NAME

uwdfif - package for high level access to UW-1/2 data files

GENERAL

#include "uwdfif.h"
Functions starting with UWDFdh... pass data to/from UW-1 style master header block.
Functions starting with UWDFch... pass data to/from UW-2 style channel header blocks.

CHANNEL NAMING CONVENTIONS

Channel names for UW-1 format can contain up to 5 arbitrary characters. There are no component flags or channel ID's in UW-1 data. UW-2 data files can have channel names of up to 7 arbitrary characters. Channel component flags for UW-2 are by convention 3 characters following the naming conventions described by the SEED manual, Appendix A. Typical examples are: SHZ - short period, high gain, vertical component SHN - short period, high gain, north component BHE - broadband, high gain, east component Channel ID's are basically user assignable, with the exception of the following convention. Analog time channels have a descriptive channel name such as WWVB or TCG (time code generator). Since channel names typically have no significance, the component flag TIM is used to designate analog time channels. Since more than one time channel may be present, the channel ID strings 0, 1, etc. are used to prioritize time channels so that applications software such as xped may preferably display one or more time channels for reference.

WRITING NEW FILES

New data files are always written in UW-2 format only. The writing process is sequential only, so a general order for function calls must be followed. First the new data file must be initialized for writing with the function UWDFinit_for_new_write(). This function initializes prototype structures for both master header, and for channel headers. The channel headers are changed for each channel by altering the prototype before each channel is written out. Either the master header prototype or the channel header prototype may be set by (a) duplicating from a currently read data file/channel or (b) setting critical fields by calls to functions such as UWDFset... When a channel is actually written, the header for that channel is frozen using whatever data is in the channel header prototype.

Before writing any channel data, a master header block (structure) must be initialized and then written for the new data file through the use of several functions. If a UW1/2 style data file is currently read, then you can duplicate the master header from that file using UWDFdup_masthead(). If no UW1/2 file has been read, then fields in the master header must be set using calls to UWDFset_dh...(). In any event, for master header values that must be specified, calls to the appropriate UWDFset_dh...() must be used. The master header is the first block of data actually written and this is done through the function: UWDFwrite_new_head().

Following the writing of the master header block, sequential writing of each channel of data takes place. First, a channel header template is initialized by: (a) duplicating from an existing channel that has been read from a UW1/2 file using UWDFdup_chead_into_proto() followed by appropriate calls to UWDFset_ch...() to change specific fields in the channel header or (b) all fields in the channel header can be set using calls to UWDFset_ch...() function calls. After the channel header template has been filled with correct data for that channel, then the actual channel header and data are then written out with a call to UWDFwrite_new_chan(). After all channels have been written, the new data file must be buttoned up with a call to UWDFclose_new_file() to complete the writing. After that, another new data file may be written using the same sequence. See the example program near the end of this document.

PROTOTYPE FILE LISTING

Below is a list of the prototype file that is included with the users program through the header: uwdfif.h. The prototype gives a convenient list of the functions available in the package with their argument syntax, and provides a summary list of the functions for the programmer. Detailed function descriptions are provided in the next section.

```
/*
 * This file was automatically generated by version 1.7 of cextract.
 * Manual editing not recommended.
```

```
 *
 *   Created: Mon Nov 14 15:49:44 1994
 */
#ifndef __CEXTRACT__
#if __STDC__

int UWDFchbias ( int );
char *UWDFchcompflg ( int );
char UWDFchfmt ( int );
char *UWDFchid ( int );
int UWDFchlen ( int );
int UWDFchlta ( int );
char *UWDFchname ( int );
int UWDFchno ( char *, char *, char *, int );
int UWDFchref_stime ( int, struct Time * );
int UWDFchret_trace ( int, char, void * );
float UWDFchsrate ( int );
char *UWDFchsrc ( int );
double UWDFchtime_corr ( int );
int UWDFchtrig ( int );
int UWDFclose_file ( void );
int UWDFclose_new_file ( void );
int UWDFconv_to_doy ( int, int, int, int * );
int UWDFconv_to_mon_day ( int, int, int *, int * );
char *UWDFdhcomment ( void );
int UWDFdhevno ( void );
char *UWDFdhext ( void );
short int *UWDFdhflgs ( void );
int UWDFdhfmt_type_for_read ( void );
int UWDFdhnchan ( void );
int UWDFdhref_stime ( struct Time * );
int UWDFdhtapeno ( void );
int UWDFdup_chead_into_proto ( int );
int UWDFdup_masthead ( void );
char *UWDFfile_opened ( void );
int UWDFinit_for_new_write ( char * );
int UWDFinit_rev_table ( char * );
int UWDFis_uw2 ( void );
int UWDFmax_trace_len ( void );
int UWDFopen_file ( char * );
struct chhead2 *UWDFret_chead_proto_struct ( void );
struct chhead2 *UWDFret_chead_struct ( int );
double UWDFret_chtime_corr_proto ( void );
struct masthead *UWDFret_mast_struct ( void );
struct masthead *UWDFret_new_mhead_struct ( void );
int UWDFset_chbias ( int );
int UWDFset_chcompflg ( char * );
int UWDFset_chid ( char * );
int UWDFset_chlta ( int );
int UWDFset_chname ( char * );
int UWDFset_chref_stime ( struct Time );
int UWDFset_chsrate ( float );
int UWDFset_chsrc ( char * );
int UWDFset_chtime_corr ( double );
```

```
int UWDFset_chtrig ( int );
int UWDFset_dhcomment ( char * );
int UWDFset_dhevno ( short int );
int UWDFset_dhextra ( char [] );
int UWDFset_dhflgs ( short int [] );
int UWDFset_dhref_stime ( struct Time );
int UWDFset_dhtapeno ( short int );
int UWDFsta_mapping_on ( char * );
float UWDFtime_diff ( struct Time, struct Time );
int UWDFwrite_new_chan ( char, int, void *, char );
int UWDFwrite_new_head ( void );

#endif /* __STDC__ */
#endif /* __CEXTRACT__ */
```

FUNCTION DESCRIPTIONS

The syntax and use of each function is described below. In general, it is a good idea (even necessary) to include the header "uwdfif.h" noted above in calling programs to ensure that prototypes are declared correctly and arguments are handled correctly. This include file also contains all of the internal structure declarations used by uwdfif. For example, the "Time" structure can be used to return and set the date-time for the header or channel. Functions returning integer values are generally true (1) or false (0) depending on whether the execution was successful or failed. The failure return may not be operative at present. The module name is given on the first line, followed by the description, with the syntax given last. Functions are arranged alphabetically. The integer variable "chno" passed to the routines always refers to the channel ordinal number (0 to N-1, where N is the number of channels).

Function: **UWDFchbias**

```
/*
Return the bias value for the specified channel as an int.  The bias
is a running average of the DC level of the channel as it is digitized,
and is a good measure of the channel state of health.  Normally bias
values should be less than 20 or 30 counts.
*/
int UWDFchbias ( int chno );
```

Function: **UWDFchcompflg**

```
/*
Return a character pointer to the 4 length string that specifies
the component for channel "chno".  The possible strings are given
in the SEED definition manual, Appendix A.  Typically the string
is a three letter code specifying the Band Code, Source Code, and
the Orientation Code.  An example is "EHV" for extremely short-period,
high-gain, vertical component.  A null terminator exists within
the field width.
*/
char *UWDFchcompflg ( int chno );
```

Function: **UWDFchfmt**

```
/*
Return the format type of the seismogram pointed to by index "chano".
This will be a single character with a value 'S', 'L', or 'F' for
short integer, long integer, or float type respectively.  This
function allows the application program to ascertain the native
```

data type for any data channel, in case, for example, the application wants to retain the original data type. This could be the case if the application was merging or splitting data files. Note that the conversion of float or long int data types to short int format is highly unadvisable, although the interface presently permits this type of conversion. Such conversion could result in loss of significance or data distortion.
/
char UWDFchfmt (int chno **);**

Function: **UWDFchid**

*/**
Return a character pointer to the 4 length string available to specify channel id in more detail. Exact use of this field is not defined by the format, but it is available as a user defined field to specify things such as high or low gain version of a particular component, alternate telemetry paths, etc. A null terminator exists within the field width.
/
char *UWDFchid (int chno **);**

Function: **UWDFchlen**

*/**
Return the length (number of samples or points) of the seismogram for the channel pointed to by index "chno". The length may vary with each seismogram in UW-2 style format, but will be the same for all seismograms for UW-1. The calling software should be written to always check the size of each trace prior to returning the trace, and to ensure that buffer space is allocated, of the correct type, in each case.
/
int UWDFchlen (int chno **);**

Function: **UWDFchlta**

*/**
Return the LTA (long term average) for channel specified by "chno". The LTA is a measure of the average RMS value of the signal over a running window, and is used in the acquisition system triggering algorithm.
/
int UWDFchlta (int chno **);**

Function: **UWDFchname**

*/**
Return a character string pointer to the name field for specified channel. "chno" is the channel number. This name is the station name for the site (point on the ground). There may be several channels with the same station name, for multicomponent stations (in UW-2 format). Component flags and id flags should be used for discriminating components, etc., at stations.
/
char *UWDFchname (int chno **);**

Function: **UWDFchno**

```
/*
Return the integer channel index (ordinal number) for the channel
characterized by the three strings: "chname", "compflg", and "chid".
The boolean switch "force_on" forces full exact comparison with
all three strings.  If no such channel exists, or a match cannot
be found, then return -1 (an illegal channel index number).  This
is the way that we can pull out a channel number by its string
specifiers.  Note: If force_on is set FALSE, then if either "compflg"
or "chid" are null strings, the search just ignores that string
comparison for the null case.  If however force_on is TRUE, then
null strings must match null strings for all three strings.  In
every case, "chname" MUST be non-null.  In the present version,
the first encounter of the correctly matching channel will be
returned, even if later matches can be made.
*/
int UWDFchno ( char *chname, char *compflg, char *chid, int force_on );
```

Function: **UWDFchref_stime**

```
/*
Returns the reference time for the first point of channel specified
by "chno".  "time" in this case is stored in the form of the
structure "Time" defined in "uwdfif.h".  "Time" contains: yr, mon,
day, hr, min, sec.  Remember to pass the POINTER to the time
structure, since data must be returned to the calling program.
*/
int UWDFchref_stime ( int chno, struct Time *time );
```

Function: **UWDFchret_trace**

```
/*
This is the basic routine for returning channel data in several
formats.  "chno" is an input integer that selects the data channel
to return.  This number should be between 0 and (N-1) where N is
the total number of channels (available with another function call).
"fmt" is a character variable that indicates the format that you
want the data returned in.  "fmt" can be be 'S' for short int, 'F'
for float, and 'L' for long integer.  There can be no default, so
this character must be set with one of those values.  "seis" is a
non-specified (void) pointer to a seismogram buffer that must be
big enough to hold one trace at full (4 byte) precision.  Even if
you want to return data as short (2 byte) integers, you MUST declare
the buffer length to be long enough for the same number of sample
points as long (4 byte) integers.  It is the users responsibility
to pass a buffer of sufficient size (to hold one entire trace).
It is also the users responsibility to ensure that the routine does
not convert data from 'F' (float) or 'L' (long int) formats to 'S'
(short int), unless it is known to be safe to do so.  This conversion
(either to or from a data file) CAN RESULT IN SIGNIFICANT LOSS OF
DATA.  No partial trace option is available at present.  The
seismogram trace length in number of samples may be checked for
each trace with the function UWDFchlen(i) for the i'th channel,
and the native format of the data in the data file may be checked
channel-by-channel with the function UWDFchfmt().
```

```
*/
int UWDFchret_trace ( int chno, char fmt, void *seis );
```

Function: **UWDFchsrate**

```
/*
Returns floating point value of sample rate in samples-per-second
for the specified channel number "chno". The sample rate is assumed to
be constant over the duration of the trace.
*/
float UWDFchsrate ( int chno );
```

Function: **UWDFchsrc**

```
/*
Return a character pointer to the 4 length string to specify
the "src" field of the channel header. Exact use of this field
is not defined by the format, but it is intended to specify
the source of the data in a three-character code. For example,
data from the Hawk system would use the designator: HWK for
this field. A null terminator exists within the field width.
*/
char *UWDFchsrc ( int chno );
```

Function: **UWDFchtime_corr**

```
/*
** Returns double precision value of the time correction in seconds
** for the specified channel number "chno".
*/
double UWDFchtime_corr ( int chno );
```

Function: **UWDFchtrig**

```
/*
Return the trigger value for the specified data channel as an int. The
trigger value is a measure of the departure of the short term average
from the long term average, and is used in the acquisition system triggering
algorithm.
*/
int UWDFchtrig ( int chno );
```

Function: **UWDFclose_file**

```
/*
Closes data file that was previously opened by UWDFopen_file.
Called before new data file is opened for reading. OK to call this
if no file is open - in this case, a FALSE value is returned.
*/
int UWDFclose_file ( void );
```

Function: **UWDFclose_new_file**

```
/*
Closes new data file after writing is complete. This function takes
care of cleanup operations, and must be performed before a new file
is opened for writing.
```

```
*/
int UWDFclose_new_file ( void );
```

Function: **UWDFconv_to_doy**

```
/*
Time conversion utility to convert from month & day time to
day-of-year.  Input variables are year (yr), month (mon), and
day-of-month (dom);  output is day-of-year (doy).
*/
int UWDFconv_to_doy ( int yr, int mon, int dom, int *doy );
```

Function: **UWDFconv_to_mon_day**

```
/*
Time conversion utiltity to convert from day-of-year to month &
day.  Input variables are year (yr), day-of-year (doy); output is
month (mon), and day-of-month (dom).
*/
int UWDFconv_to_mon_day ( int yr, int doy, int *mon, int *dom );
```

Function: **UWDFdhcomment**

```
/*
Return pointer to header "comment" string in master header; maximum
length is 80 characters
*/
char *UWDFdhcomment ( void );
```

Function: **UWDFdhevno**

```
/*
Return event number as integer; this is a short int in the header struct.
*/
int UWDFdhevno ( void );
```

Function: **UWDFdhext**

```
/*
Return character pointer to "extra" character string in master
header; maximum length is 10 characters.
*/
char *UWDFdhext ( void );
```

Function: **UWDFdhflgs**

```
/*
Return pointer to array of short int "flags"; array is length 10.
*/
short int *UWDFdhflgs ( void );
```

Function: **UWDFdhfmt_type_for_read**

```
/*
Return integer giving format type for current data file open for read.
1 is returned if UW-1 style format, 2 is returned if UW-2 style format.
*/
```

int UWDFdhfmt_type_for_read (void)**;**

Function: **UWDFdhnchan**

```
/*
Return integer value with number of channels.  This value is picked
up from the master header block for UW-1 style format, and from
the structure descriptor block for UW-2 style format.
*/
```

int UWDFdhnchan (void **);**

Function: **UWDFdhref_stime**

```
/*
Return the master reference time in the structure defined by "struct
Time" (which is declared in the "uwdfif.h" header).  The structure
contains the information: yr, mon, day, hr, min, sec.  This time
is defined as the start of all traces for UW-1 style data (all
channels have the same starting time); for UW-2, it will be the
earliest start time of all traces as determined at the time the
file is read (min of the start time of all traces).
*/
```

int UWDFdhref_stime (struct Time *time **);**

Function: **UWDFdhtapeno**

```
/*
Return tape (or run) number as integer; this is a short int in the
header struct.
*/
```

int UWDFdhtapeno (void **);**

Function: **UWDFdup_chead_into_proto**

```
/*
This function sets the channel header prototype for writing the
channel to be written next by duplicating it from the channel "chno"
of the file presently open for reading.  If you are writing a data
file from scratch, then you would not normally use this function
since there would be no currently valid data to duplicate from.
To use this function, you must have a currently valid data file
(either UW-1 or UW-2) active for reading.  Following this call,
you may reset any individual fields prior to writing.
*/
```

int UWDFdup_chead_into_proto (int chno **);**

Function: **UWDFdup_masthead**

```
/*
This function duplicates the current master header structure into
the master header structure available for writing new data file.
It may be used to simplify the process of correctly initializing
the master header.  For example, we may first duplicate the master
header from a file that has been read (this function), then change
any fields that require change by individual calls to the appropriate
functions.
```

that
**/*
int UWDFdup_masthead (void **);**

Function: **UWDFfile_opened**

*/**
Returns the complete path name of the last data file that was
actually opened. If no file was opened, then the returned string is null.
**/*
char *UWDFfile_opened (void **);**

Function: **UWDFinit_for_new_write**

/ The newfilename is squirreled away until file is closed */*
int UWDFinit_for_new_write (char *newfilename **);**

Function: **UWDFinit_rev_table**

*/**
Itialize the station reversal table using the file name provided
as an argument. If this function is NOT called, no reversal
information is used. If the reversal table is invoked, then affected
channels are automatically reversed as they are returned via the
call to UWDFchret_trace(). May be called at any time to initialize
to a new reversal table (file). Provision is made for a default
station reversal file. If the function is called with a null string ("")
as an argument, then the default mode is invoked. If the function
is called with in the default mode, then either (a) the reversal
table path is taken from the environmental variable USER_STA_REV_TABLE
if it is set, or (b) the default defined by DEFAULT_STA_REV_TABLE
is used for the path.
**/*
int UWDFinit_rev_table (char *name **);**

Function: **UWDFis_uw2**

*/**
Return the value TRUE (1) if the data format for reading (current
file) is UW-2; otherwise return FALSE (0) - currently this implies
UW-1.
**/*
int UWDFis_uw2 (void **);**

Function: **UWDFmax_trace_len**

*/**
Return the maximum trace length (in samples) for current data file.
This utility is useful for allocating space for the seismogram
buffer(s).
**/*
int UWDFmax_trace_len (void **);**

Function: **UWDFopen_file**

*/**
Opens UW-1 or UW-2 style data files for reading. Only one data

file at a time may be opened with the present version. "name" is character string that is either the name of the big "D" file in the case of UW-1 style data, or and arbitrary data file name in the case of UW-2 style data. In the case of UW-1 style data, the little "d" file name is derived from the string given. The environment variable USER_DATA_PATH can be used to specify a colon separated list of directories to search for data files. If the file "name" cannot be opened, this routine will attempt to find the file in the directories indicated by the search path. Returns TRUE (1) if successful, otherwise FALSE (0).
/
int UWDFopen_file (char *name **);**

Function: **UWDFret_chead_proto_struct**

*/**
Returns pointer to the prototype channel header structure used for writing.
/
struct chhead2 *UWDFret_chead_proto_struct (void **);**

Function: **UWDFret_chead_struct**

*/**
Returns pointer to the channel header (UW-2 style) for the i'th channel.
/
struct chhead2 *UWDFret_chead_struct (int i **);**

Function: **UWDFret_chtime_corr_proto**

*/**
*** Returns double precision value of the time correction in seconds*
*** of the prototype channel used for writing.*
/
double UWDFret_chtime_corr_proto (void **);**

Function: **UWDFret_mast_struct**

*/**
Returns pointer to the structure for the master header used for reading.
/
struct masthead *UWDFret_mast_struct (void **);**

Function: **UWDFret_new_mhead_struct**

*/**
Returns pointer to the master header used for writing.
/
struct masthead *UWDFret_new_mhead_struct (void **);**

Function: **UWDFset_chbias**

*/**
Set the channel "bias" variable. This is a moving average parameter normally generated by the UW HAWK data acquisition system. This

would not normally be reset by the user. Since it is only a 2 byte integer, it cannot be set to greater than about +/- 32000. If the application tries to set it larger, the interface sets it to zero.
/
int UWDFset_chbias (int bias **);**

Function: **UWDFset_chcompflg**

*/**
Set the component flag for the channel to be written next. This string is normally three characters in length, following the convention of the SEED defining document, Appendix A. The field can be at most 3 characters since the format allows only 4 characters including the null terminator.
/
int UWDFset_chcompflg (char *compflg **);**

Function: **UWDFset_chid**

*/**
Set the separate channel ID string. This field can be of the users definition. It could be used for example to specify different flavors of the same component from the same station (e.g., different telemetry routes). The field can be at most 3 characters long.
/
int UWDFset_chid (char *chanid **);**

Function: **UWDFset_chlta**

*/**
Set the channel "lta" variable. This is the "long term average" normally generated by the UW HAWK data acquisition system. This would not normally be reset by the user. Since it is only a 2 byte integer, it cannot be set to greater than about +/- 32000. If the application tries to set it larger, the interface sets it to zero.
/
int UWDFset_chlta (int lta **);**

Function: **UWDFset_chname**

*/**
Set the station name for the channel to be written next. This string is normally 4 or fewer characters long, although the UW-2 format will accommodate up to 7 characters plus a null terminator. It is the string specifying the "point on the ground" for the station. The string should be correctly null terminated.
/
int UWDFset_chname (char *chaname **);**

Function: **UWDFset_chref_stime**

*/**
Set the channel reference time for the channel to be written next. The reference time is the time of the first sample in the channel. "time" is a "struct Time" containing: yr, mon, day, hr, min, sec. See the Time structure declaration in "uwdfif.h". Note that the

"time" structure is passed by value.
**/*
int UWDFset_chref_stime (struct Time time **);**

Function: **UWDFset_chsrate**

*/**
Set the channel sample rate for the channel to be written next.
"new_samprate" is a float variable, with units of samples/sec.
**/*
int UWDFset_chsrate (float new_samprate **);**

Function: **UWDFset_chsrc**

*/**
Set the separate channel source string. This field can be of the users definition. It is intended to specify the source of the data; e.g., it would be set to HWK for data form the Hawk system.
**/*
int UWDFset_chsrc (char *chansrc **);**

Function: **UWDFset_chtime_corr**

*/**
*** Set the channel time correction for the channel to be written next.*
*** "new_chtime_corr" is a double variable, with units of seconds.*
**/*
int UWDFset_chtime_corr (double new_chtime_corr **);**

Function: **UWDFset_chtrig**

*/**
Set the channel "trig" variable. This is a trigger parameter normally generated by the UW HAWK data acquisition system. This would not normally be reset by the user. Since it is only a 2 byte integer, it cannot be set to greater than about +/- 32000. If the application tries to set it larger, the interface sets it to zero.
**/*
int UWDFset_chtrig (int trig **);**

Function: **UWDFset_dhcomment**

*/**
Set the comment character field in the master header. This field is a string of length 80 (79 + terminator) or less.
**/*
int UWDFset_dhcomment (char *comment **);**

Function: **UWDFset_dhevno**

*/**
Set the event number in the master header
**/*
int UWDFset_dhevno (short int new_evno **);**

Function: **UWDFset_dhextra**

```
/*
Set the character array "extra" in the master header.  "extra" is
a character array of length 10.  See design document for definitions
of defined elements.
*/
int UWDFset_dhextra ( char extra[] );
```

Function: **UWDFset_dhflgs**

```
/*
Set the integer flags array in the master header.  There are 10
of elements in the flags array.  See design document for definitions
of defined elements.
*/
int UWDFset_dhflgs ( short int new_flags[] );
```

Function: **UWDFset_dhref_stime**

```
/*
Set the master header reference time from the time structure "time".
The structure "Time" is defined in the uwdfif.h include file.  Note
that the argument is not a pointer; the structure is passed by
value.  Note also that this value is used in UW-1 data files as the
master time reference, but it essentially ignored in UW-2 where the
reference time for each channel is carried in the channel headers.
*/
int UWDFset_dhref_stime ( struct Time time );
```

Function: **UWDFset_dhtapeno**

```
/*
Set the tape number field in the master header.
*/
int UWDFset_dhtapeno ( short int new_tapenum );
```

Function: **UWDFsta_mapping_on**

```
/*
Turn on station file name mapping for reading (and writing) UW-1
style data or converting UW-1 data to UW-2 data.  This allows the
user to initialize automatic station name mapping using the specified
station mapping file.  If this function is not called, station name
mapping is not done for UW-1 type data, and original station names
are retained by default.  If station name mapping is not on when
writing new UW-2 data, then original UW-1 station names are retained
in the UW-2 style channel headers.  A default station name mapping
file is defined.  If UWDFsta_mapping_on() is called with a non-null
file name, then that name overrides any defaults and is always
used.  If UWDFsta_mapping_on() is called with a null string as the
file name, then either (a) the path defined by the environment
variable USER_STA_MAP is used for the station name mapping file if
it is defined, or (b) the default path defined by DEFAULT_STA_MAP
is used.
*/
int UWDFsta_mapping_on ( char *map_filename );
```

Function: **UWDFtime_diff**

/
Return the time difference in seconds between the time structure defined by time1 and time2 (time1 - time2). These times must be defined by the structure "Time", which is defined in the uwdfif.h header file. Always use #include "uwdfif.h" to obtain the proper structure declarations. This is just a handy utility routine for applications to avoid having to unfold the date-time structure. Note that the structure arguments are passed by value.
/
float UWDFtime_diff (struct Time time1, struct Time time2 **);**

Function: **UWDFwrite_new_chan**

/
Write next seismogram channel in sequence. This is the basic seismogram writing function. "wr_fmt" is a character variable specifying how you want the file written. The choices are 'S' (short int), 'L' (long int), and 'F' (float). "len" is the length of the data buffer in sample points, "seis" is the actual seismogram data buffer passed to function, and "seis_fmt" is actual format of the data that you passed to the routine (again, 'S', 'L', or 'F'). The routine can accept any of the three formats, and will do internal conversion as needed prior to actually writing data. Appropriate fields in the channel header such as the length and format type are filled prior to writing the data.
/
int UWDFwrite_new_chan (char wr_fmt, int len, void *seis, char seis_fmt **);**

Function: **UWDFwrite_new_head**

/
Write new master header. This function must be the first called after a new file has been opened (UWDFinit_for_new_write()), and the master header has been initialized (e.g., with UWDFdup_masthead()). Note that the new master header structure may be filled several ways.
/
int UWDFwrite_new_head (void **);**

EXAMPLE PROGRAM

The following example code generates a new data file in the UW-2 data format, from existing UW-1 style data. Although brief, it is basically all that is required to do the full conversion. Of course, most of the work is done by the uwdfif package.

```
/*  Usage: uw1-2 file ...
   where "file" is the name of first file to convert */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "uwdfif.h"
#define MAXLEN 100000 /* longest seismogram about 1000 secs (100 sps) */
int seis[MAXLEN];
main(argc, argv)
int argc;
char *argv[];
```

```
{
   char infilename[50], outfilename[50];
   int i, j, chlen, nchan;

   UWDFsta_mapping_on(""); /* Initialize sta name mapping */
   UWDFinit_rev_table(""); /* Initialize station
      reversal table; note that this is really not used for conversion,
      it is given here as an examples of its usage */
   for (i = 1; i < argc; ++i) { /* Loop over file names as arguments */
         strcpy(infilename,argv[i]);
         if (infilename[strlen(infilename)-1] != 'D')
            infilename[strlen(infilename)-1] = 'D';
         strcpy(outfilename, infilename);
         outfilename[strlen(outfilename)-1] = 'W';
         if (!UWDFopen_file(infilename)) { /* Open data file(s)
          without using station or reversal tables */
            printf(" error opening data file0);
            exit(-1);
         }
         nchan = UWDFdhnchan(); /* Get total number of channels */
         UWDFinit_for_new_write(outfilename); /* Init for new file write */
         UWDFdup_masthead(); /*Duplicate original master header into proto */
         UWDFwrite_new_head(); /* Actually write out master header */
         for (j = 0; j < nchan; ++j) {
            chlen = UWDFchlen(j); /* for UW-1, these are all same */
            if (chlen > MAXLEN) {
                  printf(" channel length too long0);
                  exit(-1);
            }
            UWDFdup_chead_into_proto(j); /* Duplicate chan header into prototype */
            UWDFchret_trace(j, 'L', seis); /* Get seismogram */
            UWDFwrite_new_chan('S', chlen, seis, 'L'); /* Write actual data
                  and fill channel header array */
         }
         UWDFclose_new_file(); /* Flushes out channel headers and cleans up */
      UWDFclose_file(); /* Close the current input file */
   }
}
```

HEADER STRUCTURES

```
/*  Master header block from UW-1 style files  */

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 (chhead1) and UW-2 (chhead2)  */

struct chhead1 {  /* UW-1 station 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 src[4]; /* "source" of data field */
}; /* length of chhead2 = 56 bytes */

/* Specification of structure for indexing expansion structures */

struct expanindex {
   char structag[4]; /* tag to indicate specific structure;  currently:
            "CH2" for UW-2 channel header structures OR
            "TC2" for UW-2 time correction structures */
   int numstructs; /* number of structures to read; for:
            "CH2" - number of channels
            "TC2" - number of channels with time corrections */
   int offset; /* file offset for beginning of structure; for:
            "CH2" -  beginning of channel header structures block
            "TC2" -  beginning of time correction structures block */
}; /* length of expanstruct = 12 bytes */

/* Specification of structure for channel time correction */

struct {
   int chno;         /* Channel number */
   int corr;         /* Additive time correction in u-sec */
} tc_struct;
```

```
/* Time structure for UWDFdhref_stime(), UWDFchref_stime(),
   UWDFsetch_ref_stime(), UWDFsetdh_ref_stime(), and UWDFtime_diff() */

struct Time {
    int yr, mon, day, hr, min;
    float sec;
}; /* length of Time = 24 bytes */
```

AUTHOR

High level interface version: Bob Crosson (bob@geophys.washington.edu) with help from Steve Malone

BUGS

Report bugs author.