NAME

ESPS Sampled-Data Feature File Subtype − (.fsd)

SYNOPSIS

#include <esps/esps.h>
#include <esps/fea.h>
#include <esps/feasd.h>

DESCRIPTION

The FEA_SD file is a subtype of the FEA file. If *hd* is a pointer to the header of such a file, then *hd−>hd.fea−> fea_type*== FEA_SD. FEA_SD files are used for storing sampled data, such as the output of an A/D converter. The data may be of any ESPS numerical data type, and multichannel data is supported. (Some ESPS programs, however, may require single-channel data.)

Typically the part of the file following the header is a simple stream of data values, and a record consists merely of a single sample value or (for multichannel data) a fixed number of samples. There are functions, such as *get_feasd_recs*(3-ESPS) and *put_feasd_recs*(3-ESPS), that can take advantage of this simple structure by reading or writing an entire multi-record array of data in one operation, rather than one record at a time. It is possible to create FEA_SD files with a more complicated record structure since *add_fea_field*(3-ESPS) can be used to add extra field definitions to any FEA file header. The function *get_feasd_recs*(3-ESPS) will still work on such files, ignoring the extra fields, but will not work as fast as if the extra fields were not present.

Samples as stored in memory may have a different data type from their type as stored in the file. The input and output routines, *get_feasd_recs*(3-ESPS) and *put_feasd_recs*(3-ESPS), perform the necessary conversions automatically.

FEA_SD files must be untagged. (See *ESPS*(5-ESPS) and *FEA*(5-ESPS) for an explanation of tags and tagged FEA files.)

A FEA_SD file header is created by calling *init_feasd_hd*(3-ESPSu) after *new_header*(3-ESPSu)

The header of a FEA_SD file may contain the following generic header items. The first 2 are always present. The remaining one is not present in every file, but if present has the meaning given below.

Name	Size	Type	Enums
start_time	1 or no. of channels	double	NULL
record_freq	1	double	NULL
max_value	1 or no. of channels	double	NULL

The items have the following meanings.

start_time
: Time corresponding to the first record in the file——given in seconds. If *start_time* has more than one element, each applies to a separate channel. Separate starting times might be appropriate if, for example, samples from the various channels were recorded in rotation, rather than simultaneously.

record_freq
: The sampling frequency in Hz.

max_value
: This item is optional. If it is present, no sample may exceed this value in magnitude. It is permissible for all samples to be smaller——that is, *max_value* is an upper bound rather than an exact maximum. For example, if the data are from an A/D converter, *max_value* may be used to indicate the maximum value that can be represented by the converter. In multichannel files, *max_value* may have one component for each channel or may be a single value applying to all channels.

The header of a FEA_SD file also contains a definition for one record field given by the following table:

Name	Size	Rank	Dimen	Type	Enums
samples	no. of channels	1	NULL or {size}	any numeric	NULL

The field can be created by calling *init_feasd_hd*(3-ESPSu). When the size is 1, there is a single channel; if no additional fields are present, each record is a single sample of the given type, and the part of the file after the header is just a sequence of such samples. When the size is greater than 1, the file contains multi-channel data. In general, each record in the file contains a vector with as many components as there are channels.

Most programs that deal with FEA_SD files will use the support routines in the ESPS library and will not directly use the information in the tables above. The library routine *init_feasd_hd*(3-ESPSu) creates the record field and the required generic header items.

Programs that deal with FEA_SD files do so in terms of structures of type (struct feasd) ——pointers to structures of this type are returned by *allo_feasd_recs*, and the FEA_SD read and write routines, *get_feasd_recs* and *put_feasd_recs*, have parameters of type (struct feasd). Here is the definition of the *feasd* structure as given in *<esps/feasd.h>:*

```
struct feasd {
        short           data_type;
        long            num_records, num_channels;
        char            *data, *ptrs;
};
```

The structure members have the following meanings.

data_type
: A code indicating the data type of the samples as stored in memory. Legal values are give by the type-code constants BYTE, SHORT, LONG, FLOAT, DOUBLE, BYTE_CPLX, SHORT_CPLX, LONG_CPLX, FLOAT_CPLX, DOUBLE_CPLX, defined in the include file *esps/esps.h.* (Codes for non-numeric ESPS types such as CHAR, CODED, EFILE, and AFILE are not allowed.)

num_records
: The number of consecutive records that may be stored in the data part of this *feasd* structure (see below).

num_channels
: The number of channels of sampled data.

data
: A pointer through which to access storage for the sampled data from *num_records* consecutive FEA_SD records. For single-channel files, the storage is just an array with *num_records* elements of the type indicated by *data_type.* The pointer *data* when cast to the appropriate type points to the first element of the array. The appropriate types are (char *) for BYTE data, (short *) for SHORT data, (double_cplx *) for DOUBLE_CPLX data, etc. For example, with declarations

*struct feasd   *rec;*
*short   *s_data;*

suppose *rec* points to a *feasd* structure properly initialized to hold SHORT data. (So *rec−>data_type==SHORT*.) Then, after the assignment

*s_data = (short *) rec−>data;*

sample number *s* may be accessed as *s_data*[*s*]. For multi-channel files, the storage is conceptually a 2-dimensional array with *num_records* rows and as many columns as there are channels. If the number of channels is fixed and known in advance (say *num_channels*==3) then a declaration

*short   (*s_arr)[3];*

and a cast

s_arr = (short (*)[3]) rec−>data;

allow sample number *s* of channel *c* to be accessed as *s_arr*[*s*][*c*]. Unfortunately the constant 3 here cannot be replaced with a variable such as *rec−>num_channels* in legal C code. One procedure that works with a variable number of channels is to treat the data array as 1-dimensional and access sample number *s* of channel *c* as *s_data*[*s***rec−>num_channels*+*c*], where *s_data* is defined as above. This way of accessing multichannel data is somewhat awkward; a more convenient method involves the structure member *ptrs* discussed next.

ptrs When appropriately cast, *ptrs*, if not NULL, points to the first element of an array of *num_records* pointers, each of which points to the first element of a row of the data array. Appropriate types for the cast are (char **) for BYTE data, (short **) for SHORT data, etc. To continue the example begun above under *data,* suppose a declaration

*short **s_ptr;*

and a cast

*s_ptr = (short **) rec−>ptrs;*

then sample number *s* of channel *c* can be accessed as *s_ptr*[*s*][*c*]. The function *allo_feasd_recs*(3-ESPSu), which creates *feasd* structures, has a parameter that determines whether to set up the pointer array or to make the *ptrs* structure member NULL. For single-channel data, *ptrs* should probably be made NULL except to avoid special-case code for single-channel data in cases where multichannel data must be handled as well. For multichannel data, the pointer array should probably be used except when considerations such as space limitations warrant forgoing the convenience.

Since the structure of the data storage depends on values in the file header, it is important to be sure that a given FEA_SD struct is consistent with the header of the file it is being used with. Specifically, all headers used with a given FEA_SD struct must have the same number of channels.

SEE ALSO

init_feasd_hd(3-ESPSu), allo_feasd_recs(3-ESPS), get_feasd_recs(3-ESPSu), get_feasd_orecs, put_feasd_recs(3-ESPSu), get_genhd(3-ESPSu), ESPS(5-ESPS), FEA(5-ESPS), SD(5-ESPS)

RECORD ELEMENT FILE STRUCTURE

FILES

/usr/esps/include/esps/fea.h
/usr/esps/include/esps/feasd.h

FUTURE CHANGES

Support for tagged files.

AUTHOR

Manual page by Rodney Johnson, ESI.