB zeros, $r_1, r_2, \ldots, r_{LB}$; LA poles $p_1, p_2, \ldots, p_{LA}$, and a normalization factor A_0. The transfer function for the stage is:

$$G(z) = S_d A_0 \frac{\prod_{n=1}^{LB} (z - r_n)}{\prod_{m=1}^{LA} (z - p_m)} = S_d A_0 \boldsymbol{H}_p(z) \quad (12)$$

where z is as defined above. Choose A_0 so that $A_0 \cdot | \boldsymbol{H}_p(z_n) | = 1.0$, where $z_n = e^{2\pi i f_n \Delta t}$. Here f_n is the f_n from Blockette [53], and should be equal to the f_s in Blockette [58]. The S_d, specified in Blockette [58], is then the stage gain at that frequency, with $G(f_n) = G(f_s) = S_d$.

The zeros rn are the solutions of the equation:

$$b_0 + b_1 z^{-1} + \ldots + b_{LB} z^{-LB} = 0 \quad (13)$$

while the poles pm are the solutions of:

$$1 + a_1 z^{-1} \ldots + a_{LA} z^{-LA} = 0 \quad (14)$$

If Blockette [53] is omitted, A_0 will be considered to equal 1.0 (this would be appropriate for a pure multiplication).

Decimation

Blockette [57] specifies this operation. If the input series is y_m, the output series is w_n, with:

$$w_n = y_{Ln+l}, \; n = 0, 1, 2, \ldots \quad (15)$$

where L is the decimation factor and l is the offset (both are integers). The output sample interval is L times the input sample interval.

Time-shifting

As the data stream w_n emerges from the decimator, at time t_T each term is tagged (at least implicitly) with a nominal time t_N. Blockette [57] gives the time shift $\delta = t_N - t_T$ implied by this, in seconds. The effect of this time shift is to introduce a phase shift of $e^{i 2 p f \delta}$.

Examples

In the following three examples, we will assume we have a seismometer (Stage 1) followed by a digitizer (stage 2) followed by an FIR filter (Stage 3). We will then show an example Stage 0 specification summarizing these 3 stages.

Example of Specifying an Analog Stage 1.

Suppose we have a seismometer with a natural frequency f_o of 1 Hz ± 1%, a damping factor $\lambda = 0.7 \pm 3\%$, and a sensitivity of 150 volts per meter per second per second at 1 Hz. The acceleration transfer function would be (ignoring any constant gains):

$$H(s) = \frac{s}{s^2 + 2\lambda\omega_0 s + \omega_0^2} \quad (16)$$

There is one zero of H (s) at s = 0.The two poles of H (s) are at:

$$s = \lambda\omega_0 \pm i\,\omega_0\sqrt{1-\lambda^2} \quad (17)$$

In our example, $\omega_{0} = 2\pi \bullet (1)$ rad/sec, $\lambda = 0.7$, so we have the poles:

$$p_1 = -4.3982 + i\,4.4871 \quad (18)$$
$$p_2 = -4.3982 - i\,4.4871$$

and the zero:

$$r_1 = 0 + i\,0 \quad (19)$$

Note that both the real and imaginary parts of p_1 and p_2 may be in error by 4%, because f_0 was ±1% and λ was ±3%. However, it is known that both parts of r_1 are exactly 0. These errors are specified in Blockette [53], along with the real and imaginary parts of the poles and zeros. For this example, Blockette [53] would be filled out as follows:

Note	Field name	Type	Length	Mask or Flags	
1	Blockette type — 053	D	3	053	
2	Length of blockette			(length in bytes)	
3	Transfer function type	A	1	A	
4	Stage sequence number	D	2	01	
5	Stage signal input units	D	3	[M/S ** 2]*	
	*NOTE: What goes here is not “M/S **2”, but rather a 3 digit unit look-up code such as “004” that refers to the corresponding (field #3) code in Blockette [34], the Units Abbreviations Blockette where “M/S**2” is defined.				
6	Stage signal output units	D	3	[V]*	
	*NOTE: see note above				
7	AO normalization factor (1.0 if none)	F	12	+0.87964E+01	
	NOTE: See EXAMPLE OF CALCULATING AN ANALOG STAGE RESPONSE for information on how to calculate AO.				
8	Normalization frequency fn(Hz)	F	12	+0.10000E+01	
9	Number of complex zeros	D	3	001	
10	Real zero	F	12	+0.00000E+00	} From (19)
11	Imaginary zero	F	12	+0.00000E+00	
12	Real zero error	F	12	+0.00000E+00	} Because both parts of r1 are exactly zero
13	Imaginary zero error	F	12	+0.00000E+00	
14	Number of complex poles	D	3	002	
15	Real pole #1	F	12	-0.43982E+01	} P1 from (18)
16	Imaginary pole #1	F	12	+0.44871E+01	
17	Real pole error #1	F	12	+0.17593E+00 (= 4% of 0.43982E+01)	
18	Imaginary pole error #1	F	12	+0.17948E+00 (= 4% of 0.44871E+01)	
15	Real pole #2	F	12	-0.43982E+01	} P2 from (18)
16	Imaginary pole #2	F	12	+0.44871E+01	
17	Real pole error #2	F	12	+0.17593E+00 (= 4% of 0.43982E+01)	
18	Imaginary pole error #2	F	12	+0.17948E+00 (= 4% of 0.44871E+01)	

Note that the errors listed are positive, but represent a plus/minus (±) error expressed in the same units (either rad/sec or Hz) as the units of the real and imaginary parts of the pole listed in fields 15 and 16. Blockette [58] would be filled as follows:

Note	Field name	Type	Length	Mask or Flags
1	Blockette type — 058	D	3	058
2	Length of blockette	D	4	(length in bytes)
3	Stage sequence number	D	2	01
4	Sensitivity/gain (Sd)	F	12	+0.15000E+03
5	Frequency (Hz) (fs)	F	12	0.10000E+01 NOTE: fs = fn of Blockette [53].
6	Number of history values	D	2	00

This blockette stops here because there are no history values.

Example of Specifying a Digital Stage 2

ANALOG DIGITAL CONVERTER:

Suppose the analog-to-digital converter (ADC) we are using is a 24-bit ADC for which full scale is ±20v = $\pm 2^{23}$ counts. We use Blockettes [54], [57], and [58] to describe this stage. We assume the ADC is producing 40 samples per second. Blockette [54] would be filled out as follows:

Note	Field name	Type	Length	Mask or Flags
1	Blockette type — 054	D	3	054
2	Length of blockette	D	4	(length in bytes)
3	Response type	A	1	D
4	Stage sequence number	D	2	02
5	Signal input units	D	3	[V] (by reference)
6	Signal output units	D	3	[counts] (by reference)
7	Number of numerators	D	4	0000
	REPEAT fields 8 — 9 for the Number of numerators:			
8	Not Present			
9	Not Present			
10	Number of denominators	D	4	0000
11	Not Present			
12	Not Present			

Blockette [57] (We need this one to specify the sample rate.)

Note	Fieldname	Type	Length	Mask or Flags
1	Blockette type	D	3	057
2	Length of blockette	D	4	(Length in bytes.]
3	Stage sequence number	D	2	02
4	Input sample rate(Hz)	F	10	0.4000E+02
5	Decimation factor	D	5	00001
6	Decimation offset	D	5	00000
7	Estimated delay (seconds)	F	11	+0.0000E+00
8	Correction applied (seconds)	F	11	+0.0000E+00

Blockette [58]:

Note	Field name	Type	Length	Mask or Flags
1	Blockette type — 058	D	3	058
2	Length of blockette	D	4	(length in bytes)
3	Stage sequence number	D	2	02
4	Sensitivity/gain (Sd)	F	12	+4.19430E+05*
		*NOTE: This number equals 223 counts divided by 20 volts, in this case.		
5	Frequency (Hz) (fs)	F	12	+0.10000E+01*
		*NOTE: We have specified the sensitivity at fs = 1 Hz, the same frequency at which we specified the seismometer sensitivity and the same frequency at which we calculated the normalization constant AO in stage 1.		
6	Number of history values	D	2	00

We end the blockette here if there are no history values.

Example of Specifying a Digital Stage 3

FIR FILTER:

Suppose the ADC in stage 2 is followed by a simple running 2-point averager, and we throw away every other sample (decimate by 2) at the output of this stage 3. A simple 2- point average is an example of a FIR filter, with both coefficients equal to 0.5. Suppose further that we want the gain of this FIR filter to be 2 at 0 Hz. One way to accomplish this in a real implementation is to let both of the coefficients be equal to 1.0 instead of 0.5. (This results in a gain of 2.00 at 0 Hz, or a gain of 1.9938 at 1 Hz. See "EXAMPLE OF CALCULATING A DIGITAL STAGE RESPONSE" below for an example of calculating this gain.)

In the form of equation (8), this FIR filter may be written as

$$\begin{aligned} y_0 &= b_0 x_0 + b_1 x_{-1} = b_0 x_0, \text{ (assume } x_{-1} = 0) \\ y_1 &= b_0 x_1 + b_1 x_0, \\ &\vdots \quad\quad \vdots \quad\quad \vdots \end{aligned} \tag{20}$$

where $b_0 = 1.0$ and $b_2 = 1.0$. The decimation process would keep y_0, throw away y_1, keep y_2, and so on. The delay of this filter would appear to be about one-half of the original sample interval of 0.025 seconds, or 0.0125 seconds (the midway point of a plot of the symmetrical coefficients).

We would specify this stage 3 FIR filter by using Blockettes [54], [57], and [58]. Since Blockette [58] needs to specify the gain separately, assuming that the coefficients listed in Blockette [54] have been normalized to produce a gain of 1.00 at $f_s = 1$ Hz, we list $b_0 = b_1 = 0.50155 = 1/1.9938$.

Blockette [54]

Note	Field name	Type	Length	Mask or Flags
1	Blockette type — 054	D	3	054
2	Length of blockette	D	4	[Length in bytes.]
3	Response type	A	1	D
4	Stage sequence number	D	2	03
5	Signal input units	D	3	[counts] (by reference)
6	Signal output units	D	3	[counts] (by reference)
7	Number of numerators	D	4	0002
8	Numerator coefficient #1	F	12	+0.50155E+00(b0)
9	Numerator error #1	F	12	+0.00000*
		*(error in b0 -- assume zero for accurately stored digital values.)		
8	Numerator coefficient #2	F	12	+0.50155E+00 (b1)
9	Numerator error #2	F	12	+0.00000 (error in b1)
10	Number of denominators	D	4	0000*
		*NOTE: Even though we list zero denominator coefficients for FIR filters, we assume that there is a non-zero denominator value of 1.0, to avoid division by zero, when evaluating the filter transfer function.		

Blockette [57]

Note	Field name	Type	Length	Mask or Flags
1	Blockette type — 057	D	3	057
2	Length of blockette	D	4	[Length in bytes.]
3	Stage sequence number	D	2	03
4	Input sample rate (Hz)	F	10	0.4000E+02
5	Decimation factor	D	5	00002 (we are throwing away every other sample)
6	Decimation offset	D	5	00000 (we are keeping the first sample)
7	Estimated delay (seconds)	F	11	+0.1250E-01
8	Correction applied (seconds)	F	11	-0.1250E-01*
				*NOTE: We are assuming here that the data acquisition system is time tagging the data at the output of this FIR filter in such a way as to correct for the estimated delay.

Blockette [58]

Note	Field name	Type	Length	Mask or Flags
1	Blockette type — 058	D	3	058
2	Length of blockette	D	4	[Length in bytes.]
3	Stage sequence number	D	2	03
4	Sensitivity/gain (S_d)	F	12	+0.19938E+014
5	Frequency (Hz) (f_s)	F	12	+0.10000E+01*
				*NOTE: We are again quoting the gain at the same frequency as in previous stages. We could also have quoted the gain as 2.00 at 0 Hz, because it is within 1% of the gain at 1 Hz. (The gain is easy to calculate at 0 Hz because it is just the sum of the coefficients b_i.)
6	Number of history values	D	2	00

Example Stage O Specification

Blockette [58] may be used to summarize the overall (stages 1 through 3 in this case) gain, or system sensitivity, at a given frequency. It is best to specify this sensitivity at the same frequencies f_s and f_n used in the previous stages. Then the stage 0 sensitivity should be equal to the product of the stage 1 through K sensitivities, if there are K stages in total.

For our 3-stage example, Blockette [58] for stage 0 should be filled in as follows:

Note	Field name	Type	Length	Mask or Flags	
1	Blockette type — 058	D	3	058	
2	Length of blockette	D	4	[Length in bytes.]	
3	Stage sequence number	D	2	00	
4	Sensitivity/gain (Sd)	F	12	+1.25439E+084	
				*Note: This sensitivity is assumed to be expressed in counts per m/s * * 2; that is, in terms of output units for stage K per input units for stage 1 at fs = 1 Hz. In this case, it is equal to	

$$150\ \frac{V}{m/s^2} \cdot 4.19430E+05\ \frac{Count}{V} \cdot 0.19938E+01\ \frac{Count}{Count} = 1.25439E+08\ \frac{Count}{m/s^2}$$

Stage 1 gain · Stage 2 gain · Stage 3 gain

5	Frequency (Hz) (fs)	F	12	+0.10000E+01	(21)
6	Number of history values	D	2	00	

Example of Calculating Analog Stage 1 Gain and Phase

For our 1 Hz seismometer in the stage 1 example given, we have, using the form in equation (6):

$$H_p(s) = \frac{s + 0}{(s + 4.3982\ i\ 4.4871)\ (s + 4.3982 - i\ 4.4871)} \quad (22)$$

$$A_0 = 0.87964E+03 \ @ \ f_n = 1 \text{ Hz} \quad (23)$$

$$S_d = 0.15000E+03 \ @ \ f_s = 1 \text{ Hz} \quad (24)$$

How did we find A_0? To evaluate H_p (s) at $f_n = 1$ Hz, we substitute for s the value $s = i\omega_n = i\ 2\ \pi\ f_n$, and then calculate the modulus of H_p ($i\ 2\ \pi\ f_n$):

$$| H_p (i\ 2\ \pi \cdot f_n) | = \left| \frac{0 + i\ 2\ \pi \cdot fn}{[4.3982 + i\ (2\ \pi\ f_n + 4.4871)]\ [4.3982 + i\ (2\ \pi\ f_n - 4.4871)]} = 0.11368 \right|_{f_n = 1} \quad (25)$$

Then

$$A_0 = \frac{1}{0.11368} = 8.79640 \quad (26)$$

Of course, equation (25) may be used to evaluate H_p(s) at any frequency other than f_n. The phase of H_p (s) at f may be obtained by:

$$\text{Ø}\ (f) = \tan^{-1} \left(\prod_{n=0}^{N} \frac{\text{Im}\ (s - r_n)}{\text{Re}\ (s - r_n)} \right) - \tan^{-1} \left(\prod_{m=0}^{M} \frac{\text{Im}\ (s - p_m)}{\text{Re}\ (s - p_m)} \right) \Bigg|_{s = i\ \pi\ f} \quad (27)$$

Where "Im" denotes the imaginary part of the argument and "Re" denotes the real part.

The symbol $\rfloor$ means everything to the left of the symbol is evaluated at the equation that follows it.

Example of Calculating Digital (FIR Filter) Stage Gain and Phase

For the FIR filter in the stage 3 example above, we have $b_1 = b_0 = 0.50155$ and $S_d = 1.9938$ at $f_s = 1$ Hz. Using the form of equation (9), the transfer function of this filter is

$$G(f) = S_d \sum_{n=0}^{L} b_n z^{-n} = S_d \boldsymbol{H}_c (z) = 1.9938\ (0.50155\ z0 + 0.50155\ z\text{-}1) \quad (28)$$

So

$$\boldsymbol{H}_c (z) = 0.50155 + 0.50155\ (z^{-1}) \quad (29)$$

In order to evaluate $\boldsymbol{H}_c$ (z), we substitute $z = e^{2\ \pi\ i\ f\ \Delta t}$, where f is the frequency at which we wish to evaluate $\boldsymbol{H}_c$ (z) and Δt is the sample interval, defined as the inverse of the sample rate listed in Blockette [57] for the stage. Using f = 1 Hz and Δt =1/40 sec = 0.025 secs, we have

$$\boldsymbol{H}_c (e^{2\ \pi\ i\ (1)\ (.025)}) = 0.50155\ (1 + e^{-2\ \pi\ i\ (1)\ (.025)}) \quad (30)$$

Using the indentity

$$e^{i\Theta} = \cos \Theta + i \sin \Theta \tag{31}$$

Equation (30) can be written

$$Hc\Big|_{f=1} = 0.50155 \{1 + \cos [-2 \pi (1) (.025)] + i \sin [-2 \pi (1) (.025)]\} \tag{32}$$

So the real part of $\boldsymbol{H}_c$ at f = 1 is

$$Re(\boldsymbol{H}_c)\Big|_{f=1} = 0.50155 \{1 + \cos (-.05 \pi) \} = 0.996925 \tag{33}$$

Ant the imaginary part of $\boldsymbol{H}_c$ at f = 1 is

$$Im(\boldsymbol{H}_c)\Big|_{f=1} = 0.50155 \{ \sin (-.05\pi \} = 0.07846 \tag{34}$$

The magnitude of $\boldsymbol{H}_c$ at f = 1 is then

$$| \boldsymbol{H}_c |_{f=1} = \{[Re(\boldsymbol{H}_c)]_{f=1}^2 + [Im (\boldsymbol{H}_c)]_{f=1}^2\}^{1/2} = \{(.996925)^2 + (0.07846)^2\}^{1/2} = 1.00000 \tag{35}$$

So we see that the coefficients $b_0 = b_1 = 0.50155$ chosen in the example really did normalize the magnitude of $\boldsymbol{H}_c$ (z) to a value of 1.0 at f = 1 Hz.

How did we know to choose the coefficients to be $b_0 = b_1 = 0.50155$? If we express $\boldsymbol{H}_c$ (z) in equation (29) in its more general form we have:

$$\boldsymbol{H}_c (z) = b_0 + b_1 z^{-1} \tag{36}$$

Equation (30) then becomes:

$$\boldsymbol{H}_c (e^{i 2 \pi f \Delta t}) = b_0 + b_1 ^{-i 2 \pi f \Delta t} \tag{37}$$

and (32) becomes

$$\boldsymbol{H}_c\Big|_{f=1} = b_0 + b_1 \cos [-2 \pi f \Delta t] + \sin [-2 \pi f \Delta t]\}_{f=1} \tag{38}$$

If we then substitute in the actual FIR filter coefficient values of $b_0 = b_1 = 1$ from our example, we find that actual magnitude of $\boldsymbol{H}_c$ at f = 1 is

$$\text{Actual } | \boldsymbol{H}_c |_{f=1} = 1.9938 \tag{39}$$

To normalize the coefficients b_i so that the resulting $\boldsymbol{H}_c$ has a value of 1.0 at $f = f_s = 1$ then, we must divide all of the bi by this actual magnitude value in equation (39). These new values of bi are then used in Blockette [54]:

$$\text{New } b_0 = \frac{\text{Actual } b_0}{1.9938} = 0.50155$$

$$\text{New b1} = \frac{\text{Actual } b_1}{1.9938} = 0.50155 \tag{40}$$

Note that this step of normalization before entry of the coefficients into the SEED blockettes is equivalent to the introduction of the A_O normalization constant for analog stages (A_O is the inverse of $| H_p (i 2 \pi f_n) |$).

If we write equation (37) for L + 1 terms we have

$$\boldsymbol{H}_c\,(e^{i\,2\pi\,f\,\Delta t}) = b_0 + b_1\,e^{-i\,2\pi\,f\,\Delta t} + b_2\,e^{-i\,2\pi\,f\,\Delta t} + \ldots + b_L\,e^{-i\,2\pi\,L\,f\,\Delta t} \qquad (41)$$

If we now let f = 0 in equation (41), we see that the magnitude of $\boldsymbol{H}_c$ is just the sum of the coefficients:

$$\boldsymbol{H}_c\,(e^0) = b_0 + b_1 + \ldots b_L \qquad (42)$$

Appendix D:
The Data Description Language

Contributed by Ray Buland and Scott Halbert

Most existing data distribution formats limit data producers to creating only one of a few data formats. Producers must contrive to produce data in the intended format, or convert data from the native format into the distribution format. Conversion is generally unpalatable because it is costly to perform and results in a format that is either less compact or less precise than the original format. Also, some data problems can only be dealt with in the original recording format. Unfortunately, adding new recording formats creates problems for data users as knowledge of the supported recording formats is usually implied.

The data description language or DDL used with SEED lets the data producer use the native data format by describing it in an unambiguous language that will ultimately drive a data parser and disassembler. The data producer may then place data directly into the SEED format with much less processing and manipulation.

Data Format Dictionary Blockette [30] uses the data description language. One blockette [30] must appear in the Abbreviation Dictionary Control Header for each different data format appearing on the volume. As many different formats may be defined as are needed. When defining the Channel ID Blockette [52], the unique number of the correct Data Format Dictionary Blockette must be placed in the Data Format ID Code field.

The actual language is composed of several records, called keys. Each key describes some aspect of the language for that family. Each family has its own arrangement and interpretation of keys. A key is made up of different fields that contain the actual parser information. A field is typically a single letter code, followed by numeric parameters determined by punctuation. Numeric parameters are always base 10. Braces ("{" and "}") in the fields imply optional sections. Two special codes specify exponentiation: #n denotes 2^n, and %n denotes 10^n. Keys are separated by tildes; fields within keys are separated by spaces. White space should not be embedded within fields (see the listings of Blockette [30] in Appendix E for sample data).

Several different data families are supported, including integer, gain ranged, integer differences compression, and text. For each family, the keys are arranged in a logical sequence from the least specific to the most specific. For example, for integer and gain ranged formats (families 0 and 1), key 1 describes the multiplexing of data from different channels. Within this multiplexing scheme, key 2 describes how to extract and interpret the subsequent bits as a signed, fixed point number. For integer formats, this value is the datum. For gain-ranged formats, it is the datum mantissa and keys 3 and 4 describe how to acquire and interpret the characteristic. In both cases, it is implied that the rules for extracting and interpreting the bit stream will be repeated until the number of samples specified in the Fixed Section of the Data Header have been converted.

For the integer differences compression (family 50), key 1 gives instructions on accessing the integration constants, key 2 shows how to interpret the compression keys, and keys 3-m describe how to decompress data for all possible compression key values. Optional key m+1 describes the grouping of frames (if necessary). In this case, keys 2-m provide a complete description for interpreting all of the data packed into one compression frame. The interpretation of compression frames is then repeated until a group of frames is completed. The interpretation of groups of frames is repeated until all data values have been decompressed. Note that multiplexing is not currently supported for compression formats.

The text formats (families 80, etc.), describe how to interpret free form text material embedded into a data record (such as a console log). The interpretation is straightforward as shown below, primarily because the character codes used to represent text are generally byte aligned and already have their own well-publicized international standards.

All of the binary data types (families 0, 1, and 50) rely on the fundamental operations of copying the next group of bits into a temporary “working buffer” and then interpreting subsets of these bits as signed fixed point numbers. Because this basic set of operations is family and key independent, it will be described here. The description of the family dependent keys will be given with each family. The fields defined for these extraction primitives fall into four groups: 1) copy and reorder bytes or bits from the input data stream into a temporary “working buffer”, 2) extract a subset of bits from the working buffer with optional scaling and offset, 3) add sign information, and 4) miscellaneous operations. The extraction primitive fields are common to all binary data type families and should not be redefined in these families.

By convention, all DDL fields operate on bytes and bits in big-endian order as described elsewhere in this manual. That is, bytes are encountered in big-endian or most significant byte first while bits are numbered in little endian order with the least significant bit (LSB) in a word being numbered zero. Thus, big-endian bit numbering is only meaningful relative to a particular word length, say in consecutive bits, in which case these bits are numbered 0, 1, 2, …, n+1 from the LSB to the most significant bit (MSB). For the purpose of the copy/reorder operations, the binary data portion of a data record is considered to be a stream of bytes that are processed in the order encountered. Alternatively, the byte stream can be considered to be a bit stream. In this case, the first bit in the bit stream will be the MSB of the first byte. Successive bits are taken from the MSB to the LSB with the LSB of the first byte followed by the MSB of the second byte and so forth. Note that bits in the bit stream are encountered in the opposite order from that implied by the big-endian bit numbering within each byte. For example, if both the bytes and bits in the byte/bit streams were numbered consecutively from zero we would have:

Bit Stream Bit Order

bit stream bit #	byte stream byte #	big-endian bit # within each byte
0	0	7
1	0	6
:	:	:
7	0	0
8	1	7
9	1	6
:	:	:

Extraction Primitives

These primitives are used in the binary data families to interpret fixed point values from the data stream.

Copying/Reordering Primitives:

Wx{,n{,...}}

Wx{,n{,...}} - x bytes are copied from the input data stream:

Copy the next x 8-bit bytes from the input data stream into the working buffer, optionally reordering them as specified by the n's. If the n's are given, there must be x of them specified. The first n specifies which byte to take from the data stream first, the second n specifies which byte to take second, etc. The next x bytes in the byte stream are numbered 0, 1, 2, ..., x-1 for the purpose of specifying the n's.

For example:

W4, 3, 2, 1, 0: Copy 4 bytes reversing them end-for-end in the working buffer. This would reorder a little-endian longword into big-endian order.

W3: Copy 3 bytes into the working buffer without reordering. This is equivalent to W3, 0, 1, 2

Bx{,t{,n{-m},...}}

Bx{,t{,n{-m},...}} - x bits are copied from the input data stream:

t=0: Groups of bits crossing byte boundaries take the remaining bits from the left side of the next byte. This implies 68000 bit order (as described above).

t≠1: Reserved for future use.

Copy the next x bits from the input data stream into the working buffer, optionally reordering groups of bits as specified by the "n-m's". If the "n-m" specifications are given, they must reference each bit copied to the working buffer once and only once. Omitting the "-m" implies "n-n". Note that for the purpose of specifying the n-m's, the bits are numbered in Motorola standard order relative to the x-bit long nibble just copied (i.e. the first bit copied will be bit number x-1 and the last will be bit number 0.

For example:

B3, 0, 0-1, 2-3, 4-5: Copy six bits into the working buffer, last two bits first, middle two bits next, and first two bits last.

B4, 0, 0, 1, 2, 3: Copy four bits into the working buffer in reverse order.

B5: Copy 5 bits into the working buffer without reordering. Equivalent to B5, 0 or B5, 0, 4, 3, 2, 1, 0 or B5, 0, 4-4, 3-3, 2-2, 1-1, 0-0

Extraction primitive:

D{n{-m}} {:b{:o:a}}

D{n{-m}} {:b{:o:a}} - extract bits n-m from the working buffer to form unsigned integer value k and then apply offset a and scale factors b as specified by code o.

If o=0, then add a to k and multiply the sum by b. If o=1, then multiply k by b and add a. If b is negative, then divide by | b | instead of multiplying. Note that either the offset or both the offset and the scale may be omitted. If n-m is omitted, extract all bits from the working buffer (and optionally apply the offset and scale). If -m is omitted, extract the "next" n bits from the working buffer from the "current position" (and optionally apply the offset and scale.) By default these, "relative" mode extractions are done in bit stream order beginning from the MSB of the working buffer. For more details, see the discussion following the Miscellaneous Primitives. The extraction operator may be applied more than once to the same working buffer. If the extraction operator is applied more than once to the working buffer in the same key, it is implied that all extracted (and possibly offset and scaled) values are to be added together. If the extraction operator is applied more than once to the working buffer in successive keys (or via the repeat operator), then each extracted value will be a distinct datum.

For example:

W1 D0-5: Copy one byte into the working buffer and extract the low order six bits as an unsigned integer.

W2 D8-15:26:0:-65 D0-7:1:0:-65: Copy two bytes into the working buffer and extract each byte separately interpreting the pair such that two upper case alphabetic ASCII characters becomes a two digit base 26 unsigned number.

B10 D: Copy 10 bits into the working buffer and extract them all. This could also be written as B10 D0-9.

W1 D4 ~ D4: Copy one byte into the working buffer and extract two successive 4-bit quantities. This is equivalent to W1 D4-7 ~ D0-3 or to B4 D ~ B4 D. Note that because the relative mode D operation works in bit stream order, it is a close analogue of the B operation.

Sign primitives:

C {t{,n}}

C {t{,n}} — the number has a "complement" sign:

t = 1: use 1's complement

t = 2: use 2's complement

n = 1: the resultant number is always multiplied by -1

n ≠ 1 or omitted: the resultant number is left unchanged

Note that the C specification refers to the bits extracted by the preceding D specification. It only makes sense if the D specification includes no offset or scale that might change the sign bit.
For example
B12 D C2: Interpret the next 12 -bits as a 2's complement number.

S b{,n}

S b{,n} — the number has a sign bit:

if bit b is set: the number is negative

n = 1: the resultant number is multiplied by -1

n ≠ 1 or omitted: the resultant number is left unchanged

Note that the sign bit designated by the S specification is a bit in the working buffer. The sign then applies to the aggregate of all extraction operations performed in the same key. The sign bit itself should not be extracted in any D specification.

For example:

W1 D0-6 S7: Interpret the next byte as a signed integer.

W1 D4-6:%1 D0-3 S7: Interpret the next byte as a two digit BCD integer using the high order bit as a sign bit for the sum of the two digits.

A b

A b — the number uses a bias type sign:

The factor b is added to the formed number. (This number is almost always negative, and is usually a power of 2 minus 1.)

The A specification will generally apply to the aggregate of all extraction operations applied in the same key. Note that in some cases, the A specification can be redundant with the D
specification offset.

For example:

W1 D A-127: Interpret the next byte as an unsigned 8-bit integer with an offset of -127
(yielding a range of [-127, 128]). Equivalent to W1 D:1:0:-127.

W1 D4-7%1 D0-3 A-49: Interpret the next byte as a two digit unsigned BCD integer and then apply the sign bias (yielding a range of [-49, 50]).

A complete specification must include one copying/reordering primitive followed by at least one extraction specification. As noted above, if multiple extraction primitives are given in one key, it is implied that the integers extracted should be added together. It is possible for an extraction to reference bits copied into the working buffer in a previous key. For example, in the gain-ranged family, the working buffer is filled with both the characteristic and the mantissa during the mantissa extraction, while the characteristic extraction is performed in the next key. Another example would be to copy/reorder enough bits to represent three fixed point numbers and then to extract them in three successive keys or in one repeat operation. Only one of the possible sign specification will usually be used in any one key and it will usually appear only once. However, there are common cases in which no sign field is needed (e.g., an unsigned integer or a sign offset included in the extraction field.

Miscellaneous Primitives:

Yx

Yx - repeat the following fields (to the end of the key) x times and interpret the results as x distinct and successive data values. Repeating is never necessary but is very convenient for complex specifications that would otherwise require many keys.

For example:

Y2 W1 D C2: Interpret the next two bytes as two 8-bit, twos complement, fixed point numbers. This could also be written as W2 Y2 D8 C2

X

X - discard the result of the following operation. If a copy/reorder operation follows, discard the contents of the working buffer. If an extraction operation follows, discard the result.

For example:

X W1: Skip one byte

X B2 B6 D C2: Skip two bits and then interpret the following six bits as a twos complement, fixed point number. This could also be written as W1 X D2 D6 C2.

O{t}

O{t} - specify the manner in which the relative mode extraction current position is updated:

t = 0 or omitted use bit stream order
t = 1 use big-endian bit order

The O field affects the operation of all relative mode D fields that follow until another O field is encountered. O 0 is the default until the first O field in encountered. In bit stream order, the current position defaults to the MSB following each copy/reorder operation and successive relative mode extractions proceed from MSB toward LSB (from left to right). In 68000 bit order, the current position defaults to the LSB following each copy/reorder operation and successive relative mode extractions proceed from LSB toward MSB (from right to left).

For Example:

W1 Y2 D4 C2 is equivalent to W1 D4-7 C2 ~ D0-3 C2. While O1 W1 Y2 D4 C2 is equivalent to W1 D0-3 C2 ~ D4-7 C2.

Jx

Jx - set the relative mode extraction current position to big-endian bit number x within the working buffer. Note that in bit stream order, the current position refers to the high order bit of the next nibble extracted while in big-endian bit order it refers to the low order bit of the next nibble extracted.

For example:

W1 J5 D4 C2 is equivalent to W1 D2-5 C2. This would also be equivalent to O1 W1 J2 D4 C2.

The repeat, discard, relative mode extract, and the relative mode extraction direction and position fields provide a powerful facility in DDL for compactly describing non-byte aligned data words packed into bytes under different assumptions. Although these packings are natural under particular computer architectures, many of them were previously not describable using DDL. For example, 6-bit twos complement data words packed in bit stream order into successive bytes (on either a big-endian or little-endian style machine) can be described by:

B6 D C2

The same specification would be appropriate had the data words been packed into successive 32-bit (long) words on a big-endian style machine. However, if the data was packed into Vax long words in bit stream order, the data is not describable at all unless blocks of 16 data words are first copied and byte reordered as follows:

W12, 3, 2, 1, 0, 7, 6, 5, 4, 11, 10, 9, 8, Y16 D6 C2

In this case, the extraction of successive independent data values from one working buffer makes interpretation possible, while the repeat specification makes it compact. If the same data was packed into long words in big-endian bit order, the specification would be:

W12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0 O1 Y16 D6 C2

Here the order instruction, supported by the byte reordering, makes the repeat possible. Finally, if the data was packed into big-endian 32 bit words in big-endian bit order, then the specification would be:

W12, 8, 9, 10, 11, 4, 5, 6, 7, 0, 1, 2, 3, O1 Y16 D6 C2

Establishing and modifying the relative mode extraction current position also requires some additional discussion. The initial current position is set depending on the bit orders selected as described above. This position is updated by each relative mode extraction as described. However, the current position could also be modified by an absolute mode extraction (i.e. where n-m is specified). In bit stream order, the new current position would be the bit just to the right of the bits first extracted (equivalent to issuing a Jn-1). In big-endian bit order, the new current position would be the bit just to the left of the bits first extracted (equivalent to issuing a Jm+1). The current position set operation is provided when the default or the resetting due to an absolute mode extraction is not what is desired. Note that the current position could also be reset by discarding an absolute mode extraction except when it is necessary to return to the default value.

Integer Format — Family 0

The integer format allows you to describe many kind of integer data. The multiplexing key fields described are also used in the gain ranged family. This family has two keys:

Key 1

In integer format, key 1 describes sample multiplexing That is, samples, all recorded at the same time and sample rate, but from different channels appearing in the same data record. We discourage sample multiplexing in the SEED format, but you can describe it. If you use multiplexing, you must describe one sub-channel in the Channel ID Blockette [52] for each multiplexed channel. "1" denotes the first multiplexed channel, "2" the second, and so on. Multiplexed samples will appear in the data record in order of increasing sub-channel numbers.

M x

M x — Multiplexed data code:

x ≠ 0 or 1: data are multiplexed for x subchannels

x = 0 or 1: data are not multiplexed, and the data record contains data from only one channel

I x

I x — Data interleave (mandatory if more than 1 subchannel):

x = 0: data are interleaved

x = 1: data are non-interleaved

When data are interleaved, the first sample of each sub-channel are written sequentially into the data record to form the first data frame. Then the second sample of each sub-channel are written sequentially to form the second data frame and so on. Multiplexed, but non-interleaved data means that all samples from the first sub-channel for this record are written into the data record followed by all samples for the second sub-channel and so forth. The number of samples must always be an integer multiple of the number of sub-channels. That is, there must be the same number of samples for each sub-channel. Thus, if there are three multiplexed sub-channels with 100 samples each , the number of samples shown in the Fixed Section of the Header will be 300.

L x

L x — Interleave size (optional):

x = number of bytes for each subchannel

Each subchannel begins on an x-byte boundary. (The x-byte block does not have to be full.)

If not specified, SEED assumes that the first sample of the next subchannel exists immediately after the last sample of the previous subchannel

Key 2

Key 2 describes the actual interpretation of the integer values using the extraction primitives defined above.

The following examples illustrate the use of the integers format family.

Here is an example of DWWSSN data usage:

Key 1: M0

Key 2: W2 D0-15 C2 (or W2 D C2)

Here is a 4-nibble (2-byte) unsigned BCD format:

Key 1: M0

Key 2: W2 D0-3 D4-7:%1 D8-11:%2 D12-15:%3 or (W2 D4 D4:%1 D4:%2 D4:%3)

Gain Ranged Format — Family 1

Use this format to describe data stored as a fraction and multiplied by a gain factor. You can also use this format to describe the native floating point systems of most computers. This family has four keys:

Key 1

This key is identical to key 1 of the integer type family.

Key 2

The second key describes how to form the mantissa, and is identical to key 2 of the integer family except that the characteristic is copied into the working buffer here, but is interpreted in key 3.

Key 3

Use this key to describe how to form the exponent (gain code). It, too, uses the extraction primitives. Note that the W or B fields are established in key 2 and cannot be set again in key 3.

Key 4

This key describes the evaluation of the exponent. It uses the following fields to describe the rules:

P gc:ml

P gc:ml,... — describes the multiplier tables:

gc =a possible gain code value

ml > 0: If the gain code value extracted by key 3 equals qc, the mantissa extracted by key 2 is multiplied by the multiplier factor, ml.

ml < 0: If the gain code value extracted by key 3 equals, gc, the mantissa extracted by key 2 is divided by |ml|.

Specify any number of code/multiplier combinations. If a gain code is seen that was not defined, SEED assumes it to be a multiplier of 1.

E b{:a{:m{:p}}}

E b{:a{:m{:p}}} — describes the exponent:

b = the base

a = an optional value added to the exponent (usually used as a bias in floating point systems)

m = an optional value multiplying the sum of the exponent and a

p = an optional value added to the result of the operations just described

In other words:

sample = mantissa X $b^{m(exponent+a)+p}$.

H

H — the mantissa has a hidden bit:

The number has been normalized such that the mantissa always has the high bit set; it has therefore been implied without explicitly appearing in the data word. Therefore, the hidden bit must be restored to the mantissa before applying the characteristic.

Z e{:m}

Z e{:m} — the number system uses a “clean zero”:

Normalized floating point cannot normally express zero without extra information. If the exponent of the number being decoded is e, and optionally, the mantissa is equal to m, then this is a special code for exactly zero.

Here are some examples of the use of the gain ranged family:

- CDSN example (without multiplexing):

```
Key 1:  M0
Key 2:  W2 D0-13 A-8191
Key 3:  D14-15
Key 4:  P0:#0,1:#2,2:#4,3:#7
```

- SRO example (without multiplexing):

```
Key 1:  M0
Key 2:  W2 D0-11 C2
Key 3:  D12-15
Key 4:  E2:0:-1:10
```

- DEC F floating format:

```
Key 1:  M0
Key 2:  W4, 1, 0, 3, 2, D0-22 S31,0
Key 3:  D23-30
Key 4:  E2:-#7 H Z0
```

Integer Differences Compression — Family 50

This language describes some possible schemes for integer differences compression. While it cannot describe them all, it does describe those that resemble the Steim Compression Algorithm. This family uses 2 keys, plus a number of control type keys (keys 3—n), which carry out the action required by the control code derived in key 2.

Key 1

The first key describes the integration constants and where they will be found:

P n

P n — determines which byte, relative to the start of the data, is the first byte of the next integration constant.

F n

F n — forward integration constant for the difference of order n:

n = 1: the integration constant for the first difference

n = 2: the integration constant for the second difference

The fields that follow will describe interpreting this difference using the extraction primitives until another F or R field is encountered. Note that, by default, the F1 and R1 constants are data values while the F2 and R2 constants are first differences, etc.

R n

R n — the reverse integration constant for difference n; same as F above, except that it is used primarily for error recovery and error checking.

Key 2

This key provides the description, location, and configuration of the control code (compression key) bits for one compression frame. Note that a group of one or more control codes are accessed in this key. Keys 3-m provide the interpretation for the value of each key and result in the decompression of the data. The control codes and the data associated with them make up one compression frame. Summation to undo the differencing is implied.

P x

P x — the first control code section will be found x bytes after the start of data. Use this to skip over header information that precedes data. The P specification is followed by a copying/ reordering primitive to place all of the control bits for this compression frame into the working buffer.

S n,l{,s}

S n,l{,s} — the control code bits are n bits wide:

l = 0: control codes start at the leftmost control code read left to right

l = 1: start at the rightmost code and read right to left

$s \neq 0$: skip s control code positions before starting to extract control codes

N x

N x — the number of usable control codes in this group, not including the codes skipped over by the command above.

Together the S and N specifications provide a description of all of the control code groups for the compression frame at the same time. Note that S and N refer to the bits placed in the working buffer by the previous copying/reordering primitive. Operationally, the control code fields are interpreted one at a time in the order specified in the S field. Each control code is interpreted as an unsigned integer value. The keys 3-m provide instructions for decompressing data corresponding to each control code value.

Keys 3-m Select the key corresponding to each control code value derived above:

Tx

T x — describes how to decode data when a control code with the value x is encountered (derived by using key 2).

I I - indirection. The next value extracted will be interpreted as unsigned sub-control code.

Kx Kx - analogous to Tx, but referring to the sub-control code value derived in the indirect specification.

N x N x — the sequence identifier for the next difference to be decoded. One control code or sub- control code value can result in unpacking many first differences. The instructions for decoding the first difference value is preceded by N 0 to identify it. The instructions for decoding the second difference value is preceded by N1, etc. Note that if successive Nx fields in the same key use the same interpretation specification, then they can all be replaced by a repeat operation.

Key m + 1 Optionally specify the action to take at the end of a block of compression frames.

Gx Gx - a block is x compression frames long. The following extraction primitives describe what action to take in collecting the block trailer information.

Notice the structure provided by DDL for the integer compression family: 1) data records are divided into blocks, 2) blocks are divided into compression frames, 3) compression frames are divided into a control code section and a data section. The control code section may have many different control code values that are interpreted one at a time. Each control code results in the interpretation of one or more difference values from the data section of the compression frame. The control code section and all associated difference values in the data section define the length of a compression frame. Each compression frame is followed immediately by another until a block is completed at which point the block trailer information must be skipped. One block follows another until all samples have been decompressed. Note that the last block and compression frame may be incomplete. Control codes after the control code in which the data is completed may be meaningless and their associated difference values may be missing. Note that not all family 50 formats use the block structure concept making key n + 1 optional.

Some examples of the integer compression formats are:

The Steim 1 data format was originally described this way:

Key 1: F1 P4 W4 D0-31 C2 R1 P8 W4 D0-31 C2
Key 2: P0 W4 N15 S2,0,1
Key 3: T0 X N0 W4 D0-31 C2
Key 4: T1 N0 W1 D0-7 C2 N1 W1 D0-7 C2 N2 W1 D0-7 C2 N3 W1 D0-7 C2
Key 5: T2 N0 W2 D0-15 C2 N1 W2 D0-15 C2
Key 6: T3 N0 W4 D0-31 C2

The Steim 1 format could be described more compactly by using the repeat operation:

Key 1: F1 P4 W4 D C2 R1 P8 W4 D C2
Key 2: P0 W4 N15 S2, 0, 1
Key 3: T0 X W4
Key 4: T1 Y4 W1 D C2
Key 5: T2 Y2 W2 D C2
Key 6: T3 N0 W4 D C2

The Steim 2 compression format can be described by:

```
Key 1:    F1 P4 W4 D C2 R1 P8 W4 D C2
Key 2:    P0 W4 N15 S2, 0,1
Key 3:    T0 X W4
Key 4:    T1 Y4 W1 D C2
Key 5:    T2 W4  I D2
Key 6:    K0 X D30
Key 7:    K1 N0 D30 C2
Key 8:    K2 Y2 D15 C2
Key 9:    K3 Y3 D10 C2
Key 10:   T3 W4 I D2
Key 11:   K0 Y5 D6 C2
Key 12:   K1 Y6 D5 C2
Key 13:   K2 X D2 Y7 D4 C2
Key 14:   K3 X D30
```

The USNSN data format looks like this:

```
Key 1:    F1  P0  W4  D  C2
Key 2:    P6  W2  N2  S4, 0, 0
Key 3:    T0  Y4  B4  D  C2
Key 4:    T1  Y8  B4  D  C2
Key 5:    T2  Y12  B4  D  C2
Key 6:    T3  Y4  B6  D  C2
Key 7:    T4  Y8  B6  D  C2
Key 8:    T5  Y4  W1  D  C2
Key 9:    T6  Y8  W1  D  C2
Key 10:   T7  Y4  B10  D  C2
Key 11:   T8  Y8  B10  D  C2
Key 12:   T9  Y4  B12  D  C2
Key 13:   T10  Y4  B14  D  C2
Key 14:   T11  Y4  W2  D  C2
Key 15:   T12  Y4  B20  D  C2
Key 16:   T13  Y4  W3  D  C2
Key 17:   T14  Y4  B28  D  C2
Key 18:   T15  Y4  W4  D  C2
Key 19:   G7  X W1
```

ASCII text — Family 80

You can also use the data records to record any ASCII text. Such data can come from console interaction by the operator, from error logs, or from modem and telemetry transactions and audits. The number of samples in the fixed data header simply refers to the number of text bytes used. The time of the data is approximately the time of the first bytes in the record.

Use combinations of carriage returns (CR — ASCII 13) or line feeds (LF —ASCII 10) for the end-of-line characters. We recommend using CRLF, LFCR, CR or LF. SEED allows nulls(NUL — ASCII 0), form feeds (FF —ASCII 12) and bells (BEL — ASCII 7), but we discourage using other control characters.

Non-ASCII text — Family 81

This data type is reserved for the non-ASCII text sequences of various languages.

Appendix E: Sample Logical Volumes

Contributed by IRIS DMC

This appendix reproduces some actual blockettes from a sample SEED volume. It contains data for two stations in two logical volumes. Each logical volume is one day in length. The output has been edited to clarify the contents. It shows the relative placement of volume index control headers, abbreviation dictionary control headers, station control headers and time span control headers.

The records listed display the important information. "Type" refers to the blockette type, and "len" is the length of the blockette (including the type and length fields).

We hope this reproduction will clearly illustrate most details of the SEED implementation, and that it will assist your organization in developing SEED reading and writing utilities.

Appendix E

```
LOGICAL VOLUME 1 BEGINS HERE

logrec 1 type V

        type 010 len 0059 : 02.1121992,001,00:00:00.0000~1992,002,00:00:00.0000~
        type 011 len 0054 : 004AAK  000003ANMO 000007ANTO 000010BJI  000012
        type 012 len 0063 : 00011992,001,00:00:00.0000~1992,002,00:00:00.0000~000014

logrec 2 type A

        type 030 len 0232 : Steim Integer Compression Format~000105006F1 P4 W4 D0-31 C2 R1 P8 W4 D0-31 C2~P0 W4 N15 S2,0,1~T0 X
                : N0 W4 D0-31 C2~T1 N0 W1 D0-7 C2 N1 W1 D0-7 C2 N2 W1 D0-7 C2 N3 W1 D0-7 C2~T2 N0 W2 D0-15 C2 N1 W2 D0
                : -15 C2~T3 N0 W4 D0-31 C2~
        type 030 len 0087 : CDSN Gain-Ranged Format~000200104M0~W2 D0-13 A-8191~D14-15~P0:#0,1:#2,2:#4,3:#7~
        type 030 len 0072 : SRO Gain-Ranged Format~000300104M0~W2 D0-11 C2~D12-15~E2:0:-1:10~
        type 030 len 0232 : Steim Integer Compression Format~000405006F1 P4 W4 D0-31 C2 R1 P8 W4 D0-31 C2~P0 W4 N15 S2,0,1~T0 X
                : N0 W4 D0-31 C2~T1 N0 W1 D0-7 C2 N1 W1 D0-7 C2 N2 W1 D0-7 C2 N3 W1 D0-7 C2~T2 N0 W2 D0-15 C2 N1 W2 D0
                : -15 C2~T3 N0 W4 D0-31 C2~
        type 031 len 0043 : 0740STime correction is unknown.~000
        type 031 len 0072 : 0750STime correction does not include leap second, (-1000ms).~000
        type 033 len 0055 : 001(GSN) Global Seismograph Network (IRIS/USGS)~
        type 033 len 0051 : 002(CDSN) China Digital Seismograph Network~
        type 033 len 0039 : 003IRIS/IDA Network (UCSD/IGPP)~
        type 033 len 0050 : 004Geotech KS-36000-I Borehole Seismometer~
        type 033 len 0041 : 005Streckeisen STS-1V Seismometer~
        type 033 len 0041 : 006Streckeisen STS-1H Seismometer~
        type 033 len 0048 : 007Geotech KS-36000 Borehole Seismometer~
        type 033 len 0045 : 008Streckeisen STS-1H/VBB Seismometer~
        type 033 len 0045 : 009Streckeisen STS-1V/VBB Seismometer~
        type 034 len 0044 : 001M/S~Velocity in Meters Per Second~
        type 034 len 0020 : 002A~Amperes~
        type 034 len 0018 : 003V~Volts~ type 034 len 0018 : 004V~Volts~
        type 034 len 0032 : 005COUNTS~Digital Counts~
        type 034 len 0041 : 006M~Earth Displacement in Meters~
        type 034 len 0044 : 007M/S~Velocity in Meters Per Second~
        type 034 len 0018 : 008V~Volts~
        type 034 len 0018 : 009V~Volts~
        type 034 len 0032 : 010COUNTS~Digital Counts~
        type 034 len 0032 : 011COUNTS~Digital Counts~

logrec 3 type S

        type 050 len 0083 : AAK   42.639000  74.494000 1645.00003000Ala Archa, USSR~0033210101991,254~~N
        type 052 len 0123 :   BHE   0008S/N #39028~007008 42.639000  74.494000 1645.0  0.0 90.0  0.00004122.0000E+015.0000E-0500
                : 00CG~1991,254~~N
        type 053 len 0382 : B 1007008 7.87395E+00 5.00000E-02  3 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.0
                : 0000E+00 0.00000E+00 0.00000E+00-1.27000E+01 0.00000E+00 0.00000E+00 0.00000E+00  4-1.96418E-03 1.96
                : 418E-03 0.00000E+00 0.00000E+00-1.96418E-03-1.96418E-03 0.00000E+00 0.00000E+00-6.23500E+00 7.81823E
                : +00 0.00000E+00 0.00000E+00-6.23500E+00-7.81823E+00 0.00000E+00 0.00000E+00
        type 058 len 0035 :  1 1.15100E+03 5.00000E-02 0
        type 058 len 0035 :  2 1.21822E+02 0.00000E+00 0
        type 054 len 0024 : D 3009010   0   0
        type 057 len 0051 :  32.0000E+02    1    0 0.0000E+00 0.0000E+00
        type 058 len 0035 :  3 3.27680E+03 0.00000E+00 0
        type 054 len 2400 : D 4011010  99 6.67466E-06 0.00000E+00 1.09015E-05 0.00000E+00 1.49367E-05 0.00000E+00 1.36129E-05 0.
                : 00000E+00 1.68905E-06 0.00000E+00-2.55129E-05 0.00000E+00-6.96400E-05 0.00000E+00-1.26610E-04 0.0000
                : 0E+00-1.84580E-04 0.00000E+00-2.23689E-04 0.00000E+00-2.18583E-04 0.00000E+00-1.44157E-04 0.00000E+0
                : 0 1.60165E-05 0.00000E+00 2.60152E-04 0.00000E+00 5.60233E-04 0.00000E+00 8.58722E-04 0.00000E+00 1.
                : 07275E-03 0.00000E+00 1.10758E-03 0.00000E+00 8.78519E-04 0.00000E+00 3.38276E-04 0.00000E+00-4.9646
                : 2E-04 0.00000E+00-1.52660E-03 0.00000E+00-2.56818E-03 0.00000E+00-3.37140E-03 0.00000E+00-3.66272E-0
                : 3 0.00000E+00-3.20570E-03 0.00000E+00-1.87049E-03 0.00000E+00 3.03131E-04 0.00000E+00 3.06640E-03 0.
                : 00000E+00 5.96158E-03 0.00000E+00 8.37105E-03 0.00000E+00 9.61594E-03 0.00000E+00 9.09321E-03 0.0000
                : 0E+00 6.42899E-03 0.00000E+00 1.61822E-03 0.00000E+00-4.88235E-03 0.00000E+00-1.21360E-02 0.00000E+0
                : 0-1.87996E-02 0.00000E+00-2.32904E-02 0.00000E+00-2.40261E-02 0.00000E+00-1.97035E-02 0.00000E+00-9.
                : 56741E-
        type 057 len 0051 :  42.0000E+02    5    0 0.0000E+00 0.0000E+00
        type 058 len 0035 :  4 1.00000E+00 0.00000E+00 0
        type 054 len 1008 : D 5011010  41 3.38704E-05 0.00000E+00 1.02582E-04 0.00000E+00 4.74744E-05 0.00000E+00-3.47027E-04 0.
                : 00000E+00-8.15713E-04 0.00000E+00-3.49874E-04 0.00000E+00 1.64030E-03 0.00000E+00 3.37187E-03 0.0000
                : 0E+00 1.19598E-03 0.00000E+00-5.54789E-03 0.00000E+00-1.00841E-02 0.00000E+00-2.75277E-03 0.00000E+0
                : 0 1.53704E-02 0.00000E+00 2.51354E-02 0.00000E+00 4.76501E-03 0.00000E+00-3.94161E-02 0.00000E+00-6.
                : 10678E-02 0.00000E+00-6.52309E-03 0.00000E+00 1.28631E-01 0.00000E+00 2.76326E-01 0.00000E+00 3.4056
                : 9E-01 0.00000E+00 2.76326E-01 0.00000E+00 1.28631E-01 0.00000E+00-6.52309E-03 0.00000E+00-6.10678E-0
                : 2 0.00000E+00-3.94161E-02 0.00000E+00 4.76501E-03 0.00000E+00 2.51354E-02 0.00000E+00 1.53704E-02 0.
                : 00000E+00-2.75277E-03 0.00000E+00-1.00841E-02 0.00000E+00-5.54789E-03 0.00000E+00 1.19598E-03 0.0000
                : 0E+00 3.37187E-03 0.00000E+000004S*00 1.64030E-03 0.00000E+00-3.49874E-04 0.00000E+00-8.15713E-04 0.
                : 00000E+00-3.47027E-04 0.00000E+00 4.74744E-05 0.00000E+00 1.02582E-04 0.00000E+00 3.38704E-05 0.0000
                : 0

logrec 4 type S*

        type 057 len 0051 :  54.0000E+01    2    0 0.0000E+00 0.0000E+00
        type 058 len 0035 :  5 4.00000E+00 0.00000E+00 0
        type 058 len 0035 :  0 1.84776E+09 1.00000E+00 0
        type 052 len 0123 :   BHN   0008S/N #39027~007008 42.639000  74.494000 1645.0  0.0  0.0  0.00004122.0000E+015.0000E-0500
                : 00CG~1991,254~~N
        type 053 len 0382 : B 1007008 7.87395E+00 5.00000E-02  3 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.0
                : 0000E+00 0.00000E+00 0.00000E+00-1.27000E+01 0.00000E+00 0.00000E+00 0.00000E+00  4-1.96418E-03 1.96
                : 418E-03 0.00000E+00 0.00000E+00-1.96418E-03-1.96418E-03 0.00000E+00 0.00000E+00-6.23500E+00 7.81823E
                : +00 0.00000E+00 0.00000E+00-6.23500E+00-7.81823E+00 0.00000E+00 0.00000E+00
        type 058 len 0035 :  1 1.11301E+03 5.00000E-02 0
        type 058 len 0035 :  2 1.21825E+02 0.00000E+00 0
        type 054 len 0024 : D 3009010 0   0
```

type 057 len 0051 : 32.0000E+02 1 0 0.0000E+00 0.0000E+00
type 058 len 0035 : 3 3.27680E+03 0.00000E+00 0
type 054 len 2400 : D 4011010 99 6.67466E-06 0.00000E+00 1.09015E-05 0.00000E+00 1.49367E-05 0.00000E+00 1.36129E-05 0.
: 00000E+00 1.68905E-06 0.00000E+00-2.55129E-05 0.00000E+00-6.96400E-05 0.00000E+00-1.26610E-04 0.0000
: 0E+00-1.84580E-04 0.00000E+00-2.23689E-04 0.00000E+00-2.18583E-04 0.00000E+00-1.44157E-04 0.00000E+0
: 0 1.60165E-05 0.00000E+00 2.60152E-04 0.00000E+00 5.60233E-04 0.00000E+00 8.58722E-04 0.00000E+00 1.
: 07275E-03 0.00000E+00 1.10758E-03 0.00000E+00 8.78519E-04 0.00000E+00 3.38276E-04 0.00000E+00-4.9646
: 2E-04 0.00000E+00-1.52660E-03 0.00000E+00-2.56818E-03 0.00000E+00-3.37140E-03 0.00000E+00-3.66272E-0
: 3 0.00000E+00-3.20570E-03 0.00000E+00-1.87049E-03 0.00000E+00 3.03131E-04 0.00000E+00 3.06640E-03 0.
: 00000E+00 5.96158E-03 0.00000E+00 8.37105E-03 0.00000E+00 9.61594E-03 0.00000E+00 9.09321E-03 0.0000
: 0E+00 6.42899E-03 0.00000E+00 1.61822E-03 0.00000E+00-4.88235E-03 0.00000E+00-1.21360E-02 0.00000E+0
: 0-1.87996E-02 0.00000E+00-2.32904E-02 0.00000E+00-2.40261E-02 0.00000E+00-1.97035E-02 0.00000E+00-9.
: 56741E-
type 057 len 0051 : 42.0000E+02 5 0 0.0000E+00 0.0000E+00
type 058 len 0035 : 4 1.00000E+00 0.00000E+00 0
type 054 len 1008 : D 5011010 41 3.38704E-05 0.00000E+00 1.02582E-04 0.00000E+00 4.74744E-05 0.00000E+00-3.47027E-04 0.
: 00000E+00-8.15713E-04 0.00000E+00-3.49874E-04 0.00000E+00 1.64030E-03 0.00000E+00 3.37187E-03 0.0000
: 0E+00 1.19598E-03 0.00000E+00-5.54789E-03 0.00000E+00-1.00841E-02 0.00000E+00-2.75277E-03 0.00000E+0
: 0 1.53704E-02 0.00000E+00 2.51354E-02 0.00000E+00 4.76501E-03 0.00000E+00-3.94161E-02 0.00000E+00-6.
: 10678E-02 0.00000E+00-6.52309E-03 0.00000E+00 1.28631E-01 0.00000E+00 2.76326E-01 0.00000E+00 3.4056
: 9E-01 0.00000E+00 2.76326E-01 0.00000E+00 1.28631E-01 0.00000E+00-6.52309E-03 0.00000E+00-6.10678E-0
: 2 0.00000E+00-3000005S*.94161E-02 0.00000E+00 4.76501E-03 0.00000E+00 2.51354E-02 0.00000E+00 1.5370
: 4E-02 0.00000E+00-2.75277E-03 0.00000E+00-1.00841E-02 0.00000E+00-5.54789E-03 0.00000E+00 1.19598E-0
: 3 0.00000E+00 3.37187E-03 0.00000E+00 1.64030E-03 0.00000E+00-3.49874E-04 0.00000E+00-8.15713E-04 0.
: 00000E+00-3.47027E-04 0.00000E+00 4.74744E-05 0.00000E+00 1.02582E-04 0.00000E+00 3.38704E-05 0.0000
: 0

logrec 5 type S*

type 057 len 0051 : 54.0000E+01 2 0 0.0000E+00 0.0000E+00
type 058 len 0035 : 5 4.00000E+00 0.00000E+00 0
type 058 len 0035 : 0 1.78681E+09 1.00000E+00 0
type 052 len 0123 : BHZ 0009S/N #28733~007008 42.639000 74.494000 1645.0 0.0 0.0-90.00004122.0000E+015.0000E-0500
: 00CG~1991,254~~N
type 053 len 0382 : B 1007008 7.87395E+00 5.00000E-02 3 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.0
: 0000E+00 0.00000E+00 0.00000E+00-1.27000E+01 0.00000E+00 0.00000E+00 0.00000E+00 4-1.96418E-03 1.96
: 418E-03 0.00000E+00 0.00000E+00-1.96418E-03-1.96418E-03 0.00000E+00 0.00000E+00-6.23500E+00 7.81823E
: +00 0.00000E+00 0.00000E+00-6.23500E+00-7.81823E+00 0.00000E+00 0.00000E+00
type 058 len 0035 : 1 1.23200E+03 5.00000E-02 0
type 058 len 0035 : 2 1.21765E+02 0.00000E+00 0
type 054 len 0024 : D 3009010 0 0
type 057 len 0051 : 32.0000E+02 1 0 0.0000E+00 0.0000E+00
type 058 len 0035 : 3 3.27680E+03 0.00000E+00 0
type 054 len 2400 : D 4011010 99 6.67466E-06 0.00000E+00 1.09015E-05 0.00000E+00 1.49367E-05 0.00000E+00 1.36129E-05 0.
: 00000E+00 1.68905E-06 0.00000E+00-2.55129E-05 0.00000E+00-6.96400E-05 0.00000E+00-1.26610E-04 0.0000
: 0E+00-1.84580E-04 0.00000E+00-2.23689E-04 0.00000E+00-2.18583E-04 0.00000E+00-1.44157E-04 0.00000E+0
: 0 1.60165E-05 0.00000E+00 2.60152E-04 0.00000E+00 5.60233E-04 0.00000E+00 8.58722E-04 0.00000E+00 1.
: 07275E-03 0.00000E+00 1.10758E-03 0.00000E+00 8.78519E-04 0.00000E+00 3.38276E-04 0.00000E+00-4.9646
: 2E-04 0.00000E+00-1.52660E-03 0.00000E+00-2.56818E-03 0.00000E+00-3.37140E-03 0.00000E+00-3.66272E-0
: 3 0.00000E+00-3.20570E-03 0.00000E+00-1.87049E-03 0.00000E+00 3.03131E-04 0.00000E+00 3.06640E-03 0.
: 00000E+00 5.96158E-03 0.00000E+00 8.37105E-03 0.00000E+00 9.61594E-03 0.00000E+00 9.09321E-03 0.0000
: 0E+00 6.42899E-03 0.00000E+00 1.61822E-03 0.00000E+00-4.88235E-03 0.00000E+00-1.21360E-02 0.00000E+0
: 0-1.87996E-02 0.00000E+00-2.32904E-02 0.00000E+00-2.40261E-02 0.00000E+00-1.97035E-02 0.00000E+00-9.
: 56741E-
type 057 len 0051 : 42.0000E+02 5 0 0.0000E+00 0.0000E+00
type 058 len 0035 : 4 1.00000E+00 0.00000E+00 0
type 054 len 1008 : D 5011010 41 3.38704E-05 0.00000E+00 1.02582E-04 0.00000E+00 4.74744E-05 0.00000E+00-3.47027E-04 0.
: 00000E+00-8.15713E-04 0.00000E+00-3.49874E-04 0.00000E+00 1.64030E-03 0.00000E+00 3.37187E-03 0.0000
: 0E+00 1.19598E-03 0.00000E+00-5.54789E-03 0.00000E+00-1.00841E-02 0.00000E+00-2.75277E-03 0.00000E+0
: 0 1.53704E-02 0.00000E+00 2.51354E-02 0.00000E+00 4.76501E-03 0.00000E+00-3.94161E-02 0.00000E+00-6.
: 106000006S*78E-02 0.00000E+00-6.52309E-03 0.00000E+00 1.28631E-01 0.00000E+00 2.76326E-01 0.00000E+0
: 0 3.40569E-01 0.00000E+00 2.76326E-01 0.00000E+00 1.28631E-01 0.00000E+00-6.52309E-03 0.00000E+00-6.
: 10678E-02 0.00000E+00-3.94161E-02 0.00000E+00 4.76501E-03 0.00000E+00 2.51354E-02 0.00000E+00 1.5370
: 4E-02 0.00000E+00-2.75277E-03 0.00000E+00-1.00841E-02 0.00000E+00-5.54789E-03 0.00000E+00 1.19598E-0
: 3 0.00000E+00 3.37187E-03 0.00000E+00 1.64030E-03 0.00000E+00-3.49874E-04 0.00000E+00-8.15713E-04 0.
: 00000E+00-3.47027E-04 0.00000E+00 4.74744E-05 0.00000E+00 1.02582E-04 0.00000E+00 3.38704E-05 0.0000
: 0

logrec 6 type S*

type 057 len 0051 : 54.0000E+01 2 0 0.0000E+00 0.0000E+00
type 058 len 0035 : 5 4.00000E+00 0.00000E+00 0
type 058 len 0035 : 0 1.97686E+09 1.00000E+00 0

logrec 7 type S

type 050 len 0096 : ANMO +34.946200-106.456700+1740.00006001Albuquerque, New Mexico, USA~0013210101989,241~~N
type 051 len 0035 : 1992,001~1992,002~0740000000
type 052 len 0119 : BHE0000004~001002+34.946200-106.456700+1740.0100.0090.0+00.0000112 2.000E+01 2.000E-030000CG~1991,
: 042,20:48~~N
type 053 len 0478 : A01001003 6.27190E+04 2.00000E-02003 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.0
: 0000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00006-8.98500E+01 0.00
: 000E+00 0.00000E+00 0.00000E+00-1.84300E+01 1.89100E+01 0.00000E+00 0.00000E+00-1.84300E+01-1.89100E
: +01 0.00000E+00 0.00000E+00-1.23400E-02 1.23400E-02 0.00000E+00 0.00000E+00-1.23400E-02-1.23400E-02
: 0.00000E+00 0.00000E+00-4.21900E-03 0.00000E+00 0.00000E+00 0.00000E+00
type 054 len 1560 : D020040050064-1.74668E+00-3.49337E-02 3.22140E+00 6.44279E-02 2.23562E+02 4.47125E+00 4.03169E+02 8.
: 3E+00 5.24351E+02 1.04870E+01-7.20810E+02-1.44162E+01 8.04317E+02 1.60863E+01-6.94529E+02-1.38906E+0 06337E+00-1.
: 98553E+02-3.97106E+00 7.34851E+01 1.46970E+00 8.52640E+01 1.70528E+00-2.92866E+02-5.8573 1 3.30693E+02 6.61387E+00 3.
:06430E+02 6.12859E+00-1.17554E+03-2.35107E+01 2.16453E+03 4.32905E+01-3.09238E+03-6.18476E+01 3.72622E+03 7.45243
:E+01-3.81300E+03-7.62600E+01 3.12290E+03 6.24580E+01-1.4983 2E+03-2.99664E+01-1.09920E+03-2.19841E+01 4.55209E+03
:9.10417E+01-8.56639E+03-1.71328E+02 1.26699E+0 4 2.53397E+02-1.62246E+04-3.24491E+02 1.84351E+04 3.68702E+02-1.83067
:E+04-3.66134E+02 1.44012E+04 2.88023E+02-3.77208E+03-7.54416E+01-2.41884E+04-4.83768E+02 3.27369E+05 6.54737E+03 1.
:55321E+05 3.10641E+03-6.67714E+04-1.33543E+03 4.02846E+04 8.05691E+02-2.42897E+04-4.85794E+02 1.26981E+04 2.53962
:E+02-4.06987E+03-8.13974E+01-2.04590E+03-4.09180E+01 5.89922E+03 1.17984E+02-7.76783E+03-1.55357E+02 8.01300E+

```
type 058 len 0035 : 00 1.00690E+09 2.00000E-0200
type 052 len 0119 :  BHN0000004~001002+34.946200-106.456700+1740.0100.0000.0+00.0000112 2.000E+01 2.000E-030000CG~1991,
     : 042,20:48~~N
type 053 len 0478 : A01001003 6.27190E+04 2.00000E-02003 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.0
     : 0000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00
     : 000E+00 0.00000E+00 0.00000E+00-1.84300E+01 1.89100E+01 0.00000E+00 0.00000E+00-1.84300E+01-1.89100E
     : +01 0.00000E+00 0.00000E+00-1.23400E-02 1.23400E-02 0.00000E+00 0.00000E+00-1.23400E-02-1.23400E-02
     : 0.00000E+00 0.00000E+00-4.21900E-03 0.00000E+00 0.00000E+00 0.00000E+00
type 054 len 1560 : D020040050064-1.74668E+00-3.49337E-02 3.22140E+00 6.44279E-02 2.23562E+02 4.47125E+00 4.03169E+02 8.
     : 06337E+00-1.98553E+02-3.97106E+00 7.34851E+01 1.46970E+00 8.52640E+01 1.70528E+00-2.92866E+02-5.8573
     : 3E+00 5.24351E+02 1.04870E+01-7.20810E+02-1.44162E+01 8.04317E+02 1.60863E+01-6.94529E+02-1.38906E+0
     : 1 3.30693E+02 6.61387E+00 3.06430E+02 6.12859E+00-1.17554E+03-2.35107E+01 2.16453E+03 4.32905E+01-3.
     : 09238E+03-6.18476E+01 3.72622E+03 7.45243E+01-3.81300E+03-7.62600E+01 3.12290E+03 6.24580E+01-1.4983
     : 2E+03-2.99664E+01-1.09920E+03-2.19841E+01 4.55209E+03 9.10417E+01-8.56639E+03-1.71328E+02 1.26699E+0
     : 4 2.53397E+02-1.62246E+04-3.24491E+02 1.84351E+04 3.68702E+02-1.83067E+04-3.66134E+02 1.44012E+04 2.
     : 88023E+02-3.77208E+03-7.54416E+01-2.41884E+04-4.83768E+02 3.27369E+05 6.54737E+03 1.55321E+05 3.1064
     : 1E+03-6.67714E+04-1.33543E+03 4.02846E+04 8.05691E+02-2.42897E+04-4.85794E+02 1.26981E+04 2.53962E+0
     : 2-4.06987E+03-8.13974E+01-2.04590E+03-4.09180E+01 5.89922E+03 1.17984E+02-7.76783E+03-1.55357E+02 8.
     : 01300E+
```

logrec 8 type S*

```
type 058 len 0035 : 00 1.00690E+09 2.00000E-0200
type 052 len 0119 :  BHZ0000004~001002+34.946200-106.456700+1740.0100.0000.0-90.0000112 2.000E+01 2.000E-030000CG~1991,
     : 042,20:48~~N
type 053 len 0478 : A01001003 6.27190E+04 2.00000E-02003 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.0
     : 0000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00
     : 000E+00 0.00000E+00 0.00000E+00-1.84300E+01 1.89100E+01 0.00000E+00 0.00000E+00-1.84300E+01-1.89100E
     : +01 0.00000E+00 0.00000E+00-1.23400E-02 1.23400E-02 0.00000E+00 0.00000E+00-1.23400E-02-1.23400E-02
     : 0.00000E+00 0.00000E+00-4.21900E-03 0.00000E+00 0.00000E+00 0.00000E+00
type 054 len 1560 : D020040050064-1.74668E+00-3.49337E-02 3.22140E+00 6.44279E-02 2.23562E+02 4.47125E+00 4.03169E+02 8.
     : 06337E+00-1.98553E+02-3.97106E+00 7.34851E+01 1.46970E+00 8.52640E+01 1.70528E+00-2.92866E+02-5.8573
     : 3E+00 5.24351E+02 1.04870E+01-7.20810E+02-1.44162E+01 8.04317E+02 1.60863E+01-6.94529E+02-1.38906E+0
     : 1 3.30693E+02 6.61387E+00 3.06430E+02 6.12859E+00-1.17554E+03-2.35107E+01 2.16453E+03 4.32905E+01-3.
     : 09238E+03-6.18476E+01 3.72622E+03 7.45243E+01-3.81300E+03-7.62600E+01 3.12290E+03 6.24580E+01-1.4983
     : 2E+03-2.99664E+01-1.09920E+03-2.19841E+01 4.55209E+03 9.10417E+01-8.56639E+03-1.71328E+02 1.26699E+0
     : 4 2.53397E+02-1.62246E+04-3.24491E+02 1.84351E+04 3.68702E+02-1.83067E+04-3.66134E+02 1.44012E+04 2.
     : 88023E+02-3.77208E+03-7.54416E+01-2.41884E+04-4.83768E+02 3.27369E+05 6.54737E+03 1.55321E+05 3.1064
     : 1E+03-6.67714E+04-1.33543E+03 4.02846E+04 8.05691E+02-2.42897E+04-4.85794E+02 1.26981E+04 2.53962E+0
     : 2-4.06987E+03-8.13974E+01-2.04590E+03-4.09180E+01 5.89922E+03 1.17984E+02-7.76783E+03-1.55357E+02 8.
     : 01300E+
type 058 len 0035 : 00 1.00690E+09 2.00000E-0200
type 052 len 0119 :  LHE0000004~001002+34.946200-106.456700+1740.0100.0090.0+00.0000112 1.000E+00 1.000E-040000CG~1991,
     : 042,20:48~~N
type 053 len 0478 : A01001003 6.27190E+04 2.00000E-02003 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.0
     : 0000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00
     : 000E+00 0.00000E+00 0.00000E+00-1.84300E+01 1.89100E+01 0.00000E+00 0.00000E+00-1.84300E+01-1.89100E
     : +01 0.00000E+00 0.00000E+00-1.23400E-02 1.23400E-02 0.00000E+00 0.00000E+00-1.23400E-02-1.23400E-02
     : 0.00000E+00 0.00000E+00-4.21900E-03 0.00000E+00 0.00000E+00 0.00000E+00
type 054 len 1104 : D020040050045-4.80207E-05-9.60415E-07-9.99033E-03-1.99807E-04 3.45482E+00 6.90964E-02-3.18539E+02-6.
     : 37078E+00-3.70084E+03-7.40168E+01-6.15216E+03-1.23043E+02 3.84297E+03 7.68593E+01-3.05587E+02-6.1117
     : 3E+00-4.85151E+03-9.70302E+01 1.05400E+04 2.10800E+02-1.38658E+04-2.77317E+02 1.13306E+04 2.26613E+0
     : 2-6.09784E+02-1.21957E+01-1.76762E+04-3.53525E+02 3.89026E+04 7.78052E+02-5.46632E+04-1.09326E+03 5.
     : 43908E+04 1.08782E+03-2.78663E+04-5.57325E+02-3.27114E+04-6.54229E+02 1.33794E+05 2.67588E+03-2.9372
     : 8E+05-5.87456E+03 6.84551E+05 1.36910E+04 1.28735E+06 2.57469E+04-5.28881E+04-1.05776E+03-9.24154E+0
     : 4-1.84831E+03 1.24696E+05 2.49391E+03-1.08734E+05-2.17468E+03 7.10849E+04 1.42170E+03-2.95894E+04-5.
     : 91788E+02-3.64620E+03-7.29240E+01 2.26317E+04 4.52634E+02-2.71833E+04-5.43666E+02 2.13163E+04 4.2632
     : 6E+02-1.08475E+04-2.16950E+02 1.03216E+03 2.06432E+01 5.01723E+03.09326E+03 5.43908E+04 1.08782E+03-
     : 2.78663E+04-5.57325E+02-3.27114E+04-6.54229E+02 1.33794E+05 2.67588E+03-2.93728E+05-5.87456E+03 6.84
     : 551E+05
```

logrec 9 type S*

```
type 058 len 0035 : 00 4.02650E+09 2.00000E-0200
type 052 len 0119 :  LHN0000004~001002+34.946200-106.456700+1740.0100.0000.0+00.0000112 1.000E+00 1.000E-040000CG~1991,
     : 042,20:48~~N
type 053 len 0478 : A01001003 6.27190E+04 2.00000E-02003 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.0
     : 0000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00
     : 000E+00 0.00000E+00 0.00000E+00-1.84300E+01 1.89100E+01 0.00000E+00 0.00000E+00-1.84300E+01-1.89100E
     : +01 0.00000E+00 0.00000E+00-1.23400E-02 1.23400E-02 0.00000E+00 0.00000E+00-1.23400E-02-1.23400E-02
     : 0.00000E+00 0.00000E+00-4.21900E-03 0.00000E+00 0.00000E+00 0.00000E+00
type 054 len 1104 : D020040050045-4.80207E-05-9.60415E-07-9.99033E-03-1.99807E-04 3.45482E+00 6.90964E-02-3.18539E+02-6.
     : 37078E+00-3.70084E+03-7.40168E+01-6.15216E+03-1.23043E+02 3.84297E+03 7.68593E+01-3.05587E+02-6.1117
     : 3E+00-4.85151E+03-9.70302E+01 1.05400E+04 2.10800E+02-1.38658E+04-2.77317E+02 1.13306E+04 2.26613E+0
     : 2-6.09784E+02-1.21957E+01-1.76762E+04-3.53525E+02 3.89026E+04 7.78052E+02-5.46632E+04-1.09326E+03 5.
     : 43908E+04 1.08782E+03-2.78663E+04-5.57325E+02-3.27114E+04-6.54229E+02 1.33794E+05 2.67588E+03-2.9372
     : 8E+05-5.87456E+03 6.84551E+05 1.36910E+04 1.28735E+06 2.57469E+04-5.28881E+04-1.05776E+03-9.24154E+0
     : 4-1.84831E+03 1.24696E+05 2.49391E+03-1.08734E+05-2.17468E+03 7.10849E+04 1.42170E+03-2.95894E+04-5.
     : 91788E+02-3.64620E+03-7.29240E+01 2.26317E+04 4.52634E+02-2.71833E+04-5.43666E+02 2.13163E+04 4.2632
     : 6E+02-1.08475E+04-2.16950E+02 1.03216E+03 2.06432E+01 5.01723E+03 1.00345E+02-6.75238E+03-1.35048E+0
     : 2 5.45243E+03 1.09049E+02-2.29507E+03-4.59013E+01-6.04042E+03-1.20809E+02-1.34924E+03-2.69849E+01-1.
     : 77083E+
type 058 len 0035 : 00 4.02650E+09 2.00000E-0200
type 052 len 0119 :  LHZ0000004~001002+34.946200-106.456700+1740.0100.0000.0-90.0000112 1.000E+00 1.000E-040000CG~1991,
     : 042,20:48~~N
type 053 len 0478 : A01001003 6.27190E+04 2.00000E-02003 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.0
     : 0000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00
     : 000E+00 0.00000E+00 0.00000E+00-1.84300E+01 1.89100E+01 0.00000E+00 0.00000E+00-1.84300E+01-1.89100E
     : +01 0.00000E+00 0.00000E+00-1.23400E-02 1.23400E-02 0.00000E+00 0.00000E+00-1.23400E-02-1.23400E-02
     : 0.00000E+00 0.00000E+00-4.21900E-03 0.00000E+00 0.00000E+00 0.00000E+00
type 054 len 1104 : D020040050045-4.80207E-05-9.60415E-07-9.99033E-03-1.99807E-04 3.45482E+00 6.90964E-02-3.18539E+02-6.
     : 37078E+00-3.70084E+03-7.40168E+01-6.15216E+03-1.23043E+02 3.84297E+03 7.68593E+01-3.05587E+02-6.1117
     : 3E+00-4.85151E+03-9.70302E+01 1.05400E+04 2.10800E+02-1.38658E+04-2.77317E+02 1.13306E+04 2.26613E+0
     : 2-6.09784E+02-1.21957E+01-1.76762E+04-3.53525E+02 3.89026E+04 7.78052E+02-5.46632E+04-1.09326E+03 5.
     : 43908E+04 1.08782E+03-2.78663E+04-5.57325E+02-3.27114E+04-6.54229E+02 1.33794E+05 2.67588E+03-2.9372
     : 8E+05-5.87456E+03 6.84551E+05 1.36910E+04 1.28735E+06 2.57469E+04-5.28881E+04-1.05776E+03-9.24154E+0
```

: 4-1.84831E+03 1.24696E+05 2.49391E+03-1.08734E+05-2.17468E+03 7.10849E+04 1.42170E+03-2.95894E+04-5.
: 91788E+02-3.64620E+03-7.29240E+01 2.26317E+04 4.52634E+02-2.71833E+04-5.43666E+02 2.13163E+04 4.2632
: 6E+02-1.08475E+04-2.16950E+02 1.03216E+03 2.06432E+01 5.01723E+03 1.00345E+02-6.75238E+03-1.35048E+0
: 2 5.45243E+03 1.09049E+02-2.29507E+03-4.59013E+01-6.04042E+03-1.20809E+02-1.34924E+03-2.69849E+01-1.
: 77083E+
type 058 len 0035 : 00 4.02650E+09 2.00000E-0200

logrec 10 type S

type 050 len 0090 : ANTO +39.868900+032.793599+0883.00003000Ankara, Turkey~0013210101978,244~1992,275~N
type 052 len 0121 : LHE0000007~006003+39.868900+032.793599+0883.0195.0090.0+00.0000312 1.000E+00 1.000E-040000CG~1980,
: 277~1992,275~N
type 053 len 1390 : A01006005 9.70000E+04 4.00000E-02009-1.26000E-01 0.00000E+00 0.00000E+00 0.00000E+00-5.01000E+01 0.0
: 0000E+00 0.00000E+00 0.00000E+00 0.00000E+00 1.05000E+00 0.00000E+00 0.00000E+00 0.00000E+00-1.05000
: E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
: 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.0
: 0000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00019-4.65000E+00 3.46000E+00 0.00
: 000E+00 0.00000E+00-4.65000E+00-3.46000E+00 0.00000E+00 0.00000E+00-1.18000E-01 0.00000E+00 0.00000E
: +00 0.00000E+00-4.07000E+01 0.00000E+00 0.00000E+00 0.00000E+00-1.00000E+02 0.00000E+00 0.00000E+00
: 0.00000E+00-1.50000E-01 0.00000E+00 0.00000E+00 0.00000E+00-2.64000E+02 0.00000E+00 0.00000E+00 0.00
: 000E+00-3.93000E+00 0.00000E+00 0.00000E+00 0.00000E+00-2.82000E-01 0.00000E+00 0.00000E+00 0.00000E
: +00-2.01000E-01 2.41000E-01 0.00000E+00 0.00000E+00-2.01000E-01-2.41000E-01 0.00000E+00 0.00000E+00-
: 1.34000
type 058 len 0035 : 00 5.34368E+09 4.00000E-0200
type 052 len 0121 : LHN0000007~006003+39.868900+032.793599+0883.0195.0000.0+00.0000312 1.000E+00 1.000E-040000CG~1980,
: 277~1992,275~N
type 053 len 1390 : A01006005 9.70000E+04 4.00000E-02009-1.26000E-01 0.00000E+00 0.00000E+00 0.00000E+00-5.01000E+01 0.0
: 0000E+00 0.00000E+00 0.00000E+00 0.00000E+00 1.05000E+00 0.00000E+00 0.00000E+00 0.00000E+00-1.05000
: E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
: 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.0
: 0000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00019-4.65000E+00 3.46000E+00 0.00
: 000E+00 0.00000E+00-4.65000E+00-3.46000E+00 0.00000E+00 0.00000E+00-1.18000E-01 0.00000E+00 0.00000E
: +00 0.00000E+00-4.07000E+01 0.00000E+00 0.00000E+00 0.00000E+00-1.00000E+02 0.00000E+00 0.00000E+00
: 0.00000E+00-1.50000E-01 0.00000E+00 0.00000E+00 0.00000E+00-2.64000E+02 0.00000E+00 0.00000E+00 0.00
: 000E+00-3.93000E+00 0.00000E+00 0.00000E+00 0.00000E+00-2.82000E-01 0.00000E+00 0.00000E+00 0.00000E
: +00-2.01000E-01 2.41000E-01 0.00000E+00 0.00000E+00-2.01000E-01-2.41000E-01 0.00000E+00 0.00000E+00-
: 1.34000
type 058 len 0035 : 00 5.08284E+09 4.00000E-0200
type 052 len 0121 : LHZ0000007~006003+39.868900+032.793599+0883.0195.0000.0-90.0000312 1.000E+00 1.000E-040000CG~1980,
: 277~1992,275~N
type 053 len 1390 : A01006005 9.70000E+04 4.00000E-02009-1.26000E-01 0.00000E+00 0.00000E+00 0.00000E+00-5.01000E+01 0.0
: 0000E+00 0.00000E+00 0.00000E+00 0.00000E+00 1.05000E+00 0.00000E+00 0.00000E+00 0.00000E+00-1.05000
: E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
: 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.0
: 0000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00019-4.65000E+00 3.46000E+00 0.00
: 000E+00 0.00000E+00-4.65000E+00-3.46000E+00 0.00000E+00 0.00000E+00-1.18000E-01 0.00000E+00 0.00000E
: +00 0.00000E+00-4.07000E+01 0.00000E+00 0.00000E+00 0.00000E+00-1.00000E+02 0.00000E+00 0.00000E+00
: 0.00000E+00-1.50000E-01 0.00000E+00 0.00000E+00 0.00000E+00-2.64000E+02 0.0000000011S*0E+00 0.00000E
: +00 0.00000E+00-3.93000E+00 0.00000E+00 0.00000E+00 0.00000E+00-2.82000E-01 0.00000E+00 0.00000E+00
: 0.00000E+00-2.01000E-01 2.41000E-01 0.00000E+00 0.00000E+00-2.01000E-01-2.41000E-01 0.00000E+00 0.00000E+00

logrec 11 type S*

type 058 len 0035 : 00 4.49909E+09 4.00000E-0200

logrec 12 type S

type 050 len 0093 : BJI +40.040300+116.175000+0043.00006002Baijatuan, Beijing, China~0023210101986,204~~N
type 051 len 0035 : 1992,001~1992,093~0740000000
type 051 len 0035 : 1992,001~1992,012~0740000000
type 052 len 0113 : BHE0000006~006002+40.040300+116.175000+0043.0003.0090.0+00.0000212 2.000E+01 2.000E-030000TG~1986,204~~N
type 053 len 0622 : A01006005 4.77000E+09 1.00000E+00003 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.0
: 0000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00009-2.22100E-01 2.22
: 100E-01 0.00000E+00 0.00000E+00-2.22100E-01-2.22100E-01 0.00000E+00 0.00000E+00-1.95700E+01 2.45700E
: +01 0.00000E+00 0.00000E+00-1.95700E+01-2.45700E+01 0.00000E+00 0.00000E+00-2.54200E+01 1.84700E+01
: 0.00000E+00 0.00000E+00-2.54200E+01-1.84700E+01 0.00000E+00 0.00000E+00-9.70800E+00 2.98800E+01 0.00
: 000E+00 0.00000E+00-9.70800E+00-2.98800E+01 0.00000E+00 0.00000E+00-3.14200E+01 0.00000E+00 0.00000E
: +00 0.00000E+00
type 058 len 0035 : 00 6.91000E+09 1.00000E+0000
type 052 len 0113 : BHN0000006~006002+40.040300+116.175000+0043.0003.0000.0+00.0000212 2.000E+01 2.000E-030000TG~1986,204~~N
type 053 len 0622 : A01006005 4.77000E+09 1.00000E+00003 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.0
: 0000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00009-2.22100E-01 2.22
: 100E-01 0.00000E+00 0.00000E+00-2.22100E-01-2.22100E-01 0.00000E+00 0.00000E+00-1.95700E+01 2.45700E
: +01 0.00000E+00 0.00000E+00-1.95700E+01-2.45700E+01 0.00000E+00 0.00000E+00-2.54200E+01 1.84700E+01
: 0.00000E+00 0.00000E+00-2.54200E+01-1.84700E+01 0.00000E+00 0.00000E+00-9.70800E+00 2.98800E+01 0.00
: 000E+00 0.00000E+00-9.70800E+00-2.98800E+01 0.00000E+00 0.00000E+00-3.14200E+01 0.00000E+00 0.00000E
: +00 0.00000E+00
type 058 len 0035 : 00 6.89000E+09 1.00000E+0000
type 052 len 0113 : BHZ0000005~006002+40.040300+116.175000+0043.0003.0000.0-90.0000212 2.000E+01 2.000E-030000TG~1986,204~~N
type 053 len 0622 : A01006005 4.77000E+09 1.00000E+00003 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.0
: 100E-01 0.00000E+00 0.00000E+00-2.22100E-01-2.22100E-01 0.00000E+00 0.00000E+00-1.95700E+01 2.45700E
: 0000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00009-2.22100E-01 2.22
: +01 0.00000E+00 0.00000E+00-1.95700E+01-2.45700E+01 0.00000E+00 0.00000E+00-2.54200E+01 1.84700E+01
: 0.00000E+00 0.00000E+00-2.54200E+01-1.84700E+01 0.00000E+00 0.00000E+00-9.70800E+00 2.98800E+01 0.00
: 000E+00 0.00000E+00-9.70800E+00-2.98800E+01 0.00000E+00 0.00000E+00-3.14200E+01 0.00000E+00 0.00000E
: +00 0.00000E+00
type 058 len 0035 : 00 7.20000E+09 1.00000E+0000
type 052 len 0113 : LHE0000006~006002+40.040300+116.175000+0043.0003.0090.0+00.0000212 1.000E+00 1.000E-040000CG~1986,
: 204~~N
type 053 len 0718 : A01006005 4.93000E-04 4.00000E-02004 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.0

: 0000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000
: E+00 0.00000E+00 0.00000E+00010-2.22100E-01 2.22100E-01 0.00000E+00 0.00000E+00-2.22100E-01-2.22100E
: -01 0.00000E+00 0.00000E+00-7.40500E-03 7.40500E-03 0.00000E+00 0.00000E+00-7.40500E-03-7.40500E-03
: 0.00000E+00 0.00000E+00-2.02300E-01 5.42000E-02 0.00000E+00 0.00000E+00-2.02300E-01-5.42000E-02 0.00
: 000E+00 0.00000E+00-1.48100E-01 1.48100E-01 0.00000E+00 0.00000E+00-1.48100E-01-1.48100E-01 0.00000E

: +00 0.00000E+00-5.42000E-02 2.02300E-01 0.00000E+00 0.00000E+00-5.42000E-02-2.02300E-01 0.00000E+00
: 0.00000E+00
type 058 len 0035 : 00 1.86000E+09 4.00000E-0200
type 052 len 0113 : LHN0000006~006002+40.040300+116.175000+0043.0003.0000.0+00.0000212 1.000E+00 1.000E-040000CG~1986,
: 204~~N
type 053 len 0718 : A01006005 4.93000E-04 4.00000E-02004 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.0
: 0000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000
: E+00 0.00000E+00 0.00000E+00010-2.22100E-01 2.22100E-01 0.00000E+00 0.00000E+00-2.22100E-01-2.22100E
: -01 0.00000E+00 0.00000E+00-7.40500E-03 7.40500E-03 0.00000E+00 0.00000E+00-7.40500E-03-7.40500E-03
: 0.00000E+00 0.00000E+00-2.02300E-01 5.42000E-02 0.00000E+00 0.00000E+00-2.02300E-01-5.42000E-02 0.00
: 000E+00 0.00000E+00-1.48100E-01 1.48100E-01 0.00000E+00 0.00000E+00-1.48100E-01-1.48100E-01 0.00000E
: +00 0.00000E+00-5.42000E-02 2000013S*.02300E-01 0.00000E+00 0.00000E+00-5.42000E-02-2.02300E-01 0.00
: 000E+00 0.0

logrec 13 type S*

type 058 len 0035 : 00 1.84000E+09 4.00000E-0200
type 052 len 0113 : LHZ0000005~006002+40.040300+116.175000+0043.0003.0000.0-90.0000212 1.000E+00 1.000E-040000CG~1986,
: 204~~N
type 053 len 0718 : A01006005 4.93000E-04 4.00000E-02004 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.0
: 0000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000
: E+00 0.00000E+00 0.00000E+00010-2.22100E-01 2.22100E-01 0.00000E+00 0.00000E+00-2.22100E-01-2.22100E
: -01 0.00000E+00 0.00000E+00-7.40500E-03 7.40500E-03 0.00000E+00 0.00000E+00-7.40500E-03-7.40500E-03
: 0.00000E+00 0.00000E+00-2.02300E-01 5.42000E-02 0.00000E+00 0.00000E+00-2.02300E-01-5.42000E-02 0.00
: 000E+00 0.00000E+00-1.48100E-01 1.48100E-01 0.00000E+00 0.00000E+00-1.48100E-01-1.48100E-01 0.00000E
: +00 0.00000E+00-5.42000E-02 2.02300E-01 0.00000E+00 0.00000E+00-5.42000E-02-2.02300E-01 0.00000E+00
: 0.00000E+00
type 058 len 0035 : 00 1.91000E+09 4.00000E-0200

logrec 14 type T

type 070 len 0054 : P1992,001,00:00:00.0000~1992,002,00:00:00.0000~
type 074 len 0082 : AAK BHE1992,001,00:00:00.0350~000015011992,001,23:02:55.5376~00045401000
type 074 len 0082 : AAK BHN1992,001,00:00:00.0350~000455011992,001,23:02:52.3376~00089401000
type 074 len 0082 : AAK BHZ1992,001,00:00:00.0350~000895011992,001,23:02:35.0375~00133401000
type 074 len 0082 : ANTO LHE1992,001,00:00:00.9100~001335011992,001,23:08:47.9100~00137601000
type 074 len 0082 : ANTO LHN1992,001,00:00:00.9100~001377011992,001,23:08:47.9100~00141801000
type 074 len 0082 : ANTO LHZ1992,001,00:00:00.9100~001419011992,001,23:08:47.9100~00146001000
type 074 len 0082 : BJI LHE1992,001,00:00:00.9300~001461011992,001,23:08:47.9300~00150201000
type 074 len 0082 : BJI LHN1992,001,00:00:00.9300~001503011992,001,23:08:47.9300~00154401000
type 074 len 0082 : BJI LHZ1992,001,00:00:00.9300~001545011992,001,23:08:47.9300~00158601000
type 074 len 0082 : ANMO BHN1992,001,00:01:57.7810~001587011992,001,23:00:31.0310~00204401000
type 074 len 0082 : ANMO BHE1992,001,00:02:27.6810~002045011992,001,23:00:59.9310~00250201000
type 074 len 0082 : ANMO BHZ1992,001,00:02:29.6810~002503011992,001,23:01:01.4300~00296001000
type 074 len 0082 : ANMO LHZ1992,001,00:03:27.1310~002961011992,001,23:10:52.1310~00300601000
type 074 len 0082 : ANMO LHE1992,001,00:20:24.1310~003007011992,001,23:30:07.1310~00305201000
type 074 len 0082 : ANMO LHN1992,001,00:23:56.1310~003053011992,001,23:03:57.1310~00309701000
type 074 len 0082 : BJI BHE1992,001,07:53:03.3700~003098011992,001,08:03:45.5200~00310401000
type 074 len 0082 : BJI BHE1992,001,08:59:31.7700~003105011992,001,09:10:30.8200~00311101000
type 074 len 0082 : BJI BHE1992,001,13:58:56.4700~003112011992,001,14:09:55.5200~00311801000
type 074 len 0082 : BJI BHE1992,001,16:03:09.3700~003119011992,001,16:13:51.5200~00312501000
type 074 len 0082 : BJI BHE1992,001,19:54:58.0700~003126011992,001,20:05:40.2200~00313201000
type 074 len 0082 : BJI BHE1992,001,20:18:54.5700~003133011992,001,20:29:36.7200~00313901000
type 074 len 0082 : BJI BHN1992,001,07:53:03.3700~003140011992,001,08:03:45.5200~00314601000
type 074 len 0082 : BJI BHN1992,001,08:59:31.7700~003147011992,001,09:10:30.8200~00315301000
type 074 len 0082 : BJI BHN1992,001,13:58:56.4700~003154011992,001,14:09:55.5200~00316001000
type 074 len 0082 : BJI BHN1992,001,16:03:09.3700~003161011992,001,16:13:51.5200~00316701000
type 074 len 0082 : BJI BHN1992,001,19:54:58.0700~003168011992,001,20:05:40.2200~00317401000
type 074 len 0082 : BJI BHN1992,001,20:18:54.5700~003175011992,001,20:29:36.7200~00318101000
type 074 len 0082 : BJI BHZ1992,001,07:53:03.3700~003182011992,001,08:03:45.5200~00318801000
type 074 len 0082 : BJI BHZ1992,001,08:59:31.7700~003189011992,001,09:10:30.8200~00319501000
type 074 len 0082 : BJI BHZ1992,001,13:58:56.4700~003196011992,001,14:09:55.5200~00320201000
type 074 len 0082 : BJI BHZ1992,001,16:03:09.3700~003203011992,001,16:13:51.5200~00320901000
type 074 len 0082 : BJI BHZ1992,001,19:54:58.0700~003210011992,001,20:05:40.2200~00321601000
type 074 len 0082 : BJI BHZ1992,001,20:18:54.5700~003217011992,001,20:29:36.7200~00322301000

Examples of Log and Opaque Data Records

Example Log Channel Blockette (header) Summary

logrec 2 type ‘A ‘
type 030 len 0067 : ASCII Quanterra Baler14 log files, State of Health~000208000
type 033 len 0055 : 002Streckeisen STS-2/Quanterra 330 Linear Phase~

logrec 3 type ‘S ‘
type 050 len 0100 : C16A -77.124400+167.898400+2.00001000Iceberg C16 Station A~0013210102003,291,00:00:00~~NXV
t□

Example Log Channel Fixed Section Data DATA RECORD

STATION	LOCATION	CHANNEL	NETWORK	TIME
C16A		LOG	XV	2003,353,032406.6200

# samples in record:	4021
sample_rate:	0
multiplier:	1
activity flags:	
I/O and clock flags:	
data quality flags:	
# of blockettes:	1
time correction:	0
begin data offset:	56
begin blkette offset:	48
BLOCKETTE 1000:	
encoding format:	ASCII text (val:0)
word order:	68000/SPARC word order
data record length:	12
reserved:	0

Example of an Ace Channel, which has blockette 500:

logrec 2 type 'A '
type 030 len 0038 : Quanterra ACE Channel~000109100

STATION	LOCATION	CHANNEL	NETWORK	TIME
C16A		ACE	XV	2003,355,000000.0000 [1]

# samples in record:	0
sample_rate:	0
multiplier:	1
activity flags:	
I/O and clock flags:	
data quality flags:	
# of blockettes:	2
time correction:	0
begin data offset:	0
begin blkette offset:	48
BLOCKETTE 1000:	
encoding format:	ASCII text (val:0)
word order:	68000/SPARC word order
data record length:	12
reserved:	0
BLOCKETTE 500:	
VCO correction:	35.9863
time of exception:	2003,355,00:00:00.0000
clock time in usec:	1
reception quality:	100
exception count:	3085
exception type:	Daily Timemark
clock mode:l	P133T11NR1

Example of an "Opaque" Channel, which has blockette 2000's:

STATION	LOCATION	CHANNEL	NETWORK	TIME
C16A		OCF	XV	2003,355,03:08:01.0000 [1]:

# samples in record	0
sample_rate	0
multiplier	1
activity flags	
I/O and clock flags	
data quality flags	
# of blockettes	7
time correction	0

Appendix F: Cross Reference for Fields in Abbreviation Dictionaries

Contributed by Tim Ahern

Some abbreviation dictionaries in a SEED volume contain fields that reference other blockettes. Here is a list:

			References		
Blockette	Field	Blockette Name	Blockette	Field	Blockette Name
31	6	Comment Description	34	3	Units Abbreviations
41	6	FIR Dictionary	34	3	Units Abbreviations
41	7	FIR Dictionary	34	3	Units Abbreviations
43	6	Response (Poles & Zeros) Dictionary	34	3	Units Abbreviations
43	7	Response (Poles & Zeros) Dictionary	34	3	Units Abbreviations
44	6	Response (Coefficients) Dictionary	34	3	Units Abbreviations
44	7	Response (Coefficients) Dictionary	34	3	Units Abbreviations
45	5	Response List Dictionary	34	3	Units Abbreviations
45	6	Response List Dictionary	34	3	Units Abbreviations
46	5	Generic Response Dictionary	34	3	Units Abbreviations
46	6	Generic Response Dictionary	34	3	Units Abbreviations
50	10	Station Identifier	33	3	Generic Abbreviation
51	5	Station Comment	31	3	Comment Description
52	6	Channel Identifier	33	3	Generic Abbreviation
52	8	Channel Identifier	34	3	Units Abbreviations
52	9	Channel Identifier	34	3	Units Abbreviations
52	16	Channel Identifier	30	4	Data Format Dictionary
53	5	Response (Poles & Zeros)	34	3	Units Abbreviations
53	6	Response (Poles & Zeros)	34	3	Units Abbreviations
54	5	Response (Coefficients)	34	3	Units Abbreviations
54	6	Response (Coefficients	34	3	Units Abbreviations
55	4	Response List	34	3	Units Abbreviations
55	5	Response List	34	3	Units Abbreviations
56	4	Generic Response	34	3	Units Abbreviations
56	5	Generic Response	34	3	Units Abbreviations
59	5	Channel Comment	31	3	Comment Description
60	6	Response Reference	41	3	FIR Dictionary
60	6	Response Reference	43	3	Response (Poles & Zeros) Dictionary
60	6	Response Reference	44	3	Response (Coefficients) Dictionary

60	6	Response Reference	45	3	Response List Dictionary
60	6	Response Reference	46	3	Generic Response Dictionary
60	6	Response Reference	47	3	Decimation Dictionary
60	6	Response Reference	48	3	Channel Sensitivity/Gain Dictionary
61	6	FIR Response	34	3	Units Abbreviations
61	7	FIR Response	34	3	Units Abbreviations
71	4	Hypocenter Information	32	3	Cited Source Dictionary
71	11	Hypocenter Information	32	3	Cited Source Dictionary
71	11+pX3	Hypocenter Information	32	3	Cited Source Dictionary
72	11	Event Phases	32	3	Cited Source Dictionary
400	5	Beam	35	3	Beam Configuration

For example, field 4 of the Hypocenter Info Blockette [71] references field 3 of the Cited Source Dictionary Blockette [32].

NOTE: Field 11+pX3 of the Hypocenter Info Blockette [71] is the last field in the blockette's group. In the equation, p = field 8 of the blockette.

Some fields are referenced by more than one blockette:

Blockette	Field	Blockette Name	Is Referenced by Blockette	Field	Blockette Name
30	4	Data Format Dictionary	52	16	Channel Identifier
31	3	Comment Description	51	5	Station Identifier
31	3	Comment Description	59	5	Channel Comment
32	3	Cited Source Dictionary	71	4	Hypocenter Information
32	3	Cited Source Dictionary	71	11	Hypocenter Information
32	3	Cited Source Dictionary	71	11+pX3	Hypocenter Information
32	3	Cited Source Dictionary	72	11	Event Phase Blockette
33	3	Generic Abbreviation	50	10	Station Identifier
33	3	Generic Abbreviation	52	6	Channel identifier
34	3	Units Abbreviations	31	6	Comment Description
34	3	Units Abbreviations	41	6	FIR Dictionary
34	3	Units Abbreviations	41	7	FIR Dictionary
34	3	Units Abbreviations	43	6	Response (Poles & Zeros) Dictionary
34	3	Units Abbreviations	43	7	Response (Poles & Zeros) Dictionary
34	3	Units Abbreviations	44	6	Response (Coefficients) Dictionary
34	3	Units Abbreviations	44	7	Response (Coefficients) Dictionary
34	3	Units Abbreviations	45	5	Response List Dictionary
34	3	Units Abbreviations	45	6	Response List Dictionary
34	3	Units Abbreviations	46	5	Generic Response Dictionary
34	3	Units Abbreviations	46	6	Generic Response Dictionary
34	3	Units Abbreviations	52	8	Channel Identifier
34	3	Units Abbreviations	52	9	Channel Identifier
34	3	Units Abbreviations	53	5	Response (Poles & Zeros)
34	3	Units Abbreviations	53	6	Response (Poles & Zeros)
34	3	Units Abbreviations	54	5	Response (Coefficients)
34	3	Units Abbreviations	54	6	Response (Coefficients)
34	3	Units Abbreviations	55	4	Response List
34	3	Units Abbreviations	55	5	Response List
34	3	Units Abbreviations	56	4	Generic Response
34	3	Units Abbreviations	56	5	Generic Response
34	3	Units Abbreviations	61	6	FIR Response
34	3	Units Abbreviations	61	7	FIR Response
35	3	Beam Configuration	400	5	Beam
41	3	FIR Dictionary	60	6	Response Reference
43	3	Response (Poles & Zeros) Dictionary	60	6	Response Reference
44	3	Response (Coefficients) Dictionary	60	6	Response Reference
45	3	Response List Dictionary	60	6	Response Reference
46	3	Generic Response Dictionary	60	6	Response Reference
47	3	Decimation Dictionary	60	6	Response Reference
48	3	Channel Sensitivity/Gain Dictionary	60	6	Response Reference

Appendix G: Data Only SEED Volumes (Mini-SEED)

Contributed by Tim Ahern

The SEED format consists of Volume Control Headers, Abbreviation Control Headers, Station Control Headers, Time Span Control Headers and finally Data Records. At the 1991 FDSN meeting in Vienna, Austria the concept of Dataless SEED volumes was introduced and accepted. The structure of SEED data records is simple, straightforward, and much simpler to understand than the control header structure of SEED. Some data loggers offer SEED data records as a method of transferring waveform information. The term Data Only SEED Volumes (Mini-SEED) has come to be used to identify SEED data records without any of the associated control header information. Data Only and Dataless SEED volumes are to a certain extent the two parts of a complete SEED volume. Only Time Span Control Headers are not included in either of these components, however Time Span Control Headers can be derived from the Data Only SEED.

The SEED format standard is defined by the FDSN Working Group on Data Exchange. This working group has recognized the need to more specifically address the definition and use of Data Only SEED as a data exchange format. Data Only SEED also has potential for use as a data analysis format. In the SEED format, much of the information needed to specify the time series in the data records is in the SEED control headers. In fact the data record portion of the SEED format does not contain information about the organization of the data in the Data Only SEED records. Missing information includes

1) specification of the data encoding format as normally specified in the DDL

2) the byte swap order of the data as either VAX like or Motorola like.

3) the data record length

With the inclusion of the above information, the Data Only format can be used to completely decode the time series information in the data records. Of course response information and some other information remains unavailable and the need to retain full SEED volume production is encouraged.

The Data Only SEED data blockette has been designed to include the needed information. The data blockette is defined as follows;

[1000] Data Only SEED Blockette (8 bytes)

Note	Field name	Type	Length	Mask or Flags
1	Blockette type - 1000	B	2	
2	Next blockette's byte number	B	2	
3	Encoding Format	B	1	
4	Word order	B	1	
5	Data Record Length	B	1	
6	Reserved	B	1	

1 UWORD : Blockette type (1000): Data Only SEED

2 UWORD : Byte number of next blockette. (Calculate this as the byte offset from the beginning of the logical record - including the fixed section of the data header; use 0 if no more blockettes will follow.)

3 BYTE : A code indicating the encoding format. This number is assigned by the FDSN Data Exchange Working Group. To request that a new format be included contact the FDSN through the FDSN Archive at the IRIS Data Management Center. To be supported in Data Only SEED, the data format must be expressible in SEED DDL. A list of valid codes at the time of publication follows.

CODES 0-9	GENERAL
0	ASCII text, byte order as specified in field 4
1	16 bit integers
2	24 bit integers
3	32 bit integers
4	IEEE floating point
5	IEEE Double precision floating point
CODES 10 - 29	FDSN Networks
10	STEIM (1) Compression
11	STEIM (2) Compression
12	GEOSCOPE Multiplexed Format 24 bit integer
13	GEOSCOPE Multiplexed Format 16 bit gain ranged, 3 bit exponent
14	GEOSCOPE Multiplexed Format 16 bit gain ranged, 4 bit exponent
15	US National Network compression
16	CDSN 16 bit gain ranged
17	Graefenberg 16 bit gain ranged
18	IPG - Strasbourg 16 bit gain ranged
19	STEIM (3) Compression
CODES 30 - 49	OLDER NETWORKS
30	SRO Format
31	HGLP Format
32	DWWSSN Gain Ranged Format
33	RSTN 16 bit gain ranged

4 The byte swapping order for 16 bit and 32 bit words. A 0 indicates VAX or 8086 order and a 1 indicates 68000 or SPARC word order. See fields 11 and 12 of blockette 50.

5 The exponent (as a power of two) of the record length for these data. The data record can be as small as 256 bytes and, in Data Only SEED format as large as 2 raised to the 256 power.

Additional Considerations in Data Only SEED

1 Any SEED data blockette can be included in the Data Only SEED format except those that refer to abbreviation dictionary blockettes. For instance blockette 100 is permitted but blockette 400 is not.

2 The Data Only SEED data blockette can be present in a full SEED volume. In this case the values in the Data Only SEED blockette take precedence over values in the SEED control headers.

3 When combining Data Only SEED data records with a Dataless SEED volume to produce a SEED volume, Time Span Control Headers must be constructed. The only other major consideration is that if the Data Only SEED record length exceeds the maximum length of 4096 bytes allowed in SEED, then the longer Data Only SEED data records must be blocked into data records of valid length.

4 Much of the necessary information needed to make the time series decipherable is already well defined in the fixed section of the data header. These fields remain unchanged.

5 Each data record must have blockette 1000.

6 The order of the fixed section of the data header must be the same as field 4 of blockette 1000 implies.

The IRIS SEED reader, RDSEED, is now capable of processing Dataless SEED volumes and Data Only SEED volumes simultaneously and therefore may provide a method of reading Mini- SEED data records that do not include blockette 1000.

Appendix H: Effective Times and Update Records

Contributed by Tim Ahern

By properly using effective times in blockettes [50] and [52] and the update flag (field 15 of blockette [50] and field 24 of blockette [52]), you can use SEED to help maintain your databases.

Effective times can specify different degrees of precision. If effective times to the nearest hour are sufficient, they can be so specified: for example, 1990,032,09~. If you need times specified to the minute, then use 1990,032,09:57~. In all cases, effective times must follow the SEED mask properly for the TIME structure: YYYY,DDD,HH:MM:SS.TTTT, where Y=year's digits; D=day of year; H=hours digits; M=minutes digits; S=seconds digits; and T=fractions of seconds digits.

Effective times should represent the date and time when a change actually occurred at a station or channel. For example, if a specific channel has a response determined at 1989,033,12:25 and then re-determined at 1990,032,09:57, then the effective times for the relevant blockette [52] written before 1990,032,09:57 should appear this way:

0520113 BHE ----------1989,033,12:25~~N

0530718 etc.

0580035 etc.

This blockette can also be represented symbolically, where "B" represents the beginning effective time, "E" the ending effective time, ">" an unspecified ending effective time, "-" time in a blockette with the "N" update flag, and "^" time in a blockette with the "U" update flag:

B-------->

A SEED volume containing data for day 1990,032 may have data with different responses, one set before and one set after the responses were determined. Designate this situation this way:

0520113 BHE ----------1989,033,12:25~1990,032,09:57~N
0530718 etc.
0580035 etc.

0520113 BHE ----------1990,032,09:57~~N
0530745 etc.
0580035 etc.

Using our symbols described above, we would write:

B -------- E
B-------->

Note that the first blockette [52] has both starting and ending effective times, whereas the second blockette [52] has just the beginning effective time. If the above situation did not exist, include only the second sequence of blockettes [52] — [59]. This use is station dependent, and the institution generating the SEED volume must decide whether the above case or the following case applies.

For the next SEED volume it is likely that only data after 1990,032,09:57 exist, so later volumes would only have a blockette [52] such as:

0520113 BHE ----------1990,032,09:57~~N
0530745 etc.
0580035 etc.

When reading a SEED volume that contain blockettes with a new beginning effective time, the reading institution may need to modify its database accordingly. For instance, the SEED volume recipient would fill in the ending effective time for the earlier blockette [52] and append the new blockette [52] to the database. Before accepting the new blockette, the receiving database should indicate the following situation:

0520113 BHE ----------1989,033,12:25~~N
0530718 etc.
0580035 etc.

upon receiving a SEED volume containing:

0520113 BHE ----------1990,032,09:57~~N
0530745 etc.
0580035 etc.

The recipient of the above SEED blockettes would update the local database to contain:

0520113 BHE ----------1989,033,12:25~1990,032,09:57~N
0530718 etc.
0580035 etc.

0520113 BHE ----------1990,032,09:57~~N

0530745 etc.

0580035 etc.

When using effective times, you should make the assumptions that the starting effective time includes the time specified in the blockette, and the ending effective times indicate times greater than the time specified. Adopting this convention will minimize confusion as to which blockettes apply to any particular sample in the time series.

When the update flags in blockettes [50] and [52] are set to "N," no information in blockettes [50] — [59] previously transmitted and stored in the data base needs to be modified. The only exception to this would be if the starting effective time is different than the previous effective time. In that case, the database should be modified to insert the ending effective time into the relevant blockettes as shown above. The order in which "N" records are inserted into the database is not important. However, if "N" type blockettes are received out of sequence, the following situation might be encountered:


```
B ---------- >
        B -------- >
B ---------- >
```

The database would be modified to reflect the following situation

```
B -------- E
        B ------- >
```

A blockette with a later effective starting time must always override an unspecified ending effective time. The above situation should only happen if SEED volumes are processed in an order other than the one in which they were written.

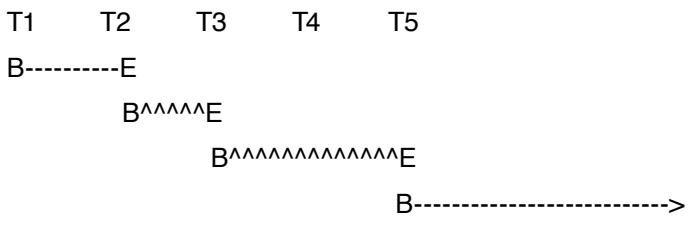
The "U" option for the update flag indicates that information previously transmitted was incorrect and the previous information should be corrected. The relevant blockettes [50] — [59] that are transmitted with the update flag set to "U" are intended to replace existing entries in the database. The order in which "U" records are processed is important if the effective times of the "U" records overlap. We expect "U" records to be transmitted infrequently, so overlapping "U" records do not need to be handled automatically but can rely upon operator intervention. The information transmitted with the "U" update flag will physically replace the relevant "N" records in the database, so you should assume that the information in the "N" records will be lost. Furthermore, if a later "N" record overlaps a previously received "U" record, the "U" record takes precedence.

An illustration may help to clarify this situation. (Remember that "-" specifies an "N" type blockette and "^" specifies a "U" type blockette.)

```
T1      T2      T3      T4      T5
B---------------------------------------------------------->
B---------------------------------------------------------->
                                B------------------------->
                                B------------------------->
        B^^^^^^^^^^^^^^E
                                B------------------------->
                B^^^^^^^^^^^^^^E
                B------------------------------------------>
```

The above example, while complicated, illustrates how to use update flags. Two blockettes with starting effective times T_1 are sent, then two blockettes with starting effective times T_5.Then the sender determines that incorrect information was transmitted for a very short time period T_2-T_4. An update record is transmitted as a correction. The sender then transmits another record with an "N" update flag and starting effective time T_5. Then, the sender realizes that the event occurring at T_5 actually occurred at T_3. The sender transmits another update record to correct the database. Subsequent blockettes are transmitted with starting effective time T_3.

The recipient would use the information transmitted as follows after first determining the correct order of the two update records to produce the following representation:



The information in the last "U" record and the last "N" record above would be identical except for different effective times. For this reason, it is not significant that the last "N" record begins at T_5 and not T_3 as originally transmitted. Although the specific method of storing the above representation may differ between database implementations, the symbolic representation above must be derivable from that implementation.

Here is a clear way of creating the proper view of the situation:

Take all "N" records received and place them on a time line. The beginning effective time of the blockettes determines where the various "N" records are placed. Place the "N" records with the earliest beginning effective time first, and place the remainder in order of the beginning effective time. Now place any "U" records on top of the time line created by the "N" records. The "U" records must be placed in the order they were written, not in the order they were received — nor in the order of earliest beginning effective time. The timeline produced this way will represent the true state of the station or channel at any given time.

Remember that the database retains the value of the update flag "N" or "U" as received. SEED volumes not yet received may contain overlapping "U" type blockettes whose presence must be detected. The "U" flags must be translated to "N" flags when output volumes are generated.

Here are the basic rules governing the use of "U" update blockettes:

1. Blockettes with "U" update flags must have both starting and ending effective times.
2. "U" blockettes always supersede "N" blockettes.
3. Multiple "U" blockettes are applied in the same order as generated. For this reason, we advise that any recipient of SEED data that intends to maintain a local database has a mechanism to determine this order. The SEED format itself cannot accomplish this.
4. "U" blockettes can only be used to modify information previously transmitted in "N" or "U" blockettes. It is illegal to send a "U" blockette if an "N" blockette has not previously been written.
5. The value of the update flag must be retained in the database unmodified.
6. Blockettes [53] through [58] are coupled to the corresponding blockette [52].

Appendix J

Appendix J: Network Codes

Contributed by Tim Ahern

The following network codes are assigned by the FDSN archive (IRIS DMC) to provide uniqueness to seismological data streams.

The first line provides the network code and the network name. The second line of each entry provides the name of the network operator or responsible organization. For example:

AA Anchorage Strong Motion Network
Geophysical Institute, University of Alaska, Fairbanks

The current list of Network Codes is available online at:
http://www.iris.edu/stations/networks.txt

Appendix K

Appendix K: Flinn-Engdahl Seismic Regions

Code	Seismic Region	Description
1	1	Central Alaska
2	1	Southern Alaska
3	1	Bering Sea
4	1	Komandorsky Islands Region
5	1	Near Islands, Aleutians Islands
6	1	Rat Islands, Aleutian Islands
7	1	Andreanof Islands, Aleutian Islands
8	1	Pribilof Islands
9	1	Fox Islands, Aleutians Islands
10	1	Unimak Island Region
11	1	Bristol Bay
12	1	Alaska Peninsula
13	1	Kodiak Island Region
14	1	Kenai Peninsula, Alaska
15	1	Gulf of Alaska
16	1	Aleutian Islands Region
17	1	South of Alaska
18	2	Southern Yukon Territory, Canada
19	2	Southeastern Alaska
20	2	Off Coast of Southeastern Alaska
21	2	West Vancouver Island
22	2	Queen Charlotte Islands Region
23	2	British Columbia
24	2	Alberta Province, Canada
25	2	Vancouver Island Region
26	2	Off Coast of Washington
27	2	Near Coast of Washington
28	2	Washington-Oregon Border Region
29	2	Washington
30	3	Off Coast of Oregon

31	3	Near Coast of Oregon
32	3	Oregon
33	3	Western Idaho
34	3	Off Coast of Northern California
35	3	Near Coast of Northern California
36	3	Northern California
37	3	Nevada
38	3	Off Coast of California
39	3	Central California
40	3	California-Nevada Border Region
41	3	Southern Nevada
42	3	Western Arizona
43	3	Southern California
44	3	California-Arizona Border Region
45	3	California-Mexico Border Region
46	3	W. Arizona-Mexico Border Region
47	4	Off W. Coast of Baja California
48	4	Baja California
49	4	Gulf of California
50	4	Northwestern California
51	4	Off Coast of Central Mexico
52	4	Near Coast of Central Mexico
53	5	Revilla Gigedo Islands Region
54	5	Off Coast of Jalisco, Mexico
55	5	Near Coast of Jalisco, Mexico
56	5	Near Coast of Michoacan, Mexico
57	5	Michoacan, Mexico
58	5	Near Coast of Guerrero, Mexico
59	5	Guerrero, Mexico
60	5	Oaxaca, Mexico
61	5	Chiapas, Mexico
62	5	Mexico-Guatemala Border Region
63	5	Off Coast of Mexico
64	5	Off Coast of Michoacan, Mexico
65	5	Off Coast of Guerrero. Mexico
66	5	Near Coast of Oaxaca
67	5	Off Coast of Oaxaca
68	5	Off Coast of Chiapas
69	5	Near Coast Chiapas
70	5	Guatemala
71	5	Near Coast of Guatemala
72	6	Honduras
73	6	El Salvador
74	6	Near Coast of Nicaragua
75	6	Nicaragua
76	6	Off Coast of Central America
77	6	Off Coast of Costa Rica
78	6	Costa Rica
79	6	North of Panama
80	6	Panama-Costa Rica Border Region
81	6	Panama
82	6	Panama-Columbia Border Region
83	6	South of Panama
84	7	Yucatan Peninsula
85	7	Cuba Region
86	7	Jamaica Region
87	7	Haiti Region
88	7	Dominican Republic Region
89	7	Mona Passage
90	7	Puerto Rico Region
91	7	Virgin Islands
92	7	Leeward Islands
93	7	Belize
94	7	Caribbean Sea
95	7	Windward Islands
96	7	Near North Coast of Columbia
97	7	Near Coast of Venezuela
98	7	Trinidad
99	7	Northern Columbia
100	7	Lake Maracaibo
101	7	Venezuela
102	7	Near West Coast of Columbia
103	8	Columbia
104	8	Off Coast of Ecuador
105	8	Near Coast of Ecuador
106	8	Columbia-Ecuador Border Region
107	8	Ecuador
108	8	Off Coast of Northern Peru
109	8	Near Coast of Northern Peru
110	8	Peru-Ecuador Border Region
111	8	Northern Peru
112	8	Peru-Brazil Border Region
113	8	Western Brazil
114	8	Off Coast of Peru
115	8	Near Coast of Peru
116	8	Peru
117	8	Southern Peru
118	8	Peru-Bolivia Border Region
119	8	Northern Bolivia
120	8	Bolivia
121	8	Off Coast of Northern Chile
122	8	Near Coast of Northern Chile
123	8	Northern Chile
124	8	Chile-Bolivia Border Region
125	8	Southern Bolivia
126	8	Paraguay
127	8	Chile-Argentina Border Region
128	8	Jujuy Province, Argentina
129	8	Salta Province, Argentina
130	8	Catamarca Province, Argentina
131	8	Tucuman Province, Argentina
132	8	Santiago Del Estero Province, Argentina
133	8	Northeastern Argentina
134	8	Off Coast of Central Chile
135	8	Near Coast of Central Chile
136	8	Central Chile
137	8	San Juan Province, Argentina
138	8	La Rioja Province, Argentina
139	8	Mendoza Province, Argentina
140	8	San Luis Province, Argentina
141	8	Cordoba Province, Argentina
142	8	Uruguay
143	9	Off Coast of Southern Chile
144	9	Near Coast of Southern Chile

145	9	South Chile-Argentina Border Region
146	10	Argentina
147	10	Tierra del Fuego
148	10	Falkland Islands Region
149	10	Drake Passage
150	10	Scotia Sea
151	10	South Georgia Island Region
152	10	South Georgia Rise
153	10	South Sandwich Islands Region
154	10	South Shetland Islands Region
155	10	Antarctic Peninsula
156	10	Southwestern Atlantic Ocean
157	10	Weddell Sea
158	11	Off West Coast of North Island New Zealand
159	11	North Island, New Zealand
160	11	Off East Coast of North Island, New Zealand
161	11	Off West Coast of South Island, New Zealand
162	11	South Island, New Zealand
163	11	Cook Straight, New Zealand
164	11	Off East Coast of South Island, New Zealand
165	11	North of MacQuarie Island
166	11	Aukland Islands Region
167	11	MacQuarie Islands Region
168	11	South of New Zealand
169	12	Samoa Islands Region
170	12	Samoa Islands
171	12	South of Fiji Islands
172	12	West of Tonga Islands
173	12	Tonga Islands
174	12	Tonga Islands Region
175	12	South of Tonga Islands
176	12	North of New Zealand
177	12	Kermadec Islands Region
178	12	Kermadec Islands
179	12	South of Kermadec Islands
180	13	North of Fiji Islands
181	13	Fiji Islands Region
182	13	Fiji Islands
183	14	Santa Cruz Islands Region
184	14	Santa Cruz Islands
185	14	Vanuatu Islands Region
186	14	Vanuatu Islands
187	14	New Caledonia
188	14	Loyalty Islands
189	14	Loyalty Islands Region
190	15	New Ireland Region
191	15	North of Solomon Islands
192	15	New Britain Region
193	15	Solomon Islands
194	15	Dentrecasteaux Islands Region
195	15	Solomon Islands Region
196	16	West Irian Region
197	16	Near North Coast of West Irian
198	16	Papua, New Guinea Region
199	16	Admiralty Islands Region
200	16	Near North Coast of Papua, New Guinea
201	16	West Irian
202	16	Papua New Guinea
203	16	Bismarck Sea
204	16	Aroe Islands Region
205	16	Near South Coast of West Irian
206	16	Near South Coast of Papua, New Guinea
207	16	East Papua, New Guinea Region
208	16	Arafura Sea
209	17	West Caroline Islands
210	17	South of Mariana Islands
211	18	South of Honshu, Japan
212	18	Bonin Islands Region
213	18	Volcano Islands Region
214	18	West of Mariana Islands
215	18	Mariana Islands Region
216	18	Mariana Islands
217	19	Kamchatka
218	19	Near East Coast of Kamchatka
219	19	Off East Coast of Kamchatka
220	19	Northwest of Kuril Islands
221	19	Kuril Islands
222	19	Kuril Islands Region
223	19	Eastern Sea of Japan
224	19	Hokkaido, Japan Region
225	19	Off Coast of Hokkaido, Japan
226	19	Near West Coast of Honshu, Japan
227	19	Honshu, Japan
228	19	Near East Coast of Honshu, Japan
229	19	Off East Coast of Honshu, Japan
230	19	Near South Coast of Honshu, Japan
231	20	South Korea
232	20	Southern Honshu, Japan
233	20	Near South Coast of Southern Honshu
234	20	East China Sea
235	20	Kyushu, Japan
236	20	Shikoku, Japan
237	20	Southeast of Shikoku, Japan
238	20	Ryukyu Islands
239	20	Ryukyu Islands Region
240	20	East of Ryukyu Islands
241	20	Philippine Sea
242	21	Near Southeastern Coast of China
243	21	Taiwan Region
244	21	Taiwan
245	21	Northeast of Taiwan
246	21	Southwestern Ryukyu Islands
247	21	Southeast of Taiwan
248	22	Philippine Islands Region
249	22	Luzon, Philippine Islands
250	22	Mindoro, Philippine Islands
251	22	Samar, Philippine Islands
252	22	Palawan, Philippine Islands
253	22	Sulu Sea
254	22	Panay, Philippine Islands
255	22	Cebu, Philippine Islands
256	22	Leyte, Philippine Islands
257	22	Negros, Philippine Islands
258	22	Sulu Archipelago

259	22	Mindanao, Philippine Islands
260	22	East of Philippine Islands
261	23	Kalimantan
262	23	Celebes Sea
263	23	Talaud Islands
264	23	North of Halmahera
265	23	Minahassa Peninsula
266	23	Molucca Passage
267	23	Halmahera
268	23	Sulawesi
269	23	Molucca Sea
270	23	Ceram Sea
271	23	Buru
272	23	Ceram
273	24	Southwest of Sumatera
274	24	Southern Sumatera
275	24	Java Sea
276	24	Sunda Strait
277	24	Java
278	24	Bali Sea
279	24	Flores Sea
280	24	Banda Sea
281	24	Tanimbar Islands Region
282	24	South of Java
283	24	Bali Island Region
284	24	South of Bali Island
285	24	Sumbawa Island Region
286	24	Flores Island Region
287	24	Sumba Island region
288	24	Savu Sea
289	24	Timor
290	24	Timor Sea
291	24	South of Sumbawa Island
292	24	South of Sumba Island
293	24	South of Timor
294	25	Burma-India Border Region
295	25	Burma-Bangladesh Border Region
296	25	Burma
297	25	Burma-China Border Region
298	25	South Burma
299	25	Southeast Asia
300	25	Hainan Island
301	25	South China Sea
302	26	Eastern Kashmir
303	26	Kashmir-India Border Region
304	26	Kashmir-Tibet Border Region
305	26	Tibet-India Border Region
306	26	Tibet
307	26	Sichuan Province, China
308	26	Northern India
309	26	Nepal-India Border Region
310	26	Nepal
311	26	Sikkim
312	26	Bhutan
313	26	India - China Border Region
314	26	India
315	26	India-Bangladesh Border Region
316	26	Bangladesh
317	26	Eastern India
318	26	Yunnan Province, China
319	26	Bay of Bengal
320	27	Kirghiz-Xinjiang Border Region
321	27	Southern Xinjiang, China
322	27	Gansu Province China
323	27	Northern China
324	27	Kashmir-Xinjiang Border Region
325	27	Qinghai Province, China
326	28	Central USSR
327	28	Lake Baikal Region
328	28	East of Lake Baikal
329	28	Eastern Kazakh SSR
330	28	Alma-Ata Region
331	28	Kazakh-Xinjiang Border Region
332	28	Northern Xinjiang Region
333	28	USSR-Mongolia Border Region
334	28	Mongolia
335	29	Ural Mountains Region
336	29	Western Kazakh SSR
337	29	Eastern Caucasus
338	29	Caspian Sea
339	29	Uzbek SSR
340	29	Turkmen SSR
341	29	Iran-USSR Border Region
342	29	Turkmen-Afghanistan Border Region
343	29	Turkey-Iran Border Region
344	29	North West Iran-USSR Border Region
345	29	Northwestern Iran
346	29	Iran-Iraq Border Region
347	29	Western Iran
348	29	Iran
349	29	Northwestern Afghanistan
350	29	Southwestern Afghanistan
351	29	Eastern Arabian Peninsula
352	29	Persian Gulf
353	29	Southern Iran
354	29	Pakistan
355	29	Gulf of Oman
356	29	Near Coast of Pakistan
357	30	Southwestern USSR
358	30	Romania
359	30	Bulgaria
360	30	Black Sea
361	30	Crimea Region
362	30	Western Caucasus
363	30	Greece-Bulgaria Border Region
364	30	Greece
365	30	Aegean Sea
366	30	Turkey
367	30	Turkey-USSR Border Region
368	30	Southern Greece
369	30	Dodecanese Islands
370	30	Crete
371	30	Eastern Mediterranean Sea
372	30	Cyprus

373	30	Dead Sea Region
374	30	Jordan - Syria Region
375	30	Iraq
376	31	Portugal
377	31	Spain
378	31	Pyrenees
379	31	Near South Coast of France
380	31	Corsica
381	31	Central Italy
382	31	Adriatic Sea
383	31	Yugoslavia
384	31	West of Gibraltar
385	31	Strait of Gibraltar
386	31	Balearic Islands
387	31	Western Mediterranean Sea
388	31	Sardinia
389	31	Tyrrhenian Sea
390	31	Southern Italy
391	31	Albania
392	31	Greece-Albania Border Region
393	31	Madeira Islands Region
394	31	Canary Islands Region
395	31	Morocco
396	31	Algeria
397	31	Tunisia
398	31	Sicily
399	31	Ionian Sea
400	31	Mediterranean Sea
401	31	Near Coast of Libya
402	32	North Atlantic Ocean
403	32	North Atlantic Ridge
404	32	Azores Islands Region
405	32	Azores Islands
406	32	Central Mid-Atlantic Ridge
407	32	North of Ascension Islands
408	32	Ascension Islands Region
409	32	South Atlantic Ocean
410	32	South Atlantic Ridge
411	32	Tristan Da Cunha Region
412	32	Bouvet Island Region
413	32	Southwest of Africa
414	32	Southeastern Atlantic Ocean
415	33	Eastern Gulf of Aden
416	33	Socotra Region
417	33	Arabian Sea
418	33	Laccadive Islands Region
419	33	Northeastern Somalia
420	33	North Indian Ocean
421	33	Carlsberg Ridge
422	33	Maldive Islands Region
423	33	Laccadive Sea
424	33	Sri Lanka
425	33	South Indian Ocean
426	33	Chagos Archipelago Region
427	33	Mascarene Islands Region
428	33	Atlantic-Indian Rise
429	33	Mid-Indian Rise
430	33	South of Africa
431	33	Prince Edward Islands Region
432	33	Crozet Islands Region
433	33	Kerguelen Islands Region
434	33	Amsterdam-Naturaliste Ridge
435	33	Southeast Indian Rise
436	33	Kerguelen-Gaussberg Rise
437	33	South of Australia
438	34	Saskatchewan Province, Canada
439	34	Manitoba Province, Canada
440	34	Hudson Bay
441	34	Ontario
442	34	Hudson Strait Region
443	34	Northern Quebec
444	34	Davis Strait
445	34	Labrador
446	34	East of Labrador
447	34	Southern Quebec
448	34	Gaspe Peninsula
449	34	Eastern Quebec
450	34	Anticosti Island, Canada
451	34	New Brunswick
452	34	Nova Scotia
453	34	Prince Edward Island, Canada
454	34	Gulf of Saint Lawrence
455	34	Newfoundland
456	34	Montana
457	34	Eastern Idaho
458	34	Hebgen Lake Region
459	34	Yellowstone National Park, Wyoming
460	34	Wyoming
461	34	North Dakota
462	34	South Dakota
463	34	Nebraska
464	34	Minnesota
465	34	Iowa
466	34	Wisconsin
467	34	Illinois
468	34	Michigan
469	34	Indiana
470	34	Southern Ontario
471	34	Ohio
472	34	New York
473	34	Pennsylvania
474	34	Northern New England
475	34	Maine
476	34	Southern New England
477	34	Gulf of Maine
478	34	Utah
479	34	Colorado
480	34	Kansas
481	34	Iowa-Missouri Border Region
482	34	Missouri-Kansas Border Region
483	34	Missouri
484	34	Missouri-Arkansas Border Region
485	34	Eastern Missouri
486	34	New Madrid, Missouri Region

487	34	Cape Girardeau, Missouri Region
488	34	Southern Illinois
489	34	Southern Indiana
490	34	Kentucky
491	34	West Virginia
492	34	Virginia
493	34	Chesapeake Bay Region
494	34	New Jersey
495	34	Eastern Arizona
496	34	New Mexico
497	34	Texas Panhandle Region
498	34	West Texas
499	34	Oklahoma
500	34	Central Texas
501	34	Arkansas-Oklahoma Border Region
502	34	Arkansas
503	34	Louisiana-Texas Border Region
504	34	Louisiana
505	34	Mississippi
506	34	Tennessee
507	34	Alabama
508	34	Western Florida
509	34	Georgia
510	34	Florida-Georgia Border Region
511	34	South Carolina
512	34	North Carolina
513	34	Off East Coast of United States
514	34	Florida Peninsula
515	34	Bahama Islands
516	34	Eastern Arizona-Mexico Border Region
517	34	Mexico-New Mexico Border Region
518	34	Texas-Mexico Border Region
519	34	Southern Texas
520	34	Texas Gulf Coast
521	34	Chihuahua, Mexico
522	34	Northern Mexico
523	34	Central Mexico
524	34	Jalisco, Mexico
525	34	Vera Cruz, Mexico
526	34	Gulf of Mexico
527	34	Gulf of Campeche
528	35	Brazil
529	35	Guyana
530	35	Suriname
531	35	French Guiana
532	35	Eire
533	36	United Kingdom
534	36	North Sea
535	36	Southern Norway
536	36	Sweden
537	36	Baltic Sea
538	36	France
539	36	Bay of Biscay
540	36	Netherlands
541	36	Belgium
542	36	Denmark
543	36	Germany
544	36	Switzerland
545	36	Northern Italy
546	36	Austria
547	36	Czechoslovakia
548	36	Poland
549	36	Hungary
550	37	Northwest Africa
551	37	Southern Algeria
552	37	Libya
553	37	Arab Republic of Egypt
554	37	Red Sea
555	37	Western Arabian Peninsula
556	37	Central Africa
557	37	Sudan
558	37	Ethiopia
559	37	Western Gulf of Aden
560	37	Northwestern Somalia
561	37	Off South Coast of Northwest Africa
562	37	Cameroon
563	37	Equatorial Guinea
564	37	Central African Republic
565	37	Gabon
566	37	Congo Republic
567	37	Zaire Republic
568	37	Uganda
569	37	Lake Victoria Region
570	37	Kenya
571	37	Southern Somalia
572	37	Lake Tanganyika Region
573	37	Tanzania
574	37	Northwest of Madagascar
575	37	Angola
576	37	Zambia
577	37	Malawi
578	37	Namibia
579	37	Botswana Republic
580	37	Zimbabwe
581	37	Mozambique
582	37	Mozambique Channel
583	37	Malagasay Republic
584	37	Republic of South Africa
585	37	Lesotho
586	37	Swaziland
587	37	Off Coast of South Africa
588	38	Northwest of Australia
589	38	West of Australia
590	38	Western Australia
591	38	Northern Territory, Australia
592	38	South Australia
593	38	Gulf of Carpenteria
594	38	Queensland, Australia
595	38	Coral Sea
596	38	South of Solomon Islands
597	38	New Caledonia Region
598	38	Southwest of Australia
599	38	Off South Coast of Australia
600	38	Near South Coast of Australia

601	38	New South Wales, Australia
602	38	Victoria, Australia
603	38	Near South East Coast of Australia
604	38	Near East Coast of Australia
605	38	East of Australia
606	38	Norfolk Island Region
607	38	Northwest of New Zealand
608	38	Bass Strait
609	38	Tasmania Region
610	38	Southeast of Australia
611	39	North Pacific Ocean
612	39	Hawaii Region
613	39	Hawaii
614	39	Caroline Islands Region
615	39	Marshall Islands Region
616	39	Eniwetok Atoll Region
617	39	Bikini Atoll Region
618	39	Gilbert Islands
619	39	Johnston Island Region
620	39	Line Islands Region
621	39	Palmyra Island Region
622	39	Christmas Island Region
623	39	Ellice Islands Region
624	39	Phoenix Islands Region
625	39	Tekelau Islands Region
626	39	Northern Cook Islands
627	39	Cook Islands Region
628	39	Society Islands Region
629	39	Tubuai Islands Region
630	39	Marquesas Islands Region
631	39	Tuamotu Archipelago Region
632	39	South Pacific Ocean
633	40	Lomonosov Ridge
634	40	Arctic Ocean
635	40	Near North Coast of Greenland
636	40	Eastern Greenland
637	40	Iceland Region
638	40	Iceland
639	40	Jan Mayen Island region
640	40	Greenland Sea
641	40	North of Svalbard
642	40	Norwegian Sea
643	40	Svalbard Region
644	40	North of Franz Josef Land
645	40	Franz Josef Land
646	40	Northern Norway
647	40	Barents Sea
648	40	Novaya Zemlya
649	40	Kara Sea
650	40	Near Coast of Western Siberia
651	40	North of Severnaya Zemlya
652	40	Severnaya Zemlya
653	40	Near Coast of Central Zemlya
654	40	East of Severnaya Zemlya
655	40	Laptev Sea
656	41	Eastern USSR
657	41	East USSR-North East China Border Region
658	41	Northeastern China
659	41	North Korea
660	41	Sea of Japan
661	41	Near East Coast of Eastern USSR
662	41	Sakhalin Island
663	41	Sea of Okhotsk
664	41	Eastern China
665	41	Yellow Sea
666	41	Off Coast of Eastern China
667	42	North of New Siberian Islands
668	42	New Siberian Islands
669	42	East Siberian Sea
670	42	Near North Coast of Eastern Siberia
671	42	Eastern Siberia
672	42	Chukchi Sea
673	42	Bering Strait
674	42	Saint Lawrence Island Region
675	42	Beaufort Sea
676	42	Alaska
677	42	Northern Yukon Territory, Canada
678	42	Queen Elizabeth Islands
679	42	Northwest Territories, Canada
680	42	Western Greenland
681	42	Baffin Bay
682	42	Baffin Island Region
683	43	Southeast Central Pacific Ocean
684	43	Easter Island Cordillera
685	43	Easter Island Region
686	43	West Chile Rise
687	43	Juan Fernandez Islands Region
688	43	East of North Island, New Zealand
689	43	Chatham Islands Region
690	43	South of Chatham Islands
691	43	South of Pacific Cordillera
692	43	Southern Pacific Ocean
693	44	East Central Pacific Ocean
694	44	Northern Easter Island Cordillera
695	44	West of Galapagos Islands
696	44	Galapagos Islands Region
697	44	Galapagos Islands
698	44	Southwest of Galapagos Islands
699	44	Southeast of Galapagos Islands
700	45	South of Tasmania
701	45	West of MacQuarie Island
702	45	Balleny Islands Region
703	46	Andaman Islands Region
704	46	Nicobar Islands Region
705	46	Off West Coast of Northern Sumatera
706	46	Northern Sumatera
707	46	Malay Peninsula
708	46	Gulf of Thailand
709	47	Afghanistan
710	47	Pakistan
711	47	Southwestern Kashmir
712	47	India-Pakistan Border Region
713	48	Central Kazakh SSR
714	48	Southeastern Uzbek SSR

715	48	Tajik SSR
716	48	Kirghiz SSR
717	48	Afghanistan-USSR Border Region
718	48	Hindu Kush Region
719	48	Tajik-Xinjiang Border Region
720	48	Northwestern Kashmir
721	49	Finland
722	49	Norway-USSR Border Region
723	49	Finland-USSR Border Region
724	49	European USSR
725	49	Western Siberia
726	49	Central Siberia
727	49	Victoria Land, Antarctica
728	50	Ross Sea
729	50	Antarctica

Appendix L: FDSN Usage

Contributed by Ray Buland

The SEED format was created by seismologists, primarily interested in teleseismically recorded earthquakes, for the purpose of facilitating the exchange of digitally recorded ground motion. The development and subsequent evolution of SEED had been performed under the auspices of the Federation of Digital Seismographic Networks (FDSN). This development was based on extensive practical experience in digital waveform data exchange. However, the subsequent evolution has been heavily influenced by numerous real-world problems in the implementation of FDSN data collection and exchange, some that were anticipated at the outset and some that were encountered later. Despite this orientation, the original SEED designers had the foresight to make SEED very general. In fact, SEED should already encompass (or be easily extended to encompass) any type of equally sampled time series data recorded at discrete points on the surface of a planet. For example, SEED is already being used to exchange temperature, barometric pressure, and wind speed data as part of the state-of -health information recorded at a seismic station. The format could as easily be used to exchange tilt, strain, creep, magnetic field, or other geophysically interesting time series.

While experience indicates that the generality and flexibility designed into SEED have and will continue to serve us well in warding off premature obsolescence, they also have their price. For example, the flexibility of SEED has contributed significantly to its complexity. The generality of SEED has also created (unexpected) practical problems. In particular, SEED is so general that access to similar types of information exchanged among the members of a user group (such as the FDSN) cannot be easily automated without additional usage conventions. The most obvious example is in naming station channels. Without a standard (such as that described in Appendix A) it would be very difficult (although possible in principle) to formulate and fill requests for similar types of data contributed by various members of the FDSN to a unified data management center.

To understand this problem, consider a request for broadband, seismometer data. This request would be generally understood by any FDSN member to refer to data with a sample rate between 10 and 80 samples per second, a sensitive bandwidth of at least two decades in frequency, and a sensitivity sufficient to record earth noise over a significant portion of the bandwidth.

Although all of this information is encoded in the Station Control Headers, it would be very difficult (and time consuming) to analyze the response information, in particular, to glean the needed information each time a request for data was formulated. The channel naming convention summarizes this information in a human readable mnemonic shorthand that makes the request process natural and straightforward. In essence, by sacrificing some of the generality of SEED, a practical data distribution problem has been solved in such a way as to contribute to mutual understanding among users and to the automation and efficiency of the data request retrieval process.

The FDSN has found it convenient to codify a number of these usage conventions in order to facilitate the flow of data among FDSN members and scientists using FDSN data. These conventions will be described in the following. Note that it is very important to distinguish between SEED definitions and FDSN usage conventions. FDSN usage conventions in no way restrict the generality of the SEED format for groups of users who deal with other types of data or who serve different communities. However, it is important to understand the concept of such conventions as it is highly probable that other groups will find it convenient or even necessary to formulate their own usage conventions for reasons similar to the FDSN's.

In general, information found in the main body of this Manual describes SEED standards. Information found in the appendices, however, falls into several categories. For example, Appendices C (Specifying and Using Channel Response Information), D (The Data Description Language), F (Cross Reference for Fields in Abbreviation Dictionaries), H

(Effective Times and Update Records), and K (Flinn-Engdahl Seismic Regions) all seek to explain, organize, or codify the usage of various SEED blockettes or fields and should be considered to be integral parts of the SEED specification. On the other hand, Appendix A

(Channel Naming) and Appendix J (Network Codes) describe FDSN usage conventions while Appendix B (The Steim Compression Algorithm) describes a data compression format used primarily by some FDSN members. Appendix E (Sample Logical Volumes) shows an example drawn from one FDSN member.

FDSN usages conventions fall into two classes: 1) common terminology and 2) channel description standards. The channel naming and network code conventions falls into the first class. Other terminology conventions include standards for unit naming (particularly non-SI units such as COUNTS) and standards for naming Data Description Language (DDL) specifications. The former contributes to automation in interpreting instrument response descriptions by end users for channels from various FDSN member networks. The latter provides a processing short cut as a DDL parser need not be invoked for the handful of currently extant FDSN binary data formats. In other words, the DDL name is used to select a binary data interpretation routine, by-passing the more cumbersome DDL parser driven interpretation.

Channel description standards fall into two chronically troublesome areas: 1) time keeping and 2) response specification. These seem to be subjects that each seismological network operator has had to think a great deal about and, consequently, about which each operator has formed very definite ideas (and operational procedures). Some of the divergence in usage among FDSN members has resulted in more precise definitions of the SEED standard itself. However, other issues were resolved by additional usage conventions. For time keeping, it was decided that the basic SEED definitions were generally adequate with minor extensions. In particular, the clarification that the channel sample rate defined in the Channel Identifier Blockette [52] and in the Fixed Section of the Data Header should be the design sample rate resulted from this discussion. A new blockette was added for cases where the actual (average) sample rate deviated significantly from the design rate. Further, the FDSN usage convention that short term deviations from the actual sample rate must be handled through the time correction fields in the Fixed Section of the Data Header rather than by frequent changes of the design sample rate (through the channel effective date mechanism) was adopted. These clarifications act to make time keeping more uniform among FDSN members and more comprehensible to end users. The convention concerning rate changes makes channel related information more manageable.

Usage conventions for response information were adopted both to make access to the data more consistent and convenient and to establish minimum acceptable standards for completeness. In the former case, the clarification of the SEED standards that the A0 constant must correctly normalize the relative transfer function at the given reference frequency in the Response (Poles & Zeros) Blockette [53] coupled with the FDSN convention that a stage 0 Channel Sensitivity/ Gain Blockette [58] must be given greatly simplifies the usage of response information by the end user. In the latter case, FDSN usage requires that all digital FIR and IIR filter coefficients be given in addition to the poles and zeros of the Laplace transform of the analogue response. This convention is highly specific to the FDSN. It is currently the most complete and

precise method of defining the hybrid analogue/digital transfer function of a seismological instrument. This convention was considered appropriate for the FDSN as FDSN networks are specifically designed to provide data for the most demanding waveform analysis work.

In summary, while generality and flexibility of the SEED standard have many benefits, various SEED user groups will probably find it desirable to superimpose usage conventions that limit the generality of SEED for their specific purposes. The above discussion presents current FDSN usage conventions as a guide to FDSN members and as an example of how such conventions might arise for other SEED user communities. As we have seen, usage conventions can have the following benefits: 1) establishing a minimum acceptable level of station/ channel documentation, 2) providing mechanisms to facilitate the automation of data requests and retrieval, and 3) providing a consistent framework to facilitate the analysis of the data by the end user.

Bibliography

Duncan Carr Agnew, "Conventions for Seismometer Transfer Functions" 1987 Personal Correspondence with C. R. Hutt of the Albuquerque Seismological Laboratory.

Robert R. Blandford, David Racine, and Raleigh Romine, *Single Channel Seismic Event Detection*, 1981 VELA Seismological Center Report VSC-TR-81-8.

James N. Murdock, and Charles R. Hutt, *A New Event Detector Designed for the Seismic Research Observatories*, 1983 USGS Open File Report 83-785.

James N. Murdock, and Scott E. Halbert, *A C Language Implementation of the SRO (Murdock) Detector/Analyzer*, 1987 USGS Open File Report 87-158.

Bruce W. Presgrave, Russell E. Needham, and John H. Minsch, *Seismograph Station Codes and Coordinates — 1985 Edition*, 1985 USGS Open File Report 85-714.

Samuel D. Stearns, *Digital Signal Analysis*, 1975 Hayden Book Company, New Jersey.

Samuel D. Stearns and Ruth A. David, *Signal Processing Algorithms*, 1987 Prentice—Hall, New Jersey.