

Biomedical Interchange File Format (BIFF)

INTRODUCTION	2
DATA TYPES	3
Alignment	3
Numbers	3
Characters	3
Chunks	3
PREDEFINED CHUNK ID'S: BIFF, LIST, CAT, PROP, GRP	4
BIFF	4
CAT	4
LIST	4
PROP	4
GRP	5
Problem	5
DEFINITION OF THE SEMG FORM	6
Group ID 'VERS'	6
Group ID 'DINF'	6
Group ID 'EXPI'	7
Group ID 'MEAS'	7
Group ID 'SGRD'	8
Group ID 'PATI'	8
Group ID 'TRIG'	8
Group ID 'EVNT'	9

Introduction

After a search reviewing different data formats we came to the conclusion that none of the defined formats met our demands. Some came very close and were almost perfect but then there was one big disadvantage that made us choose otherwise. Therefore, we constructed a 'new' type of data format for use in a biomedical environment. Because we were very impressed by the IFF format described in 1985 by Jerry Morris at Electronic Arts, we tried to use this format as a base for BIFF. We are not the first who made a new IFF format there are more IFF-like formats in use today e.g. RIFF, RIFX, TIFF, AIFF etc.

The differences with the standard IFF will be pointed out in this document. The following chapters you will find a global description of the BIFF format. As will become clear this format is, like the IFF, a framework for a general data format with many advantages. The main advantage is its facility to extend without breaking compatibility with previous software versions.

Data types

Alignment

Although to conform to the standard IFF every chunk (see below) must be even (16 bit) aligned, we think there is no longer a good reason to use this kind of alignment. This means all chunks in the BIFF format are byte (8-bit) aligned.

Numbers

The data types supported are listed in table 1. Here also, we differ with the IFF standard, since nowadays there are good standards for floating point numeric types, there is no reason to exclude these types anymore. Another big difference with the IFF standard is the byte order of the 16-bit and 32-bit integer type. We store these data in little endian byte order because all our systems run on Intel machines and most of our users will use this type of machine as well. One way or the other, one of the system types has to swap bytes. We prefer it to be the system that is least used.

Table 1.

UBYTE	8 bits unsigned
SHORT	16 bits signed
WORD	16 bits unsigned
LONG	32 bits signed
ULONG	32 bits unsigned
FLOAT	16 bits according to IEEE standard
DOUBLE	32 bits according to IEEE standard

Characters

Characters that are used in text strings are assumed to conform to the 8-bit ASCII (ISO-8859 Latin 1) code.

Chunks

Chunks are the building blocks of the IFF format. For the BIFF format this is the same. A chunk is a piece of data with a name and a value telling you how large the connected data is. The principle of a chunk is especially convenient for maintaining backward compatibility. This means an older version of the program can still read the new "data format" version by skipping the chunks he does not recognise. We tried to set-up the BIFF format in a way the user will benefit maximal from this principle (see GRP chapt. 3). Every chunk must start with a 4-byte character string identifying the chunk. Then a 32-bit unsigned long indicates the size of data compartment that will follow (see table 2).

Table 2.

<CHUNKNAME>	CHAR	4 byte character ID string
<n-bytes>	ULONG	Number of bytes to follow this ULONG
<data>		The data belonging to this chunk

Predefined chunk ID's: BIFF, LIST, CAT, PROP, GRP

BIFF

Every data file in the IFF standard should start with the ID "FORM", "LIST" or "CAT ". The BIFF standard differs here from the IFF standard. In order to prevent confusion the "FROM" ID is replaced by the "BIFF" ID. Hence, every BIFF file should start with the ID "BIFF", "LIST" or "CAT". In case of a singleton file this must be the "BIFF" ID. The next field should contain the number of bytes inside this BIFF and must be followed by a 4byte character field that will identify the BIFF type (see table 3).

Table 3.

'BIFF'	This is a BIFF file and it is a singleton file because it starts with BIFF
12002	This means there are 12002 byte to come after this ULONG
'SEMG'	It is a SEMG file

CAT

In order to make it possible to concatenate several different types of files together you can use the "CAT " ID instead of the "BIFF" ID field as the first field of the file. This is the same as in the original IFF standard. To connect two different files, every file inside the BIFF file should start with the "BIFF" ID (see table 4).

Table 4.

'CAT '	This is a concatenated file because it starts with CAT
10500	This means there are 10500 bytes to come after this ULONG.
' '	"A grab bag" of BIFFs are inside this CAT
'BIFF'	Start of the first BIFF file
490	This means this BIFF is 490 bytes long
'FORC'	This means this is a FORC file (force recording)
'BIFF'	Second different BIFF file
9982	After this ULONG there are 9982 in this BIFF
'SEMG'	This means this is a SEMG file

LIST

When concatenating several files of the same type, e.g. several epochs or trials of electrophysiological data, you can use the "LIST" ID instead of the "CAT " ID. This will have the same effect except you will have a second possible group ID. That ID is PROP. Note that you can store the same type of files in a "CAT ", but you will not be able to use the PROP ID.

PROP

A "LIST" can have a "PROP" where all the common chunks of the concatenated files are located. This is a convenient way to concatenate several epochs or trials together. The common information is stored only once under the "PROP" ID (see table 5).

Table 5.

'LIST'	This is a LIST of files of the same type
n-bytes	This means there are n-bytes to come after this ULONG
'SEMG'	A list of SEMG files
'PROP'	All common chunks are put here
100	This means in this PROP are 100 bytes
'SEMG'	Type of the property
'BIFF'	first BIFF file
n-bytes	After this ULONG there are n-bytes in this BIFF
'SEMG'	This means this is a SEMG file

'BIFF'	Second BIFF file
n-bytes	After this ULONG there are n-bytes in this BIFF
'SEMG'	This means this is a SEMG file

GRP

Every "BIFF" is made up of chunks. The "GRP" ID can be used to group a number of chunks together. This allows nesting of chunks which was not possible in the original IFF standard. Again a 32-bit integer field that contains the number of bytes inside this group must follow the "GRP" ID. After this a group identification field must be added (see table 6).

Table 6.

'BIFF'	This is a singleton file because it starts with BIFF
171	This means there are 171 bytes to come after this ULONG
'SEMG'	It is a SEMG (surface EMG) file
'GRP '	This means a set of chunks is coming after this
163	Total size of all chunks is 163 bytes
'EXPI'	The group is called EXPI (experiment information)
<CHUNK>	Chunk 1 (i.e. 'NAME' for name of the experiment)
50	This chunk is 50 bytes long
<DATA>	The data of this chunk is here
<CHUNK>	Chunk 1 (i.e. 'RDOC' for physicians name)
25	This chunk is 25 bytes long
<DATA>	The data of this chunk is here
<CHUNK>	Chunk 1 (i.e. 'MUSC' for examined muscle)
72	This chunk is 72 bytes long
<DATA>	The data of this chunk is here

Problem

Because the BIFF is a little endian data format a problem arises by using the same chunk ID's as used by the IFF standard, i.e. CAT, LIST and PROP. This means a IFF compatible reader will be able to read the first ID but will get wrong information about the file size because according to the IFF standard it should be in Big endian. As far as I know the RIFF standard has the same problem. Also the size of a CAT, LIST or PROP should be WORD aligned if we at least want to comply with the RIFF standard. Another possibility is to rename these chunk-ID's.

Definition of the SEMG form

High-density surface ElectroMyoGraphy (sEMG) (sEMG using multiple electrodes, up to 130) was the reason for our search after a suitable data format. When we could not find one we liked we came up with the general BIFF format. Within this frame we will define our SEMG format Version 1.0.

The chunk ID "**DATA**" is the only predefined chunk ID. It contains the physiological data.

Predefined group (GRP) ID's are:

'VERS'	File format version information
'DINF'	Data information
'EXPI'	Experiment related information
'MEAS'	Measurement related information
'SGRD'	Surface EMG electrode grid/matrix information
'PATI'	Patient information
'TRIG'	Triggers connected to the recorded data
'EVNT'	Events connected to the recorded data

Group ID 'VERS'

This group contains SEMG format information and the license number of the equipment used. This must always be the first group in the SEMG file

ID	Type	Description
'ID '	UBYTE	Version ID example: 1.0
'LIC '	UBYTE	License number

Group ID 'DINF'

This group contains all the information necessary to read the data chunk. Storage mode indicates in what sequence data is stored, first sample one of all channels then sample two of all channels (Channel_Time_based) or first all samples of channel 1 then all samples of channel two (Time_Channel_Based). This must always be the second group in the SEMG file.

ID	Type	Description
'CHAN'	ULONG	Number of channels (NrCh)
'MODE'	UBYTE	Storage mode: 0=Channel_Time_Based, S=sample, ch=channel (i.e. S1: ch1,ch2,ch3 ;S2: ch1,ch2,ch3 ...) 1=Time_Channel_Based, (i.e. ch1: S1,S2,S3 ;ch2: S1,S2,S3 ...)
'TYPE'	UBYTE	Data type: 0=BYTE 1=UBYTE 2=SHORT 3=WORD 4=LONG 5=ULONG 6=FLOAT 7=DOUBLE
'FS '	FLOAT	Sample frequency will be the same for all channels in the data chunk.
'REF '	SHORT	Reference channel.
'SYNC'	SHORT	Synchronisation channel.

'TRIG'	SHORT	Physical trigger channel; This is also the status channel containing status information about the amplifier during the recording.
'TGNR'	SHORT	Triggers are assumed to be coded in the lower 8 bits of the trigger channel.
'ACQM'	SHORT	Three acquisition modes are implemented: 0=continuous data storage; 1=Triggered (epoch) with a constant width determined by WIDT; 2=Triggered (epoch) with a width that depends on the trigger length.
'ACTI'	SHORT	1=Trigger is active high; 2=Trigger is active low.
'WIDT'	SHORT	Width of the epochs, only valid if ACQM is 1.
'PRET'	SHORT	Pre-trigger: number of samples stored before trigger.
'LABL'	UBYTE	NrCh * <character string> label describing the channels, the length of the string is determined by dividing the number of bytes by the number of channels. All labels must be equal of length.
'UFAC'	FLOAT	NrCh * <FLOAT> Multiplication factor to convert to SI Unit
'UTYP'	UBYTE	NrCh * <UBYTE>, SI standard physical units e.g. V,s,A,N,m etc. (see the BIFF.H file for definitions)
'FNHP'	UBYTE	NrCh * <UBYTE>, name of the High Pass filter
'FVHP'	FLOAT	NrCh * <UBYTE>, Value of the High Pass filter [Hz]
'FNLP'	UBYTE	NrCh * <UBYTE>, Name of the Low Pass filter
'FVLP'	FLOAT	NrCh * <UBYTE>, Value of the High Pass filter [Hz]

Group ID 'EXPI'

Experiment and session related information

ID	Type	Description
'NAME'	UBYTE	Character string description of the type of experiment
'EXAM'	UBYTE	Name of the person who performed the examination
'EXRC'	UBYTE	Indicates the type of exercise that was used: 0=unknown, 1=Voluntary isometric contraction, 2=Voluntary dynamic contraction, 3=Motor point stimulated contraction, 4=Nerve stimulated contraction
'DATE'	UBYTE	Date of the experiment, character string: YYYY:MM:DD
'TIME'	UBYTE	Time of the start of the experiment, character string: HH:MM:SS
'RPHS'	UBYTE	Name of the physician who ordered the examination

Group ID 'MEAS'

Measurement related information, these fields may change during one session but not during the measurement.

ID	Type	Description
'TIME'	UBYTE	Time of the start of the measurement, character string: HH:MM:SS
'MUSC'	UBYTE	Name of the muscle examined
'ORFN'	UBYTE	Original file name
'FNUM'	UBYTE	File number of this measurement within one session
'SIDE'	UBYTE	Which side of the patient or subject was examined: 0=unkown, 1=right, 2=left
'TEMP'	UBYTE	Character string, unit should be included. E.g. 30.0 degrees C.
'FORC'	UBYTE	String of characters describing the amount of force
'ANOT'	UBYTE	Annotation

* The chunks 'LABL', 'UFAC', 'UTYP', 'FNHP', 'FVHP', 'FNLP' and 'FVLP' all are vectors with length NrCh. In case of 'LABL', 'FNHP', 'FNLP' the number of bytes reserved for the labels/names is fixed but can vary in next versions. The length can be derived by deviding the total number of bytes of the chunk by the number of channels.

Group ID 'SGRD'

Describing the electrode matrix/grid used for the measurement or session. In principle this can change during one session.

ID	Type	Description
'NPD '	UBYTE	Number of electrodes from proximal to distal
'NML '	UBYTE	Number of electrodes from medial to lateral
'IED '	FLOAT	Inter Electrode Distance [mm]
'FIBD'	UBYTE	Direction of the muscle fibres (needed?)
'POS '	UBYTE	Position of the top left electrode of the matrix 1=Proximal-Medial, 2=Proximal-Lateral 3=Distal-Medial, 4=Distal-Lateral
'CABL'	UBYTE	Position of the cable: 1=top, 2=bottom, 3=left, 4=right
'GRID'	SHORT	Corresponding channel numbers (NrCh*<SHORT>) electrode (positions) that are not connected must contain -1
'SPAT'	UBYTE	Name of the spatial filter filename that was used

Group ID 'PATI'

Patient or subject information, the unique ID is the connection with the systems database.

ID	Type	Description
'UPID'	UBYTE	String of numbers containing a unique patient ID for every patient or subject. E.g.: [<Lic. Nr.> + <YYYYMMDDHHMMSS>].
'GEN '	UBYTE	Gender, 0=unknown, 1=female, 2=male.
'BDAT'	UBYTE	Birthdate of the patient/subject <YYYYMMDD>
'HPID'	UBYTE	Hospital ID, the ID given to the patient by the hospital.
'ANAM'	UBYTE	Anamnesis of the patient/subject.

Group ID 'TRIG'

Trigger information is stored only for the trigger that is used to save an epoch file. Therefore, a file containing the chunk ID TRIG is a non-contiguous file.

ID	Type	Description
'NTRG'	ULONG	Number of triggers in the list.
'TIME'	ULONG	List of sample time moments.
'INDX '	ULONG	List of index values pointing to data chunk connecting the trigger event to the data stored.

Group ID 'EVNT'

Events are stored as a list much like in the trigger chunk. Events are user defined keystrokes or program initiated events such as stimuli events. Event 0 is reserved and is used for electrical stimuli, the value stored is the stimulation current in mA. Events 1-255 are user definable. Events can be stored both in a continuous or in an epoch file.

ID	Type	Description
'NLBL'	UBYTE	Number of labels/defined event types in LABL list.
'LABL'	UBYTE	List of strings defining the events types.
'NRE '	ULONG	Number of events in the list of event numbers, time and index.
'TYPE'	UBYTE	List of event numbers giving the type of event that occurred at TIME.
'INDX'	ULONG	List of index values pointing to the data chunk.
'VAL '	FLOAT	Value that can be added to the eventtype.