$ K + The ABF Header Section

The ABF header is the first block of data at the start of an ABF data file. The header contains parameters that describe the stimulation, the acquisition and the hierarchy of the data. It describes the contents of the data file and contains entries to describe the settings in effect when the data file was acquired.

In version 1.8, the header is 6144 bytes long. This will most likely be increased in subsequent versions. Third party programs should NOT rely on the size of the header, or retrieve information based on a byte offset. Only use the documented variables defined in the header.

See the file ABFHEADR.H for a "C" definition of the ABFFileHeader structure.

ABF Header Sub Sections

1. File ID and Size information
2. File Structure
3. Trial Hierarchy Information
4. Display Parameters
5. Hardware Information
6. Environmental information
7. Multi-channel information
8. Synchronous timer outputs
9. Epoch Waveform and Pulses
10. DAC Output File
11. Conditioning pulse train
12. Variable Parameter User List
13. Autopeak measurement
14. Channel Arithmetic
15. On-line Subtraction
16. Miscellaneous Parameters
17. Extended File Structure
18. Extended Multi-Channel Information
19. Train Parameters
20. Extended Epoch Waveform and Pulses
21. Extended DAC Output File
22. Extended Pre-sweep (Conditioning) Pulse Train
23. Extended Variable Parameter User List
24. Extended On-Line Subtraction
25. Extended Environmental Information
26. Extended Statistics Measurements
27. Application Version Data
28. LTP Protocol
29. Digidata 132x Trigger Out
30. Epoch Resistance
31. Alternating Episodic Mode
32. Post-Processing Actions

The total header length = 6144 bytes.
Offsets in the lists referenced above are Byte offsets.

Notes:

Data Types
Version Numbers
ADC Channel Numbering
Composite Values
Indexing Arrays in the ABF Header

The_ABF_Header_Section

$ The ABF Header Section

K The ABF Header Section

+ ABFFUNC

$ K + ADC Channel Numbering

Axon's data acquisition programs distinguish between physical and logical channel numbers. Physical channel numbers are the channel numbers used internally to communicate with the acquisition hardware. Logical channel numbers are the external connector labels on the front panel of the acquisition hardware. Logical channel numbers are used only for presentation to the user. Physical channel numbers are used everywhere else. For example, parameters are stored using physical channel number order (0 to 15) for such structures as the sampling sequence array and the entries for the external lowpass and highpass filters. Similarly, a physical channel number is used for the Trigger channel. At the time of printing this document, the only digitizer known to have different physical and logical channel numbering is the (now obsolete) TL-2 interface.

[#] ADC_Channel_Numbering

[$] ADC Channel Numbering

[K] ADC Channel Numbering

[+] ABFFUNC

$ K + Indexing Arrays in the ABF Header

To get a Logical channel number from a Physical channel number, simply index the nADCPtoLchannelMap array by the Channel number you wish to convert. Thus nADCPtoLchannelMap[1] provides the Logical Channel Number for Physical Channel Number 1. This array is always symmetrical, so it can be used in the same way to convert back to Physical Channel Numbers from Logical Channel Numbers.

The first thing to look at is the nADCSamplingSeq array. This tells you which physical ADC channels were acquired and in what order. The first entry in this array is the Physical channel number of the first ADC channel acquired, followed by the second etc. There are nADCNumChannels channels in this array. All ADC arrays except for the nADCSamplingSeq are indexed through Physical channel numbers. These include: sADCChannelName, sADCUnits, etc.

Special Note: All array indexing within the header and within the ABF routines start at 0, except Sweep number, which starts at 1.

```
void ShowFirstAcquiredChannelInfo( int nFile, ABFFileHeader *pFH )
{
   // Get first physical channel number and name
   int nFirstPhysicalChannel = pFH->nADCSamplingSeq[0];
   char *psSignalName = pFH->sADCChannelName[nFirstPhysicalChannel];

   // Get channel number to show to user
   int nFirstLogicalChannel = pFH->nADCPtoLchannelMap[nFirstPhysicalChannel];
   fprint("The first acquired channel (%s) comes from ADC channel %d\n",
          psSignalName, nFirstLogicalChannel);

   // Get start time of first episode of first channel
   int nFirstEpisode = 1;
   float fStartTime;
   ABF_GetStartTime(nFile, pFH, nFirstPhysicalChannel, nFirstEpisode,
                    &fStartTime, NULL);
}
```

Indexing_Arrays_in_the_ABF_Header

$ Indexing Arrays in the ABF Header

K Indexing Arrays in the ABF Header

+ ABFHELP

$ K + Composite Values

Axon Instruments' acquisition programs can automatically read the gain and lowpass filter settings of the analog instrument (e.g. patch clamp) on upto 16 ADC channels. If the ABF reading routines are used the telegraphed gain must be taken into account when calculating the signal scale factor for the telegraphed channel, as must the telegraphed lowpass filter setting. Both of these factors are automatically taken into account by the ABF reading routines.

In some cases, several parameters must be considered together to correctly interpret the data.

Composite ADC scale factor	fInstrumentScaleFactor[n] * fTelegraphAdditGain * fADCProgrammableGain[n] * fSignalGain[n]. Note, the fTelegraphAdditGain only applies to the telegraphed ADC channel.
Composite ADC offset	fInstrumentOffset[n] + fSignalOffset[n]
Composite lowpass filter -3 dB frequency	A complex function of fSignalLowpassFilter and fTelegraphFilter, for the telegraphed ADC channel. On other ADC channels, only the fSignalLowpassFilter contributes to the composite value.

The functions ABFH_GetADCtoUUFactors() and ABFH_GetDACtoUUFactors() should be used to get the combined scale factors required for converting ADC/DAC values to UserUnits and vice versa.

Composite_Values

$ Composite Values

K Composite Values

+ ABFFUNC

$ K + Data Types

Unused real parameters should be filled with zeros. Unused strings should be filled with the space character (ASCII #32).

Parameters for unsampled ADC channels should be filled with the indicated default.

Data_Types

$ Data Types

K Data Types

+ ABFFUNC

$ K + Version Numbers

The file version number consists of a major and a minor number. For example, the "1" in Version 1.0 is the major number, and the "0" is the minor number.

In general, the major version number is updated when changes affect the byte offset of the existing parameters or would otherwise make the file unusable by existing programs. The minor version number is updated when unused parameters are utilized. In most cases, existing programs will not be affected since they should not be dependent upon the unused parameters.

[#] Version_Numbers

[$] Version Numbers

[K] Version Numbers

[+] ABFFUNC

$ K + ABF Hardware and Storage Limits

Some of the more important limitations in the range of hardware supported and the size of components in ABF formatted files are listed here.

- Sixteen physical ADC channels, numbered 0-15.
- Up to sixteen bits per ADC word.
- Up to 1,032,258 multiplexed samples per sweep in high-speed oscilloscope mode, fixed-length event-driven mode and episodic stimulation mode.
- Up to 2 G multiplexed samples per segment in variable-length event-driven mode and gap-free mode.
- Stimulus waveform can be generated on two DACs channel simultaneously.
- Pre-sweep train (was previously called conditioning train) can be generated on two DACs channel simultaneously.
- One Math Channel.
- Upto 16 telegraphed instruments
- Leak subtraction can be applied to ADC channels simultaneously.
- One set of display amplifications and offsets.
- Only one averaged Run per file!

ABF_Hardware_and_Storage_Limits

$ ABF Hardware and Storage Limits

K ABF Hardware and Storage Limits

+ ABFHELP

Special Note: A header entry name prefixed with underscore "_" indicates that this parameter is discontinued or has been replaced with an updated version. In such cases the parameter is also tagged as Discontinued in the description field and where appropriate a link has been added to the updated version of the parameter in the Word version of this document.

$ K + File ID and Size information

(40 bytes).

These ten entries represent two types of information: (1) File identification information; (2) Parameters whose actual value may be different to the value requested before the acquisition commenced, e.g. due to a user abort. The requested values are located elsewhere.

Offset	Header Entry Name	Type	Description
0	lFileSignature	long	File type used for format identification. Possible values are: "ABF ", "CLPX" and "FTCX". Used to create the numbers in nFileType. (In old pCLAMP and Axotape data files, the first four bytes were a float: 1 = CLAMPEX, 10 = FETCHEX/AxoTape. This is translated on reading into either CLPX or FTCX as appropriate.)
4	fFileVersionNumber	float	File format version stored in the data file during acquisition. The present version is 1.5. (In old pCLAMP and Axotape data files, this parameter is in the range 2.0-5.3)
8	nOperationMode	short	Operation mode: 1 = Event-driven, variable length; 2 = Oscilloscope, loss free (Same as Event-driven, fixed length); 3 = Gap-free; 4 = Oscilloscope, high-speed; 5 = episodic stimulation (Clampex only).
10	lActualAcqLength	long	Actual number of ADC samples (aggregate) in data file. See lAcqLength. Averaged sweeps are included.
14	nNumPointsIgnored	short	Number of points ignored at data start. Normally zero, but non-zero for gap-free acquisition using AXOLAB configurations with one or more ADS boards.
16	lActualEpisodes	long	Actual number of sweeps in the file including averaged sweeps. See lEpisodesPerRun. If nOperationMode = 3 (gap-free) the value of this parameter is 1.
20	lFileStartDate	long	Date when data portion of this file was first written to. Stored as YYMMDD. If YY is in the range 80-99, prefix with "19" to get the year. If YY is in the range 00-79, prefix with "20" to get the year.
24	lFileStartTime	long	Time of day in seconds past midnight when data portion of this file was first written to.

File_ID_and_Size_information

$ File ID and Size information

K File ID and Size information;lFileSignature;fFileVersionNumber;nOperationMode;lActualAcqLength;nNumPointsIgnored;lActualEpisodes;lFileStartDate;lFileStartTime;lStopwatchTime;fHeaderVersionNumber;nFileType;nMSBinFormat

+ ABFFUNC

28	lStopwatchTime	long	Time since the stopwatch was zeroed that the data portion of this file was first written to. Not supported by all programs. Default = 0.
32	fHeaderVersionNumber	float	Version number of the header structure returned by the ABF_ReadOpen function. Currently 1.8. This parameter does not identify the data file format. See fFileVersionNumber.
36	nFileType	short	Numeric equivalent of file type. 1 = ABF file; 2 = Old FETCHEX file (FTCX); 3 = Old Clampex file (CLPX). See sFileType.
38	nMSBinFormat	short	Storage method for real numbers in the header. Also see nDataFormat. 0 = IEEE format; 1 = Microsoft Binary format (old files only).

$ K + File Structure

(78 bytes).

Header entries describing the structure of the file. See Extended File Structure section for new variables

Offset	Header Entry Name	Type	Description
40	lDataSectionPtr	long	Block number of start of Data section.
44	lTagSectionPtr	long	Block number of start of Tag section.
48	lNumTagEntries	long	Number of Tag entries.
52	lScopeConfigPtr	long	Block number of the ABF Scope Config section. (was block number of start of Long Description section.)
56	lNumScopes	long	Number of ABFScopeConfig structures in the ABF Scope Config section. (was number of lines of Long Description.)
60	_lDACFilePtr	long	Block number of start of DAC file section. No longer used. Discontinued: Use lDACFilePtr
64	_lDACFileNumEpisodes	long	Number of sweeps in the DAC file section. Sweeps are not multiplexed. Discontinued: Use lDACFileNumEpisodes
68	SUnused001	4char	Unused.
72	lDeltaArrayPtr	long	Block number of start of Delta Array section.
76	lNumDeltas	long	Number of entries in Delta Array section.
80	lVoiceTagPtr	long	Block number of start of Voice Tag section.
84	lVoiceTagEntries	long	Number of Voice Tag entries.
88	lUnused002	long	(was number of automatic entries in Notebook section)
92	lSynchArrayPtr	long	Block number of start of the Synch Array section.
96	lSynchArraySize	long	Number of pairs of entries in the Synch Array section. If averaging is enabled, this includes the entry for the averaged sweep.
100	nDataFormat	short	Data representation. 0 = 2-byte integer; 1 = IEEE 4 byte float.
102	nSimultaneousScan	short	ADC Channel Scanning Mode: 0=Multiplexed; 1=Simultaneous Scanning (currently unimplemented)
104	lStatisticsConfigPtr	long	Block number of start of Scope Config section.
108	lAnnotationSectionPtr	long	Block number of start of annotations section
112	lNumAnnotations	long	Number of annotations
116	sUnused003	2char	Unused.

File_Structure

$ File Structure

K File Structure;lDataSectionPtr;lTagSectionPtr;lNumTagEntries;lScopeConfigPtr;lNumScopes;lDACFilePtr;lDACFileNumEpisodes;lDeltaArrayPtr;lNumDeltas;lSynchArrayPtr;lSynchArraySize;nDataFormat;

+ ABFFUNC

$ K + Trial Hierarchy Information

(82 bytes).

Header entries describing the trial hierarchy.

Offset	Header Entry Name	Type	Description
118	channel_count_acquired	short	Number of analog input channels acquired. Can be less than nADCNumChannels. Currently unimplemented.
120	nADCNumChannels	short	Number of analog input channels sampled.
122	fADCSampleInterval	float	Interval between multiplexed A/D samples (us). In Clampex, this is also known as the first clock interval.
126	fADCSecondSampleInterval	float	Second definition of the interval between multiplexed A/D samples (us). If this interval is zero, the first clock interval is used for the whole of each sweep. If this interval is non-zero, the second clock interval starts at the sample number specified by lClockChange.
130	fSynchTimeUnit	float	Time unit for start time in the Synch Array section: 0 = Value in sample intervals; nn = Value in µs. See notes below for the Synch Array section. pCLAMP 6.0 and AxoTape 2.0 use value in sample intervals only.
134	fSecondsPerRun	float	Requested acquisition length in seconds. 0 means use available disk space. -1 means ignore this parameter and refer to lEpisodesPerRun.
138	lNumSamplesPerEpisode	long	Number of multiplexed ADC samples per sweep if nOperationMode is 2, 4, or 5. Undefined if nOperationMode is 1 or 3.
142	lPreTriggerSamples	long	Pre-trigger interval stored as number of multiplexed ADC samples (note that this is the underlying number of ADC samples, not the number of pre-trigger samples in the trigger channel). FETCHEX uses the same value for the post-trigger interval. Undefined if nOperationMode is 3 or 5.
146	lEpisodesPerRun	long	Requested number of sweeps/run. 0=Run until terminated by user. If nOperationMode = 3 (gap free), this parameter is 1 and the requested acquisition length is set in fSecondsPerRun.
150	lRunsPerTrial	long	Requested number of runs/trial. 0=Run until terminated by user. Runs are averaged. If nOperationMode = 3 (gap free), the value of this parameter is 1. See lAverageCount.
154	lNumberOfTrials	long	Number of trials to be acquired. Note, only one trial is contained in a data file. This parameter is used for Clampex acquisition control. -1 = continuous.

Trial_Hierarchy_Information

$ Trial Hierarchy Information

K Trial Hierarchy Information;nADCNumChannels;fADCSampleInterval;fADCSecondSampleInterval;fSynchTimeUnit;fSecondsPerRun;lNumSamplesPerEpisode;lPreTriggerSamples;lEpisodesPerRun;lRunsPerTrial;lNumberOfTrials;nAveragingMode;nUndoRunCount;nFirstEpisodeInRun;fTriggerThreshold;nTriggerSource;nTriggerAction;nTriggerPolarity;fScopeOutputInterval;fEpisodeStartToStart;fRunStartToStart;fTrialStartToStart;lAverageCount;lClockChange;nAutoTriggerStrategy

+ ABFFUNC

158	nAveragingMode	short	Data averaging strategy: 0 = No averaging; 1 = Average but don't save raw data; 2 = Average and save raw data.
160	nUndoRunCount	short	Frequency with which temporary undo file created during averaging is updated. -1 = Disabled; 0 = after each pseudo-doubling; N = after every N runs.
162	nFirstEpisodeInRun	short	First sweep number (normally 1).
164	fTriggerThreshold	float	Trigger threshold value for Event or Oscilloscope mode (user units).
168	nTriggerSource	short	Trigger source: N (>=0) = Physical channel number selected for threshold detection; -1 = external trigger; -2 = keyboard; -3 = use start-to-start interval. If nOperationMode=3 (gap-free) 0= start immediately.
170	nTriggerAction	short	Trigger start strategy: 0 = start one sweep; 1 = start one run; 2 = start one trial.
172	nTriggerPolarity	short	Trigger on: 0 = rising edge of waveform; 1 = falling edge. The terms "rising" and "falling" refer to the direction of the data transitions on the screen during acquisition.
174	fScopeOutputInterval	float	Interval between digital output used for external oscilloscope, or equivalent, and the start of each sweep (ms)
178	fEpisodeStartToStart	float	Time between start of sweeps (seconds). Use when nTriggerSource = "start-to-start".
182	fRunStartToStart	float	Time between start of runs (seconds).
186	fTrialStartToStart	float	Time between trials (seconds).
190	lAverageCount	long	The actual number of runs contributing to each average. See lEpisodesPerRun or lRunsPerTrial.
194	lClockChange	long	The multiplexed ADC sample number after which the second sampling interval commences. 0 = lNumSamplesPerEpisode/2. See fADCSecondSampleInterval.
198	nAutoTriggerStrategy	short	0 = Do not auto trigger; 1 = Autotrigger. (Only significant when nOperationMode == ABF_HIGHSPEEDOSC)

$ K + Display Parameters

(44 bytes).

Header entries describing display options in effect during acquisition.

Offset	Header Entry Name	Type	Description
200	nDrawingStrategy	short	Strategy for the drawing of raw data: 0 = not at all; 1 = update immediately as data is acquired; 2 = update at the end of each sweep or trace; 3 = update at the end of each run;
202	nTiledDisplay	short	ADC channel display arrangement: 0 = Superimposed; 1 = Tiled.
204	nEraseStrategy	short	Automatically erase screen: 0 = not at all; 1 = before each sweep; 2 = before each run. Discontinued: Use scope config entry instead.
206	nDataDisplayMode	short	Data display mode: 0 = Points; 1 = Lines.
208	lDisplayAverageUpdate	long	Display averaged data: -1 = at end of trial; 0 = after each pseudo-doubling; N = after N runs.
212	nChannelStatsStrategy	short	Show channel statistics in gap-free mode: 0 = No; 1 = Yes.
214	lCalculationPeriod	long	Length of the real-time statistics update period in samples. Conventionally a multiple of 1024. Default = 16384. Discontinued: Use fStatisticsPeriod
218	lSamplesPerTrace	long	Number of multiplexed ADC samples in displayed trace.
222	lStartDisplayNum	long	Starting sample number for sweep display: N = starting sample number. (Use N = 1 to start from the first sample.)
226	lFinishDisplayNum	long	Finishing sample number for sweep display: 0 = finish at end of sweep; N = finishing sample number.
230	nMultiColor	short	Color control for multi-trace displays. 0 = single color for all traces; 1 = two or more colors for traces.
232	nShowPNRawData	short	0 = display corrected data; 1 = display raw data.
234	fStatisticsPeriod	float	Length of the real-time statistics update period in seconds. Not used.
238	lStatisticsMeasurements	long	Bit mask for statistics measurements to display. This applies to trigger statistics only. For episodic statistics use lStatsMeasurements Above Threshold: 0x00000001 Event Frequency: 0x00000002

Display_Parameters

$ Display Parameters

K Display Parameters;nDrawingStrategy;nTiledDisplay;nEraseStrategy;nDataDisplayMode;lDisplayAverageUpdate;nChannelStatsStrategy;lCalculationPeriod;lSamplesPerTrace;lStartDisplayNum;lFinishDisplayNum;nMultiColor;nShowPNRawData;fStatisticsPeriod

+ ABFFUNC

			Mean Open Time: 0x00000004
			Mean Closed Time: 0x00000008
242	nStatisticsSaveStrategy	short	Strategy used to save statistics: No Auto Save = 0; Auto Save = 1

[#] [$] [K] [+] Hardware Information

(16 bytes).

Description of hardware properties of digitizer used for acquisition.

Offset	Header Entry Name	Type	Description
244	fADCRange	float	ADC positive full-scale input in volts (e.g. 10.00V).
248	fDACRange	float	DAC positive full-scale range in volts.
252	lADCResolution	long	Number of ADC counts corresponding to the positive full-scale voltage in ADCRange (e.g. 2000, 2048, 32000 or 32768).
256	lDACResolution	long	Number of DAC counts corresponding to the positive full-scale voltage in DACRange.

[#] Hardware_Information

[$] Hardware Information

[K] Hardware Information;fADCRange;fDACRange;lADCResolution;lDACResolution

[+] ABFFUNC

[#] [$] [K] [+] Environmental information

(118 bytes).

Description of telegraph properties and comment info. See Extended Environmental Information for new parameters.

Offset	Header Entry Name	Type	Description
260	nExperimentType	short	Experiment type: 0 = Voltage Clamp; 1 = Current Clamp.
262	_nAutosampleEnable	short	Enable storage of autosample information: 0 = Disabled; 1 = Automatic; 2 = Manual. Discontinued: Use nTelegraphEnable
264	_nAutosampleADCNum	short	Physical ADC channel number to which autosampled parameters apply. Discontinued.
266	_nAutosampleInstrument	short	Autosample instrument: (IHS-1 telegraphs are not supported.) Note, for most programs this is an information-only field. For example, in Clampex the autosample instrument is chosen as a configuration item and copied into this field. Discontinued: Use nTelegraphInstrument
268	_fAutosampleAdditGain	float	Additional gain multiplier of Instrument connected to nAutosampleADCNum. (Optionally autosampled by some acquisition programs.) (Default = 1) Discontinued: Use fTelegraphAdditGain
272	_fAutosampleFilter	float	Lowpass filter cutoff frequency of Instrument connected to nAutosampleADCNum. (Optionally autosampled by some acquisition programs.) (Default = 100000) Discontinued: Use fTelegraphFilter
276	_fAutosampleMembraneCap	float	Patch-clamp membrane capacitance compensation. (Optionally autosampled by some acquisition programs.) Discontinued: Use fTelegraphMembraneCap
280	nManualInfoStrategy	short	Strategy for writing the manually entered information: 0 = Do not write; 1 = Write each trial; 2 = Prompt each trial..
282	fCellID1	float	Numeric identifier #1, e.g. cell identifier.
286	fCellID2	float	Numeric identifier #2, e.g. temperature in °C.
290	fCellID3	float	Numeric identifier #3.
294	sCreatorInfo	16char	Name and version of program used to create the file. For example, "AxoTape 2.0" or "Clampex 6.0".

[#] Environmental_information

[$] Environmental information

[K] Environmental information;nExperimentType;nAutosampleEnable;nAutosampleADCNum;nAutosampleInstrument;fAutosampleAdditGain;fAutosampleFilter;fAutosampleMembraneCap;nManualInfoStrategy;fCellID1;fCellID2;fCellID3;sCreatorInfo;sFileComment

[+] ABFFUNC

310	_sFileComment	56char	56 byte ASCII comment string. Discontinued: Use sFileComment
366	nFileStartMillisecs	short	Milliseconds portion of lFileStartTime
368	nCommentsEnable	short	Enable comments field
370	sUnused003a	8char	Unused.

$ K + Multi-channel information

(1044 bytes).

Description of ADC channel properties.

Offset	Header Entry Name	Type	Description
378	nADCPtoLChannelMap(0-15)	short	ADC physical-to-logical channel map. The entries are in the physical order 0, 1, 2, ... 14, 15. If there are fewer than 16 logical channels in the system, the array is padded with -1. All channels supported by the hardware are present, even if only a subset is used. For example, for the TL-2 the entries would be 7, 6, 5, 4, 3, 2, 1, 0, -1 ... -1.
410	nADCSamplingSeq(0-15)	short	ADC channel sampling sequence. This is the order in which the physical ADC channels are sampled. If fewer than the maximum number of channels are sampled, pad with -1. For example, if two channels are sampled on the TL-2, this array will contain 6, 7, -1 ... -1. If two channels are sampled on the TL-1, this array will contain 14, 15, -1 ... -1.
442	sADCChannelName(0-15)	10char	ADC channel name in physical channel number order. Default = spaces.
602	sADCUnits(0-15)	8char	The user units for ADC channels in physical channel number order. Default = spaces.
730	fADCProgrammableGain(0-15)	float	ADC programmable gain in physical channel number order (dimensionless). Default = 1.
794	fADCDisplayAmplification(0-15)	float	ADC channel display amplification in physical channel number order (dimensionless.) Default = 1.
858	fADCDisplayOffset(0-15)	float	ADC channel display offset in physical channel number order (user units). Default = 0.
922	fInstrumentScaleFactor(0-15)	float	Instrument scale factor in physical ADC channel number order (Volts at ADC / user unit). (Programs would normally display this information to the user as user units / volt at ADC).
986	fInstrumentOffset(0-15)	float	Instrument offset in physical ADC channel number order (user units corresponding to 0 V at the ADC). Default is zero.
1050	fSignalGain(0-15)	float	Signal conditioner gain in physical ADC channel number order (dimensionless). Default = 1.
1114	fSignalOffset(0-15)	float	Signal conditioner offset in physical ADC channel number order (user units). Default = 0.
1178	fSignalLowpassFilter(0-15)	float	Signal-conditioner lowpass filter corner frequency in physical ADC channel number order (Hz). 100000 means lowpass filter is bypassed (i.e. wideband). Default = 100000.

Multi_channel_information

$ Multi-channel information

K Multi-channel information;nADCPtoLChannelMap;nADCSamplingSeq;sADCChannelName;sADCUnits;fADCProgrammableGain;fADCDisplayAmplification;fADCDisplayOffset;fInstrumentScaleFactor;fInstrumentOffset;fSignalGain;fSignalOffset;fSignalLowpassFilter;fSignalHighpassFilter;sDACChannelName;sDACChannelUnits;fDACScaleFactor;fDACHoldingLevel;nSignalType

+ ABFFUNC

1242	fSignalHighpassFilter(0-15)	float	Signal-conditioner highpass filter corner frequency in physical ADC channel number order (Hz). 0 means highpass filter is bypassed (i.e. DC coupled). -1 means inputs are grounded. Default = 0.
1306	sDACChannelName(0-3)	10char	DAC channel name. Default = spaces.
1346	sDACChannelUnits(0-3)	8char	The user units for this DAC channel. Default = spaces.
1378	fDACScaleFactor(0-3)	float	DAC channel gain (user units / V at DAC). Default = 1
1394	fDACHoldingLevel(0-3)	float	DAC channel holding level (user units). Default = 0.
1410	nSignalType	short	Type of signal conditioner that was used. 0 = None; 1 = CyberAmp 320/380.
1412	SUnused004	10char	Unused.

$ K + Synchronous timer outputs

(14 bytes).

Synchronous timer outputs were dropped from pCLAMP version 6. These parameters have been kept in the ABF 1.0 specification for compatibility when reading old data files. New programs should write zeros to all of the parameters in this group.

Offset	Header Entry Name	Type	Description
1422	nOUTEnable	short	Enable synchronous timer outputs: 0 = No; 1 = Yes.
1424	nSampleNumberOUT1	short	Sample number for pulse on synchronous timer OUT #1.
1426	nSampleNumberOUT2	short	Sample number for pulse on synchronous timer OUT #2.
1428	nFirstEpisodeOUT	short	First sweep at which synchronous timer OUT #1 and #2 fire.
1430	nLastEpisodeOUT	short	Last sweep at which synchronous timer OUT #1 and #2 fire.
1432	nPulseSamplesOUT1	short	Duration and polarity of pulse on synchronous timer OUT #1 (DAC samples).
1434	nPulseSamplesOUT2	short	Duration and polarity of pulse on synchronous timer OUT #2 (DAC samples).

[#] Synchronous_timer_outputs

[$] Synchronous timer outputs

[K] Synchronous timer outputs;nOUTEnable;nSampleNumberOUT1;nSampleNumberOUT2;nFirstEpisodeOUT;nLastEpisodeOUT;nPulseSamplesOUT1;nPulseSamplesOUT2;

[+] ABFFUNC

$ K + Epoch Waveform and Pulses

(184 bytes).

Description of epoch waveform properties. See Extended Epoch Waveform and Pulse for new parameters.

Offset	Header Entry Name	Type	Description
1436	nDigitalEnable	short	Enable digital outputs: 0 = No; 1 = Yes.
1438	_nWaveformSource	short	Analog waveform source: 0 = Disable; 1 = Generate waveform from epoch definitions; 2 = Generate waveform from a DAC file. Discontinued: Use nWaveformSource
1440	nActiveDACChannel	short	Active DAC channel, i.e. the one used for waveform generation. Redundant in ABF v1.8 except for control of the active digital output. To control each DAC waveform use nWaveformEnable
1442	_nInterEpisodeLevel	short	Inter-sweep holding level: 0 = Use holding level; 1 = Use last epoch amplitude. Discontinued: Use nInterEpisodeLevel
1444	_nEpochType(0-9)	short	Epoch type: 0 = Disabled; 1 = Step; 2 = Ramp. Discontinued: Use nEpochType
1464	_fEpochInitLevel(0-9)	float	Epoch initial level (user units). Discontinued: Use fEpochInitLevel
1504	_fEpochLevelInc(0-9)	float	Epoch level increment (user units). Discontinued: Use fEpochLevelInc
1544	_nEpochInitDuration(0-9)	short	Epoch initial duration (in sequence counts). Discontinued: Use lEpochInitDuration
1564	_nEpochDurationInc(0-9)	short	Epoch duration increment (in sequence counts) Discontinued lEpochDurationInc
1584	nDigitalHolding	short	Holding value for digital output.
1586	nDigitalInterEpisode	short	Inter-sweep digital holding value: 0 = Use holding value; 1 = Use last epoch value.
1588	nDigitalValue(0-9)	short	Epoch value for digital output (0..9).
1610	sUnavailable1608	4char	This parameter was used by CLAMPFIT 6.0, do not reuse. Was fWaveformOffset. Offset (in active DAC user units) in instrument command pathway, usually due to a non-zero holding potential or current.
1612	nDigitalDACChannel	short	Not used.
1614	sUnused005	6char	Unused.

Epoch_Waveform_and_Pulses

$ Epoch Waveform and Pulses

K Epoch Waveform and Pulses;nDigitalEnable;nWaveformSource;nActiveDACChannel;nInterEpisodeLevel;nEpochType;fEpochInitLevel;fEpochLevelInc;nEpochInitDuration;nEpochDurationInc;nDigitalHolding;nDigitalInterEpisode;nDigitalValue

+ ABFFUNC

[#] [$] [K] [+] DAC Output File

(98 bytes).

Description of DAC File properties. See Extended DAC File for new parameters.

Offset	Header Entry Name	Type	Description
1620	_fDACFileScale	float	Scaling factor to apply to DAC file contents.
			Discontinued: Use fDACFileScale
1624	_fDACFileOffset	float	Offset (in user units) to apply to DAC file contents.
			Discontinued: Use fDACFileOffset
1628	sUnused006	2char	Unused.
1630	_nDACFileEpisodeNum	short	Sweep (or column) number to replay from waveform file: -1 = all except the first (which is skipped), repeating last if necessary; 0 = all sweeps; N = sweep number.
			Discontinued: Use lDACFileEpisodeNum
1632	_nDACFileADCNum	short	Logical ADC channel number to replay from waveform file.
			Discontinued: Use nDACFileADCNum
1634	_sDACFilePath	84char	Drive and directory for sDACFileName.
			Discontinued: Use sDACFilePath

[#] DAC_Output_File

[$] DAC Output File

[K] DAC Output File;fDACFileScale;fDACFileOffset;nDACFileEpisodeNum;nDACFileADCNum;sDACFileName;sDACFilePath;

[+] ABFFUNC

[#] [$] [K] [+] Conditioning pulse train

(44 bytes).

This section is replaced by Extended Conditioning Pulse Train variables. Also Conditioning Pulse Train has been renamed Pre-sweep Train in the application but variable names remain the same.

Offset	Header Entry Name	Type	Description
1718	_nConditEnable	short	Conditioning pulse train activation status: 0 = Disable; 1 = Enable.
			Discontinued: Use nConditEnable
1720	_nConditChannel	short	DAC channel used for conditioning pulses.
			Discontinued.
1722	_lConditNumPulses	long	Number of pulses in conditioning pulse train.
			Discontinued: Use lConditNumPulses
1726	_fBaselineDuration	float	A single pulse in the conditioning train consists of a baseline followed by a step. This parameter is the baseline duration in ms.
			Discontinued: Use fBaselineDuration
1730	_fBaselineLevel	float	Baseline level (user units).
			Discontinued: Use fBaselineLevel
1734	_fStepDuration	float	Step duration (ms).
			Discontinued: Use fStepDuration
1738	_fStepLevel	float	Step level (user units).
			Discontinued: Use fStepLevel
1742	_fPostTrainPeriod	float	At the end of the conditioning train there is a post-train steady-state output. This parameter is the post-train duration in ms.
			Discontinued: Use fPostTrainPeriod
1746	_fPostTrainLevel	float	Post-train level (user units).
			Discontinued: Use fPostTrainLevel
1750	sUnused007	12char	Unused.

[#] Conditioning_pulse_train

[$] Conditioning pulse train

[K] Conditioning pulse train;nConditEnable;nConditChannel;lConditNumPulses;fBaselineDuration;fBaselineLevel;fStepDuration;fStepLevel;fPostTrainPeriod;fPostTrainLevel;

[+] ABFFUNC

[#] [$] [K] [+] Variable Parameter User List

(82 bytes).

This section is replaced by the Extended Variable Parameter User List.

Offset	Header Entry Name	Type	Description
1762	_nParamToVary	short	Holds the index of the parameter that varies from sweep to sweep in one run. Discontinued: Use nULParamToVary
1764	_sParamValueList	80char	List of comma-separated values. If the number of entries in the list is fewer than the requested number of sweeps , the last list value is re-used. If there are more values in the list, the excess list values are ignored. Discontinued: Use sULParamValueList

[#] Variable_Parameter_User_List

[$] Variable Parameter User List

[K] Variable Parameter User List;nParamToVary;sParamValueList

[+] ABFFUNC

[#] [$] [K] [+] Statistics measurement

(36 bytes).

This section is replaced by the Extended Statistics Measurement.

Offset	Header Entry Name	Type	Description
1844	_nAutopeakEnable	short	Autopeak activation status: 0 = Disabled; 1 = Enabled. Discontinued: Use nStatsEnable
1846	_nAutopeakPolarity	short	-1 = search for negative peaks; 0 = search for absolute peak; 1 = search for positive peaks. Discontinued: Use nStatsChannelPolarity
1848	_nAutopeakADCNum	short	Physical ADC channel number to measure. Discontinued: Use bit flag nStatsActiveChannels for selecting multiple channels
1850	_nAutopeakSearchMode	short	Search mode: 0-9 = epoch A-J; -1 = All; -2 = Use specified region. Discontinued: Use nStatsSearchMode
1852	_lAutopeakStart	long	Start of specified statistics region. Only valid if nAutopeakSearchMode = -2. Discontinued: Use lStatsStart
1856	_AutopeakEnd	long	End of specified statistics region. Only valid if nAutopeakSearchMode = -2. Discontinued: Use lStatsEnd
1860	_nAutopeakSmoothing	short	Number of samples averaged in boxcar smoothing window when looking for a peak. Discontinued: Use nStatsSmoothing
1862	_nAutopeakBaseline	short	Baseline for statistics measurements: -2 = None; -1 = Average the first (lNumSamplesPerEpisode/64) samples; 0-9 = epoch A-J Discontinued: Use nStatsBaseline
1864	_nAutopeakAverage	short	0 = search average and data; 1 = search average only. Discontinued.
1866	sUnavailable1866	2char	Was nAutopeakSaveStrategy, use nStatisticsSaveStrategy
1868	_lAutopeakBaselineStart	long	Start of baseline used in statistics measurements. Discontinued: Use lStatsBaselineStart
1872	_lAutopeakBaselineEnd	long	End of baseline used in statistics measurements. Discontinued: Use lStatsBaselineEnd
1876	_lAutopeakMeasurements	long	Bit mask indicating which statistics measurements to perform. Discontinued: Use lStatsMeasurements

---

[#] Autopeak_measurement

[$] Autopeak measurement

[K] Autopeak measurement;nAutopeakEnable;nAutopeakPolarity;nAutopeakADCNum;nAutopeakSearchMode;lAutopeakStart;lAutopeakEnd;nAutopeakSmoothing;nAutopeakBaseline;nAutopeakAverage;

[+] ABFFUNC

[#] [$] [K] [+] Channel Arithmetic

(52 bytes).

Several standard arithmetic expressions are supported:

Expression #0 (general purpose)
Result = (K1*A + K2) <op> (K3*B + K4)

Expression #1 (ratio dyes)
Result = (K1*R + K2) <op> (K3*R + K4)
where
R = (A + K5)/(B + K6)

In both cases:
A and B are ADC channel numbers. A = B is allowed.
K1 .. K6 are constants.
<op> is an arithmetic operator.

Offset	Header Entry Name	Type	Description
1880	nArithmeticEnable	short	Arithmetic activation status: 0 = Disable; 1 = Enable
1882	fArithmeticUpperLimit	float	Display upper limit for arithmetic Results channel.
1886	fArithmeticLowerLimit	float	Display lower limit for arithmetic Results channel.
1890	nArithmeticADCNumA	short	Physical ADC channel number used for A.
1892	nArithmeticADCNumB	short	Physical ADC channel number used for B.
1894	fArithmeticK1	float	Numeric constant K1
1898	fArithmeticK2	float	Numeric constant K2
1902	fArithmeticK3	float	Numeric constant K3
1906	fArithmeticK4	float	Numeric constant K4
1910	sArithmeticOperator	2char	Arithmetic operator: "+ ", "- ", "* " or "/ ".
1912	sArithmeticUnits	8char	Units for Results channel.
1920	fArithmeticK5	float	Numeric constant K5
1924	fArithmeticK6	float	Numeric constant K6
1928	NArithmeticExpression	short	The expression to evaluate: N = Expression #N.
1930	sUnused1930	2char	Unused.

[#] Channel_Arithmetic

[$] Channel Arithmetic

[K] Channel Arithmetic;nArithmeticEnable;fArithmeticUpperLimit;fArithmeticLowerLimit;nArithmeticADCNumA;nArithmeticADCNumB;fArithmeticK1;fArithmeticK2;fArithmeticK3;fArithmeticK4;sArithmeticOperator;sArithmeticUnits;fArithmeticK5;fArithmeticK6;nArithmeticExpression;

[+] ABFFUNC

$ K + On-line Subtraction

(34 bytes).

Parts of this section have been replaced by the Extended On-line subtraction.

Offset	Header Entry Name	Type	Description
1932	_nPNEnable	short	P/N activation status: 0 = Disable; 1 = Enable Discontinued: Use nPNEnable
1934	nPNPosition	short	P/N subpulse execution: 0 = Before waveform; 1 = After waveform.
1936	_nPNPolarity	short	-1 = opposite polarity to the waveform; 1 = same polarity as the waveform. Discontinued: Use nPNPolarity
1938	nPNNumPulses	short	Number of P/N subpulses.
1940	_nPNADCNum	short	Physical ADC channel number for P/N subtraction. Discontinued: Use nPNADCNum
1942	_fPNHoldingLevel	float	Subpulse holding level. Discontinued: Use fPNHoldingLevel
1946	fPNSettlingTime	float	Settling time at subpulse holding level (ms). Wait this interval after changing to subpulse holding level before starting subpulses; wait this interval after returning to normal holding level before starting epochs.
1950	fPNInterpulse	float	Timing interval between the start of one subpulse and the start of the next (ms).
1954	sUnused009	12char	Unused.

On_line_Subtraction

$ On-line Subtraction

K On-line Subtraction

nPNEnable;nPNPosition;nPNPolarity;nPNNumPulses;nPNADCNum;fPNHoldingLevel;fPNSettlingTime;fPNInterpulse;

+ ABFFUNC

$ K + Miscellaneous Parameters

(82 bytes).

Offset	Header Entry Name	Type	Description
1966	_nListEnable	short	Parameter list activation status: 0 = Disable; 1 = Enable. Discontinued: Use nULEnable
1968	nBellEnable(0-1)	short	Auditory tone activation status: 0 = Disable; 1 = Enable
1972	nBellLocation(0-1)	short	Location of bell relative to trial: 0 = Before; 1 = After
1976	nBellRepetitions(0-1)	short	Number of sounds to produce.
1980	nLevelHysteresis	short	Amount of level hysteresis to use when detecting events. This is the amount that the data has to go past the trigger level before it is considered triggered. Re-arming of the trigger level is always done at the actual nominated trigger level. (See fTriggerThreshold)
1982	lTimeHysteresis	long	Amount of time hysteresis to use when detecting events. This is the number of samples that have to be blow the trigger point before the trigger is said to be rearmed.
1986	nAllowExternalTags	short	0 = Do not scan for external tags during acquisition. 1 = Scan for external tags.
1988	nLowpassFilterType(0-15)	char	Type of Low Pass filter for each ADC channel: 0 = None; 1 = External; 2 = Simple RC; 3 = Bessell; 4 = Butterworth.
2004	nHighpassFilterType(0-15)	char	Type of High Pass filter for each ADC channel: 0 = None; 1 = External; 2 = Simple RC; 3 = Bessell; 4 = Butterworth.
2020	nAverageAlgorithm	short	Algorithm used for calculating averages: 0 = Cumulative Averaging; 1 = Most Recent Averaging (uses fAverageWeighting below)
2022	fAverageWeighting	float	Weighting Factor for Most Recent Averaging. This is the proportion of the incoming sweep to include in the average.
2026	nUndoPromptStrategy	short	Strategy for Prompting to create an Undo file: 0 = On Abort; 1 = Always;
2028	nTrialTriggerSource	short	Trigger source for start trial: -3 = Spacebar; -2 = External Trigger; -1 = None
2030	nStatisticsDisplayStrategy	short	Strategy for displaying statistics: 0 = Display Statistics; 1 = Do Not display Statistics
2032	nExternalTagType	short	Type of External Tag: 0 = Time Tag; 1 = External Tag; 2 = External Tag (from BNC input) 3 = Voice Tag (from audio input); 4 = New File Tag
2034	lHeaderSize	long	Total size of the header. Currently 6144 bytes.

[#] Miscellaneous_parameters

[$] Miscellaneous parameters

[K] Miscellaneous parameters

nListEnable;nBellEnable;nBellLocation;nBellRepetitions;nLevelHysteresis;lTimeHysteresis;nAllowExternalTags;

[+] ABFFUNC

2038	dFileDuration	double	Not used.
2046	nStatisticsClearStrategy	short	Strategy for clearing statistics from display: 0 = Do not clear statistics; 1 = Clear Statistics.

$ K + Extended File Structure

(26 bytes).

Offset	Header Entry Name	Type	Description
2048	lDACFilePtr(0-1)	long	Pointer to location of DAC stimulus file data in physical DAC channel order.
2056	lDACFileNumEpisodes(0-1)	long	Number of episodes for each DAC stimulus file in physical DAC channel order.
2064	sUnused010	10char	Unused.

Extended_File_Structure

$ Extended File Structure

K Extended File Structure

lDACFilePtr;lDACFileNumEpisodes;

+ ABFFUNC

[#] [$] [K] [+] Extended Multi-Channel Information

(62 bytes)

Offset	Header Entry Name	Type	Description
2074	fDACCalibrationFactor(0-1)	float	Calibration factor for the current digitizer in physical DAC channel order.
2090	fDACCalibrationOffset(0-1)	float	Calibration offset for the current digitizer in physical DAC channel order.
2106	sUnused011	30char	Unused.

[#] Extended_MultiChannel_Information

[$] Extended MultiChannel Information

[K] Extended MultiChannel Information

fDACCalibrationFactor;fDACCalibrationOffset;

[+] ABFFUNC

$ K + Train Parameters

(160 bytes).

Offset	Header Entry Name	Type	Description
2136	lEpochPulsePeriod(0-1)(0-9)	long	Train period in physical DAC channel order then epoch order (samples).
2216	lEpochPulseWidth(0-1)(0-9)	long	Train pulse width in physical DAC channel order then epoch order (samples).

Train_Parameters

$ Train Parameters

K Train Parameters

lEpochPulsePeriod;lEpochPulseWidth;

+ ABFFUNC

$ K + Extended Epoch Waveform and Pulses

(412 bytes).

Offset	Header Entry Name	Type	Description
2296	nWaveformEnable(0-1)	short	Analog waveform output status for each physical DAC channel: 0 = Disabled, 1 = Enabled.
2300	nWaveformSource(0-1)	short	Analog waveform source in physical DAC channel order: 0 = Disable; 1 = Generate waveform from epoch definitions; 2 = Generate waveform from a DAC file.
2304	nInterEpisodeLevel(0-1)	short	Inter-sweep holding level in physical DAC channel order: 0 = Use holding level; 1 = Use last epoch amplitude.
2308	nEpochType(0-1)(0-9)	short	Epoch type in physical DAC channel order then epoch order: 0 = Disabled; 1 = Step; 2 = Ramp.
2348	FEpochInitLevel(0-1)(0-9)	float	Epoch initial level in physical DAC channel order then epoch order (user units).
2428	fEpochLevelInc(0-1)(0-9)	float	Epoch level increment in physical DAC channel order then epoch order (user units).
2508	lEpochInitDuration(0-1)(0-9)	long	Epoch initial duration in physical DAC channel order then epoch order (in sequence counts).
2588	lEpochDurationInc(0-1)(0-9)	long	Epoch duration increment in physical DAC channel order then epoch order (in sequence counts)
2668	nDigitalTrainValue(0-9)	short	Epoch value for digital train output in epoch order. 0000 = Disabled; 0*000 = Generates digital train on bit 3. Train period and pulse width can be controlled by the user list. ASCII digital train pattern 0000 :0x00000000 000* :0x00000001 00*0 :0x00000002 0*00 :0x00000004 *000 :0x00000008
2688	nDigitalTrainActiveLogic	short	Digital train polarity. 0 = train starts with LOW pulse. 1 = trains starts with HIGH pulse.
2690	sUnused012	18char	Unused.

Extended_Epoch_Waveform_and_Pulses

$ Extended Epoch Waveform and Pulses

K Extended Epoch Waveform and Pulses

nWaveformEnable;nWaveformSource;nInterEpisodeLevel;nEpochType;fEpochInitLevel;fEpochLevelInc;lEpochInitDuration;lEpochDurationInc;nDigitalTrainValue;nDigitalTrainActiveLogic;

+ ABFFUNC

$ K + Extended DAC Output File

(552 bytes).

Offset	Header Entry Name	Type	Description
2708	fDACFileScale(0-1)	float	Scaling factor to apply to DAC file contents in physical DAC channel order.
2716	fDACFileOffset(0-1)	float	Offset (in user units) to apply to DAC file contents in physical DAC channel order..
2724	lDACFileEpisodeNum(0-1)	long	Sweep (or column) number to replay from waveform file in physical DAC channel order: -1 = all except the first (which is skipped), repeating last if necessary; 0 = all sweeps; N = sweep number.
2732	nDACFileADCNum(0-1)	short	Logical ADC channel number to replay from waveform file in physical DAC channel order.
2736	sDACFilePath(0-1)	256char	Drive and directory for sDACFileName in physical DAC channel order.
3248	sUnused013	12char	Unused.

Extended_DAC_Output File

$ Extended DAC Output File

K Extended DAC Output File

fDACFileScale;fDACFileOffset;lDACFileEpisodeNum;nDACFileADCNum;sDACFilePath;

+ ABFFUNC

$ K + Extended Pre-sweep (Conditioning) Pulse Train

(100 bytes).

Offset	Header Entry Name	Type	Description
3260	nConditEnable(0-1)	short	Pre-sweep pulse train activation status in physical DAC channel order: 0 = Disable; 1 = Enable.
3264	lConditNumPulses(0-1)	long	Number of pre-sweep train pulses in physical DAC channel order.
3272	fBaselineDuration(0-1)	float	A single pulse in the pre-sweep train consists of a baseline followed by a step. This parameter is the baseline duration in ms in physical DAC channel order.
3280	fBaselineLevel(0-1)	float	Baseline level in physical DAC channel order (user units).
3288	fStepDuration(0-1)	float	Step duration in physical DAC channel order (ms).
3296	fStepLevel(0-1)	float	Step level in physical DAC channel order (user units).
3304	fPostTrainPeriod(0-1)	float	At the end of the conditioning train there is a post-train steady-state output. This parameter is the post-train duration in ms in physical DAC channel order.
3312	fPostTrainLevel(0-1)	float	Post-train level in physical DAC channel order (user units).
3320	sUnused014	40char	Unused.

[#] Extended_Presweep_(Conditioning)_Pulse_Train

[$] Extended Presweep (Conditioning) Pulse Train

[K] Extended Presweep (Conditioning) Pulse Train;
fDACFileScale;fDACFileOffset;lDACFileEpisodeNum;nDACFileADCNum;sDACFilePath;

[+] ABFFUNC

$ K + Extended Variable Parameter User List

(1096 bytes).

Offset	Header Entry Name	Type	Description
3360	nULEnable(0-3)	short	Parameter list activation status in physical DAC channel order: 0 = Disable; 1 = Enable.
3368	nULParamToVary(0-3)	short	Holds the index of the parameter that varies from sweep to sweep in one run in physical DAC channel order.
			CONDITNUMPULSES 0
			CONDITBASELINEDURATION 1
			CONDITBASELINELEVEL 2
			CONDITSTEPDURATION 3
			CONDITSTEPLEVEL 4
			CONDITPOSTTRAINDURATION 5
			CONDITPOSTTRAINLEVEL 6
			EPISODESTARTTOSTART 7
			INACTIVEHOLDING 8
			DIGITALINTEREPISODE 9
			PNNUMPULSES 10
			PARALLELVALUE(0-9) 11-20
			EPOCHINITLEVEL(0-9) 21-30
			EPOCHINITDURATION(0-9) 31-40
			EPOCHTRAINPERIOD(0-9) 41-50
			EPOCHTRAINPULSEWIDTH(0-9) 51-60
3376	sULParamValueList(0-3)	256char	List of comma-separated values in physical DAC channel oder. If the number of entries in the list is fewer than the requested number of sweeps , the last list value is re-used but only if nULRepeat = 0. If there are more values in the list, the excess list values are ignored.
4400	nULRepeat(0-3)	short	Repeat the list when the current sweep exceeds the number of entries in the list. 0 = Disable, 1 = Repeat the list.
4408	sUnused015	48char	Unused.

[#] Extended_Variable_Parameter_User_List

[$] Extended Variable Parameter User List

[K] Extended Variable Parameter User List

nULEnable;nULParamToVary;sULParamValueList;nULRepeat;

[+] ABFFUNC

[#] [$] [K] [+] Extended On-Line Subtraction

(56 bytes).

Offset	Header Entry Name	Type	Description
4456	nPNEnable(0-1)	short	P/N activation status in physical DAC channel order: 0 = Disable; 1 = Enable
4460	nPNPolarity(0-1)	short	Polarity of P/N waveform in physical DAC channel order: -1 = opposite polarity to the waveform; 1 = same polarity as the waveform.
4464	nPNADCNum(0-1)	short	Physical ADC channel number for P/N subtraction in physical DAC channel order.
4468	fPNHoldingLevel(0-1)	float	Subpulse holding level in physical DAC channel order.
4476	sUnused016	36char	Unused.

[#] Extended_On_Line_Subtraction

[$] Extended On-Line Subtraction

[K] Extended On-Line Subtraction

nPNEnable;nPNPolarity;nPNADCNum;fPNHoldingLevel;

[+] ABFFUNC

[#] [$] [K] [+] Extended Environmental Information

(898 bytes).

Offset	Header Entry Name	Type	Description
4512	nTelegraphEnable(0-15)	short	Enable storage of telegraph information in physical ADC channel order: 0 = Disabled; 1 = Automatic; 2 = Manual.
4544	nTelegraphInstrument(0-15)	short	Telegraphed instrument in physical ADC channel order. (IHS-1 telegraphs are not supported.) Note, for most programs this is an information-only field. For example, in Clampex the telegraphed instrument is chosen as a configuration item and copied into this field.

Unknown instrument	0
Axopatch-1 with CV-4-1/100	1
Axopatch-1 with CV-4-0.1/100	2
Axopatch-1B(inv.) CV-4-1/100	3
Axopatch-1B(inv) CV-4-0.1/100	4
Axopatch 200 with CV 201	5
Axopatch 200 with CV 202	6
GeneClamp	7
Dagan 3900	8
Dagan 3900A	9
Dagan CA-1 Im=0.1	10
Dagan CA-1 Im=1.0	11
Dagan CA-1 Im=10	12
Warner OC-725	13
Warner OC-725C	14
AxoPatch 200B	15
Dagan PC-ONE Im=0.1	16
Dagan PC-ONE Im=1.0	17
Dagan PC-ONE Im=10	18
Dagan PC-ONE Im=100	19
Warner BC-525C	20
Warner PC-505	21
Warner PC-501	22
Dagan CA-1 Im=0.05	23

[#] Extended_Environmental_Information

[$] Extended Environmental Information

[K] Extended Environmental Information

nTelegraphEnable;nTelegraphInstrument;fTelegraphAdditGain;fTelegraphFilter;fTelegraphMembraneCap;nTelegraphMode;nManualTelegraphStrategy;nAutoAnalyseEnable;sAutoAnalysisMacroName;sProtocolPath;sFileComment;

[+] ABFFUNC

			MultiClamp 700 24 Turbo Tec 25 OpusXpress 6000 26
4576	fTelegraphAdditGain(0-15)	float	Additional gain multiplier of telegraphed instrument in physical ADC channel order. (Optionally autosampled by some acquisition programs.) (Default = 1)
4640	fTelegraphFilter(0-15)	float	Lowpass filter cutoff frequency of telegraphed instrument in ADC physical channel order. (Optionally autosampled by some acquisition programs.) (Default = 100000)
4704	fTelegraphMembraneCap(0-15)	float	Patch-clamp membrane capacitance compensation of telegraphed instrument in physical ADC channel order. (Optionally autosampled by some acquisition programs.)
4768	nTelegraphMode(0-15)	short	Telegraph mode in physical ADC channel order: 1 = Voltage Clamp; 2 = Current Clamp; 4 = Current Clamp Zero (Automatically set by MultiClamp instrument)
4800	nTelegraphDACScaleFactor Enable(0-3)	short	Determines whether fDACScaleFactor was telegraphed: 1 = telegraphed; 0 = not telegraphed
4808	sUnused016a	24char	Unused.
4832	nAutoAnalyseEnable	short	Discontinued.
4834	sAutoAnalysisMacroName	64char	Discontinued.
4898	sProtocolPath	256char	The name and path of the protocol
5154	sFileComment	128char	128 byte ASCII comment string.
5282	sUnused017	128char	Unused.

[#] [$] [K] [+] Extended Statistics Measurements

(388 bytes).

Offset	Header Entry Name	Type	Description
5410	nStatsEnable	short	Statistics activation status: 0 = Disabled; 1 = Enabled.
5412	nStatsActiveChannels	unsigned short	Bit flag for selecting statistics in each physical ADC channel. Channel 0: 0x0001 Channel 1: 0x0002 Channel 2: 0x0004 Channel 3: 0x0008 Channel 4: 0x0010 Channel 5: 0x0020 Channel 6: 0x0040 Channel 7: 0x0080 Channel 8: 0x0100 Channel 9: 0x0200 Channel 10: 0x0400 Channel 11: 0x0800 Channel 12: 0x1000 Channel 13: 0x2000 Channel 14: 0x4000 Channel 15: 0x8000 All channels: 0xFFFF
5414	nStatsSearchRegionFlags	unsigned short	Bit flag for selecting statistics in each region. Region 0: 0x0001 Region 1: 0x0002 Region 2: 0x0004 Region 3: 0x0008 Region 4: 0x0010 Region 5: 0x0020 Region 6: 0x0040 Region 7: 0x0080

[#] Extended_Statistics_Measurements

[$] Extended Statistics Measurements

[K] Extended Statistics Measurements

nStatsEnable;nStatsActiveChannels;nStatsSearchRegionFlags;nStatsSelectedRegion;_nStatsSearchMode;nStatsSmoothing;nStatsSmoothingEnable;nStatsBaseline;lStatsBaselineStart;lStatsBaselineEnd;lStatsMeasurements;lStatsStart;lStatsEnd;nRiseBottomPercentile;RiseTopPercentile;nDecayBottomPercentile;nDecayTopPercentile;nStatsChannelPolarity;nStatsSearchMode;

[+] ABFFUNC

			All regions: 0x00FF
5416	nStatsSelectedRegion	short	The currently active statistics region: 0 = region 0; 1 = region 1 ... 7 = region 7.
5418	_nStatsSearchMode	short	Not used
5420	nStatsSmoothing	short	Number of samples averaged in boxcar smoothing window when looking for a peak.
5422	nStatsSmoothingEnable	short	Not used
5422	nStatsBaseline	short	Baseline for statistics measurements: -2 = None; -1 = Average the first (lNumSamplesPerEpisode/64) samples; 0-9 = epoch A-J
5426	lStatsBaselineStart	long	Start of baseline used in statistics measurements (samples).
5430	lStatsBaselineEnd	long	End of baseline used in statistics measurements (samples)
5434	lStatsMeasurements(0-7)	long	Bit mask indicating which statistics measurements to perform in region order.
			Peak: 0x00000001
			Peak time: 0x00000002
			Antipeak: 0x00000004
			Antipeak Time: 0x00000008
			Mean: 0x00000010
			Standard Deviation: 0x00000020
			Integral: 0x00000040
			Max Rise Slope: 0x00000080
			Max Rise Slope Time: 0x00000100
			Max Decay Slope: 0x00000200
			Max Decay Slope Time: 0x00000400
			Rise Time: 0x00000800
			Decay Time: 0x00001000
			Half Width: 0x00002000
			Baseline: 0x00004000
			Rise Slope: 0x00008000
			Decay Slope: 0x00010000
			Region Slope: 0x00020000
			All Measurements: 0x0002FFFF
5466	lStatsStart(0-7)	long	Start of specified statistics region in region order (Samples). Only valid if corresponding value in nStatsSearchMode = -2.
5498	lStatsEnd(0-7)	long	End of specified statistics region in region order (Samples). Only valid if corresponding value in nStatsSearchMode = -2.
5530	nRiseBottomPercentile(0-7)	short	Nominated percentage of the peak amplitude (relative to the baseline) that determines the left most point of region for measuring rise slope parameters: 0 = at baseline; 100 = at peak.

5546	nRiseTopPercentile(0-7)	short	Nominated percentage of the peak amplitude (relative to the baseline) that determines the right most point of region for measuring rise slope parameters. 0 = at baseline; 100 = at peak.
5562	nDecayBottomPercentile(0-7)	short	Nominated percentage of the peak amplitude (relative to the baseline) that determines the right most point of region for measuring decay slope parameters. 0 = at baseline; 100 = at peak.
5578	nDecayTopPercentile(0-7)	short	Nominated percentage of the peak amplitude (relative to the baseline) that determines the left most point of region for measuring decay slope parameters. 0 = at baseline; 100 = at peak.
5594	nStatsChannelPolarity(0-15)	short	Polarity of peaks to find in physical ADC channel order. -1 = search for negative peaks; 0 = search for absolute peak; 1 = search for positive peaks.
5626	nStatsSearchMode(0-7)	short	Search mode in region order: 0-9 = epoch A-J; -1 = All; -2 = Use specified region.
5642	sUnused018	156char	Unused.

[#] [$] [K] [+] Application Version Data

(16 bytes).

Offset	Header Entry Name	Type	Description
5798	nMajorVersion	short	Major version of application: **x**.0.0.0
5800	nMinorVersion	short	Minor version of application: 0.**x**.0.0
5802	nBugfixVersion	short	Bug fix version of application: 0.0.**x**.0
5804	nBuildVersion	short	Build version of application: 0.0.0.**x**
5806	sUnused019	8char	Unused.

[#] Application_Version_Data

[$] Application Version Data

[K] Application Version Data

nMajorVersion;nMinorVersion;nBugfixVersion;nBuildVersion;

[+] ABFFUNC

[#] [$] [K] [+] LTP Protocol

(14 bytes).

Offset	Header Entry Name	Type	Description
5814	nLTPType	char	Type of LTP protocol: 0 = None; 1 = Baseline; 2 = Induction/Conditioning
5816	nLTPUsageOfDAC(0-1)	short	Indicates how specified DAC is used: 0 = Presynaptic stimulus; Postsynaptic stimulus.
5820	nLTPPresynapticPulses(0-1)	short	The number pulses in presynaptic pulse train.
5824	sUnused020	4char	Unused.

[#] LTP_Protocol

[$] LTP Protocol

[K] LTP Protocol

nLTPType;nLTPUsageOfDAC;nLTPPresynapticPulses;

[+] ABFFUNC

[#] [$] [K] [+] Digidata 132x Trigger Out

(8 bytes).

Offset	Header Entry Name	Type	Description
5828	nDD132xTriggerOut	short	The trigger enable for DD132x series digitiziers
5830	sUnused021	6char	Unused.

[#] Digidata_132x_Trigger_Out

[$] Digidata 132x Trigger Out

[K] Digidata 132x Trigger Out nDD132xTriggerOut;

[+] ABFFUNC

[#] [$] [K] [+] Epoch Resistance

(40 bytes).

Offset	Header Entry Name	Type	Description
5836	sEpochResistanceSignalName(0-1)	10char	Name of the signal on which to measure epoch resistance in physical DAC channel order.
5856	nEpochResistanceState(0-1)	short	Epoch resistance status in physical DAC channel order: 0 = Disabled; 1 = Enabled.
5860	sUnused022	16char	Unused.

[#] Epoch_Resistance

[$] Epoch Resistance

[K] Epoch Resistance

sEpochResistanceSignalName;nEpochResistanceState;

[+] ABFFUNC

[#] [$] [K] [+] Alternating Episodic Mode

(58 bytes).

Offset	Header Entry Name	Type	Description
5876	nAlternateDACOutputState	short	Alternating DAC output status: 0 = Disabled; 1 = Enabled. For the first sweep DAC 0 is set by waveform 0 and DAC 1 is set to holding level. For the second sweep DAC 1 is set by waveform 1 and DAC 0 is set to holding level etc.
5878	nAlternateDigitalValue(0-9)	short	Digital value held by the alternate or non-active digital output: The active digital output is set by nActiveDACChannel.
5898	nAlternateDigitalTrainValue(0-9)	short	Digital train value held by the alternate or non-active digital output: The active digital output is set by nActiveDACChannel.
5918	nAlternateDigitalOutputState	short	Alternating digital output status: 0 = Disabled; 1 = Enabled. . For the first sweep the digital pattern is set by waveform 0. For the second sweep the digital pattern is set by waveform 1 etc.
5920	sUnused023	14char	Unused.

[#] Alternating_Episodic_Mode

[$] Alternating Episodic Mode

[K] Alternating Episodic Mode

nAlternateDACOutputState;nAlternateDigitalValue;nAlternateDigitalTrainValue;nAlternateDigitalOutputState;

[+] ABFFUNC

[#] [$] [K] [+] Post-Processing Actions

(210 bytes).

Offset	Header Entry Name	Type	Description	
5934	fPostProcessLowpassFilter(0-15)	float	Post processing lowpass filter cutoff for single io channel data (Hz)	
5998	nPostProcessLowpassFilterType(0-15)	char	Post processing lowpass filter type	
			None	0
			Combination	1
			Bessel	2
			Boxcar	3
			Butterworth	4
			Chebyshev	5
			Gaussian	6
			Rc	7
			Rc8	8
			Adaptive	9
6014	sUnused2048	130char	Unused.	

TOTAL 6144 bytes

[#] Post_Processing_Actions

[$] Post-Processing Actions

[K] Post-Processing Actions

fPostProcessLowpassFilter;nPostProcessLowpassFilterType;

[+] ABFFUNC

$ K + The ABF Scope Config Section

If present, the ABF Scope Config section will contain one or more ABFScopeConfig structures describing the attributes of the scope windows used for data display during the data acquisition.

The ABFScopeConfig Structure

Each ABFScopeConfig structure contains configuration information that describes the setup of a scope window. This structure in turn contains a structure (ABFLogFont) to define the font properties used to draw textual items such as tic labels, and an array of structures (of type ABFSignal), one for each ADC channel being acquired.

Offset	Header Entry Name	Type	Description
0	dwFlags	DWORD	Flags that are meaningful to the scope.
4	rgbColor[ABF_SCOPECOLORS]	DWORD	Colors for the components of the scope.
44	fDisplayStart	float	Start of the display area in ms.
48	fDisplayEnd	float	End of the display area in ms.
52	wScopeMode	WORD	The display mode: 0=sweeps; 1=continuous.
54	bMaximized	char	TRUE = Scope parent is maximized.
55	bMinimized	char	TRUE = Scope parent is minimized.
56	xLeft	short	Coordinate of the left edge.
58	yTop	short	Coordinate of the top edge.
60	xRight	short	Coordinate of the right edge.
62	yBottom	short	Coordinate of the bottom edge.
64	LogFont	ABFLogFont	Description of current font.
104	TraceList[ABF_ADCCOUNT]	ABFSignal	List of traces in current use - see nADCNumChannels in the ABFFileHeader for the number of channels actually used.
648	nYAxisWidth	short	Width of the YAxis region.
650	nTraceCount	short	Number of traces described in TraceList. (this should always match nADCNumChannels in the ABFFileHeader)
652	nEraseStrategy	short	Erase strategy: 0=Erase before each sweep; 1=Erase before each run; 2=Erase before each trial; 3=Do not erase
654	nDockState	short	Docking state: 0=Not docked; 1=Top; 2=Left; 3=Right; 4=Bottom.

TOTAL 656 bytes

The ABFLogFont Structure

The ABFLogFont structure is a subset of the Windows LogFont structure, containing information to describe the characteristics of the font to be used in the Scope window.

The_ABF_Scope_Config_Section

$ The ABF Scope Config Section

K The ABF Scope Config Section;ABFLogFont;ABFSignal;ABFScopeConfig

+ ABFFUNC

Offset	Header Entry Name	Type	Description
0	nHeight	short	N.B. Height of the font in *points*.
2	nWeight	short	MSWindows font weight value.
4	cPitchAndFamily	char	MSWindows pitch and family mask.
5	Unused[3]	3char	Unused space to maintain 4-byte packing.
8	szFaceName[ABF_FACESIZE]	char	Face name of the font.

TOTAL 40 bytes

The ABFSignal Structure

The ABFSignal structure describes the characteristics of a single data trace on the screen, corresponding to a particular acquired signal.

Offset	Header Entry Name	Type	Description
0	szName[ABF_ADCNAMELEN+2]	char	ABF name length + '\0' + 1 for alignment.
12	nMxOffset	short	Offset of the signal in the sampling sequence.
14	RgbColor	DWORD	Pen color used to draw trace.
18	nPenWidth	char	Pen width in pixels.
19	bDrawPoints	char	TRUE = Draw disconnected points.
20	bHidden	char	TRUE = Hide the trace.
21	bFloatData	char	TRUE = Floating point pseudo channel.
22	fVertProportion	float	Relative proportion of the client are to use.
26	fDisplayGain	char	Display gain of trace in UserUnits.
30	fDisplayOffset	char	Display offset of trace in UserUnits.

TOTAL 34 bytes

$ K + The ABF Data Section

Acquired data samples are stored as multiplexed two-byte binary integers. A special four-byte floating point version is used by some analysis programs for storage of analysis results (see nDataFormat in the file header section). If the data file contains multiple sweeps or segments of data, these are stored end for end without a gap. That is, no parameters are stored between sweeps of the data; all parameters are stored in the header or in the specialized sections described below. There is only one data section.

[#] The_ABF_Data_Section

[$] The ABF Data Section

[K] The ABF Data Section

[+] ABFFUNC

$ K + The ABF Synch Section

The ABF Synch array is an important array that stores the start time and length of each portion of the data if the data are not part of a continuous gap-free acquisition. The data section might contain equal length or variable length sweeps of data. The Synch Array contains a record to indicate the start time and length of every sweep or Event in the data file. The ABF reading routines automatically decode the Synch Array when providing information about the data.

A Synch array is created and used in the following acquisition modes: ABF_VARLENEVENTS, ABF_FIXLENEVENTS & ABF_HIGHSPEEDOSC. The acquisition modes ABF_GAPFREEFILE and ABF_WAVEFORMFILE do not always use a Synch array.

The ABFSynch Structure

(8 bytes).

Offset	Header Entry Name	Type	Description
0	lStart	long	Start time of sweep in fSynchTimeUnit units.
4	lLength	long	Length of the sweep in multiplexed samples.

The_ABF_Synch_Section

$ The ABF Synch Section

K The ABF Synch Section;ABFSynch

+ ABFFUNC

$ K + The ABF Tag Section

During an acquisition, some programs allow the user to tag points of interest in the input data stream. These tags are saved in the Tag Section. Each tag consists of a time stamp, a text comment, and a tag type identifier. If the tag is a voice tag, the data is held in an ABFVoiceTagInfo struct.

The ABFTag Structure

(64 bytes).

Offset	Header Entry Name	Type	Description
0	lTagTime	long	Time at which the tag was entered in fSynchTimeUnit units.
4	sComment	56char	Optional comment to describe the tag.
60	nTagType	short	Type of tag. Valid types are ABF_TIMETAG=0, ABF_COMMENTTAG=1, ABF_EXTERNALTAG=2, ABF_VOICETAG=3
62	nVoiceTagNumber	short	If nTagType=ABF_VOICETAG, this is the number of this voice tag.

The ABFVoiceTagInfo structure

(32 bytes).

Offset	Header Entry Name	Type	Description
0	lTagNumber	long	The tag number that corresponds to this VoiceTag
4	lFileOffset	long	Offset to this tag within the VoiceTag block
8	lUncompressedSize	long	Size of the voice tag expanded.
12	lCompressedSize	long	Compressed size of the tag.
16	nCompressionType	short	Compression method used.
18	nSampleSize	short	Size of the samples acquired.
20	lSamplesPerSecond	long	Rate at which the sound was acquired.
24	dwCRC	DWORD	CRC used to check data integrity.
28	wChannels	WORD	Number of channels in the tag (usually 1).
30	wUnused	WORD	Unused space.

The_ABF_Tag_Section

$ The ABF Tag Section

K The ABF Tag Section;ABFTag

+ ABFFUNC

$ K + The ABF Deltas Section

When acquisition parameters are changed during an acquisition, the changes are tracked and entered in the ABF deltas section. Each entry is time stamped in fSynchTimeUnit units, so that the value of the parameter can be determined at any point during the acquisition.

The ABFDelta Structure

(64 bytes).

Offset	Header Entry Name	Type	Description
0	lDeltaTime	long	Time at which the parameter was changed in fSynchTimeUnit units.
4	lParameterID	long	Identifier for the parameter changed. Legal parameter values are: ABF_DELTA_XXXXXXXX
8	lNewParamValue fNewParamValue	long float	Depending on the value of lParameterID this entry may be either a float or a long.

The_ABF_Deltas_Section

$ The ABF Deltas Section

K The ABF Deltas Section;ABFDelta

+ ABFFUNC

$ K + The DAC Data Section

In some experiments, instead of the analog output stimulation waveform being described parametrically in a table it is described on a sample-by-sample basis from a file. In experiments where files are used to describe the analog output stimulation, a copy of the stimulation file is attached in the DAC Data section to ensure that the stimulus waveform is always available during analysis. The ABF routines allow reading of the DAC data with separate functions similar to those of ABF_ReadChannel (i.e. called ABF_ReadDACFileEpi).

The_DAC_Data_Section

$ The DAC Data Section

K The DAC Data Section

+ ABFFUNC

$ K + ABFHEADR.H

```
//***************************************************************************************************
//
//    Copyright (c) 1993-2003 Axon Instruments.
//    All rights reserved.
//    Permission is granted to freely use, modify and copy the code in this file.
//
//***************************************************************************************************
// HEADER:  ABFHEADR.H.
// PURPOSE: Defines the ABFFileHeader structure, and provides prototypes for
//          functions implemented in ABFHEADR.CPP for reading and writing
//          ABFFileHeader's.
// REVISIONS:
//   1.1  - Version 1.1 was released in April 1992.
//   1.2  - Added nDataFormat so that data can optionally be stored in floating point format.
//        - Added lClockChange to control the multiplexed ADC sample number after which the second
sampling interval commences.
//   1.3  - Change 4-byte sFileType string to long lFileSignature.
//        - #define ABF_NATIVESIGNATURE & ABF_REVERSESIGNATURE for byte order detection.
//        - Added support for Bells during before or after acquisitions
//        - Added parameters to describe hysteresis during event detected acquisitions:
nLevelHysteresis and lTimeHysteresis.
//        - Dropped support for BASIC and Pascal.
//        - Added the ABF Scope Config section to store scope configuration information
//   1.4  - Remove support for big-endian machines.
//   1.5  - Change ABFSignal parameters from UUTop & UUBottom to
//          fDisplayGain & fDisplayOffset.
//        - Added and changed parameters in the 'File Structure', 'Display Parameters',
//          'DAC Output File', 'Autopeak Measurements' and 'Unused space and end of header' sections
of the ABF file header.
//        - Expanded the ABF API and error return codes
//   1.6  - Expanded header to 5120 bytes and added extra parameters to support 2 waveform channels
PRC
//   1.65 - Telegraph support added.
//   1.67 - Train epochs, multiple channel and multiple region stats
//   1.68 - ABFScopeConfig expanded
//   1.69 - Added user entered percentile levels for rise and decay stats
//   1.70 - Added data reduction  - AjD
//   1.71 - Added epoch resistance
//   1.72 - Added alternating outputs
//   1.73 - Added post-processing lowpass filter settings.  When filtering is done in Clampfit it is
stored in the header.
//   1.74 - Added channel_count_acquired
//   1.75 - Added polarity for each channel
//   1.76 - Added digital trigger out flag
//   1.77 - Added major, minor and bugfix version numbers
//   1.78 - Added separate entries for alternating DAC and digital outputs
//   1.79 - Removed data reduction (now minidigi only)
//   1.80 - Added stats mode for each region: mode is cursor region, epoch etc

#ifndef INC_ABFHEADR_H
#define INC_ABFHEADR_H

#include "AxABFFIO32.h"

#ifdef __cplusplus
extern "C" {
#endif

//
// Constants used in defining the ABF file header
//

#define ABF_ADCCOUNT           16    // number of ADC channels supported.
#define ABF_DACCOUNT           4     // number of DAC channels supported.
#define ABF_WAVEFORMCOUNT      2     // number of DAC channels which support waveforms.
#define ABF_EPOCHCOUNT         10    // number of waveform epochs supported.
```

ABFHEADR_H

$ ABFHEADR.H

K ABFHEADR.H

+ ABFFUNC

```
#define ABF_BELLCOUNT          2     // Number of auditory signals supported.
#define ABF_ADCUNITLEN         8     // length of ADC units strings
#define ABF_ADCNAMELEN         10    // length of ADC channel name strings
#define ABF_DACUNITLEN         8     // length of DAC units strings
#define ABF_DACNAMELEN         10    // length of DAC channel name strings
#define ABF_VARPARAMLISTLEN    80    // length of conditioning string
#define ABF_USERLISTLEN        256   // length of the user list (V1.6)
#define ABF_USERLISTCOUNT      4     // number of independent user lists (V1.6)
#define ABF_OLDFILECOMMENTLEN  56    // length of file comment string (pre V1.6)
#define ABF_FILECOMMENTLEN     128   // length of file comment string (V1.6)

#define ABF_CREATORINFOLEN     16    // length of file creator info string
#define ABF_OLDDACFILENAMELEN  12    // old length of the DACFile name string
#define ABF_OLDDACFILEPATHLEN  60    // old length of the DACFile path string
#define ABF_DACFILEPATHLEN     84    // length of full path for DACFile
#define ABF_PATHLEN            256   // length of full path, used for DACFile and Protocol name.
#define ABF_ARITHMETICOPLEN    2     // length of the Arithmetic operator field
#define ABF_ARITHMETICUNITSLEN 8     // length of arithmetic units string
#define ABF_TAGCOMMENTLEN      56    // length of tag comment string
#define ABF_LONGDESCRIPTIONLEN 56    // length of long description entry
#define ABF_NOTENAMELEN        10    // length of the name component of a note
#define ABF_NOTEVALUELEN       8     // length of the value component of a note
#define ABF_NOTEUNITSLEN       8     // length of the units component of a note
#define ABF_BLOCKSIZE          512   // Size of block alignment in ABF files.
#define ABF_MACRONAMELEN       64    // Size of a Clampfit macro name.

#define ABF_CURRENTVERSION     1.80F          // Current file format version number
#define ABF_PREVIOUSVERSION    1.5F           // Previous file format version number (for old
header size)
#define ABF_V16                1.6F           // Version number when the header size changed.
#define ABF_HEADERSIZE         6144           // Size of a Version 1.6 or later header
#define ABF_OLDHEADERSIZE      2048           // Size of a Version 1.5 or earlier header
#define ABF_NATIVESIGNATURE    0x20464241     // PC="ABF ", MAC=" FBA"
#define ABF_REVERSESIGNATURE   0x41424620     // PC=" FBA", MAC="ABF "

#define PCLAMP6_MAXSWEEPLENGTH         16384   // Maximum multiplexed sweep length supported by
pCLAMP6 apps.
#define PCLAMP7_MAXSWEEPLEN_PERCHAN    1032258  // Maximum per channel sweep length supported by
pCLAMP7 apps.

#define ABF_MAX_TRIAL_SAMPLES  0x7FFFFFFF    // Maximum length of acquisition supported (samples)
                                             // INT_MAX is used instead of UINT_MAX because of the
signed
                                             // values in the ABF header.

#define ABF_MAX_SWEEPS_PER_AVERAGE 65500     // The maximum number of sweeps that can be combined
into a
                                             // cumulative average
(nAverageAlgorithm=ABF_INFINITEAVERAGE).

#define ABF_STATS_REGIONS     8              // The number of independent statistics regions.
#define ABF_BASELINE_REGIONS  1              // The number of independent baseline regions.

#ifdef _MAC
   #define ABF_OLDPCLAMP        ABF_REVERSESIGNATURE
#else
   #define ABF_OLDPCLAMP        ABF_NATIVESIGNATURE
#endif

//
// Constant definitions for nFileType
//
#define ABF_ABFFILE          1
#define ABF_FETCHEX          2
#define ABF_CLAMPEX          3

//
// Constant definitions for nDataFormat
//
#define ABF_INTEGERDATA      0
#define ABF_FLOATDATA        1

//
// Constant definitions for nOperationMode
//
#define ABF_VARLENEVENTS     1
#define ABF_FIXLENEVENTS     2     // (ABF_FIXLENEVENTS == ABF_LOSSFREEOSC)
#define ABF_LOSSFREEOSC      2
#define ABF_GAPFREEFILE      3
```

```
#define ABF_HIGHSPEEDOSC     4
#define ABF_WAVEFORMFILE     5

//
// Constant definitions for nParamToVary
//
#define ABF_CONDITNUMPULSES         0
#define ABF_CONDITBASELINEDURATION  1
#define ABF_CONDITBASELINELEVEL     2
#define ABF_CONDITSTEPDURATION      3
#define ABF_CONDITSTEPLEVEL         4
#define ABF_CONDITPOSTTRAINDURATION 5
#define ABF_CONDITPOSTTRAINLEVEL    6
#define ABF_EPISODESTARTTOSTART     7
#define ABF_INACTIVEHOLDING         8
#define ABF_DIGITALHOLDING          9
#define ABF_PNNUMPULSES             10
#define ABF_PARALLELVALUE           11
#define ABF_EPOCHINITLEVEL          (ABF_PARALLELVALUE + ABF_EPOCHCOUNT)
#define ABF_EPOCHINITDURATION       (ABF_EPOCHINITLEVEL + ABF_EPOCHCOUNT)
#define ABF_EPOCHTRAINPERIOD        (ABF_EPOCHINITDURATION + ABF_EPOCHCOUNT)
#define ABF_EPOCHTRAINPULSEWIDTH    (ABF_EPOCHTRAINPERIOD + ABF_EPOCHCOUNT)
// Next value is (ABF_EPOCHINITDURATION + ABF_EPOCHCOUNT)

//
// Constants for nAveragingMode
//
#define ABF_NOAVERAGING       0
#define ABF_SAVEAVERAGEONLY   1
#define ABF_AVERAGESAVEALL    2

//
// Constants for nAverageAlgorithm
//
#define ABF_INFINITEAVERAGE   0
#define ABF_SLIDINGAVERAGE    1

//
// Constants for nEpochType
//
#define ABF_EPOCHDISABLED           0     // disabled epoch
#define ABF_EPOCHSTEPPED            1     // stepped waveform
#define ABF_EPOCHRAMPED             2     // ramp waveform
#define ABF_EPOCH_TYPE_RECTANGLE    3     // rectangular pulse train
#define ABF_EPOCH_TYPE_TRIANGLE     4     // triangular waveform
#define ABF_EPOCH_TYPE_COSINE       5     // cosinusoidal waveform
#define ABF_EPOCH_TYPE_RESISTANCE   6     // resistance waveform
#define ABF_EPOCH_TYPE_BIPHASIC     7     // biphasic pulse train

//
// Constants for epoch resistance
//
#define ABF_MIN_EPOCH_RESISTANCE_DURATION 8

//
// Constants for nWaveformSource
//
#define ABF_WAVEFORMDISABLED     0              // disabled waveform
#define ABF_EPOCHTABLEWAVEFORM   1
#define ABF_DACFILEWAVEFORM      2

//
// Constants for nInterEpisodeLevel & nDigitalInterEpisode
//
#define ABF_INTEREPI_USEHOLDING    0
#define ABF_INTEREPI_USELASTEPOCH  1

//
// Constants for nExperimentType
//
#define ABF_VOLTAGECLAMP         0
#define ABF_CURRENTCLAMP         1
#define ABF_SIMPLEACQUISITION    2

//
// Constants for nAutosampleEnable
//
#define ABF_AUTOSAMPLEDISABLED   0
#define ABF_AUTOSAMPLEAUTOMATIC  1
```

```
#define ABF_AUTOSAMPLEMANUAL     2

//
// Constants for nAutosampleInstrument
//
#define ABF_INST_UNKNOWN         0   // Unknown instrument (manual or user defined telegraph table).
#define ABF_INST_AXOPATCH1       1   // Axopatch-1 with CV-4-1/100
#define ABF_INST_AXOPATCH1_1     2   // Axopatch-1 with CV-4-0.1/100
#define ABF_INST_AXOPATCH1B      3   // Axopatch-1B(inv.) CV-4-1/100
#define ABF_INST_AXOPATCH1B_1    4   // Axopatch-1B(inv) CV-4-0.1/100
#define ABF_INST_AXOPATCH201     5   // Axopatch 200 with CV 201
#define ABF_INST_AXOPATCH202     6   // Axopatch 200 with CV 202
#define ABF_INST_GENECLAMP       7   // GeneClamp
#define ABF_INST_DAGAN3900       8   // Dagan 3900
#define ABF_INST_DAGAN3900A      9   // Dagan 3900A
#define ABF_INST_DAGANCA1_1      10  // Dagan CA-1  Im=0.1
#define ABF_INST_DAGANCA1        11  // Dagan CA-1  Im=1.0
#define ABF_INST_DAGANCA10       12  // Dagan CA-1  Im=10
#define ABF_INST_WARNER_OC725    13  // Warner OC-725
#define ABF_INST_WARNER_OC725C   14  // Warner OC-725
#define ABF_INST_AXOPATCH200B    15  // Axopatch 200B
#define ABF_INST_DAGANPCONE0_1   16  // Dagan PC-ONE  Im=0.1
#define ABF_INST_DAGANPCONE1     17  // Dagan PC-ONE  Im=1.0
#define ABF_INST_DAGANPCONE10    18  // Dagan PC-ONE  Im=10
#define ABF_INST_DAGANPCONE100   19  // Dagan PC-ONE  Im=100
#define ABF_INST_WARNER_BC525C   20  // Warner BC-525C
#define ABF_INST_WARNER_PC505    21  // Warner PC-505
#define ABF_INST_WARNER_PC501    22  // Warner PC-501
#define ABF_INST_DAGANCA1_05     23  // Dagan CA-1  Im=0.05
#define ABF_INST_MULTICLAMP700   24  // MultiClamp 700
#define ABF_INST_TURBO_TEC       25  // Turbo Tec
#define ABF_INST_OPUSXPRESS6000  26  // OpusXpress 6000A

//
// Constants for nManualInfoStrategy
//
#define ABF_ENV_DONOTWRITE      0
#define ABF_ENV_WRITEEACHTRIAL  1
#define ABF_ENV_PROMPTEACHTRIAL 2

//
// Constants for nTriggerSource
//
#define ABF_TRIGGERLINEINPUT          -5   // Start on line trigger (DD1320 only)
#define ABF_TRIGGERTAGINPUT           -4
#define ABF_TRIGGERFIRSTCHANNEL       -3
#define ABF_TRIGGEREXTERNAL           -2
#define ABF_TRIGGERSPACEBAR           -1
// >=0 = ADC channel to trigger off.

//
// Constants for nTrialTriggerSource
//
#define ABF_TRIALTRIGGER_SWSTARTONLY   -6   // Start on software message, end when protocol ends.
#define ABF_TRIALTRIGGER_SWSTARTSTOP   -5   // Start and end on software messages.
#define ABF_TRIALTRIGGER_LINEINPUT     -4   // Start on line trigger (DD1320 only)
#define ABF_TRIALTRIGGER_SPACEBAR      -3   // Start on spacebar press.
#define ABF_TRIALTRIGGER_EXTERNAL      -2   // Start on external trigger high
#define ABF_TRIALTRIGGER_NONE          -1   // Start immediately (default).
// >=0 = ADC channel to trigger off.    // Not implemented as yet...

//
// Constants for nTriggerPolarity.
//
#define ABF_TRIGGER_RISINGEDGE  0
#define ABF_TRIGGER_FALLINGEDGE 1

//
// Constants for nTriggerAction
//
#define ABF_TRIGGER_STARTEPISODE 0
#define ABF_TRIGGER_STARTRUN     1
#define ABF_TRIGGER_STARTTRIAL   2    // N.B. Discontinued in favor of nTrialTriggerSource

//
// Constants for nDrawingStrategy
//
#define ABF_DRAW_NONE           0
#define ABF_DRAW_REALTIME       1
```

```
#define ABF_DRAW_FULLSCREEN      2
#define ABF_DRAW_ENDOFRUN        3

//
// Constants for nTiledDisplay
//
#define ABF_DISPLAY_SUPERIMPOSED 0
#define ABF_DISPLAY_TILED        1

//
// Constants for nDataDisplayMode
//
#define ABF_DRAW_POINTS       0
#define ABF_DRAW_LINES        1

//
// Constants for nArithmeticExpression
//
#define ABF_SIMPLE_EXPRESSION    0
#define ABF_RATIO_EXPRESSION     1

//
// Constants for nLowpassFilterType & nHighpassFilterType
//
#define ABF_FILTER_NONE          0
#define ABF_FILTER_EXTERNAL      1
#define ABF_FILTER_SIMPLE_RC     2
#define ABF_FILTER_BESSEL        3
#define ABF_FILTER_BUTTERWORTH   4

//
// Constants for nPNPosition
//
#define ABF_PN_BEFORE_EPISODE    0
#define ABF_PN_AFTER_EPISODE     1

//
// Constants for nPNPolarity
//
#define ABF_PN_OPPOSITE_POLARITY -1
#define ABF_PN_SAME_POLARITY     1

//
// Constants for nAutopeakPolarity
//
#define ABF_PEAK_NEGATIVE       -1
#define ABF_PEAK_ABSOLUTE        0
#define ABF_PEAK_POSITIVE        1

//
// Constants for nAutopeakSearchMode
//
#define ABF_PEAK_SEARCH_SPECIFIED       -2
#define ABF_PEAK_SEARCH_ALL             -1
// nAutopeakSearchMode 0..9   = epoch in waveform 0's epoch table
// nAutopeakSearchMode 10..19 = epoch in waveform 1's epoch table

//
// Constants for nAutopeakBaseline
//
#define ABF_PEAK_BASELINE_SPECIFIED    -3
#define ABF_PEAK_BASELINE_NONE             -2
#define ABF_PEAK_BASELINE_FIRSTHOLDING -1
#define ABF_PEAK_BASELINE_LASTHOLDING  -4

//
// Constants for lAutopeakMeasurements
//
#define ABF_PEAK_MEASURE_PEAK               0x00000001
#define ABF_PEAK_MEASURE_PEAKTIME           0x00000002
#define ABF_PEAK_MEASURE_ANTIPEAK           0x00000004
#define ABF_PEAK_MEASURE_ANTIPEAKTIME       0x00000008
#define ABF_PEAK_MEASURE_MEAN               0x00000010
#define ABF_PEAK_MEASURE_STDDEV             0x00000020
#define ABF_PEAK_MEASURE_INTEGRAL           0x00000040
#define ABF_PEAK_MEASURE_MAXRISESLOPE       0x00000080
#define ABF_PEAK_MEASURE_MAXRISESLOPETIME   0x00000100
#define ABF_PEAK_MEASURE_MAXDECAYSLOPE      0x00000200
#define ABF_PEAK_MEASURE_MAXDECAYSLOPETIME  0x00000400
```

```
#define ABF_PEAK_MEASURE_RISETIME           0x00000800
#define ABF_PEAK_MEASURE_DECAYTIME          0x00001000
#define ABF_PEAK_MEASURE_HALFWIDTH          0x00002000
#define ABF_PEAK_MEASURE_BASELINE           0x00004000
#define ABF_PEAK_MEASURE_RISESLOPE          0x00008000
#define ABF_PEAK_MEASURE_DECAYSLOPE         0x00010000
#define ABF_PEAK_MEASURE_REGIONSLOPE        0x00020000
#define ABF_PEAK_MEASURE_ALL                0x0002FFFF    // All of the above OR'd together.

//
// Constants for nStatsActiveChannels
//
#define ABF_PEAK_SEARCH_CHANNEL0         0x0001
#define ABF_PEAK_SEARCH_CHANNEL1         0x0002
#define ABF_PEAK_SEARCH_CHANNEL2         0x0004
#define ABF_PEAK_SEARCH_CHANNEL3         0x0008
#define ABF_PEAK_SEARCH_CHANNEL4         0x0010
#define ABF_PEAK_SEARCH_CHANNEL5         0x0020
#define ABF_PEAK_SEARCH_CHANNEL6         0x0040
#define ABF_PEAK_SEARCH_CHANNEL7         0x0080
#define ABF_PEAK_SEARCH_CHANNEL8         0x0100
#define ABF_PEAK_SEARCH_CHANNEL9         0x0200
#define ABF_PEAK_SEARCH_CHANNEL10        0x0400
#define ABF_PEAK_SEARCH_CHANNEL11        0x0800
#define ABF_PEAK_SEARCH_CHANNEL12        0x1000
#define ABF_PEAK_SEARCH_CHANNEL13        0x2000
#define ABF_PEAK_SEARCH_CHANNEL14        0x4000
#define ABF_PEAK_SEARCH_CHANNEL15        0x8000
#define ABF_PEAK_SEARCH_CHANNELSALL      0xFFFF      // All of the above OR'd together.

// Bit flag settings for nStatsSearchRegionFlags
//
#define ABF_PEAK_SEARCH_REGION0          0x01
#define ABF_PEAK_SEARCH_REGION1          0x02
#define ABF_PEAK_SEARCH_REGION2          0x04
#define ABF_PEAK_SEARCH_REGION3          0x08
#define ABF_PEAK_SEARCH_REGION4          0x10
#define ABF_PEAK_SEARCH_REGION5          0x20
#define ABF_PEAK_SEARCH_REGION6          0x40
#define ABF_PEAK_SEARCH_REGION7          0x80
#define ABF_PEAK_SEARCH_REGIONALL        0xFF        // All of the above OR'd together.

//
// Constants for lStatisticsMeasurements
//
#define ABF_STATISTICS_ABOVETHRESHOLD    0x00000001
#define ABF_STATISTICS_EVENTFREQUENCY    0x00000002
#define ABF_STATISTICS_MEANOPENTIME      0x00000004
#define ABF_STATISTICS_MEANCLOSEDTIME    0x00000008
#define ABF_STATISTICS_ALL               0x0000000F     // All the above OR'd together.

//
// Constants for nStatisticsSaveStrategy
//
#define ABF_STATISTICS_NOAUTOSAVE           0
#define ABF_STATISTICS_AUTOSAVE             1
#define ABF_STATISTICS_AUTOSAVE_AUTOCLEAR   2

//
// Constants for nStatisticsDisplayStrategy
//
#define ABF_STATISTICS_DISPLAY      0
#define ABF_STATISTICS_NODISPLAY    1

//
// Constants for nStatisticsClearStrategy
// determines whether to clear statistics after saving.
//
#define ABF_STATISTICS_NOCLEAR      0
#define ABF_STATISTICS_CLEAR        1

//
// Constants for nDACFileEpisodeNum
//
#define ABF_DACFILE_SKIPFIRSTSWEEP -1
#define ABF_DACFILE_USEALLSWEEPS    0
// >0 = The specific sweep number.

//
```

```
// Constants for nUndoPromptStrategy
//
#define ABF_UNDOPROMPT_ONABORT   0
#define ABF_UNDOPROMPT_ALWAYS    1

//
// Constants for nAutoAnalyseEnable
//
#define ABF_AUTOANALYSE_DISABLED   0
#define ABF_AUTOANALYSE_DEFAULT    1
#define ABF_AUTOANALYSE_RUNMACRO   2

//
// Constants for post nPostprocessLowpassFilterType
//
#define ABF_POSTPROCESS_FILTER_NONE          0
#define ABF_POSTPROCESS_FILTER_COMBINATION   1
#define ABF_POSTPROCESS_FILTER_BESSEL        2
#define ABF_POSTPROCESS_FILTER_BOXCAR        3
#define ABF_POSTPROCESS_FILTER_BUTTERWORTH   4
#define ABF_POSTPROCESS_FILTER_CHEBYSHEV     5
#define ABF_POSTPROCESS_FILTER_GAUSSIAN      6
#define ABF_POSTPROCESS_FILTER_RC            7
#define ABF_POSTPROCESS_FILTER_RC8           8
#define ABF_POSTPROCESS_FILTER_ADAPTIVE      9

//
// Miscellaneous constants
//
#define ABF_FILTERDISABLED  100000.0F     // Large frequency to disable lowpass filters
#define ABF_UNUSED_CHANNEL  -1            // Unused ADC and DAC channels.

//
// The output sampling sequence identifier for a seperate digital out channel.
//
#define ABF_DIGITAL_OUT_CHANNEL -1
#define ABF_PADDING_OUT_CHANNEL -2

//
// maximum values for various parameters (used by ABFH_CheckUserList).
//
#define ABF_CTPULSECOUNT_MAX           10000
#define ABF_CTBASELINEDURATION_MAX     100000.0F
#define ABF_CTSTEPDURATION_MAX         100000.0F
#define ABF_CTPOSTTRAINDURATION_MAX    100000.0F
#define ABF_SWEEPSTARTTOSTARTTIME_MAX  100000.0F
#define ABF_PNPULSECOUNT_MAX           8
#define ABF_DIGITALVALUE_MAX           0xFF
#define ABF_EPOCHDIGITALVALUE_MAX      0x0F

//
// LTP Types - Reflects whether the header is used for LTP as baseline or induction.
//
#define ABF_LTP_TYPE_NONE              0
#define ABF_LTP_TYPE_BASELINE          1
#define ABF_LTP_TYPE_INDUCTION         2

//
// LTP Usage of DAC - Reflects whether the analog output will be used presynaptically or
postsynaptically.
//
#define ABF_LTP_DAC_USAGE_NONE         0
#define ABF_LTP_DAC_USAGE_PRESYNAPTIC  1
#define ABF_LTP_DAC_USAGE_POSTSYNAPTIC 2

//
// Header Version Numbers
//
#define ABF_V166  1.66F
#define ABF_V167  1.67F
#define ABF_V168  1.68F
#define ABF_V169  1.69F
#define ABF_V170  1.70F
#define ABF_V171  1.71F
#define ABF_V172  1.72F
#define ABF_V173  1.73F
#define ABF_V174  1.74F
#define ABF_V175  1.75F
#define ABF_V176  1.76F
```

```
#define ABF_V177  1.77F
#define ABF_V178  1.78F
#define ABF_V179  1.79F
#define ABF_V180  1.80F

//
// pack structure on byte boundaries
//
#ifndef RC_INVOKED
#pragma pack(push, 1)
#endif

//
// Definition of the ABF header structure.
//

struct ABFFileHeader               // The total header length = 6144 bytes.
{
public:
   // GROUP #1 - File ID and size information. (40 bytes)
   long     lFileSignature;
   float    fFileVersionNumber;
   short    nOperationMode;
   long     lActualAcqLength;
   short    nNumPointsIgnored;
   long     lActualEpisodes;
   long     lFileStartDate;         // YYYYMMDD
   long     lFileStartTime;
   long     lStopwatchTime;
   float    fHeaderVersionNumber;
   short    nFileType;
   short    nMSBinFormat;

   // GROUP #2 - File Structure (78 bytes)
   long     lDataSectionPtr;
   long     lTagSectionPtr;
   long     lNumTagEntries;
   long     lScopeConfigPtr;
   long     lNumScopes;
   long     _lDACFilePtr;
   long     _lDACFileNumEpisodes;
   char     sUnused001[4];
   long     lDeltaArrayPtr;
   long     lNumDeltas;
   long     lVoiceTagPtr;
   long     lVoiceTagEntries;
   long     lUnused002;
   long     lSynchArrayPtr;
   long     lSynchArraySize;
   short    nDataFormat;
   short    nSimultaneousScan;
   long     lStatisticsConfigPtr;
   long     lAnnotationSectionPtr;
   long     lNumAnnotations;
   char     sUnused003[2];

   // GROUP #3 - Trial hierarchy information (82 bytes)
   /**
   The number of input channels we acquired.
   Do not access directly - use CABFHeader::get_channel_count_acquired
   */
   short    channel_count_acquired;

   /**
   The number of input channels we recorded.
   Do not access directly - use CABFHeader::get_channel_count_recorded
   */
   short    nADCNumChannels;
   float    fADCSampleInterval;
      /*{{
      The documentation says these two sample intervals are the interval between multiplexed
samples, but not all digitisers work like that.
      Instead, these are the per-channel sample rate divided by the number of channels.
      If the user chose 100uS and has two channels, this value will be 50uS.
      }}*/
   float    fADCSecondSampleInterval;
      /*{{
      // The two sample intervals must be an integer multiple (or submultiple) of each other.
      if (fADCSampleInterval > fADCSecondSampleInterval)
```

```
      ASSERT(fmod(fADCSampleInterval, fADCSecondSampleInterval) == 0.0);
   if (fADCSecondSampleInterval, fADCSampleInterval)
      ASSERT(fmod(fADCSecondSampleInterval, fADCSampleInterval) == 0.0);
   }}*/
   float    fSynchTimeUnit;
   float    fSecondsPerRun;

   /**
   * The total number of samples per episode, for the recorded channels only.
   * This does not include channels which are acquired but not recorded.
   *
   * This is the number of samples per episode per channel, times the number of recorded channels.
   *
   * If you want the samples per episode for one channel, you must divide this by
get_channel_count_recorded().
   */
   long     lNumSamplesPerEpisode;
   long     lPreTriggerSamples;
   long     lEpisodesPerRun;
   long     lRunsPerTrial;
   long     lNumberOfTrials;
   short    nAveragingMode;
   short    nUndoRunCount;
   short    nFirstEpisodeInRun;
   float    fTriggerThreshold;
   short    nTriggerSource;
   short    nTriggerAction;
   short    nTriggerPolarity;
   float    fScopeOutputInterval;
   float    fEpisodeStartToStart;
   float    fRunStartToStart;
   float    fTrialStartToStart;
   long     lAverageCount;
   long     lClockChange;
   short    nAutoTriggerStrategy;

   // GROUP #4 - Display Parameters (44 bytes)
   short    nDrawingStrategy;
   short    nTiledDisplay;
   short    nEraseStrategy;           // N.B. Discontinued. Use scope config entry instead.
   short    nDataDisplayMode;
   long     lDisplayAverageUpdate;
   short    nChannelStatsStrategy;
   long     lCalculationPeriod;       // N.B. Discontinued. Use fStatisticsPeriod.
   long     lSamplesPerTrace;
   long     lStartDisplayNum;
   long     lFinishDisplayNum;
   short    nMultiColor;
   short    nShowPNRawData;
   float    fStatisticsPeriod;
   long     lStatisticsMeasurements;
   short    nStatisticsSaveStrategy;

   // GROUP #5 - Hardware information (16 bytes)
   float    fADCRange;
   float    fDACRange;
   long     lADCResolution;
   long     lDACResolution;

   // GROUP #6 Environmental Information (118 bytes)
   short    nExperimentType;
   short    _nAutosampleEnable;
   short    _nAutosampleADCNum;
   short    _nAutosampleInstrument;
   float    _fAutosampleAdditGain;
   float    _fAutosampleFilter;
   float    _fAutosampleMembraneCap;
   short    nManualInfoStrategy;
   float    fCellID1;
   float    fCellID2;
   float    fCellID3;
   char     sCreatorInfo[ABF_CREATORINFOLEN];
   char     _sFileComment[ABF_OLDFILECOMMENTLEN];
   short    nFileStartMillisecs;    // Milliseconds portion of lFileStartTime
   short    nCommentsEnable;
   char     sUnused003a[8];

   // GROUP #7 - Multi-channel information (1044 bytes)
   short    nADCPtoLChannelMap[ABF_ADCCOUNT];
```

```
short    nADCSamplingSeq[ABF_ADCCOUNT];
char     sADCChannelName[ABF_ADCCOUNT][ABF_ADCNAMELEN];
char     sADCUnits[ABF_ADCCOUNT][ABF_ADCUNITLEN];
float    fADCProgrammableGain[ABF_ADCCOUNT];
float    fADCDisplayAmplification[ABF_ADCCOUNT];
float    fADCDisplayOffset[ABF_ADCCOUNT];
float    fInstrumentScaleFactor[ABF_ADCCOUNT];
float    fInstrumentOffset[ABF_ADCCOUNT];
float    fSignalGain[ABF_ADCCOUNT];
float    fSignalOffset[ABF_ADCCOUNT];
float    fSignalLowpassFilter[ABF_ADCCOUNT];
float    fSignalHighpassFilter[ABF_ADCCOUNT];
char     sDACChannelName[ABF_DACCOUNT][ABF_DACNAMELEN];
char     sDACChannelUnits[ABF_DACCOUNT][ABF_DACUNITLEN];
float    fDACScaleFactor[ABF_DACCOUNT];
float    fDACHoldingLevel[ABF_DACCOUNT];
short    nSignalType;
char     sUnused004[10];

// GROUP #8 - Synchronous timer outputs (14 bytes)
short    nOUTEnable;
short    nSampleNumberOUT1;
short    nSampleNumberOUT2;
short    nFirstEpisodeOUT;
short    nLastEpisodeOUT;
short    nPulseSamplesOUT1;
short    nPulseSamplesOUT2;

// GROUP #9 - Epoch Waveform and Pulses (184 bytes)
short    nDigitalEnable;
short    _nWaveformSource;
short    nActiveDACChannel;
short    _nInterEpisodeLevel;
short    _nEpochType[ABF_EPOCHCOUNT];
float    _fEpochInitLevel[ABF_EPOCHCOUNT];
float    _fEpochLevelInc[ABF_EPOCHCOUNT];
short    _nEpochInitDuration[ABF_EPOCHCOUNT];
short    _nEpochDurationInc[ABF_EPOCHCOUNT];
short    nDigitalHolding;
short    nDigitalInterEpisode;
short    nDigitalValue[ABF_EPOCHCOUNT];
char     sUnavailable1608[4];    // was float fWaveformOffset;
short    nDigitalDACChannel;
char     sUnused005[6];

// GROUP #10 - DAC Output File (98 bytes)
float    _fDACFileScale;
float    _fDACFileOffset;
char     sUnused006[2];
short    _nDACFileEpisodeNum;
short    _nDACFileADCNum;
char     _sDACFilePath[ABF_DACFILEPATHLEN];

// GROUP #11 - Presweep (conditioning) pulse train (44 bytes)
short    _nConditEnable;
short    _nConditChannel;
long     _lConditNumPulses;
float    _fBaselineDuration;
float    _fBaselineLevel;
float    _fStepDuration;
float    _fStepLevel;
float    _fPostTrainPeriod;
float    _fPostTrainLevel;
char     sUnused007[12];

// GROUP #12 - Variable parameter user list ( 82 bytes)
short    _nParamToVary;
char     _sParamValueList[ABF_VARPARAMLISTLEN];

// GROUP #13 - Autopeak measurement (36 bytes)
short    _nAutopeakEnable;
short    _nAutopeakPolarity;
short    _nAutopeakADCNum;
short    _nAutopeakSearchMode;
long     _lAutopeakStart;
long     _lAutopeakEnd;
short    _nAutopeakSmoothing;
short    _nAutopeakBaseline;
short    _nAutopeakAverage;
```

```
char     sUnavailable1866[2];     // Was nAutopeakSaveStrategy, use nStatisticsSaveStrategy
long     _lAutopeakBaselineStart;
long     _lAutopeakBaselineEnd;
long     _lAutopeakMeasurements;

// GROUP #14 - Channel Arithmetic (52 bytes)
short    nArithmeticEnable;
float    fArithmeticUpperLimit;
float    fArithmeticLowerLimit;
short    nArithmeticADCNumA;
short    nArithmeticADCNumB;
float    fArithmeticK1;
float    fArithmeticK2;
float    fArithmeticK3;
float    fArithmeticK4;
char     sArithmeticOperator[ABF_ARITHMETICOPLEN];
char     sArithmeticUnits[ABF_ARITHMETICUNITSLEN];
float    fArithmeticK5;
float    fArithmeticK6;
short    nArithmeticExpression;
char     sUnused008[2];

// GROUP #15 - On-line subtraction (34 bytes)
short    _nPNEnable;
short    nPNPosition;
short    _nPNPolarity;
short    nPNNumPulses;
short    _nPNADCNum;
float    _fPNHoldingLevel;
float    fPNSettlingTime;
float    fPNInterpulse;
char     sUnused009[12];

// GROUP #16 - Miscellaneous variables (82 bytes)
short    _nListEnable;

short    nBellEnable[ABF_BELLCOUNT];
short    nBellLocation[ABF_BELLCOUNT];
short    nBellRepetitions[ABF_BELLCOUNT];

short    nLevelHysteresis;
long     lTimeHysteresis;
short    nAllowExternalTags;

char     nLowpassFilterType[ABF_ADCCOUNT];
char     nHighpassFilterType[ABF_ADCCOUNT];
short    nAverageAlgorithm;
float    fAverageWeighting;
short    nUndoPromptStrategy;
short    nTrialTriggerSource;
short    nStatisticsDisplayStrategy;
short    nExternalTagType;
long     lHeaderSize;
double   dFileDuration;
short    nStatisticsClearStrategy;
// Size of v1.5 header = 2048

// Extra parameters in v1.6
// EXTENDED GROUP #2 - File Structure (26 bytes)
long     lDACFilePtr[ABF_WAVEFORMCOUNT];
long     lDACFileNumEpisodes[ABF_WAVEFORMCOUNT];
char     sUnused010[10];

// EXTENDED GROUP #7 - Multi-channel information (62 bytes)
float    fDACCalibrationFactor[ABF_DACCOUNT];
float    fDACCalibrationOffset[ABF_DACCOUNT];
char     sUnused011[30];

// GROUP #17 - Trains parameters (160 bytes)
long     lEpochPulsePeriod[ABF_WAVEFORMCOUNT][ABF_EPOCHCOUNT];
long     lEpochPulseWidth [ABF_WAVEFORMCOUNT][ABF_EPOCHCOUNT];

// EXTENDED GROUP #9 - Epoch Waveform and Pulses ( 412 bytes)
short    nWaveformEnable[ABF_WAVEFORMCOUNT];
short    nWaveformSource[ABF_WAVEFORMCOUNT];
short    nInterEpisodeLevel[ABF_WAVEFORMCOUNT];
short    nEpochType[ABF_WAVEFORMCOUNT][ABF_EPOCHCOUNT];
float    fEpochInitLevel[ABF_WAVEFORMCOUNT][ABF_EPOCHCOUNT];
float    fEpochLevelInc[ABF_WAVEFORMCOUNT][ABF_EPOCHCOUNT];
```

```
   long     lEpochInitDuration[ABF_WAVEFORMCOUNT][ABF_EPOCHCOUNT];
   long     lEpochDurationInc[ABF_WAVEFORMCOUNT][ABF_EPOCHCOUNT];
   short    nDigitalTrainValue[ABF_EPOCHCOUNT];                         // 2 * 10 = 20 bytes
   short    nDigitalTrainActiveLogic;                                   // 2 bytes
   char     sUnused012[18];

   // EXTENDED GROUP #10 - DAC Output File (552 bytes)
   float    fDACFileScale[ABF_WAVEFORMCOUNT];
   float    fDACFileOffset[ABF_WAVEFORMCOUNT];
   long     lDACFileEpisodeNum[ABF_WAVEFORMCOUNT];
   short    nDACFileADCNum[ABF_WAVEFORMCOUNT];
   char     sDACFilePath[ABF_WAVEFORMCOUNT][ABF_PATHLEN];
   char     sUnused013[12];

   // EXTENDED GROUP #11 - Presweep (conditioning) pulse train (100 bytes)
   short    nConditEnable[ABF_WAVEFORMCOUNT];
   long     lConditNumPulses[ABF_WAVEFORMCOUNT];
   float    fBaselineDuration[ABF_WAVEFORMCOUNT];
   float    fBaselineLevel[ABF_WAVEFORMCOUNT];
   float    fStepDuration[ABF_WAVEFORMCOUNT];
   float    fStepLevel[ABF_WAVEFORMCOUNT];
   float    fPostTrainPeriod[ABF_WAVEFORMCOUNT];
   float    fPostTrainLevel[ABF_WAVEFORMCOUNT];
   char     sUnused014[40];

   // EXTENDED GROUP #12 - Variable parameter user list (1096 bytes)
   short    nULEnable[ABF_USERLISTCOUNT];
   short    nULParamToVary[ABF_USERLISTCOUNT];
   char     sULParamValueList[ABF_USERLISTCOUNT][ABF_USERLISTLEN];
   short    nULRepeat[ABF_USERLISTCOUNT];
   char     sUnused015[48];

   // EXTENDED GROUP #15 - On-line subtraction (56 bytes)
   short    nPNEnable[ABF_WAVEFORMCOUNT];
   short    nPNPolarity[ABF_WAVEFORMCOUNT];
   short    nPNADCNum[ABF_WAVEFORMCOUNT];
   float    fPNHoldingLevel[ABF_WAVEFORMCOUNT];
   char     sUnused016[36];

   // EXTENDED GROUP #6 Environmental Information  (898 bytes)
   short    nTelegraphEnable[ABF_ADCCOUNT];
   short    nTelegraphInstrument[ABF_ADCCOUNT];
   float    fTelegraphAdditGain[ABF_ADCCOUNT];
   float    fTelegraphFilter[ABF_ADCCOUNT];
   float    fTelegraphMembraneCap[ABF_ADCCOUNT];
   short    nTelegraphMode[ABF_ADCCOUNT];
   short    nTelegraphDACScaleFactorEnable[ABF_DACCOUNT];
   char     sUnused016a[24];

   short    nAutoAnalyseEnable;
   char     sAutoAnalysisMacroName[ABF_MACRONAMELEN];
   char     sProtocolPath[ABF_PATHLEN];

   char     sFileComment[ABF_FILECOMMENTLEN];
   char     sUnused017[128];

   // EXTENDED GROUP #13 - Statistics measurements (388 bytes)
   short    nStatsEnable;
   unsigned short nStatsActiveChannels;             // Active stats channel bit flag
   unsigned short nStatsSearchRegionFlags;          // Active stats region bit flag
   short    nStatsSelectedRegion;
   short    _nStatsSearchMode;
   short    nStatsSmoothing;
   short    nStatsSmoothingEnable;
   short    nStatsBaseline;
   long     lStatsBaselineStart;
   long     lStatsBaselineEnd;
   long     lStatsMeasurements[ABF_STATS_REGIONS];  // Measurement bit flag for each region
   long     lStatsStart[ABF_STATS_REGIONS];
   long     lStatsEnd[ABF_STATS_REGIONS];
   short    nRiseBottomPercentile[ABF_STATS_REGIONS];
   short    nRiseTopPercentile[ABF_STATS_REGIONS];
   short    nDecayBottomPercentile[ABF_STATS_REGIONS];
   short    nDecayTopPercentile[ABF_STATS_REGIONS];
   short    nStatsChannelPolarity[ABF_ADCCOUNT];
   short    nStatsSearchMode[ABF_STATS_REGIONS];    // Stats mode per region: mode is cursor region,
epoch etc
   char     sUnused018[156];
```

```
   // GROUP #18 - Application version data (16 bytes)
   short    nMajorVersion;
   short    nMinorVersion;
   short    nBugfixVersion;
   short    nBuildVersion;
   char     sUnused019[8];

   // GROUP #19 - LTP protocol (14 bytes)
   short    nLTPType;
   short    nLTPUsageOfDAC[ABF_WAVEFORMCOUNT];
   short    nLTPPresynapticPulses[ABF_WAVEFORMCOUNT];
   char     sUnused020[4];

   // GROUP #20 - Digidata 132x Trigger out flag. (8 bytes)
   short    nDD132xTriggerOut;
   char     sUnused021[6];

   // GROUP #21 - Epoch resistance (40 bytes)
   char     sEpochResistanceSignalName[ABF_WAVEFORMCOUNT][ABF_ADCNAMELEN];
   short    nEpochResistanceState[ABF_WAVEFORMCOUNT];
   char     sUnused022[16];

   // GROUP #22 - Alternating episodic mode (58 bytes)
   short    nAlternateDACOutputState;
   short    nAlternateDigitalValue[ABF_EPOCHCOUNT];
   short    nAlternateDigitalTrainValue[ABF_EPOCHCOUNT];
   short    nAlternateDigitalOutputState;
   char     sUnused023[14];

   // GROUP #23 - Post-processing actions (210 bytes)
   float    fPostProcessLowpassFilter[ABF_ADCCOUNT];
   char     nPostProcessLowpassFilterType[ABF_ADCCOUNT];


   // 6014 header bytes allocated + 130 header bytes not allocated
   char     sUnused2048[130];

   ABFFileHeader();
};   // Size = 6144
// This structure is persisted, so the size MUST NOT CHANGE
STATIC_ASSERT(sizeof(ABFFileHeader) == 6144);

inline ABFFileHeader::ABFFileHeader()
{
   // Set everything to 0.
   memset( this, 0, sizeof(ABFFileHeader) );

   // Set critical parameters so we can determine the version.
   lFileSignature       = ABF_NATIVESIGNATURE;
   fFileVersionNumber   = ABF_CURRENTVERSION;
   fHeaderVersionNumber = ABF_CURRENTVERSION;
   lHeaderSize          = ABF_HEADERSIZE;
}


//
// Scope descriptor format.
//
#define ABF_FACESIZE 32
struct ABFLogFont
{
   short nHeight;               // Height of the font in pixels.
//   short lWidth;               // use 0
//   short lEscapement;          // use 0
//   short lOrientation;         // use 0
   short nWeight;               // MSWindows font weight value.
//   char bItalic;               // use 0
//   char bUnderline;            // use 0
//   char bStrikeOut;            // use 0
//   char cCharSet;              // use ANSI_CHARSET (0)
//   char cOutPrecision;         // use OUT_TT_PRECIS
//   char cClipPrecision;        // use CLIP_DEFAULT_PRECIS
//   char cQuality;              // use PROOF_QUALITY
   char cPitchAndFamily;        // MSWindows pitch and family mask.
   char Unused[3];              // Unused space to maintain 4-byte packing.
   char szFaceName[ABF_FACESIZE];// Face name of the font.
};     // Size = 40

struct ABFSignal
{
```

```
   char     szName[ABF_ADCNAMELEN+2];       // ABF name length + '\0' + 1 for alignment.
   short    nMxOffset;                      // Offset of the signal in the sampling sequence.
   DWORD    rgbColor;                       // Pen color used to draw trace.
   char     nPenWidth;                      // Pen width in pixels.
   char     bDrawPoints;                    // TRUE = Draw disconnected points
   char     bHidden;                        // TRUE = Hide the trace.
   char     bFloatData;                     // TRUE = Floating point pseudo channel
   float    fVertProportion;                // Relative proportion of client area to use
   float    fDisplayGain;                   // Display gain of trace in UserUnits
   float    fDisplayOffset;                 // Display offset of trace in UserUnits

//   float    fUUTop;                          // Top of window in UserUnits
//   float    fUUBottom;                       // Bottom of window in UserUnits
};      // Size = 34

//////////////////////////////////////////////////////////////////////////////
//// WARNING WARNING WARNING WARNING WARNING WARNING WARNING WARNING WARNING //////
//////////////////////////////////////////////////////////////////////////////
// The Following #defines appear to be largely unused in opur code base
// However there does exist a second set of #defines in AxScope32.h
// that REALLY defines what these bits in the header do.
// In particular it important to note that all 32 bits are in fact used internally
//////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////////////
// Bit flags used in dwFlags field of ABFScopeConfig.
#define ABF_OVERLAPPED      0x00000001
#define ABF_DONTERASE       0x00000002
#define ABF_MONOCHROME      0x00000004
#define ABF_CLIPPING        0x00000008
#define ABF_HIDEHORZGRIDS   0x00000010
#define ABF_HIDEVERTGRIDS   0x00000020
#define ABF_FULLSCREEN      0x00000040
#define ABF_HIDEXAXIS       0x00000080
#define ABF_HIDEYAXIS       0x00000100
#define ABF_HIDEXSCROLL     0x00000200
#define ABF_HIDEYSCROLL     0x00000400
#define ABF_HIDESIGNALNAME  0x00000800
#define ABF_ENABLEZOOM      0x00001000
#define ABF_XSPINFROMCENTER 0x00002000
#define ABF_HIDEXSPINNER    0x00004000
#define ABF_LARGESPINNERS   0x00008000
#define ABF_PERSISTENCEMODE 0x00010000
#define ABF_CARDIACMODE     0x00020000
#define ABF_HIDETWIRLER     0x00040000
#define ABF_DISABLEUI       0x00080000
//////////////////////////////////////////////////////////////////////////////
// #define ABF_INTERNALUSE  0xFFF00000
// Do not add extra bit flags ^^^ here they are used internally
//////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////////////
//// DANGER DANGER DANGER DANGER DANGER DANGER DANGER DANGER DANGER DANGER DANGER//
//////////////////////////////////////////////////////////////////////////////

// Values for the wScopeMode field in ABFScopeConfig.
#define ABF_EPISODICMODE    0
#define ABF_CONTINUOUSMODE  1
//#define ABF_XYMODE          2

// Values for the nEraseStrategy field in ABFScopeConfig.
#define ABF_ERASE_EACHSWEEP   0
#define ABF_ERASE_EACHRUN     1
#define ABF_ERASE_EACHTRIAL   2
#define ABF_ERASE_DONTERASE   3

// Indexes into the rgbColor field of ABFScopeConfig.
#define ABF_BACKGROUNDCOLOR   0
#define ABF_GRIDCOLOR         1
#define ABF_THRESHOLDCOLOR    2
#define ABF_EVENTMARKERCOLOR  3
#define ABF_SEPARATORCOLOR    4
#define ABF_AVERAGECOLOR      5
#define ABF_OLDDATACOLOR      6
#define ABF_TEXTCOLOR         7
#define ABF_AXISCOLOR         8
#define ABF_ACTIVEAXISCOLOR   9
#define ABF_LASTCOLOR         ABF_ACTIVEAXISCOLOR
#define ABF_SCOPECOLORS       (ABF_LASTCOLOR+1)

// Extended colors for rgbColorEx field in ABFScopeConfig
```

```
#define ABF_STATISTICS_REGION0 0
#define ABF_STATISTICS_REGION1 1
#define ABF_STATISTICS_REGION2 2
#define ABF_STATISTICS_REGION3 3
#define ABF_STATISTICS_REGION4 4
#define ABF_STATISTICS_REGION5 5
#define ABF_STATISTICS_REGION6 6
#define ABF_STATISTICS_REGION7 7
#define ABF_BASELINE_REGION    8
#define ABF_STOREDSWEEPCOLOR   9
#define ABF_LASTCOLOR_EX       ABF_STOREDSWEEPCOLOR
#define ABF_SCOPECOLORS_EX     (ABF_LASTCOLOR+1)

// Values for the nDockState field in ABFScopeConfig
#define ABF_SCOPE_NOTDOCKED      0
#define ABF_SCOPE_DOCKED_TOP     1
#define ABF_SCOPE_DOCKED_LEFT    2
#define ABF_SCOPE_DOCKED_RIGHT   3
#define ABF_SCOPE_DOCKED_BOTTOM  4

struct ABFScopeConfig
{
   // Section 1 scope configurations
   DWORD       dwFlags;                   // Flags that are meaningful to the scope.
   DWORD       rgbColor[ABF_SCOPECOLORS]; // Colors for the components of the scope.
   float       fDisplayStart;             // Start of the display area in ms.
   float       fDisplayEnd;               // End of the display area in ms.
   WORD        wScopeMode;                // Mode that the scope is in.
   char        bMaximized;                // TRUE = Scope parent is maximized.
   char        bMinimized;                // TRUE = Scope parent is minimized.
   short       xLeft;                     // Coordinate of the left edge.
   short       yTop;                      // Coordinate of the top edge.
   short       xRight;                    // Coordinate of the right edge.
   short       yBottom;                   // Coordinate of the bottom edge.
   ABFLogFont  LogFont;                   // Description of current font.
   ABFSignal   TraceList[ABF_ADCCOUNT];   // List of traces in current use.
   short       nYAxisWidth;               // Width of the YAxis region.
   short       nTraceCount;               // Number of traces described in TraceList.
   short       nEraseStrategy;            // Erase strategy.
   short       nDockState;                // Docked position.
   // Size 656
   // * Do not insert any new members above this point! *
   // Section 2 scope configurations for file version 1.68.
   short       nSizeofOldStructure;              // Unused byte to determine the offset of the
version 2 data.
   DWORD       rgbColorEx[ ABF_SCOPECOLORS_EX ]; // New color settings for stored sweep and cursors.
   short       nAutoZeroState;                   // Status of the autozero selection.
   DWORD       dwCursorsVisibleState;            // Flag for visible status of cursors.
   DWORD       dwCursorsLockedState;             // Flag for enabled status of cursors.
   char        sUnasigned[61];
   // Size 113
   ABFScopeConfig();
}; // Size = 769

inline ABFScopeConfig::ABFScopeConfig()
{
   // Set everything to 0.
   memset( this, 0, sizeof(ABFScopeConfig) );

   // Set critical parameters so we can determine the version.
   nSizeofOldStructure = 656;
}

//
// Definition of the ABF synch array structure
//

struct ABFSynch
{
   long    lStart;           // Start of the episode/event in fSynchTimeUnit units.
   long    lLength;          // Length of the episode/event in multiplexed samples.
}; // Size = 8

//
// Constants for nTagType in the ABFTag structure.
//
#define ABF_TIMETAG             0
#define ABF_COMMENTTAG          1
#define ABF_EXTERNALTAG         2
```

```
#define ABF_VOICETAG            3
#define ABF_NEWFILETAG          4

//
// Definition of the ABF Tag structure
//
struct ABFTag
{
   long    lTagTime;          // Time at which the tag was entered in fSynchTimeUnit units.
   char    sComment[ABF_TAGCOMMENTLEN];   // Optional tag comment.
   short   nTagType;          // Type of tag ABF_TIMETAG, ABF_COMMENTTAG, ABF_EXTERNALTAG or
ABF_VOICETAG.
   short   nVoiceTagNumber;   // If nTagType=ABF_VOICETAG, this is the number of this voice tag.
}; // Size = 64

// Comment inserted for externally acquired tags (expanded with spaces to ABF_TAGCOMMENTLEN).
#define ABF_EXTERNALTAGCOMMENT   "<External>"
#define ABF_VOICETAGCOMMENT      "<Voice Tag>"

//
// Constants for nCompressionType in the ABFVoiceTagInfo structure.
//
#define ABF_COMPRESSION_NONE     0
#define ABF_COMPRESSION_PKWARE   1
//#define ABF_COMPRESSION_MPEG     2

//
// Definition of the ABFVoiceTagInfo structure.
//
struct ABFVoiceTagInfo
{
   long  lTagNumber;          // The tag number that corresponds to this VoiceTag
   long  lFileOffset;         // Offset to this tag within the VoiceTag block
   long  lUncompressedSize;   // Size of the voice tag expanded.
   long  lCompressedSize;     // Compressed size of the tag.
   short nCompressionType;    // Compression method used.
   short nSampleSize;         // Size of the samples acquired.
   long  lSamplesPerSecond;   // Rate at which the sound was acquired.
   DWORD dwCRC;               // CRC used to check data integrity.
   WORD  wChannels;           // Number of channels in the tag (usually 1).
   WORD  wUnused;             // Unused space.
}; // Size 32

//
// Constants for lParameterID in the ABFDelta structure.
//
// NOTE: If any changes are made to this list, the code in ABF_UpdateHeader must
//       be updated to include the new items.
#define ABF_DELTA_HOLDING0         0
#define ABF_DELTA_HOLDING1         1
#define ABF_DELTA_HOLDING2         2
#define ABF_DELTA_HOLDING3         3
#define ABF_DELTA_DIGITALOUTS      4
#define ABF_DELTA_THRESHOLD        5
#define ABF_DELTA_PRETRIGGER       6

// Because of lack of space, the Autosample Gain ID also contains the ADC number.
#define ABF_DELTA_AUTOSAMPLE_GAIN   100   // +ADC channel.

// Because of lack of space, the Signal Gain ID also contains the ADC number.
#define ABF_DELTA_SIGNAL_GAIN       200   // +ADC channel.

//
// Definition of the ABF Delta structure.
//

struct ABFDelta
{
   long    lDeltaTime;        // Time at which the parameter was changed in fSynchTimeUnit units.
   long    lParameterID;      // Identifier for the parameter changed
   union
   {
      long  lNewParamValue;   // Depending on the value of lParameterID
      float fNewParamValue;   // this entry may be either a float or a long.
   };
}; // Size = 12

#ifndef RC_INVOKED
#pragma pack(pop)                     // return to default packing
```

```
#endif

//
// The size of the buffers to be passed to ABFH_GetWaveformVertor
//
#define ABFH_MAXVECTORS     30

//
// Function prototypes for functions in ABFHEADR.C
//

void WINAPI ABFH_Initialize( ABFFileHeader *pFH );

void WINAPI ABFH_InitializeScopeConfig(const ABFFileHeader *pFH, ABFScopeConfig *pCfg);

BOOL WINAPI ABFH_CheckScopeConfig(ABFFileHeader *pFH, ABFScopeConfig *pCfg);

void WINAPI ABFH_GetADCDisplayRange( const ABFFileHeader *pFH, int nChannel,
                                     float *pfUUTop, float *pfUUBottom);

void WINAPI ABFH_GetADCtoUUFactors( const ABFFileHeader *pFH, int nChannel,
                                    float *pfADCToUUFactor, float *pfADCToUUShift );
void WINAPI ABFH_ClipADCUUValue(const ABFFileHeader *pFH, int nChannel, float *pfUUValue);

void WINAPI ABFH_GetDACtoUUFactors( const ABFFileHeader *pFH, int nChannel,
                                    float *pfDACToUUFactor, float *pfDACToUUShift );
void WINAPI ABFH_ClipDACUUValue(const ABFFileHeader *pFH, int nChannel, float *pfUUValue);

BOOL WINAPI ABFH_GetMathValue(const ABFFileHeader *pFH, float fA, float fB, float *pfRval);
int  WINAPI ABFH_GetMathChannelName(char *pszName, UINT uNameLen);

BOOL WINAPI ABFH_ParamReader( HANDLE hFile, ABFFileHeader *pFH, int *pnError );
BOOL WINAPI ABFH_ParamReaderEx( HANDLE hFile, ABFFileHeader *pFH, int *pnError );
BOOL WINAPI ABFH_ParamWriter( HANDLE hFile, ABFFileHeader *pFH, int *pnError );

BOOL WINAPI ABFH_GetErrorText( int nError, char *pszBuffer, UINT nBufferSize );

// ABFHWAVE.CPP

// Constants for ABFH_GetEpochLimits
#define ABFH_FIRSTHOLDING  -1
#define ABFH_LASTHOLDING   ABF_EPOCHCOUNT

// Return the bounds of a given epoch in a given episode. Values returned are ZERO relative.
BOOL WINAPI ABFH_GetEpochLimits(const ABFFileHeader *pFH, int nADCChannel, DWORD dwEpisode,
                                int nEpoch, UINT *puEpochStart, UINT *puEpochEnd,
                                int *pnError);

BOOL WINAPI ABFH_GetEpochLimitsEx(const ABFFileHeader *pFH, int nADCChannel, UINT uDACChannel, DWORD
dwEpisode,
                                int nEpoch, UINT *puEpochStart, UINT *puEpochEnd,
                                int *pnError);

// Get the offset in the sampling sequence for the given physical channel.
BOOL WINAPI ABFH_GetChannelOffset( const ABFFileHeader *pFH, int nChannel, UINT *puChannelOffset );

// This function forms the de-multiplexed DAC output waveform for the
// particular channel in the pfBuffer, in DAC UserUnits.
BOOL WINAPI ABFH_GetWaveform( const ABFFileHeader *pFH, int nADCChannel, DWORD dwEpisode,
                              float *pfBuffer, int *pnError);

BOOL WINAPI ABFH_GetWaveformEx( const ABFFileHeader *pFH, UINT uDACChannel, DWORD dwEpisode,
                                float *pfBuffer, int *pnError);

// This function forms the de-multiplexed Digital output waveform for the
// particular channel in the pdwBuffer, as a bit mask. Digital OUT 0 is in bit 0.
BOOL WINAPI ABFH_GetDigitalWaveform( const ABFFileHeader *pFH, int nChannel, DWORD dwEpisode,
                                     DWORD *pdwBuffer, int *pnError);

// Returns vector pairs for displaying a waveform made up of epochs.
BOOL WINAPI ABFH_GetWaveformVector(const ABFFileHeader *pFH, DWORD dwEpisode, UINT uStart,
                                   UINT uFinish, float *pfLevels, float *pfTimes,
                                   int *pnVectors, int *pnError);

// Returns vector pairs for displaying the digital outs.
BOOL WINAPI ABFH_GetDigitalWaveformVector(const ABFFileHeader *pFH, DWORD dwEpisode, UINT uStart,
                                          UINT uFinish, DWORD *pdwLevels, float *pfTimes,
                                          int *pnVectors, int *pnError);

// Calculates the timebase array for the file.
```

```
void WINAPI ABFH_GetTimebase(const ABFFileHeader *pFH, float fTimeOffset, float *pfBuffer, UINT
uBufferSize);
void WINAPI ABFH_GetTimebaseEx(const ABFFileHeader *pFH, double dTimeOffset, double *pdBuffer, UINT
uBufferSize);

// Get the duration of the first holding period.
UINT WINAPI ABFH_GetHoldingDuration(const ABFFileHeader *pFH);

// Checks whether the waveform varies from episode to episode.
BOOL WINAPI ABFH_IsConstantWaveform(const ABFFileHeader *pFH);

BOOL WINAPI ABFH_IsConstantWaveformEx(const ABFFileHeader *pFH, UINT uDACChannel);

// Checks that the sample intervals in the header are valid.
BOOL WINAPI ABFH_CheckSampleIntervals(const ABFFileHeader *pFH, float fClockResolution, int
*pnError);

// Gets the closest sample intervals higher and lower than the passed interval.
void WINAPI ABFH_GetClosestSampleIntervals(float fSampleInterval, float fClockResolution,
                                           int nOperationMode, float fMinPeriod, float fMaxPeriod,
                                           float *pfHigher, float *pfLower);

// Sets up the list for the spinner to drive the sampling interval through.
UINT WINAPI ABFH_SetupSamplingList(UINT uNumChannels, float fMinPeriod, float fMaxPeriod,
                                   float *pfIntervalList, UINT uListEntries);

// Get the full sweep length given the length available to epochs or vice-versa.
int WINAPI ABFH_SweepLenFromUserLen(int nUserLength, int nNumChannels);
int WINAPI ABFH_UserLenFromSweepLen(int nSweepLength, int nNumChannels);

// Converts a display range to the equivalent gain and offset factors.
void WINAPI ABFH_GainOffsetToDisplayRange( const ABFFileHeader *pFH, int nChannel,
                                           float fDisplayGain, float fDisplayOffset,
                                           float *pfUUTop, float *pfUUBottom);

// Converts a display range to the equivalent gain and offset factors.
void WINAPI ABFH_DisplayRangeToGainOffset( const ABFFileHeader *pFH, int nChannel,
                                           float fUUTop, float fUUBottom,
                                           float *pfDisplayGain, float *pfDisplayOffset);

// Converts a time value to a synch time count or vice-versa.
void WINAPI ABFH_SynchCountToMS(const ABFFileHeader *pFH, UINT uCount, double *pdTimeMS);
UINT WINAPI ABFH_MSToSynchCount(const ABFFileHeader *pFH, double dTimeMS);

// Gets the point at which the sampling interval changes if split clock.
UINT WINAPI ABFH_GetClockChange(const ABFFileHeader *pFH);

// Gets the duration of the Waveform Episode (in us), allowing for split clock etc.
void WINAPI ABFH_GetEpisodeDuration(const ABFFileHeader *pFH, double *pdEpisodeDuration);

// Gets the duration of a P/N sequence (in us), including settling times.
void WINAPI ABFH_GetPNDuration(const ABFFileHeader *pFH, double *pdPNDuration);

void WINAPI ABFH_GetPNDurationEx(const ABFFileHeader *pFH, UINT uDAC, double *pdPNDuration);

// Gets the duration of a pre-sweep train in us.
void WINAPI ABFH_GetTrainDuration(const ABFFileHeader *pFH, double *pdTrainDuration);

void WINAPI ABFH_GetTrainDurationEx (const ABFFileHeader *pFH, UINT uDAC, double *pdTrainDuration);

// Gets the duration of a whole meta-episode (in us).
void WINAPI ABFH_GetMetaEpisodeDuration(const ABFFileHeader *pFH, double *pdMetaEpisodeDuration);

// Gets the start to start period for the episode in us.
void WINAPI ABFH_GetEpisodeStartToStart(const ABFFileHeader *pFH, double *pdEpisodeStartToStart);

// Checks that the user list contains valid entries for the protocol.
BOOL WINAPI ABFH_CheckUserList(const ABFFileHeader *pFH, int *pnError);

BOOL WINAPI ABFH_CheckUserListEx(const ABFFileHeader *pFH, UINT uListNum, int *pnError);

// Checks if the ABFFileHeader is a new (6k) or old (2k) header.
BOOL WINAPI ABFH_IsNewHeader(const ABFFileHeader *pFH);

// Demotes a 1.5 or 1.6 (or greater) ABF header to 1.5 version ABF header.
void WINAPI ABFH_DemoteHeader(ABFFileHeader *pOut, const ABFFileHeader *pIn );

// Promotes a 1.5 or 1.6 (or less) ABF header to a 1.6 ABF header.
void WINAPI ABFH_PromoteHeader(ABFFileHeader *pOut, const ABFFileHeader *pIn );
```

```
// Gets the first sample interval, expressed as a double.
double WINAPI ABFH_GetFirstSampleInterval( const ABFFileHeader *pFH );

// Gets the second sample interval expressed as a double.
double WINAPI ABFH_GetSecondSampleInterval( const ABFFileHeader *pFH );

// Counts the number of changing sweeps.
UINT WINAPI ABFH_GetNumberOfChangingSweeps( const ABFFileHeader *pFH );

// // Checks whether the digital output varies from episode to episode.
BOOL WINAPI ABFH_IsConstantDigitalOutput(const ABFFileHeader *pFH);

BOOL WINAPI ABFH_IsConstantDigitalOutputEx(const ABFFileHeader *pFH, UINT uDACChannel);

//
// Error return values that may be returned by the ABFH_xxx functions.
//

#define ABFH_FIRSTERRORNUMBER          2001
#define ABFH_EHEADERREAD               2001
#define ABFH_EHEADERWRITE              2002
#define ABFH_EINVALIDFILE              2003
#define ABFH_EUNKNOWNFILETYPE          2004
#define ABFH_CHANNELNOTSAMPLED         2005
#define ABFH_EPOCHNOTPRESENT           2006
#define ABFH_ENOWAVEFORM               2007
#define ABFH_EDACFILEWAVEFORM          2008
#define ABFH_ENOMEMORY                 2009
#define ABFH_BADSAMPLEINTERVAL         2010
#define ABFH_BADSECONDSAMPLEINTERVAL   2011
#define ABFH_BADSAMPLEINTERVALS        2012
#define ABFH_ENOCONDITTRAINS           2013
#define ABFH_EMETADURATION             2014
#define ABFH_ECONDITNUMPULSES          2015
#define ABFH_ECONDITBASEDUR            2016
#define ABFH_ECONDITBASELEVEL          2017
#define ABFH_ECONDITPOSTTRAINDUR       2018
#define ABFH_ECONDITPOSTTRAINLEVEL     2019
#define ABFH_ESTART2START              2020
#define ABFH_EINACTIVEHOLDING          2021
#define ABFH_EINVALIDCHARS             2022
#define ABFH_ENODIG                    2023
#define ABFH_EDIGHOLDLEVEL             2024
#define ABFH_ENOPNPULSES               2025
#define ABFH_EPNNUMPULSES              2026
#define ABFH_ENOEPOCH                  2027
#define ABFH_EEPOCHLEN                 2028
#define ABFH_EEPOCHINITLEVEL           2029
#define ABFH_EDIGLEVEL                 2030
#define ABFH_ECONDITSTEPDUR            2031
#define ABFH_ECONDITSTEPLEVEL          2032
#define ABFH_EINVALIDBINARYCHARS       2033
#define ABFH_EBADWAVEFORM              2034


#ifdef __cplusplus
}
#endif

#endif   /* INC_ABFHEADR_H */
```

See Also:

The ABF File I/O Functions

The ABF File I/O Functions by category

ABFFILES.H

ABFINFO.H