Design and Specifications: UW-1/2 Seismic Data Formats (10/91)

Introduction

This document defines two University of Washington (UW) data formats: UW-1 and UW-2. UW-1 has been in use since 1980, with essentially no change, and is currently in use. It's formal definition resides in C language structure definitions utilized with a number of applications programs written at the UW as well as other institutions. UW-2 is a new format, designed to replace UW-1 in such a way as to satisfy design goals outlined in this document.

Rationale

In the near future, we want to integrate broadband (event) data into our existing short period network data acquisition system. We now face the need for a more comprehensive data format. Our existing data format, UW-1, while serving us well for a number of years as long as our data were homogeneous (i.e., sampled uniformly with time windows for all stations starting and stopping at the same time), cannot accommodate data of different sample rates and different starting/stopping times. In addition, the current short integer format is inadequate when the linear dynamic range of the data exceeds 16 bits (current state-of-the-art data may be acquired in up to 24 bit format). The channel header format for UW-1 also carries only a limited number of parameters and is unsuitable for broadband data. We have basically two general options: we can abandon our format altogether in favor of one of the currently developed (or developing) formats such as "AH", "SUDS", "SAC", etc. Alternatively, we can "extend" our format to provide the additional needed functionality.

Each of these two broad options has advantages and disadvantages. Abandoning our format will have the major disadvantage of backward incompatibility. What do we do with the mass of data already accumulated? Convert it? Provide convenient conversion routines? Attempt to provide access utilities that will automatically identify two or more formats? On the other hand, changing our format could have the advantages of providing compatibility with at least some other users and a certain amount of potential "standardization". Unfortunately, there is no real "standardization" of seismic network data formats at present. Staying with but extending our present format will contribute to format "proliferation", and perhaps present some barriers to future data exchange. At the same time, the job of providing backward compatibility will be easier since we would have complete control of the format conventions. Our format offers relatively compact data storage for event oriented data since things that are common to all channels are contained in the master header block. We don't attempt to store event location data in the data format, and presently provide only a limited amount of station information (channel name, bias, long-term average, and trigger condition). This is an advantage in that erroneous associated station information (not uncommon) does not get archived with the data.

After considering these factors, and examining the major external formats such as AH, SUDS, SAC, and SEED, we have decided to develop a new data format, UW-2, that will offer backward compatibility with our existing format, UW-1. As a part of the design process, we will develop a set of interface routines that will provide a buffer between the user's applications and the details of the data format. By designing the user interface carefully, we can minimize (but not completely eliminate) the impact that this and future data format changes will have on applications programs that use the data.

Design Goals

Our objectives are to provide a format which will accommodate our immediate and future needs to incorporate heterogeneous data, such as that from dial-up broadband stations. It is assumed that the data are generally event oriented (associated with a particular interval of time) although individual seismograms may have varying start and stop times, varying sample rates, and varying intrinsic format (e.g., short or long integer, floating point). We would like to accommodate either single or dual (UW-1) file formats, and maintain complete backward compatibility so that the user need not be concerned with whether the data are in UW-1 or UW-2 formats (except where the added capabilities of UW-2 come into play). Since our data are inherently indexed (random access) at present, and we feel that this is an important feature for rapid access by applications software, UW-2 retains indexing information for random access. It is also important to provide more channel ID information than currently provided in UW-1. For example, it is currently not

possible to uniquely specify a channel polarization independently of the station name, nor is is possible to unambiguously specify two response bands derived from the same instrument. Finally, a design goal is to provide a relatively simple, evolutionary change to our current data format, retaining the desirable features that have made the format easy to use and durable, but adding needed functionality. Simplicity is a positive attribute of a format in that it speeds up the design and construction of essential applications. We feel that at this stage, unneeded complexity at the working data format level represents excess baggage that is detrimental.

Software Implementation

An important consideration in format design is the ease with which software can be designed to use it. Our intention is to build a layer of software that will stand between the application programs that use the data, and the details of the format. This software layer, which we call the "data file interface" (uwdfif), will implement the backward compatibility with UW-1, and relieve the user of handling the internal details of extracting information from the data files. In addition, the details of writing new data files will be handled by the data file interface.

Our intention is to provide a uniform set of interface calls that will return data to the users program in a relatively format independent way. As a simple example, the length of the i'th trace would be returned as a standard function call with "i" as the argument. The user need not be concerned whether this information came from the UW-1 or UW-2 formats. The data file interface will also implement desired behavior such as portability across platforms (e.g., byte reversal, and possible word alignment problems will be accounted for by the interface); it is essential that both the format and data file interface be implemented and tested simultaneously. We plan to implement and test the data file interface prior to final freezing of the data file format for UW-2. To implement backward compatibility, the data file interface will provide essential function calls used by current applications. Migration of existing software to use extensions in the newly defined interface is a goal that will provide future portability and reduce the impact of future format changes.

Since in general, trace data buffers can be of arbitrary length, often quite large, the data file interface will place the burden of memory allocation for trace data on the users application. Since we wish to isolate the channel header details from the user, the data file interface will handle the much smaller (volumewise) problem of memory allocation for the channel headers.

Single or dual file format

Our present two-file data format has advantages of simplicity (the data file is simply concatenated trace data, all in the same format), and convenience in writing (less advantage in reading). Convenience in writing is achieved since channels can be processed sequentially, and both header and data for a single channel can be written simultaneously to the two different files. A significant disadvantage of the two file format is the proliferation of files and clutter engendered by always having to deal with two files for any give time chunk. Although it may have been imagined at one time that header files could exist in a data base separate from the trace files, practice has shown that this is not desirable. The header and data files must really be "locked" together.

If we are to efficiently write single channel data files, but at the same time process the data sequentially, it is difficult to write the channel header data near the beginning of the file. Since trace data buffers are so much longer than header data buffers, it is inconvenient and inefficient to save all trace data in memory so that it can be written after all headers are written. If we mix header and data blocks alternately, the header data cannot be read as efficiently when the data files are opened for reading. Thus, placing channel headers ahead of the trace data in a single file format, while convenient for reading, makes writing very difficult and/or inefficient. In reading, the two file format is a disadvantage since two separate files must be opened and maintained.

In the design that we have developed for UW-2, efficient writing of the data to a single data file will be accomplished by first writing the master header block, as is presently done, followed by sequential trace data buffers, followed by the header blocks for all channels. As trace data are written, channel headers, which are relatively short, are sequentially saved in memory. When trace writing is completed, the N channel headers (where N is the total number of data channels written), each of which is of fixed length, are

written. The last thing to be written to the data file is indexing information that specifies the number of channels and the location of the channel headers in the file for reading. Using seek to the end of the file, the number of channels, and all of the channel header information can be picked up rapidly and stored back in memory when a data file is read. This ensures rapid, random access to the data on read operations, as well as efficient writing in single file format. To achieve these advantages, we give up the possibility of sequential reading/processing of the data, unless the entire data file is read into memory (not ruled out with present virtual memory machines). Data files must be on random access storage with the ability to seek to the end of the file.

Data written under the new data file interface will always be UW-2 format in single file layout. By using dynamic memory management in the data file interface, it should be possible to design a system in which reading and manipulating data from a number of files simultaneously is efficient and still fairly convenient.

Master Header

The initial master data header block used presently by UW-1 is retained in identical form in UW-2. We achieve backward compatibility with UW-1 by newly defining two of the ten expansion fields to self-identify the format. Attachment A lists the precise structure of both master header block and channel header blocks for the combined UW-1/2 data format as C structures. This Attachment should be consulted as the final format definition.

The first expansion field ("extra[0]") is already defined as a flag for event type. We define the second expansion field character (one byte, extra[1]) to identify the integer data as being IEEE conformant or not. The reading software must be aware of the type of hardware it is being run on. If, for example, the data don't conform to the machine hardware convention, the integer data must then be byte swapped when read in. When written, the data (and extra[1]) are always set to the byte convention of the CURRENT machine under the assumption that the data will be most frequently read by the same machine, with the resulting efficiency of not invoking byte swapping. The third expansion field character identifies the format as UW-1 or UW-2.

Remaining fields in the master header block are identically defined for both UW-1 and UW-2, but software interpretation may differ in each case. For example, the sample rate in the master header block for UW-1 is not used in UW-2 since the sample rate is defined in the channel headers for UW-2. By having relaxed specification of the expansion characters, we can achieve backward compatibility. The expansion characters have in practice been assigned blanks and not used in UW-1 (with the exception of "extra[0]"). Thus, for example, in the case of the second expansion character, either a blank (' ') or a 'I' can indicate IEEE conformant data. An 'I' is less ambiguous and should be used in the future, but a blank is accepted to provide backward compatibility. Other fields in the master header block that have different meanings in UW-1 and UW-2 are: (a) the number of channels is not used in UW-2 since with the single file format of UW-2, this value is written at the end of the data file; (b) the reference time which is the time of the first sample for all channels in UW-1 is not used in UW-2 since the reference time for each trace is given in the channel headers. (c) the channel length, which is used for all channels for UW-1, is not used, since this number is given in the individual channel headers for UW-2.

Channel Headers

The channel header blocks are the most critical part of the specification of UW-2. The table below lists the fields and brief definitions for UW-2. The actual C structure declaration for the channel headers is given in Attachment B.

The ID tags for each field are given in upper case letters. Although these names are not necessarily those to be used in the actual I/O code, they are convenient references and coordination with the code is probably desirable. To the degree possible, fairly long descriptive names in the actual code will make it more readable and self documenting. One design goal is to make the structures readable without special action across differing platforms. In particular, word boundary alignments in structures present problems. Each channel structure should have an even number of bytes, and quantities are grouped to minimize word boundary problems. Reading and writing of the formats will be tested on several platforms where problems might be expected to ensure compatibility.

center, box, tab(^); c s s s s s c | c | c | c lw(0.8i) | c | c | c | lw(2.3i). _ UW-2 CHANNEL HEADER = NAME^BYTES^TYPE^OLD/NEW^DESCRIPTION = CHLEN^4^int^new^T{ Channel length (number of data points) T} _ OFFSET^4^int^new^T{ Byte offset of start of data channel from beginning of file T} _ START_LMIN^4^int^new^T{ Starting time of channel; minutes reference defined by date/time algorithm T} _ START_LSEC^4^int^new^T{ Starting time of channel; in micro-seconds relative to lmin T} _ LRATE^4^int^new^T{ Sample rate in samples per 1000 seconds; can vary for each channel T} _ EXPAN1^4^int^new^T{ Expansion slot for 4 byte integer T} _ LTA^2^int^old^T{ Long term average at time of trigger T} _ TRIG^2^int^old^T{ Trigger parameter from HAWK system T} _ BIAS^2^int^old^T{ Channel bias or running mean T} _ FILL2^2^int^new^T{ Fill short int so that short int block is mult of 4 bytes T} _ NAME^8^char^old^T{ Station name; 4 bytes of character data unique to station and not used for channel ID; 4 bytes unused T} _ FMT^4^char^new^T{ Intrinsic data format; flag for short int, long int, or floating point data (S,L,F) T} _ COMPFLG^4^char^new^T{ Component flag; to indicate channel band, orientation, etc., SEED Appen A definitions T} _ CHID^4^char^new^T{ Channel ID; specified by user to uniquely identify channel if needed. T} _ EXPAN2^4^char^new^T{ Expansion character field T}

Channel Data

The channel data are structured as time series samples, equally spaced with a sample rate defined in the channel headers (or master header as in UW-1), and in a format which is flagged by the "fmt" character field in the channel headers. Initially we define three formats, short integer, long integer, and floating point, flagged by the characters 'S', 'L', and 'F'. This system is expandable to include other possible formats as the need arises. A given flag must specify exactly how the samples are to be interpreted. For example, gain ranged that were encoded in ASCII characters could be flagged with a format flag 'X'. We only require that the reading software must be able to recover the necessary characters to reconstruct successive sample points. We note that the actual length of each channel of data in bytes is contained in the channel pointer information in the channel headers. Thus it would be theoretically possible to reconstruct data that were actually encoded in varying numbers of bytes, such as a compression scheme, as long as the algorithm for doing so is available to the reading software. Our initial implementation of the data file interface will not provide such capability, but it should not be precluded as a future software addition.

File Organization and Conventions

UW-1 is written as two files, with file names characteristically identified by either a 10 or 11 digit numerical sequence that defines the date-time of the data in the file, and ending in the characters 'D' or 'd'. The 'D' file is the header file and its layout is as follows: The master header block of 132 bytes is written first, followed in sequence by N channel header blocks of 12 bytes each, where N is the total number of channels. The total length of the 'D' file is thus 132 + 12*N where N. The 'd' file is written as N concatenated, equal length data files where each sample is represented by a 2 byte integer. The order of the data channels is the same as the order of the channel header blocks in the 'D' file. Thus the total length of the 'd' file is N*(channel length).

UW-2 is constructed as follows. The first 132 bytes will be the master header block as defined in this document (identical in structure to the UW-1 master header). The master header block will be followed by the demultiplexed channel data, written sequentially as channel 1 followed by channel 2, ... Each channel may have variable length and variable data type as identified in the channel header for that channel. Thus the total length of data will be the sum over the number of channels of {the length of channel K times the number of bytes per sample for channel K}. Note that the number of bytes per sample may vary from channel to channel. The channel data are followed by the channel header blocks, which are written in the same sequence as the data channels. Each channel header block is K = sizeof(chhead2) (currently 56) bytes long, so the total length of this section is K*N. Finally, index data are written at the end of the file that specify the number of channels and the location of the channel headers.

We provide capability for future inclusion of other structures in the data file by placing an index at the end of the file that can be expanded to index arbitrary kinds of structures. Presently, only channel header structures are indexed in this manner. The form of this index information is fairly simple. The last 4 byte word in the file is an integer indicating how many such auxiliary structures are indexed. This

number is currently 1 since only the channel headers are indexed. Each actual index is a three word structure (12 bytes) written ahead of the last 4 byte word. The first 4 bytes of the index structure is a flag to indicate the kind of structure (e.g., channel header), the second 4 bytes is a long integer specifying the number of such structures (e.g., the number of channels), and the third 4 bytes is a long integer containing a byte offset from the beginning of the data file to the first structure. The byte offset will be necessary if other structures are added in the future.

The two file convention of UW-1 necessitates the big D and little d naming conventions if the application software is to be relieved of independently managing the header and trace file names. In the past, our data file interface routines have implicitly assumed this naming convention, so that both header and trace files could be managed by the interface. The users application needed only to provide the name of either the header or trace file, or the root file name (without the 'D' or 'd' suffix), and the interface routines could find, open, and close the appropriate data files. With the single file format of UW-2, we are in a position to relax this restriction on file naming conventions. With only a single file to manage, the data interface can be relieved of any naming assumptions, providing increased flexibility for file management and naming conventions. The price for this increased flexibility in addition to retaining backward compatibility with our existing UW-1 format, is to require the application programs to always pass the name of the HEADER file ('D' file) if UW-1 type data are being accessed. This may require that in some applications, the name of the header file must be selected in preference to the data file ('d' file) where a list of files to be processed must be passed to the application.

In practice the way this will work in the data file interface is that the data file interface will blindly accept any file name passed to it to open, assuming implicitly that this will be either a header file in UW-1 format, or a full data file in UW-2 format. If the data file interface determines that this is UW-1 format from the initial header block, it will construct a new trace data file name by replacing the last character of the file name with a 'd', opening the trace file with this new name. If, on the other hand, the data are in UW-2 format, the name of the data file is completely ignored, and can be anything the user supplies. As long as the UW-2 format is used, any file naming convention can therefore be used. We recommend that the UW-2 style data use the suffix "W", to readily distinguish this data format from UW-1.

Finally, in the data file interface, write-over capability will be provided. This will be accomplished by providing a temporary file name while a new file is being written, and after cleaning up, verifying and closing the write operation, the temporary name will be changed to the requested one. Thus an old file can be overwritten by a new file name with the same name (in UW-2) even though the old file remains open during the writing operation. This is useful if old data are to be processed sequentially into a new file, such as in a filtering operation. The responsibility for managing the file names remains with the users application.

Attachment A - Definitions and Conventions

Master Header Block: UW-1 & UW-2

The following definitions and conventions hold for the master header block. This block is identical for UW-1 and UW-2, and is designed so that software will be able to distinguish either format in reading.

NCHAN — Short integer indicating the number of data channels for UW-1. Ignored for UW-2 since number of channels is placed at end of file as last 4-byte integer. This value can optionally be set to 65536 (maximum unsigned short integer) as a flag for UW-2 format.

LRATE — Long integer specifying the sample rate for all channels for UW-1. For UW-2, this field is ignored since the sample rate may vary from channel to channel.

LMIN — Long integer specifying the reference time for all the data in file. This time is given in minutes since midnight, Dec. 31, 1599, and is decoded by the Carl Johnson date routine grgmin. This header value is for optional use by applications, since each channel has independent initial reference time. It will be carried over in conversion from UW-1 to UW-2 intact, and in this case will be the start time of all of the simultaneous multiplexer derived channels.

LSEC — Long integer specifying the reference time in fractional seconds relative to the LMIN time. This time is given in integer microseconds. This value has the same significance as the LMIN field for UW-2.

LENGTH — Long integer giving the number of data samples for all channels for UW-1 format. This field is not used for UW-2.

TAPENUM — Short integer which originally specified the tape number for data generated by the DEC 11/34 system. This field is now used as a "run" number generated by the HAWK system. Run numbers are sequentially generated by the HAWK data acquisition system; the run number is incremented when the system is reinitialized, and is useful to monitor the flow of data from the acquisition system.

EVENTNUM — Short integer which specifies the sequence of an event within a specific run or tape. Successive network triggers increment the EVENTNUM. Both TAPENUM and EVENTNUM are valuable in particular applications to monitor the sequence of events generated by automated on-line processing.

FLG — Array of 10 short integers used for various purposes. The current uses are specified in the structure definitions.

EXTRA — A 10 character field (or character array of length 10) that was originally defined for expansion. Its current definitions are:

extra[0] is user defined event type; extra[1] is set to a blank (default value in UW-1) or the character 'I' (normal) if the integer data are IEEE conformant, set to the character 'D' (Dec) if the data are DEC style byte reversed. extra[2] is set to blank (' ') or character '1' if the data format is UW-1; set to '2' if UW-2 format.

COMMENT — Eighty character user defined field for optional comments.

Channel Header Blocks: UW-1

The following definitions and conventions are used for quantities in the channel headers of the UW-1 format:

NAME — The station name is up to 4 characters, upper case alphabetic or numeric, and uniquely specify a point on the ground (station location) referenced to a station table. The station name will not be used to specify a component orientation, channel, or other attribute of the particular channel in question. The last two characters of the 6 character name field are currently ignored.

LTA — Short integer specifying the long term average of the data as provided by the HAWK data acquisition system.

TRIG Short integer specifying the trigger state provided by the HAWK data acquisition system.

BIAS Short integer specifying the bias (or mean) of the data in integer counts, coming from the digitizing system. This value is removed from the system in demultiplexing, but is useful as a state of health indicator for channel.

Channel Header Blocks: UW-2

The following definitions and conventions will be used for quantities in the channel headers of the UW-2 format:

CHLEN Long integer specifying number of data points in channel.

OFFSET Four byte integer specifying the byte offset of the channel in the data file relative to the beginning of the data file. The data channels will follow one another contiguously, but may be positioned at various places within the data file. In UW-2, the data channels will immediately follow the initial header block.

START_LMIN Four byte integer specifying absolute starting time of channel in minutes as defined by date/time algorithm (due to Carl Johnson). Each channel thus has completely independent starting time of arbitrary value. This value may be returned to the application program through the interface as either a structure or as a time array.

START_LSEC Four byte integer specifying precise starting time of channel. This time offset is given in microseconds.

LRATE Long integer specifying data sample rate as number of samples per 1000 seconds. This convention follows the convention followed in UW-1 for all channels and probably originated with Carl Johnson.

EXPAN1 Long integer expansion field. Not used at present.

LTA Short integer specifying the long term average of the data as provided by the HAWK data acquisition system. Identical to UW-1 definition.

TRIG Short integer specifying the trigger state provided by the HAWK data acquisition system. Identical to UW-1 definition.

BIAS Short integer specifying the bias (or mean) of the data in integer counts, coming from the digitizing system. This is identical to the UW-1 definition, and comes from the data acquisition system. This value is removed from the system in demultiplexing, but is useful as a state of health indicator for channel.

FILL Short integer fill field so that short integer block is 8 bytes total to alleviate word boundary problems.

NAME The station name field is 8 characters long, with the first 4 characters used for the station name, upper case alphabetic or numeric, to uniquely identify a point on the ground (station location) through a station table. The station name will not be used to specify a component orientation, channel, or other attribute of the particular channel in question. The last four characters of the 8 character name field are currently ignored.

FMT The first character of four character field specifies the machine format of the binary data of channel. The possible values are "S" (short int), "L" (long int), or "F" (floating point, implementation dependent). The second character is unused and ignored.

CHID A four character field that can be optionally used to uniquely identify a channel, especially where multiple channels are derived from the same sensor through filtering, transmission paths, etc. There is no current specification of possible field values; this field is user defined.

COMPFLG The first three characters of this four character field indicates the instrument band and orientation (or channel) according to the 3 character code specified in the IRIS Seed specification manual, Appendix A. The 4th character can be used as a channel sequence number (0, 1, 2, ...) for cases where a single channel may be included in non-contiguous chunks more than once. If this character is left blank (or null), it is assumed that only a single

representation of this channel is present. Other situations such as a single channel that is represented more than once due to multiple transmission paths, etc, are handled through the 4 character channel id field.

EXPAN2	14	Four character field for expansion; not defined or used at present.

Attachment B
C structure definitions for UW-1 and UW-2

```
/*  Master header block for both UW-1 and UW-2 is defined below */

struct masthead {
   short int nchan; /* Number of channels; not used for UW-2 */
   int lrate;   /* Sample rate, samples per 1000 seconds; not used in UW-2 */
   int lmin;    /* Reference time for Carl's date routine (grgmin) */
   int lsec;    /* Reference second from above min in microseconds */
   int length;  /* Number of samples per channel in total record
                   Note that in UW-2, this is defunct */
   short int tapenum;  /* Original 11/34 tape number; now run number */
   short int eventnum; /* Event number on 11/34 tape; now sequence number */
   short int flg[10];  /* Extra user defined flags for expansion.
         Some current usages follow:
         flg[0] is -3 when lmin is not set but the digitization
               rate and seconds modulo 10 have been set.
         flg[0] is -2 when lmin and lsec have not been set.
         flg[0] is -1 when lmin is set to within plus or minus 3.
         flg[0] is what you get from the online demultiplexer.
               it usually means  that the time is set
               to within 3 seconds and the digrate is good
               to about 3 places, but this may be in error:
            a) if the online clock was not set at reboot
                  time then the year will not be 198?.
            b) if the digrate is exactly 100 in old 64 chan
               stuff then this is not even close.
               The digrate was constant throughout the
               64-chan configuration.
         flg[0] is  1 when ping is through with it.  This
               generally means the time is set to within
               .2 second and the digrate has not been reset.
         flg[0] is  2  when the digrate has been set to 4
            significant figures and the starting time
            has been set to within one sampling interval.
         flg[1] is the logical OR of: low bit: 0 -> Squash Lock On
                                      low bit: 1 -> Squash Lock Off
                                      2nd bit: 0 -> Not Squashed
                                      2nd bit: 1 -> Squashed
         flg[2] is number of times the file has been merged.
         flg[3] is 0 usually and 1 if the station names have been modified.
         flg[4] is decimation factor if slashed
         flg[5] is the channel number (1 - hm.nchan) of the
            time code from which the time was set.
         flg[6] is used by the 5-day merge package.  It is
            set to 1 after the station names have been corrected.
         flg[7] is used by 'stack' to tell how many files have
            been stacked onto this one.
*/
   char extra[10];   /* extra codes for expansion
         extra[0]  is currently used as an event type flag
         extra[1] is set to ' ' (blank) or 'I' if integer data are IEEE
               conformant; 'D' if data are DEC style byte reversed;
               'I' or 'D' are preferred forms for new format
```

```
          extra[2] is set to ' ' (blank) or '1' if UW-1 format is used;
            '2' for UW-2 format;  '1' and '2' are preferred forms for
              new format
*/
   char comment[80]; /* 80 optional comment characters */
};

/* Channel headers for UW-1 and UW-2 are defined below */

struct chhead1 {  /* UW-1 channel headers; one per channel */
   char name[6]; /* Station name (4 characters and a null) */
   short int lta;    /* long term average */
   short int trig;   /* Trigger  (positive for trigger on) */
   short int bias;   /* DC offset */
}; /* length of chhead1 = 12 bytes */

struct chhead2 { /* UW-2 channel headers; one per channel */
   int chlen;    /* channel length in samples */
   int offset;   /* start offset of channel; bytes rel. to start of file */
   int start_lmin;    /* start time in min; same def. as lmin in masthead */
   int start_lsec;    /* start time offset relative to lmin; u-sec */
   int lrate;    /* sample rate in samples per 1000 secs */
   int expan1;        /* expansion field for long integers */
   short int lta;    /* long term average; same as chhead1 */
   short int trig;   /* Trigger (positive for trigger on); same as chhead1 */
   short int bias;   /* DC offset; same as chhead1 */
   short int fill;   /* Fill short int so short int block 8 bytes long */
   char name[8]; /* station name (4 characters and null) */
   char fmt[4];  /* first char designates data fmt (S, L, or F) */
   char compflg[4]; /* component id as per Seed Appen I + seq no. */
   char chid[4]; /* unique channel id; user defined */
   char expan2[4]; /* expansion characters */
}; /* length of chhead2 = 56 bytes */
```