GDF - A GENERAL DATAFORMAT FOR BIOSIGNALS

VERSION 1.25

Alois Schlögl[1], Oliver Filz[2], Herbert Ramoser[1], Gert Pfurtscheller[1, 3]

(1998-1999)

[1]Institute of Biomedical Engineering, University of Technology Graz

[2]Institut für Informationsverarbeitung, Österreichische Akademie der Wissenschaften

[3]Ludwig Boltzmann Institute for Medical Informatics and Neuroinformatics, Graz

Keyword(s): biomedical data format, ISO 9000, quality management, general public license GPL, Matlab,

Last modification:

April 15th, 2004: Alois Schlögl, Inclusion of event table

April 29th, 2004: Alois Schlögl, typos in table 2 fixed

May 06th, 2004: Alois Schlögl, update online links to the EDF sites.

Aug 12th, 2004: Alois Schlögl, include table of event codes

Dez 7th, 2004: Alois Schlögl, Samplerate associated with Events saved in Eventtable

Mar 29th, 2005: AS, typos fixed

ABSTRACT

Interdisciplinary and/or multi-centre research projects require a common format for data exchange. The European data format for bio-signals (EDF) is a widely accepted standard. This paper proposes modifications to overcome some limitations of EDF. In addition to 16-bit integer numbers, various other data types including floating point numbers and strings are supported. Moreover, the Y2K problem is addressed and an automatic overflow detection of the data is incorporated. Data can be stored in time-based as well as in channel-based order. Some preparatory measures for a quality management system in recording scientific data in the clinical routine are implemented, e.g. fields for the identification of the equipment provider, serial number, lab, and technician included. Since the modifications are not possible within the present definition of EDF, the proposed format is named "A General Data format for Biosignals" (GDF).

1. INTRODUCTION

Biosignals are currently stored in a wide variety of mostly proprietary data-formats. It is, therefore, difficult for research groups to exchange data. Presently, the European Data Format (EDF, Kemp et al. 1992) is the most widely used format, which overcomes this problem. Software[1] for displaying EDF data is available. It is also used for interdisciplinary research projects like SIESTA (Dorffner 1998) where it is used to exchange data between "engineers" and "clinicians". This works well for raw data (e.g. digitized biosignals) whereas the transfer

[1] http://www.hsr.nl/edf/edf_sftw.htm

of pre-processed data (e.g. Fourier coefficients) is hampered by the fact that EDF supports only one data type, namely, 2 byte signed integer. Pre-processed data, however, is usually available in floating point numbers. The restriction to 2 byte integers requires the conversion of the floating point numbers which introduces problems including the need for a proper mapping of the floating point numbers to integer (linear, logarithmic) and the definition of scaling coefficients for physical and digital minimum and maximum.

Kemp et al. (1998) proposes a method to store floating-point numbers in the EDF in int16 format. The drawback of this solution is that important information might be lost due to the conversion. Other limitations of EDF are that the year information is stored only in 2 digits (Y2K problem) and that no automatic overflow detection is possible (Schloegl et al. 1999). Furthermore, in multi-centre studies the information about the recording devices used is not available because it is not supported by the data format. This information is also required for quality management and should, therefore, be incorporated into the data format specification.

The Info2000 server[2] gives a good survey of available data exchange standards. Alternatives to EDF are EBS[3] and the "File Exchange Format for vital signs"[4] from CEN/TC251 or some scientific data formats like HDF[5], netCDF[6] and CDF[7]. Recently, a new document format

[2] http://www2.echo.lu/oii/en/science.html

[3] http://www.ipb.uni-erlangen.de/biokybernetik/ebsspec.html

[4] ftp://sigftp.cs.tut.fi/pub/eeg-data/standards/cenf060.zip

[5] http://hdf.ncsa.uiuc.edu/

[6] http://www.unidata.ucar.edu/packages/netcdf/

[7] http://nssdc.gsfc.nasa.gov/cdf/

(XML)[8] has been discussed. Medical data are also considered by the ACR-NEMA/DICOM[9] standard.

The EBS is a standard for biosignals, which is quite different to EDF. EBS uses ASCII for storing floating point numbers and allows only one sampling rate for all channels. CEN (1995) is designed to provide a normative standard for the exchange of physiological measurement date between computer systems. Although, currently there is only a draft version (V0.60) from CEN/TC251 available, it can be expected that the specification of CEN will be vast. CEN addresses the presentation and application of vital signs (layer 6 and 7 in the ISO/OSI reference model). The draft of 1997 considers also the data representation on lower levels, but uses e.g. ASCII for storing floating point numbers. The features offered by this standard will be beyond the needs for the exchange of biosignal data. XML is a document description language, which does not support numeric data types. Hence, it is not appropriate for storing biosignal data. Scientific data formats, such as HDF, netCDF and CDF support all common data types, but they are mainly used in the fields of meteorology, geography and climate research. Currently, there is no convention that defines the use of biosignals within those formats. DICOM is a standard for medical images and does not consider one-dimensional data like biosignals.

Many of these formats are promising alternatives. However, it requires a lot of development to adapt one of these formats for the purpose of an interdisciplinary, multi-centre research

[8] http://www.mcis.duke.edu/standards/HL7/sigs/sgml/index.html

[9] http://www.xray.hmc.psu.edu/dicom/dicom_home.html

project. For such research projects, a format definition for transporting measurement data from one lab to another is sufficient (transport-layer4 in the ISO/OSI reference model). A modification of the "simple" EDF format seems to be a comparatively simple solution. The most important drawbacks of EDF are (i) all data must be stored in integer16 format, (ii) dates use a two-digit year, and (iii) overflow detection can not be automated.

From a discussion with Bob Kemp and Peter Jacobi on the - recently founded - independent EDF mailinglist[10], it can be concluded that there is no possibility to overcome these limitations by introducing a revised EDF specification. In this paper we propose GDF, an extension to EDF, which overcomes most of the limitations of EDF.

2. METHOD AND DATA

The following limitations of EDF were identified and are addressed in the proposed modifications:

1) The scaling coefficients, e.g. the physical minimum and maximum are stored in 8*ASCII fields. This can lead to a rounding error or resolution up to 9%, e.g. the difference between -1.0e-09 and -1.1e-09 is 1e-10 that is 9-10% of the actual value. Again for raw data this is not a real problem because the dimension can be stored in the "units" field of the EDF header. But in case of pre-processed data, when these scaling coefficients are generated automatically, this may cause problems.

[10] Topic: "Limitations of EDF and suggestions for improving EDF." http://groups.yahoo.com/group/EDF, Digest [4]-[9], Nov 7th-18th, 1998

2) EDF uses only one data type (integer with 16bit) but today, already 22bit Analogue-Digital Converters (ADC) are available. Besides, pre-processed data - usually floating point numbers - can not be stored without loss of accuracy.
3) Some extensions to EDF have already been published, e.g. for storing floating point numbers (Kemp et al. 1998) and annotations (Velde 1998) but have not been incorporated in the specification.
4) Automatic overflow detection is not possible with the current standard because the digital minimum and maximum “should correspond to these digital extremes", but there is no "must" for doing it. Without standardising it is impossible to perform automatic overflow detection.
5) Year2000 - problem (Y2K)

   The header information uses only two digits for storing the year information. Today, more and more institutions, like hospitals and companies, are concerned about the millenium problem and a new data format should provide a solution for this problem.
6) The blocksize is limited to 61,440 bytes (31,220 samples) and some minor inconsistencies in the specification exist. E.g. the specification states "the duration of one data record … is recommended not to exceed 61,440 bytes". In the EDF FAQ [11]Q11 the excess of this size is classified as an "error" in EDF files. It is not clear whether a "recommendation" is a weak or a strict criterion.
7) EDF provides no support for quality management of recorded data.
8) Event information can not be stored in EDF. A modification of EDF (EDF+) supports the use of event information, but the decoding of the event information requires to read the whole data file. In many cases, this is not practical.

[11] http://www.medfac.leidenuniv.nl/neurology/knf/kemp/edf/edf_faq.htm

The specification is shown below. One goal of the specification was to keep it as similar as possible to the EDF specification. According to the specification of GDF, we adapted the "EDF toolbox for Matlab" and the "EDF Viewer for Matlab"[12] in order to show the feasibility of its implementation.

3. RESULTS

Specification

1. Data is stored in little endian format.
2. The number of data records is of type int64 (8byte); the duration of data records in seconds is a rational number consisting of the numerator (uint32) and a denominator (uint32) (together 8byte). The number of signals is of type uint32 (4byte) and the size and position of the variables has not changed, only its type definition has changed.
3. The "number of samples per record" field is changed from type 8*ASCII*NS to uint32 (4bytes)*NS. The other 4*NS bytes are used for storing the type information (see next point).
4. Various data types are defined (see Table 1). The type information is stored in the variable header as uint32 * NS (NS = number of channels) after "number of samples per record". Each channel in each record is filled up to a full number of bytes. E.g. one channel of type "bit1" with 9 samples per record gives 9bits=1.125bytes per record which requires 2 bytes. Two such channels need 4 bytes.

[12] http://www.dpmi.tu-graz.ac.at/~schloegl/matlab/eeg/edf.html

[Table 1 around here]

5. Physical minimum and maximum are stored as IEEE floating point numbers with double precision (8bytes) instead of char[8] (=8*ASCII).
6. Digital minimum and maximum are of type int64.
7. Furthermore, for all integer types (1-7) the value of digital minimum and maximum must indicate the overflow (saturation) value. In case of floating point numbers, an overflow and underflow is indicated by the value "Infinity" and "-Infinity", respectively. For the character data type (0) no overflow is defined.
8. The recording date and time is changed for Y2K compliance. The format "YYYYMMDDhhmmsscc" will be used. This format needs 16 bytes as in EDF and is at the same position (bytes 169-184). All digits must be numbers ranging from '0' to '9' (ASCII(48) to ASCII(57)). Separating characters are must not be used. YYYY is the 4digit year, MM is the two digit month from 01 to 12, DD is the day from 01 to 31, hh hours from 00 to 23, mm minutes 00 to 59, ss are seconds from 00 to 59, and cc are 0.01s from 00 to 99. If cc is unspecified two blanks (ASCII 32) must be used.
9. The Version field is of type char[8] and is at the beginning of the file. The first three letters are always 'GDF'. During the testing period of GDF writing functions, 'GDF 0.11' should be used. It is planned that later when routines are generating valid GDF the field will contain "GDF 1.00"
10. The length of the header is defined in the field "number of bytes in header record". The number of bytes must be at least 256*(1+NS).
11. The table of events is stored after the data section. The starting position of the event table (event table position ETP) can be calculated in the following way.

ETP = Length_of_header + number_of_blocks * bytes_per_block. (1)

In case ETP is the filesize or large, no eventtable is included. The specification for the event table is shown in Table 2. The value of the first byte (mode of event table) can be "1" or "3". A value of "1" indicates that the event table contains event-type and position, only. A value of "3" indicates that the position, type, associated channel and the duration of the event is stored. The next three bytes are 24bit-integer value indicating the sampling rate associated with the event position and duration. Dividing the event position (and duration) by this value yields the position (duration) in seconds. The next 4 bytes store the number of events in a 32-bit integer number. Then the position of all events is saved in 32bit integers, followed by the event type as 16-bit integers. If the mode of the event-table has value 3, it follows the channel number associated with each event (a value of 0 indicates the event refers to all channels) and the duration of each event.

[Table 2 around here]

The encoding of the various event types is defined in Table 3.

[Table 3 around here]

Recommendation and Remarks:

It is recommended to have only one data type (e.g. int16, or float) in one GDF file. Current software implementation does support only GDF files with one data format.

The physical and digital minimum and maximum values are usually not used for the type 0 (char), 16 (float32) and 17 (float64). "Infinity" and "-infinity" can indicate the overflow and underflow of floating point numbers, respectively in the data section. In char-type channels no overflow detection is possible. The difference between char and int8 is that char is interpreted as character while int8 is assumed to contain numbers. If no scaling is needed, the maxima and minima can be set to 1 and 0, respectively.

The "equipment provider identification" (EP-key), "laboratory identification" (Lab-key) and the "technician identification" (T-key) fields might be useful for a posterior quality control. The default values are always 0x2020202020202020 (i.e. blank characters). It is planned that providers of recording equipment can get a unique EP-key and laboratories (e.g. for sleep) a unique Lab-key. The technician identification should indicate the responsible technician that performed the recording. It can be defined by each laboratory and must be unique. In case an EP-key is used, the first 12 bytes of the "reserved" field can be used to store the serial number of the equipment. These features can be used to certify the data generation according to the quality management standard ISO 9000.

Programming languages without support for 8 byte integer (int64 and uint64) should read and write two numbers of type int32. The first one (int32L) contains the actual number the second one (int32H) is usually (int32) 0 = 0x00000000, in case of a negative number it is (int32) (-1) = 0xFFFFFFFF.

The format definition of GDF is nearly as simple as the definition of EDF, as can be seen in Table 2. The numeric data in the header is stored using a numeric data format instead of ASCII. Various data types are possible in the data section. The big advantage of GDF is that pre-processed data - usually in float format - can be stored without loss of accuracy and it is not necessary to consider proper formatting of the scaling coefficient (and the loss of accuracy due to rounding effects at data conversion). The same precision that is used internally in computers can be used in GDF.

The proposed format specification was successfully implemented in Matlab®. The software implementation required only minor changes to upgrade from EDF to GDF. Several conversions from strings to numbers were replaced by using the appropriate type within the "fread"-function. The major change was considering the size of the different data types. The software is available "online" (see "EDF/GDF toolbox for Matlab" and "EDF/GDF Viewer for Matlab"[13]) and is "free" under the terms of the "Free Software Foundation"[14]. Hence, it is published under the term of the "General Public License" (GPL).

The event table is defined only, if the number of records is known. Consequently, no eventtable can be stored in an ongoing recording, because the number of records is usually not known (NRec == -1). The eventtable can be only written to the GDF file, once the recording length (i.e. data size, number of blocks) is known.

4. DISCUSSION AND CONCLUSION

The present definition of GDF allows up to 2^32 (ca. 4e+9) channels, up to 2^63 (ca. 9e+18) data records, and up to 2^32 (ca. 4e+9) samples per channel and per record. This makes it possible to store the data in an arbitrary order. It is possible to use either one record, where

[13] http://www.dpmi.tu-graz.ac.at/~schloegl/matlab/eeg/gdf/

[14] http://www.fsf.org/

the data are stored in channel-based order, or to define one block as one sample. In latter case the data is stored in time-based order. This is possible because the length of one record is defined as a rational number, so the duration can be one over the sampling rate (1/Fs). But also the EDF recommendation can be used where "the duration is a whole number of seconds" and the number of samples of one record "… must not exceed 61440 bytes".

GDF overcomes many limitations of the EDF format and includes several new features. The Y2K problem is solved using a 4-digit number for the year. The rounding errors in the header information as well as in the data section do not matter anymore, due to the use of ASCII/integer have been removed. Various data types are supported and automatic overflow detection is possible, because of the strict definition of the digital minimum and maximum values. Moreover, the support of event information is included.

Several measures for the implementation of a quality management system are included in this definition. The patient identification and the recording time were already defined in EDF. In GDF it is possible to store identification and serial number of the equipment, the laboratory and the responsible technician. Possible overflow and saturation effects of the amplifier and the ADC are described by the extreme values. Parameters that influence the data quality can be documented, which is a prerequisite for a quality management system of scientific data recorded within the clinical routine.

All features of EDF are implemented in GDF. It can be said that GDF is (upwards) compatible to EDF in the sense that every EDF file can be converted to GDF without loss of any information. Overall, the changes to the EDF format are relatively small. Routines for reading and writing of GDF files in Octave and Matlab are implemented in the open source package BIOSIG (Schlögl, 2003-2004). It can be assumed, implementing these routines in other programming languages is not too difficult.

5. ACKNOWLEDGMENT

The SIESTA project Biomed-2 BMH4-CT97-2040 was funded by the European Commission, DG XII and by the Austrian Ministry for Science and Technology (BMWF) GZ 650 133/2-VIII/6/97. We thank Peter Jacobi for their contribution to the discussion and Sabine Ferstl for the proofreading.

6. REFERENCES

[1] CEN (1995) Vital signs Information Representation Version 1.2, Interim Report - CEN/TC251/WG5/N95-3, European Committee for Standardisation, Brussels.

[2] Dorffner G. (1998): Towards a new standard of modeling sleep based on polysomnograms - the SIESTA project. *Proc. ECCN 98, Ljubljana, Electroenceph. clin. Neurophysiol. 106(Suppl. 1001): 28*

[3] B. Kemp, A. Värri, A.C. Rosa, K.D. Nielsen and J. Gade (1992): A simple format for exchange of digitized polygraphic recordings. *Electroenceph. clin. Neurophysiol.,* 82: 391-393.

[4] Kemp B, Penzel T, Värri AO, Sykacek P, Roberts SJ, Nielsen KD. (1998): EDF: a simple format for graphical analysis results from polygraphic SIESTA recordings. *J Sleep Research* 7, suppl. 2, 132.

[5] A. Schlögl, B. Kemp, T. Penzel, D. Kunz, S.-L. Himanen, A. Värri, G. Dorffner, G. Pfurtscheller (1998): Quality Control of polysomnographic Sleep Data by Histogram and Entropy Analysis. *Electroenceph. clin. Neurophysiol.* submitted.

[6] van de Velde M, van den Berg-Lenssen MM, van Boxtel GJ, Cluitmans PJ, Kemp B, Gade J, Thomsen CE, Varri A. (1998): Digital archival and exchange of events in a simple format for polygraphic recordings with application in event related potential studies. *Electroencephalogr Clin Neurophysiol.* 106(6):547-51.

[7] A.Schlögl, The BIOSIG project. 2003-2004. online available at: http://biosig.sf.net/.

Table 1: Data types, memory requirement, coding scheme.

channel type	size per sample [bytes]	code
char	1	0
int8	1	1
uint8	1	2
int16	2	3
uint16	2	4
int32	4	5
uint32	4	6
int64	8	7
float32	4	16
float64	8	17
bitN	N/8	255+N
ubitN	N/8	511+N
examples:		
Boolean 0=false, 1=true	1/8	256
ubit1	1/8	512
bit12	3/2	267
bit24	3	279

Table 2: Overview of the header definition

FIXED HEADER	remark	Position Start:End [bytes]	Bytes	Type
Version identification (GDF 0.12)		0	8	char[8]
Patient identification (P-id)		8	80	char[80]
Recording identification (R-id)		88	80	char[80]
Startdate and -time of recording (YYYYMMDDhhmmsscc)		168	16	char[16]
number of bytes in header record		184	8	int64
Equipment Provider identification (EP-id)	NEW	192	8	uint64
Laboratory Identification (Lab id)	NEW	200	8	uint64
Technician Identification (T-id)	NEW	208	8	uint64
reserved / serial number		216	20	char
number of data records (-1 if unknown)		236	8	int64
Duration of a data record, as a rational number in seconds (first the numerator, secondly the denominator.		244	8	uint32[2]
NS: number of signals (channels)		252	4	uint32
VARIABLE HEADER		256		
Label		256	NS*16	char[16]*NS
Transducer type		256+16*NS	NS*80	char[80]*NS
Physical dimension		256+96*NS	NS*8	char[8]*NS
Physical minimum		256+104*NS	NS*8	float64*NS
Physical maximum		"	NS*8	float64*NS
digital minimum		"	NS*8	int64
digital maximum		"	NS*8	int64
Pre-filtering		"	NS*80	char[80]*NS
nr: number of samples in each record	size only 4*NS	"	NS*4	uint32*NS
Channel TYPE	NEW	"	NS*4	uint32*NS
reserved		"	NS*32	char[32]*NS
DATA RECORD		256*(NS+1)		
nr samples from channel [1] of TYPE[1]				Type[1]
nr samples from channel [2] of TYPE[2]				Type[2]
nr samples from channel [3] of TYPE[3]				Type[3]
nr samples from channel [NS] of TYPE[NS]				Type[NS]
EVENT TABLE	NEW	ETP		
mode of event table can be 1 or 3.		ETP + 0	1	uint8
Samplerate associated with Event positions.		ETP + 1	3	uint8[3]
Number of events N		ETP + 4	4	uint32
Position [in samples]		ETP + 8	N*4	uint32
Type		ETP + 8+N*4	N*2	uint16
Channel [optional]	Only if mode is 3	ETP + 8+N*6	N*2	uint16
Duration [in samples, optional]	Only if mode is 3	ETP + 8+N*8	N*4	uint32

Table 3: Table of event codes. The most recent version of this table will be available from biosig/t200/eventcodes.txt [7].

```
### Table of event codes.
# This file is part of the biosig project http://biosig.sf.net/
# Copyright (C) 2004 Alois Schloegl <a.schloegl@ieee.org>
# $Revision: 1.3 $
# $Id: eventcodes.txt,v 1.3 2004/06/17 17:08:54 schloegl Exp $
#
### table of event codes: lines starting with # are omitted
### add 0x8000 to indicate end of event
#
### 0x010_     EEG artifacts
0x0101  artifact:EOG
0x0102  artifact:ECG
0x0103  artifact:EMG/Muscle
0x0104  artifact:Movement
0x0105  artifact:Failing Electrode
0x0106  artifact:Sweat
0x0107  artifact:50/60 Hz mains interference
0x0108  artifact:breathing
0x0109  artifact:pulse
### 0x011_     EEG patterns
0x0111  eeg:Sleep spindles
0x0112  eeg:K-complexes
0x0113  eeg:Saw-tooth waves
### 0x03__     Trigger, cues, classlabels,
0x0300  Trigger, start of Trial  (unspecific)
0x0301  Left - cue onset (BCI experiment)
0x0302  Right - cue onset (BCI experiment)
0x0303  Foot - cue onset (BCI experiment)
0x0304  Tongue - cue onset (BCI experiment)
0x0306  Down - cue onset (BCI experiment)
0x030C  Up - cue onset (BCI experiment)
0x030D  Feedback (continuous) - onset (BCI experiment)
0x030E  Feedback (discrete) - onset (BCI experiment)
0x0311  Beep (accustic stimulus, BCI experiment)
0x0312  Cross on screen (BCI experiment)
0x03ff  Rejection of whole trial
### 0x040_     Sleep-related Respiratory Events
0x0401  Obstructive Apnea/Hypopnea Event (OAHE)
0x0402  Respiratory Effort Related Arousal (RERA)
0x0403  Central Apnea/Hypopnea Event (CAHE)
0x0404  Cheyne-Stokes Breathing (CSB)
0x0405  Sleep Hypoventilation
### 0x041_     Sleep stages according to Rechtschaffen&Kales
0x0410  Wake
0x0411  Stage 1
0x0412  Stage 2
0x0413  Stage 3
0x0414  Stage 4
0x0415  REM
### 0x050_     ECG events
0x0501  ecg:Fiducial point of QRS complex
0x0502  ecg:P-wave
0x0503  ecg:Q-point
0x0504  ecg:R-point
0x0505  ecg:S-point
0x0506  ecg:T-point
0x0507  ecg:U-wave
### 0x____     OTHER
0x0000  No event
```