

LASER INTERFEROMETER GRAVITATIONAL WAVE OBSERVATORY (LIGO)
&
THE VIRGO EXPERIMENT

Document Type Technical Note	LIGO-T970130-D-E: VIRGO-SPE-LAP-5400-102	08 Dec 2000
Specification of a Common Data Frame Format for Interferometric Gravitational Wave Detectors (IGWD)		
LIGO Data Group VIRGO Data Acquisition Group		

VIRGO CNRS/INFN
Traversa H di Via Macerata
56021 S. Stefano a Macerata
Cascina (PI), Italy
Phone (39) 050 752 521
Fax (39) 050 752 550
E-mail: virgo@lapp.in2p3.fr

California Institute of Technology
LIGO Project - MS 18-34
Pasadena CA 91125
Phone (1) 626 395 2129
Fax (1) 626 304 9834
E-mail: info@ligo.caltech.edu

Massachusetts Institute of Technology
LIGO Project - MS 16NW-145
Cambridge, MA 01239
Phone (1) 617 253 4824
Fax (1) 617 253 7014
E-mail: info@ligo.mit.edu

URL: <http://www.ligo.caltech.edu/>
URL: <http://www.virgo.infn.it/>

Table of Contents

1	Introduction.....	9
1.1.	Purpose.....	9
1.2.	Scope.....	9
1.3.	Applicability	10
2	Applicable Documents.....	10
3	List of Definitions, Acronyms and Symbols	10
4	IGWD Frame Structure.....	11
4.1.	Overall.....	11
4.2.	Data types.....	11
4.3.	Composition of files and frames written to media.....	13
5	Rules for Revision.....	33
5.1.	Frame Formats	33
5.2.	Frame Library Software.....	33
5.3.	Requests for changes.....	33
5.4.	Change control	33
Appendix A	Assignment of Data Quality (DQ) Bits	35
Appendix B	Data Compression Schemes.....	36

List of Tables

Table 1:	Applicable Documents.....	10
Table 2:	Acronyms, definitions and symbols used in this document.....	10
Table 3:	IGWD frame data types as written to media.....	11
Table 4:	Byte-level descriptor for a file header	13
Table 5:	Common Elements of All Frame Structures.....	16
Table 6:	Frame Structure Header Data.....	17
Table 7:	Frame Structure Element Data.....	18
Table 8:	Frame Header Structure Definition.....	18
Table 9:	ADC Data Structure Definition	20
Table 10:	Detector Data Structure Definition,	21
Table 11:	End of File Data Structure Definition.....	22
Table 12:	End of Frame Data Structure Definition.....	22
Table 13:	History Structure Definition	22
Table 14:	Message Log Data Structure Definition	23
Table 15:	Post-Processed Data Structure Definition.....	23
Table 16:	Raw Data Structure Definition.....	24
Table 17:	Serial Data Structure Definition.....	24
Table 18:	Simulated Data Structure Definition.....	25
Table 19:	Simulated Event Data Structure Definition	25
Table 20:	Static Data Structure Definition.....	26
Table 21:	Summary Data Structure Definition	27
Table 22:	Table Data Structure Definition.....	27
Table 23:	Table of Contents Data Structure Definition	28
Table 24:	Trigger Data Structure Definition.....	30

Table 25:	Vector Data Structure Definition	31
Table 26:	Bit Assignment of dataQuality Word in FrameH	35
Table 27:	Compression Schemes Supported.....	36
Table 28:	FrVect Data Types	37

List of Figures

Figure 1	Schematic representation of data organization within a file	14
Figure 2	Schematic representation of frame structures and relative pointers (not all possible structures are shown)15	
Figure 3	Byte-level graphical representation of a frame	32

SPECIFICATION REVISION HISTORY			
REVISION	AUTHORITY	PAGES AFFECTED	Item(s) Affected
A	Initial release	-	-
01	Pending release as Rev. B	14 15 19 20 20 22	Table 8, added row 16 (overRange). Table 9, last row, delete FrDetector* element of structure. Table 18, row 5, change variable name to <i>type</i> . Table 18, rows 11, 12, interchange variable names (unitY, unitX). Table 18, footnote a, change {class#...} to {type#...}. Figure 3, renumber 3rd bytes of various elements to reflect changes described above.
02	Pending release as Rev. B	5 7 19 19 19 19 19 19 20	First version of new format changed to 3 Table 4, edited description of Byte 5 (row 2) Table 17, delete row 10, FrStatData* element of structure Table 18, add element class INT_2U, to flag data compression. Add footnote to describe implemented compression algorithms. This becomes footnote a on page 20. Table 18, change class of variable <i>type</i> to INT_2U Table 18, introduce new element, class INT_4U, variable name <i>nBytes</i> . Needed to support data compression. Table 18, footnote a becomes footnote b. Table 18, added footnote a; footnote a becomes footnote b.
03	Pending release as Rev. B	5 6 6 20 20	Table 2, Added time standard acronym definitions. Table 3, introduce a footnote (a) explaining that any of the data classes may be used as a list of such elements. The notation for this shall be *type, just as in C/C++. The number of elements in the list will be a piece of information contained elsewhere within the structure where this list object appears. At present it is exclusively used in the structure FrVect. By doing this, previous footnote a becomes footnote b. Table 3, correct C/C++ Data Type entry for INT_4U. int_U -> unsigned int. Table 18, exchange the Descriptor & Comments fields for the entries unitX and unitY (comments are reversed) Table 18, modify the compression types by editing footnote a.

SPECIFICATION REVISION HISTORY			
04	Pending release as Rev. B	6	Table 2, added in footnote b the definition of the leap second.
		13	Table 7, changed variable names UTimeXX -> GTimeXX to reflect change of time basis to GPS (atomic time). Added the new variable ULeapS, giving the time offset between UTC and TAI. Added parenthetical comment to localTime comment field.
		14	Table 8, corrected typographical error: nbits->nBits.
		21	Table 19, changed variable name from UTimeXX->GTimeXX to reflect change of time basis to GPS (atomic) time.
		22	Table 21, corrected typographical errors: nframes->nFrames; nbytes->nBytes.
05	Pending release as Rev. B	5	Section 1.1, Purpose, introduced discussion on primary intent and evolution of frames with time.
		20	Table 17, added an element of class PTR_STRUCT to accommodate use of multiple detector structures within one frame. See also new footnote a to Table 17 and text in section b.13.
		24	Figure 3, corrected various errors in previous versions. Some of these reflect changes in structure definitions which have been introduced to date.
B	DCN E970066-00-E	13	Table 7, parameter dt, frame length, redefined as seconds.

SPECIFICATION REVISION HISTORY			
06	Pending release as Rev. C	All	<p>Miscellaneous wording changes, reordering of structure definitions by alphabetical order, etc. Created Table 5 to show the first three elements common to all structures.</p> <p>Added pointers in various structures to newly defined structures (see below).</p> <p>7 Section 4, Changed Frame Format Version Number from 3 to 4.</p> <p>11 Altered Figure 1 to show Table of Contents structure</p> <p>14 Altered structure of FrameH to include dataQuality; changed run number to INT_4S.</p> <p>17 Altered FrDetector to remove reference to arm length to accommodate multiple interferometers and bars.</p> <p>18 Altered FrEndOfFile to include pointer to the new FrTOC structure.</p> <p>21 Introduced FrSimEvent to capture input parameters to a simulated event.</p> <p>22 Altered FrStatData to include a data representation variable.</p> <p>23 Introduced FrTable structure to accommodate tabular data formats.</p> <p>24 Introduced FrTOC structure to capture index (table of contents) into a file of frames.</p> <p>26 Altered FrTrigData to include new parameters to capture trigger duration.</p> <p>27 Altered FrVect to include offset, "startX", for value of parameter "x".</p> <p>31 Added Appendix A to list dataQuality bit values</p> <p>32 Added Appendix B to replace two long footnotes to FrVect.</p>
C	DCN E000023-00-E	All	<p>All changes described for Rev. 06 have been incorporated. In addition several other miscellaneous changes and corrections were included.</p>
07	Pending as release Rev. D	10	<p>Section 4.2, called out explicitly that all variables are case sensitive.</p>
		14	<p>Table 5, changed wording in the use of instance counters to be more precise.</p> <p>Sectionn 4.3b, changed "frame" to "file" in text describing uses of FrSh and FrSE.</p>
		19	<p>Section 4.3b.6, removed unneeded comma in text.</p> <p>Table 12, changed FrEndofFrame run number to INT_4S to match change made in FrHeader</p>
		23	<p>Table 20, changed wording in footnote to be more precise.</p>
		24	<p>Table 22, added pointer to next frTable structure in the definition of frTable.</p>
		33	<p>Table 27, modified table to include big/little endian machines and changed wording in table to be more precise. Also changed the numbering scheme by +1 for compression modes > 256. This was done to be consistent with text in the first paragraph. and to include uncompressed data for both platform types.</p>

SPECIFICATION REVISION HISTORY			
8	Pending as release Rev. D	16	Section 4.3, b, added paragraphs at end to discuss how attributes of structures must appear contiguously and how structures themselves appear hierarchically.
		18	Table 7, called out uniqueness of FrSE element names.
		26	Table 20, called out version numbering convention.
		27	Table 22, called out that name of table structures need not be unique.
			Section 4.3b.18, specified how and what structures appear in FrTOC.
		28	Table 23, changed the following FrTOC attributes: ULeapS->INT_2U; dt->REAL_8; runs->INT_4S; position->specified in bytes; added nFirstADC, nFirstSer, nFirstTable, nFirstMsg to index these key structures; added channelID, groupID to ADC index; corrected length of FrTrig and FrSimEvt position lists.
		31	Table 25, called out that name of vector structures need not be unique.
9	Pending as release Rev. D	21	Section 4.3b.6, changed wording to reflect that FrTOC comes after all frames in a file.
		27	Table 23, changed UleapS to INT_2S. Changed positionH to refer to FrameH. Changed or added FrStatData elements (nameStat, tStart, tEnd, positionStat). Changed or added FrADCData elements (channelID, groupID, positionADC). Changed or added FrProcData elements (nameProc, positionProc). Changed or added FrSimData elements (nameSim, positionSim). Changed or added FrSerData elements (nameSer, positionSer). Changed or added FrTrgiData elements (nameTrig, position Trig). Changed or added FrSimEvt elements (nameSimEvt, positionSimEvt). Changed or added FrSummary elements (nameSum, position Sum).

SPECIFICATION REVISION HISTORY			
10	Pending as release Rev. D	25	Table 20, added footnote noting that the combination of four elements for each FrStatData must be uniquely specified.
		27	Table 23, Made nStat variables unique. Moved location of FrSummary elements within the table.
11	Pending as release Rev. D	18	Table 8, changed variable simEvtData to simEvent
		24	Table 19, replaced word "trigger" by "event" throughout comments.
		26	Table 22, renamed linked list pointer from "table" to "next".
		29	Table 24, changed timeBefore and timeAfter datatypes to REAL_4.
D	DCN E000550-00-E	All	<p>All changes described for all revisions after Release C have been incorporated.</p> <p>Table 3, added footnote c, regarding STRING terminations.</p> <p>In addition several other miscellaneous changes and corrections were included.</p>

1 INTRODUCTION

The LIGO/VIRGO Data Frame Format for interferometric gravitational wave detectors (IGWD) is a collaborative effort which has evolved out of a frame format design originated within the VIRGO Project. This specification has evolved out of the recognition for the need of a standard definition of this frame format which can be used by individual (international) projects wishing to adhere to a common representation of data produced by IGWD. It is hoped that by using a standard design for data, future collaborative analyses of data taken by different projects can be promoted more easily.

The predominant type of data stored in frames is time series data of arbitrary duration. It is possible, however, to encapsulate in frame structures other types of data, e.g., spectra, lists, vectors or arrays, etc. However, the primary purpose of this specification is to address how (raw) data are written into frame structures.

It is the intent of the Projects collaborating internationally on this frame definition and standardization to promote a continued evolution of the standard through formal configuration control, scheduled updates and releases, and code maintenance. A Consortium or Working Group with representatives from each of the participating projects will be formed and will have formal control over the contents of this specification as it evolves.

1.1. Purpose

This specification formally defines the IGWD Frame for data structuring and exchange that is to be used where applicable (see below).

The primary intent of the Frame Format is to capture the informational content of real-time data acquisition systems associated with interferometric gravitational wave detectors to efficiently archive that information.

As experience in using frame data within the gravitational wave community develops, the informational content of Frames will need to grow to support newly identified needs through the addition of new structures. It is the intent of this specification to present a foundation to frame-based data which will only be modified to remove errors and to fill in missing components; new ways of organizing future data needs will be accommodated through the addition of new structures, rather than the evolution of existing (and working) structures. In doing this, it will be more easily possible to support older frame formats at the same time while accommodating newer ones within the same frame libraries.

1.2. Scope

Frames are written assuming IEEE/ASCII compliant hardware and software are used to read/write data.

This standard specifies the organization and content of IGWD Frame data sets, including the C structures which create a frame:

This specification also defines rules to which new extensions and revisions are required to conform.

1.3. Applicability

LIGO and VIRGO will work to ensure that all developed hardware and software systems will support IGWD Frames (“Frames”) for the interchange of binary data. All participating projects will acquire their data in Frames and make their data available, when and if data exchanges occur, in Frame formatted media. There is no restriction in media type. Reduced data still containing time-series representation of IGWD datastreams shall be made available in Frames. The Frame format shall be available in the public domain, subject only to the standards and controls defined herein.

2 APPLICABLE DOCUMENTS

Table 1: Applicable Documents

Document Identifier	Description	Comments
VIRGO-MAN-LAP-5400-103	Frame Library Users Manual	
T970100	LIGO System Software Design Issues	
T970140	LIGO Systems Software Specifications and Design Requirements	In process

3 LIST OF DEFINITIONS, ACRONYMS AND SYMBOLS

Table 2: Acronyms, definitions and symbols used in this document

Acronym	Definition
ASCII	American Standard Code Information Interchange
ANSI	American National Standards Institute
C/C++	Programming languages
GPS ^a	Global Positioning System Time
IEEE	Institute of Electrical and Electronic Engineers
IGWD	Interferometric Gravitational Wave Detector(s)
LIGO	Laser Interferometric Gravitational Laboratory
VIRGO	VIRGO Experiment sponsored by CNRS (France) - INFN (Italy)
TAI	International Atomic Time
UT	Universal Time (GMT + 12 ^h)
UTC ^b	Universal Coordinated Time

- a. GPS time uses atomic time as its basis and equals TAI within an offset defining the GPS epoch. $GPS = TAI + 19.000^s$. GPS uses as its origin the standard epoch, 1980 January 6.^{d0}, Julian date (JD) 2 444 244.5. JD = 0 corresponds to 4713 B.C., January 1.^{d5}.

- b. UTC uses the atomic second as its basis, but to keep UTC close to UT and civil time, integer leap seconds (of either sign) are added to UTC at distinct epochs. GPS and UTC were coincident at the GPS standard epoch, 1980 January 6.^{d0}. The integer number of leap seconds, N_S , between TAI and UTC in the present epoch is defined by the relationship:
- $$\text{TAI} - \text{UTC} = N_S \cdot 1^s.000.$$

4 IGWD FRAME STRUCTURE

This document specifies Frame Format Version Number 4, valid with the release of this document. Subsequent updates to this document will indicate in this paragraph the valid Format Version Number.

4.1. Overall

A Frame is a grouping of multiple C structures composed of the following elements:

- Frame header
- Dictionaries permitting reconstruction of the C structures via reading of frame data off media
- Frame history comment
- Detector/instrumental configuration
- Raw fast data
- Serial data
- Event trigger data
- Post-processed/derived data
- Simulated data
- etc.

4.2. Data types

The following C data types are used in frames

Table 3: IGWD frame data types as written to media

Data Class ^a	C/C++ Data Type	Length ^b (Bytes)	Comments
CHAR	char	1	Character
CHAR_U	unsigned char	1	Unsigned character
INT_2S	signed short	2	Signed integer, Range: $(-2^{15}, 2^{15}-1)$
INT_2U	unsigned short	2	Unsigned integer, Range: $(0, 2^{16}-1)$
INT_4S	int	4	Signed integer, Range: $(-2^{31}, 2^{31}-1)$
INT_4U	unsigned int	4	Unsigned integer, Range: $(0, 2^{32}-1)$
INT_8S	long	8	Signed integer, Range: $(-2^{63}, 2^{63}-1)$

Table 3: IGWD frame data types as written to media

Data Class ^a	C/C++ Data Type	Length ^b (Bytes)	Comments
INT_8U	long	8	Unsigned integer, Range: (0, 2 ⁶⁴ -1)
REAL_4	float	4	IEEE-defined single precision floating point number
REAL_8	double	8	IEEE-defined double precision floating point number
Composite Data Types			
STRING	char []	< 65536	Character string; first 2 bytes are interpreted as an INT_2U for length of string, exclusive of these two bytes but inclusive of the “\0” string terminator ^c .
PTR_STRUCT	void*	4	Pointer to a structure. This object replaces an actual pointer when the structure is written to media (pointer address would be meaningless). Instead, a pair of INT_2U are written, to be interpreted as (data class, data instance) => (INT_2U, INT_2U) NULL == (0, 0) The frame reading software uses these two variables to rebuild a pointer table when the frame is read into memory.
COMPLEX_8	Pair of REAL_4	8	Complex real number, two single precision floats, stored as a pair: (real, imaginary)
COMPLEX_16	Pair of REAL_8	16	Complex real number, two double precision floats, stored as a pair: (real, imaginary)

- The classes {INT_2S,..., STRING} inclusive, may also be used as lists of such objects. The notation in the specification will be to precede the data class with an asterisk (*): e.g., *INT_2U implies a list or array of INT_2U objects. The information on the length of the list will appear elsewhere within the header of the structure using such objects.
- Note: lengths indicated are the minimum lengths for these types; actual lengths must be determined by the encoding of types in the file header. Software assumes only ASCII character set usage.
- Note that ALL strings must have a “\0” terminator; even NULL strings.

Byte ordering of all integer and real types is determined by hardware and compiler options. To allow for optimal performance, the actual byte ordering in these frame data types will be free to be either big-endian (most significant byte first) or little-endian (least significant byte first). The actual ordering is encoded in the file header. It is required of the software to transparently determine and allow for translation between these conventions as needed on specific platforms. The ordering applies to individual elements of composite structures, but does NOT apply to ordering of composite elements themselves.

Code which writes and uses frames shall use the capitalized casts to the specified class variable definitions to ease the transportability of the code among platforms and operating systems to the greatest extent possible. I/O methods employed within frame libraries shall be written in a POSIX.1 compliant style. Frame structure assumes a minimum 32-bit computer architecture.

The structures and all supporting libraries shall conform to recognized standard C/C++ usage. The controlling standard for the C language is ANSI C. When the ANSI C++ standard is available, it shall be adopted for any C++ components to the frame supporting libraries. Note that all variables are cAsE sensitive.

4.3. Composition of files and frames written to media

A file consists of binary data. Figure 1 shows a schematic representation of a data file as written to media. Figure 2 presents the pointer/structure schema upon which the frame is built. Note that structures stored in RAM have pointer elements associated with them (which are needed for memory allocation and usage in the machine) which are not written as addresses to media, but rather as PTR STRUCT identifiers.

A file contains a header, frames, and an end of file:

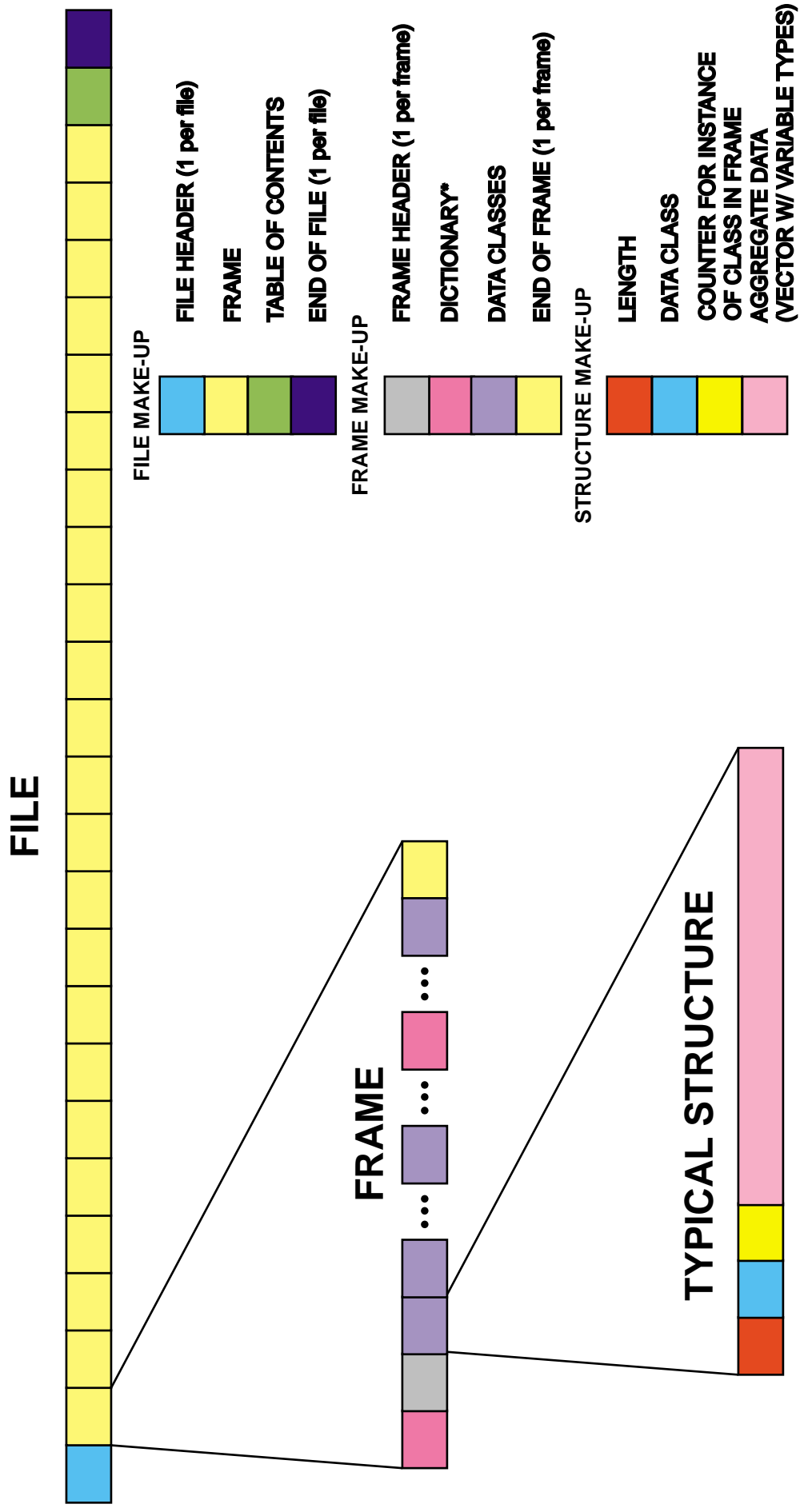
File:{FileHeader, Frame₁, Frame₂,..., Frame_m,... Frame_N, EndOfFile}

a. File Header

Table 4: Byte-level descriptor for a file header

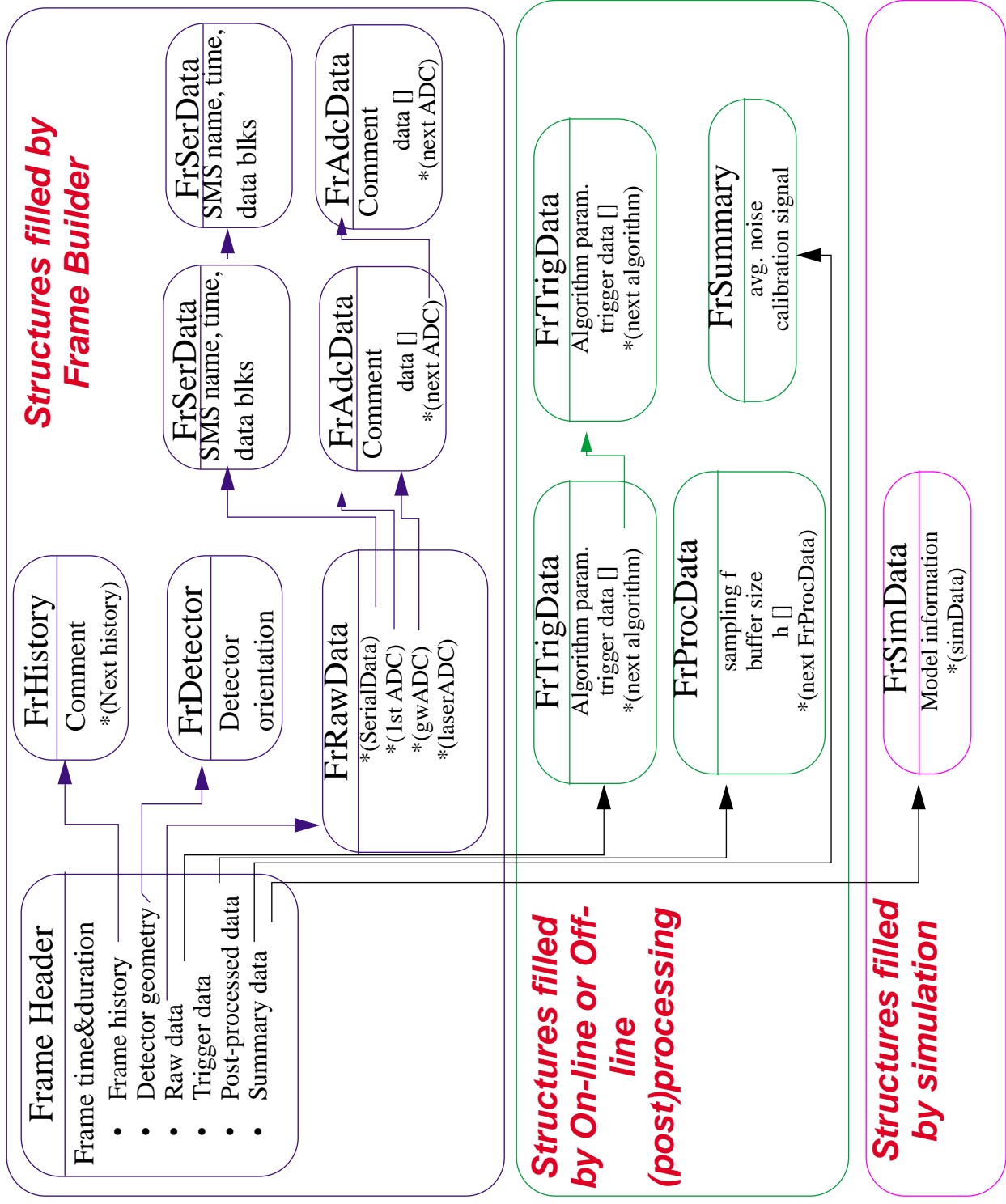
Byte(s)	Description
0 - 4	ASCII Characters "IGWD" (string terminated with a \0) or other identifier of originator of frame file
5	Data format version for this file (4 as of this release of the document)
6	Frame Library minor version number for software used to write this file.
7	Size of an INT_2 on originating hardware
8	Size of an INT_4 on originating hardware
9	Size of an INT_8 on originating hardware
10	Size of a REAL_4 on originating hardware
11	Size of a REAL_8 on originating hardware
12 - 13	2 bytes containing 0x1234. This is used to determine byte order differences between writing hardware and reading hardware
14 - 17	4 bytes containing 0x12345678. This is used to determine byte order differences between writing hardware and reading hardware
18 - 25	8 bytes containing 0x123456789abcdef. This is used to determine byte order differences between writing hardware and reading hardware
26 - 29	IEEE single precision floating point representation of $\pi = 3.1415926535897932384\dots$
30 - 37	IEEE double precision floating point representation of $\pi = 3.1415926535897932384\dots$
38 - 39	ASCII 'A' 'Z' to check for IEEE ASCII hardware standard

Figure 1 Schematic representation of data organization within a file



* Dictionary structure behavior is unique in that:
 1. It precedes header for first frame of file;
 2. Dictionary is built up incrementally as additional

Figure 2 Schematic representation of frame structures and relative pointers (not all possible structures are shown)



b. Frame: {DictionaryStructure, FrameHeader, DictionaryStructure, Structure₁, Structure₂,..., DictionaryStructure, ..., Structure_N, FrameEnd}

Data are written as frames which are composed of structures. There are a number of unique structures from which a frame may be built; not all possible structures appear in a particular frame.

Any structure which is used for the first time in a file requires that it be preceded by a corresponding dictionary-type structure describing it. Thus, each of the structure types introduced in paragraph **b.3** and following below must be described on media by one (and only one) dictionary structure containing the sequence: {FrSH, FrSE, ..., FrSE}. There are as many elements FrSE in the sequence as there are elements of a structure in its corresponding table (excepting the first three rows of each table, which are used as structure headers -- see Figure 1 and Table 5.). These dictionary structures are normally written immediately preceding the first occurrence of the corresponding structure.

Structure class numbers 1 and 2 correspond to FrSH and FrSE. The primitives FrSH and FrSE are themselves not described by dictionary structures on media, and must be known *a priori* to interpret a file on media. These primitive structures shall be maintained across revisions of frame-writing software libraries to maintain backwards compatibility with data.

With the exception of FrSh, FrSE, FrameH, FrEndOfFile, FrEndOfFrame, FrRawData and FrStatData, all other structures appearing in a frame must be referenced by other structures.

Except for FrVector and FrTable, all instances of other structures which may appear more than once in a frame must be assigned a unique name variable.

All attributes appear in the order in which they are listed in the following tables. All structure attributes must be written contiguously to file. Ordering of structures is hierarchical so that all lower level structures which are referenced by a given structure must appear contiguously before the next structure appears at the same or higher level of the frame tree. Linked lists of structures must appear in monotonic order through the file.

Table 5: Common Elements of All Frame Structures

<i>Data class</i>	<i>Variable Name</i>	<i>Descriptor & Comments</i>
INT_4U	length	Byte length of this structure, including byte count of this variable
INT_2U	class	Structure class for this particular structure.
INT_2U	instance	Counter for occurrence of this class of structure within current frame or current file, starting from 0. NOTE: All instance counters are set to 0 after end of frame AND end of file.

b.1 Frame Structure Header -- FrSH

FrSH is a dictionary-type structure. It contains the following data:

Table 6: Frame Structure Header Data

<i>Data class</i>	<i>Variable Name</i>	<i>Descriptor & Comments</i>
INT_4U	length	Byte length of this structure, including byte count of this variable
INT_2U	1	Structure class for FrSH (always 1)
INT_2U	instance	Counter for occurrence of this class of structure within current frame, starting from 0
STRING	name	Name of structure being described by this dictionary structure
INT_2U	class	Class number of structure being described
STRING	comment	Comment

b.2 Frame Structure Element -- FrSE

FrSE is a second element of a dictionary-type structure. FrSE contains the following data:

Table 7: Frame Structure Element Data

<i>Data class</i>	<i>Variable Name</i>	<i>Descriptor & Comments</i>
INT_4U	length	Byte length of this structure, including byte count of this variable
INT_2U	2	Structure class for FrSE (always 2)
INT_2U	instance	Counter for occurrence of class FrSE structure within the current frame, starting from 0
STRING	name	Name of an element of the structure being described by dictionary. All element names within the structure must be unique. NOTE: The first FrSE begins with row 4 for each of the tables below.
STRING	class	Literally contains "CHAR", INT_2U",...
STRING	comment	Comment

The structure types allowed are described below. The list will be augmented as the frame design evolves.

b.3 Frame Header -- FrameH

This is a structure containing the following data:

Table 8: Frame Header Structure Definition

<i>Data class</i>	<i>Variable Name</i>	<i>Descriptor & Comments</i>
First three elements are as shown in Table 5		
STRING	name	Name of project or other experiment description (e.g., GEO; LIGO; VIRGO; TAMA;...)
INT_4S	run	Run number (number < 0 reserved for simulated data); monotonic for experimental runs.
INT_4U	frame	Frame number, monotonically increasing until end of run, re-starting from 0 with each new run.
INT_4U	dataQuality	A logical 32-bit word to denote top level quality of data. Lowest order bits are reserved one each for various GW detectors: "1" means data for that detector is present in frame and VALID at a top level (as a minimum, locked data and valid time stamp). ^a
INT_4U	GTimeS	Frame start time, GPS time in integer seconds since GPS standard epoch, valid for approximately 143 years from time origin.
INT_4U	GTimeN	Frame start time residual, integer nanoseconds.
INT_2U	ULeapS	The integer number of leap seconds between GPS/TAI and UTC in the epoch when the frame is written: ULeapS = Int[TAI - UTC]. e.g., ULeapS was 32 at end of 1999/07

Table 8: Frame Header Structure Definition

<i>Data class</i>	<i>Variable Name</i>	<i>Descriptor & Comments</i>
INT_4S	localTime	Local seasonal time - UTC in seconds.
REAL_8	dt	Frame length in seconds
One or more of the pointers to the structures below may be NULL in any given Frame Header		
(FrVect *) PTR_STRUCT	type	Identifier for array used to store general info like the trigger type. Detailed description will Be Determined later.
(FrVect *) PTR_STRUCT	user	Identifier for array for user-provided information. Use is generic.
(FrDetector *) PTR_STRUCT	detectSim	Identifier for array storing model or simulation parameter data definition
(FrDetector *) PTR_STRUCT	detectProc	Identifier for data used in generating post-processed outputs (typically for off line processing)
(FrHistory *) PTR_STRUCT	history	Identifier for first history of post-processing with which frame may have been generated.
(FrRawData *) PTR_STRUCT	rawData	Identifier for the raw data structure
(FrProcData *) PTR_STRUCT	procData	Identifier for the first post-processed data
(FrProcData *) PTR_STRUCT	strain	Identifier for the post-processed strain data (best estimate)
(FrSimData *) PTR_STRUCT	simData	Identifier for the first simulated data buffers
(FrTrigData *) PTR_STRUCT	trigData	Identifier for the first trigger data structure
(FrSimEvent *) PTR_STRUCT	simEvent	Identifier for the first simulated event data structure
(FrSummary *) PTR_STRUCT	summaryData	Identifier for the first statistical summary data
(FrVect *) PTR_STRUCT	auxData	Identifier for the first auxiliary data
(FrTable *) PTR_STRUCT	auxTable	Identifier for the first auxiliary table data

a. See Appendix A for bit assignments.

b.4 ADC Data -- FrAdcData

This is a structure containing the following data:

Table 9: ADC Data Structure Definition^a

<i>Data class</i>	<i>Variable Name</i>	<i>Descriptor & Comments</i>
First three elements are as shown in Table 5		
STRING	name	Channel name -- must be unique with the frame
STRING	comment	Comment
INT_4U	channelGroup	Channel grouping number containing ADC ^b
INT_4U	channelNumber	Channel number ^b
INT_4U	nBits	Number of bits in A/D output
REAL_4	bias	DC bias on channel (Units @ ADC_counts = 0)
REAL_4	slope	ADC calibration: input units/ct
STRING	units	ADC calibration: input units for slope. If dimensionless, then units == <NONE>, in CAPITALS (without <...>).
REAL_8	sampleRate	Data acquisition rate, samples/s.
INT_4S	timeOffsetS ^c	Offset of 1 st sample relative to the integer seconds frame start time
INT_4U	timeOffsetN ^c	Offset of 1 st sample relative to timeOffsetS in integer residual nanoseconds
REAL_8	fShift	Frequency shift if signal has been heterodyned before ADC: fShift = (f_heterodyne - fNyquist@sampleRate)
INT_2U	dataValid	Data valid flag: dataValid = 0 -> ADC data valid; dataValid!= 0 -> ADC data suspect/not valid
(FrVect *) PTR_STRUCT	data	Identifier for vector of sampled data.
(FrVect *) PTR_STRUCT	aux	Identifier for vector for user-provided information; use is generic.
(FrAdcData *) PTR_STRUCT	next	Identifier for next ADC structure in the linked list.

- a. Whenever physical units are used, these shall be given as combinations of SI units; ADC counts shall be denoted "ct".
- b. These two variables are determined by site and must be unique over all detectors
- c. Time offsets always added together (with suitable scaling of timeOffsetN) to obtain decimal time offset.

b.5 Detector Data -- FrDetector

This is a structure containing the following data:

Table 10: Detector Data Structure Definition^{a,b}

<i>Data class</i>	<i>Variable Name</i>	<i>Descriptor & Comments</i>
First three elements are as shown in Table 5		
STRING	name	Instrument name (e.g., VIRGO; GEO; TAMA; LIGO_1, _2, _3; 40m; PNI; simulated pseudo data - model version etc.)
INT_2S	longitudeD	Detector vertex longitude, geographical coordinates: Integer degrees; DDD>0 => E of Greenwich
INT_2S	longitudeM	Detector vertex longitude, geographical coordinates: Integer minutes; 60 > MM >= 0
REAL_4	longitudeS	Detector vertex longitude, geographical coordinates: Decimal seconds; 60.0 > SS >= 0
INT_2S	latitudeD	Detector vertex latitude, geographical coordinates: Integer degrees; DDD>0 => N of Equator
INT_2S	latitudeM	Detector vertex latitude, geographical coordinates: Integer minutes; 60 > MM >= 0
REAL_4	latitudeS	Detector vertex latitude, geographical coordinates: Decimal seconds; 60.0 > SS >= 0
REAL_4	elevation	Vertex elevation, meters. relative to WGS84 ellipsoid.
REAL_4	armXazimuth	Orientation of X arm, measured in radians CCW from East.
REAL_4	armYazimuth	Orientation of Y arm, measured in radians CCW from East.
(FrVect *) PTR_STRUCT	more	Identifier for user-provided (presently undefined) structure for additional detector data.
(FrTable *) PTR_STRUCT	moreTable	Identifier for user-provided (presently undefined) table structure for additional detector data.

- a. All geographical coordinates are defined with respect to the Earth ellipsoidal model WGS84.
- b. References to interferometer arm length have been dropped to facilitate use of this structure for both 2km and 4km LIGO interferometers, as well as permitting bar detector data to be captured. This then means that GW channel calibrations must be in dimensionless units of strain.

b.6 End of File Structure - FrEndOfFile

There will be a structure to indicate end of file. Immediately before this structure, there may be an FrTOC structure. After FrEndOfFile there will be a hardware-specific End of File marker for media. This is a structure containing the following data:

Table 11: End of File Data Structure Definition

<i>Data class</i>	<i>Variable Name</i>	<i>Descriptor & Comments</i>
First three elements are as shown in Table 5		
INT_4U	nFrames	Number of frames in this file
INT_4U	nBytes	Total number of bytes in this file (0 if NOT computed)
INT_4U	chkFlag	Flag for checksum (1 = calculated; 0 = not calculated)
INT_4U	chkSum	File checksum value, calculated exclusive of chkFlaG and chkSum.
INT_4U	seekTOC	Bytes to back up to the beginning of the table of contents structure. If seekTOC == 0, then there is no TOC for this file.

b.7 End of Frame Data -- FrEndOfFrame

This is a structure containing the following data:

Table 12: End of Frame Data Structure Definition

<i>Data class</i>	<i>Variable Name</i>	<i>Descriptor & Comments</i>
First three elements are as shown in Table 5		
INT_4S	run	Run number; same as in Frame Header run number datum.
INT_4U	frame	Frame number, monotonically increasing until end of run; same as in Frame Header run number datum

b.8 History Data -- FrHistory

This is a structure containing the following data:

Table 13: History Structure Definition

<i>Data class</i>	<i>Variable Name</i>	<i>Descriptor & Comments</i>
First three elements are as shown in Table 5		
STRING	name	Name of history record.
INT_4U	time	Time of post-processing, GPS time in integer seconds since GPS standard epoch, valid for approximately 143 years from time origin.
STRING	comment	Program name and relevant comments needed to define post-processing.
(FrHistory *) PTR_STRUCT	next	Identifier for next history structure in the linked list.

b.9 Message Log Data -- FrMsg

This is a structure containing the following data:

Table 14: Message Log Data Structure Definition

<i>Data class</i>	<i>Variable Name</i>	<i>Descriptor & Comments</i>
First three elements are as shown in Table 5		
STRING	alarm	Name of message, error flag or alarm state.
STRING	message	Message body
INT_4U	severity	Message severity level (To Be Defined)
(FrMsg *) PTR_STRUCT	next	Identifier for next message structure in the linked list.

b.10 Post-processed Data -- FrProcData

This is a structure containing the following data:

Table 15: Post-Processed Data Structure Definition

<i>Data class</i>	<i>Variable Name</i>	<i>Descriptor & Comments</i>
First three elements are as shown in Table 5		
STRING	name	Data or channel name
STRING	comment	Comment
REAL_8	sampleRate	Data acquisition rate, samples/s.
INT_4U	timeOffsetS	For triggered data lasting less than one frame, integer seconds start time relative to frame start
INT_4U	timeOffsetN	For triggered data lasting less than one frame, integer residual nanoseconds start time relative to frame start
REAL_8	fShift	Frequency shift if signal has been heterodyned before ADC: $fShift = (f_{heterodyne} - f_{Nyquist}@sampleRate)$
(FrVect *) PTR_STRUCT	data	Identifier for array of sampled data.
(FrVect *) PTR_STRUCT	aux	Identifier for vector for user-provided information; use is generic.
(FrTable *) PTR_STRUCT	table	Identifier for table structure.
(FrProcData *) PTR_STRUCT	next	Identifier for next FrProcData structure in the linked list.

b.11 Raw Data -- FrRawData

This is a structure containing the following data:

Table 16: Raw Data Structure Definition

<i>Data class</i>	<i>Variable Name</i>	<i>Descriptor & Comments</i>
First three elements are as shown in Table 5		
STRING	name	Name of raw data.
(FrSerData *) PTR_STRUCT	firstSer	Identifier for first serial data structure in the linked list.
(FrAdcData *) PTR_STRUCT	firstAdc	Identifier for first ADC data structure in the linked list.
(FrTable *) PTR_STRUCT	firstTable	Identifier for first table structure in the linked list.
(FrMsg *) PTR_STRUCT	logMsg	Identifier for first error message structure in the linked list.
(FrVect *) PTR_STRUCT	more	Identifier for the additional user-defined data structures.

b.12 Serial Data -- FrSerData

This is a structure containing the following data:

Table 17: Serial Data Structure Definition

<i>Data class</i>	<i>Variable Name</i>	<i>Descriptor & Comments</i>
First three elements are as shown in Table 5		
STRING	name	Name of station producing serial data stream.
INT_4U	timeSec	Time of data acquisition, GPS time in integer seconds since GPS standard epoch, valid for approximately 143 years from time origin.
INT_4U	timeNsec	Frame start time residual, integer nanoseconds.
REAL_4	sampleRate	Sample rate, samples / s.
STRING	data	Pointer to string for ASCII-based data.
(FrVect *) PTR_STRUCT	serial	Identifier for serial data vector.
(FrTable*) PTR_STRUCT	table	Identifier for the user-defined table structure.
(FrSerData *) PTR_STRUCT	next	Identifier for next serial data structure in the linked list.

b.13 Simulated Data -- FrSimData

This is a structure containing the following data:

Table 18: Simulated Data Structure Definition

<i>Data class</i>	<i>Variable Name</i>	<i>Descriptor & Comments</i>
First three elements are as shown in Table 5		
STRING	name	Name of simulated data.
STRING	comment	Comment
REAL_4	sampleRate	Simulated data sample rate, samples / s.
(FrVect *) PTR_STRUCT	data	Identifier for array of simulated data.
(FrVect *) PTR_STRUCT	input	Identifier for input parameters for simulation.
(FrTable *) PTR_STRUCT	table	Identifier for table data structure.
(FrSimData *) PTR_STRUCT	next	Identifier for next simulated data structure in the linked list.

b.14 Simulated Event Data -- FrSimEvent

This is a structure containing the following data:

Table 19: Simulated Event Data Structure Definition

<i>Data class</i>	<i>Variable Name</i>	<i>Descriptor & Comments</i>
First three elements are as shown in Table 5		
STRING	name	Name of event.
STRING	comment	Descriptor of event.
STRING	inputs	Input channels and filter parameters to event process.
INT_4U	GTimeS	GPS time in seconds corresponding to maximum of event.
INT_4U	GTimeN	GPS time in residual nanoseconds relative to GTimeS corresponding to maximum of event.
REAL_4	timeBefore	Signal duration before GTimeS
REAL_4	timeAfter	Signal duration after GTimeS
REAL_4	amplitude	Continuous output amplitude returned by event
(FrVect *) PTR_STRUCT	data	Identifier for vector containing additional event results.
(FrTable *) PTR_STRUCT	table	Identifier for table structure containing additional event information.
(FrSimEvent *) PTR_STRUCT	next	Identifier for another event.

b.15 Static Data -- FrStatData

This is a structure containing the following data:

Table 20: Static Data Structure Definition^a

<i>Data class</i>	<i>Variable Name</i>	<i>Descriptor & Comments</i>
First three elements are as shown in Table 5		
STRING	name	Static data name ^b
STRING	comment	Comment
STRING	representation	Type of static data being represented. e.g., calibration, swept sine, pole-zero, FIR or IIR coefficients, ...
INT_4U	timeStart	Start time of static data validity, GPS time in integer seconds since GPS standard epoch, valid for approximately 143 years from time origin.
INT_4U	timeEnd	End time of static data validity (if unknown, set to 0), GPS time in integer seconds since GPS standard epoch, valid for approximately 143 years from time origin.
INT_4U	version	Version number for this static structure. i.e, the counter begins at 0 and is incremented by 1 thereafter. Updated statics for the same time window (e.g., modified calibration data) will be identified by unique version numbers.
(FrDetector *) PTR_STRUCT	detector	Identifier for the detector this static data is associated with.
(FrVect *) PTR_STRUCT	data	Identifier for vector of data.
(FrTable *) PTR_STRUCT	table	Identifier for first table structure in the linked list.

- a. A file is not required to contain any FrStatData structure; however, if at least one FrStatData structure exists in a frame, then the first one will appear immediately after its associated detector structure. Subsequent instances of FrStatData are not linked; each one must be identified at read time.
- b. Note that the combined list of FrStatData elements {name,timeStart, timeEnd,version} must be unique. To access FrStatData of a given name for a given epoch, one selects the block(s) with the desired name and with a time range spanning the epoch of interest, one then selects that block with the latest timeStart, and finally one takes the FrStatData with the highest version number.

It is possible for a frame to contain more than one structure requiring *different* FrStatData structures. For example FrDetector for raw data may require static data associated with instrumental parameters; FrProcData for processed data may require static data associated with the filtering or other analysis parameters; FrSimData for simulated data may require static data to define precisely the input parameters to the simulation. For this reason, there is included in the FrStatData definition a PTR_STRUCT object which provides information on the antecedent detector structure with which any one FrStatData structure is associated. Footnote a, Table 20, applies to each detector structure separately.

b.16 Summary Data -- FrSummary

This is a structure containing the following data:

Table 21: Summary Data Structure Definition

<i>Data class</i>	<i>Variable Name</i>	<i>Descriptor & Comments</i>
First three elements are as shown in Table 5		
STRING	name	Name of summary statistic.
STRING	comment	Comment.
STRING	test	Statistical test(s) used on raw data
(FrVect *) PTR_STRUCT	moments	Identifier for vector containing statistical descriptors
(FrTable *) PTR_STRUCT	table	Identifier for table structure containing additional summary information.
(FrSummary *) PTR_STRUCT	next	Identifier for other summary.

b.17 Table Structure - FrTable

This structure provides a mechanism to capture tabular data (i.e., groups of columns of different data types). This is a structure containing the following data:

Table 22: Table Data Structure Definition

<i>Data class</i>	<i>Variable Name</i>	<i>Descriptor & Comments</i>
First three elements are as shown in Table 5		
STRING	name	Name of this table -- <i>not</i> required to be unique within a frame.
STRING	comment	Comment
INT_2U	nColumn	Number of columns in table
INT_4U	nRow	Number of rows in table. The frame API must ensure that all columns have this length.
*STRING	columnName	Names of the columns. The frame API must then copy these names into each FrVect structure included in this FrTable
(FrVect *) PTR_STRUCT	column	First column of table (may be names of rows)
(FrTable *) PTR_STRUXT	next	Next table in linked list

b.18 Table of Contents Structure - FrTOC

This structure may be included (but is not required) immediately after the last frame in a file. It enables efficient indexing of key structures into a large number of frames by providing byte offset information. NOTE: not all structures within a file need to be indexed in FrTOC; however, if one type of structure *is indexed*, then *all* structures of that type *must* be indexed.

This is a structure containing the following data:

Table 23: Table of Contents Data Structure Definition

<i>Data class</i>	<i>Variable Name</i>	<i>Descriptor & Comments</i>
First three elements are as shown in Table 5		
INT_2S	ULeapS	From the first FrameH in the file.
INT_4S	localTime	From the first FrameH in the file.
INT_4U	nFrame	Number of frames in this file
*INT_4U	GTimeS	Array of integer GPS frame times (size of nFrame)
*INT_4U	GTimeN	Array of integer GPS residual nanoseconds for the frame (size of nFrame)
*REAL_8	dt	Array of frame durations in seconds (size of nFrame)
*INT_4S	runs	Array of run numbers (size of nFrame).
*INT_4U	frame	Array of frame numbers (size of nFrame).
*INT_8U	positionH	Array of FrameH positions, in bytes, from beginning of file (size of nFrame).
*INT_8U	nFirstADC	Array of first FrADCData positions, in bytes, from beginning of file (size of nFrame).
*INT_8U	nFirstSer	Array of first FrSerData positions, in bytes, from beginning of file (size of nFrame).
*INT_8U	nFirstTable	Array of first FrTable positions, in bytes, from beginning of file (size of nFrame). NOTE: The pointer is to the first table associated with FrRawData for each frame.
*INT_8U	nFirstMsg	Array of first FrMsg positions, in bytes, from beginning of file (size of nFrame).
INT_4U	nSH	Number of FrSH structures in the file.
*INT_2U	SHid	Array of FrSH IDs (size of nSH).
*STRING	SHname	Array of FrSH names (size of nSH).
INT_4U	nStatType	Number of static data block types in the file. Each is identified by a unique name If nStat == $2^{32}-1$, then “no data available in FrTOC”
For each FrStatData:		
STRING	nameStat	FrStatData name
STRING	detector	Detector name
INT_4U	nStatInstance	Number of instance for this FrStatData
*INT_4U	tStart	Array of GPS integer start times, in seconds (size of nStatInstance)
*INT_4U	tEnd	Array of GPS integer end times, in seconds (size of nStatInstance)
*INT_4U	version	Array of version time (size of nStatInstance)
*INT_8U	positionStat	Array of FrStatData positions from beginning of file (size of nStatInstance)
For FrAdcData:		

Table 23: Table of Contents Data Structure Definition

<i>Data class</i>	<i>Variable Name</i>	<i>Descriptor & Comments</i>
INT_4U	nADC	Number of unique FrAdcData names in file. If nADC == $2^{32}-1$, then “no data available in FrTOC”
*STRING	name	Array of FrAdcData names (size of nADC)
*INT_4U	channelID	Array of ADC channel IDs (size of nADC)
*INT_4U	groupID	Array of ADC group IDs (size of nADC)
*INT_8U	positionADC	Array of lists of FrAdcData offset positions, in bytes, from beginning of file (size of nFrame*nADC) Ordering of entries:row/column major ordering, i.e., all positions for one ADC appear sequentially.
For FrProcData:		
INT_4U	nProc	Number of unique FrProcData names in file. If nProc == $2^{32}-1$, then “no data available in FrTOC”
*STRING	nameProc	Array of FrProcData names (size of nProc)
*INT_8U	positionProc	Array of FrProcData positions, in bytes, from beginning of file (size of nFrame*nProc)
For FrSimData:		
INT_4U	nSim	Number of unique FrSimData names in file. If nSim == $2^{32}-1$, then “no data available in FrTOC”
*STRING	nameSim	Array of FrSimData names (size of nSim)
*INT_8U	positionSim	Array of FrSimData positions, in bytes, from beginning of file (size of nFrame*nSim)
For FrSerData:		
INT_4U	nSer	Number of unique FrSerData names in file. If nSer == $2^{32}-1$, then “no data available in FrTOC”
*STRING	nameSer	Array of FrSerData names (size of nSer)
*INT_8U	positionSer	Array of FrSerData positions, in bytes, from beginning of file (size of nFrame*nSer)
For FrSummary:		
INT_4U	nSummary	Number of unique FrSummary names in file. If nSummary == $2^{32}-1$, then “no data available in FrTOC”
*STRING	nameSum	Array of FrSummary names (size of nSummary)
*INT_8U	positionSum	Array of FrSummary positions, in bytes, from beginning of file (size of nFrame*nSummary)
For FrTrigData:		
INT_4U	nTrig	Number of FrTrigData in file. If nTrig == $2^{32}-1$, then “no data available in FrTOC”
*STRING	nameTrig	Array of FrTrigData names (size of nTrig)

Table 23: Table of Contents Data Structure Definition

<i>Data class</i>	<i>Variable Name</i>	<i>Descriptor & Comments</i>
*INT_4U	GTimeSTrig	GPS time in integer seconds (size of nTrig)
*INT_4U	GTimeNTrig	Residual GPS time in integer nanoseconds (size of nTrig)
*INT_8U	positionTrig	Array of FrTrigData positions, in bytes, from beginning of file (size of nTrig)
For FrSimEvent:		
INT_4U	nSimEvt	Number of FrSimEvent in file. If nSimEvt == $2^{32}-1$, then “no data available in FrTOC”
*STRING	nameSimEvt	Array of FrSimEvent names (size of nSimEvt)
*INT_4U	GTimeSSim	GPS time in integer seconds (size of nSimEvt)
*INT_4U	GTimeNSim	Residual GPS time in integer nanoseconds (size of nSimEvt)
*INT_8U	positionSimEvt	Array of FrSimEvent positions, in bytes, from beginning of file (size of nSimEvt)

b.19 Trigger Data -- FrTrigData

This is a structure containing the following data:

Table 24: Trigger Data Structure Definition

<i>Data class</i>	<i>Variable Name</i>	<i>Descriptor & Comments</i>
First three elements are as shown in Table 5		
STRING	name	Name of trigger.
STRING	comment	Descriptor of trigger.
STRING	inputs	Input channels and filter parameters to trigger process.
INT_4U	GTimeS	GPS time in seconds corresponding to maximum of trigger.
INT_4U	GTimeN	GPS time in residual nanoseconds relative to GTimeS corresponding to reference value of trigger, as defined by trigger algorithm.
REAL_4	timeBefore	Signal duration before (GTimeS.GTimeN)
REAL_4	timeAfter	Signal duration after (GTimeS.GTimeN)
INT_4U	triggerStatus	Defined by trigger.
REAL_4	amplitude	Continuous output amplitude returned by trigger
REAL_4	probability	Likelihood estimate of event, if available (probability = -1 if cannot be estimated)
STRING	statistics	Statistical description of event, if relevant or available.
(FrVect *) PTR_STRUCT	data	Identifier for vector containing additional trigger results.
(FrTable *) PTR_STRUCT	table	Identifier for table structure containing additional trigger information.
(FrTrigData *) PTR_STRUCT	next	Identifier for another trigger.

b.20 Vector Data -- FrVect

This is a structure containing the following data:

Table 25: Vector Data Structure Definition

<i>Data class</i>	<i>Variable Name</i>	<i>Descriptor & Comments</i>
First three elements are as shown in Table 5		
STRING	name	Channel name -- not required to be unique
INT_2U	compress	Compression algorithm number ^a
INT_2U	type	Vector class ^b
INT_4U	nData	Number of sample elements in data series
INT_4U	nBytes	Number of bytes in the compressed vector
See footnote a	data	nData elements of specified class
INT_4U	nDim	Dimensionality of data vector
*INT_4U	nx	nDim INT_4U elements whose values are dimension lengths
*REAL_8	dx	nDim REAL_8 elements whose values are scale factors for each coordinate;
*REAL_8	startX	Origin for each data set
*STRING	unitX	nDim strings containing scale factors in ASCII; “unit per step size along each coordinate”. If dimensionless, then unitX == <NONE>, in CAPITALS (without <...>).
STRING	unitY	String describing how to interpret the value of each element. If dimensionless, then unitY == <NONE>, in CAPITALS (without <...>).
(FrVect *) PTR_STRUCT	next	Identifier for additional data.

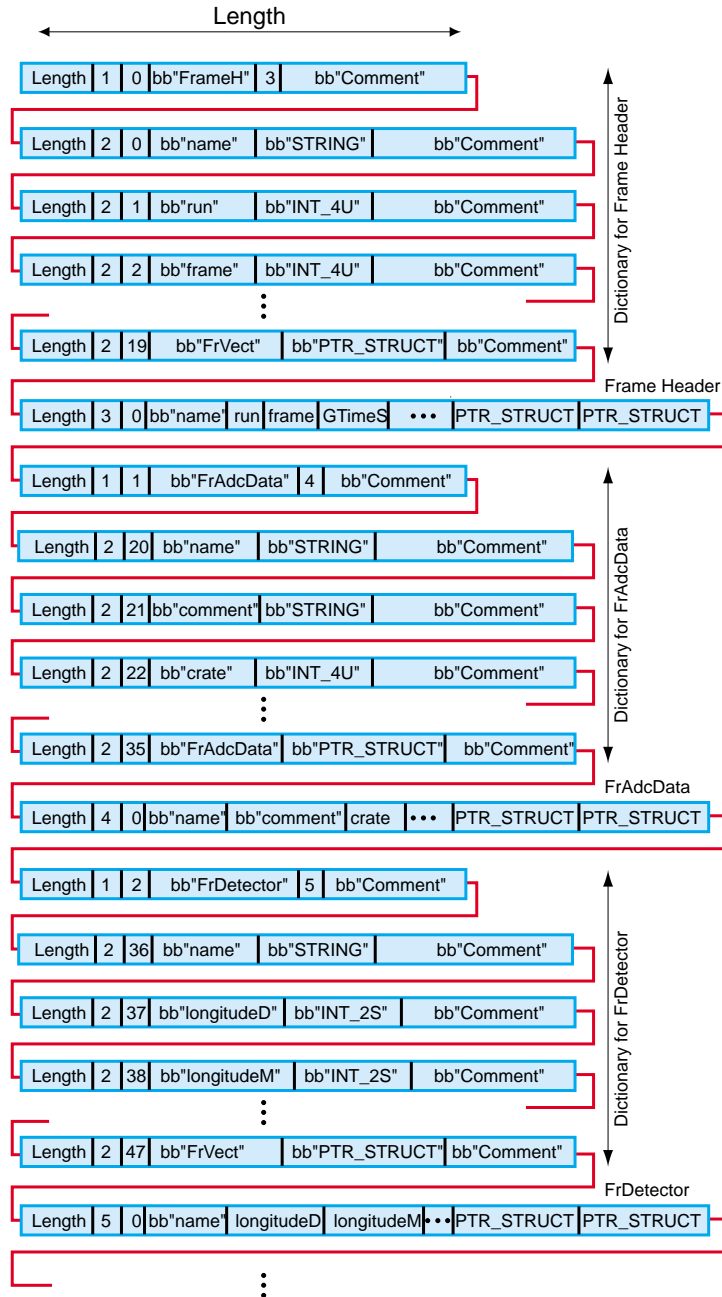
a. See Appendix B.

b. See Appendix B.

For multi-dimensional arrays, elements are stored by rows following the C convention.

Figure 3 presents a byte-level picture of how structures are linked to compose a frame.

Figure 3 Byte-level graphical representation of a frame



* bb"..." denotes a composite STRING type object defined in data type table
 * refer to structure definition tables in text for byte length of various objects above.

5 RULES FOR REVISION

LIGO and VIRGO will jointly maintain both the definition for the Frame Format and also all associated software libraries needed to write and access the frame structures.

5.1. Frame Formats

The numbering scheme for future revisions of the frame format shall be a single number as indicated in Section 4 above. There will be a database linking frame format version to frame library version creating it. This Format Version shall be incremented only when File Header or structures FrSH or FrSE are changed.

5.2. Frame Library Software

The actual software design of the frame manipulation libraries is the subject of a second document. However, for completeness, the rules for revising this software are indicated here.

There is a corresponding software specification document which provides detailed information in usage and generation of frames. As the frame format evolves, corresponding changes in the software libraries will be made, and a corresponding index of Frame format version and software library version shall be maintained. Version specification for the software libraries shall be in a form A.B.

A = version number. This is incremented whenever the frame format version number is changed. A shall be the same as the frame data format version number. If A is incremented, B is reset to 0.

B = revision number. This is incremented whenever one or more of the following changes are made: (i) software error fixes; (ii) enhancements in existing functionality; (iii) modification or addition of structures not addressed in A above.

5.3. Requests for changes

LIGO and VIRGO will maintain a web page (address **To Be Announced**) for submitting requests for changes and for providing for releases for code.

5.4. Change control

The IGWD Frame Format specification and software library specification will be placed under joint configuration control by VIRGO and LIGO using UNIX/CVS.

Updates will be provided by the following basis.

- a. Change requests will be reviewed jointly by VIRGO and LIGO on a regular basis.
- b. Those changes which are selected for incorporation shall be assigned for implementation to respective groups.
- c. All changes will be validated and verified using a prescribed test procedure.

- d. Once available, the new release will be distributed via the LIGO and VIRGO web sites. All affected documentation will be revised to show changes.
- e. A history of revisions shall be maintained and made available to users.

APPENDIX A ASSIGNMENT OF DATA QUALITY (DQ) BITS

Table 26: Bit Assignment of dataQuality Word in FrameH

Bit	Description
2^0	TAMA DQ
2^1	VIRGO DQ
2^2	GEO DQ
2^3	LIGO LHO 2 km DQ ^a
2^4	LIGO LHO 4 km DQ
2^5	LIGO LLO 4 km DQ
2^6	All prototype interferometers
2^7	ALLEGRO DQ
2^8	AURIGA DQ
2^9	EXPLORER DQ
2^{10}	NIOBE DQ
2^{11}	NAUTILUS DQ
$2^{12} - 2^{31}$	unimplemented

- a. bits 4 (2^3) and 5 (2^4) may both be set if both LIGO Hanford interferometers are written to the same frame.

APPENDIX B DATA COMPRESSION SCHEMES

The following are supported compression algorithms. Note: Every compression scheme (0 - 255) has a corresponding entry (256 - 511) for the case in which the compressing machine used a byte-swapped convention relative to the uncompressing machine.

Table 27: Compression Schemes Supported

ID	Writing Platform	Compression Description ^a
0	bigendian	uncompressed raw values
1	bigendian	gzip
2	bigendian	differential values
3	bigendian	gzip, differential values
4	bigendian	not used
5	bigendian	differentiation and zero suppression for integer types only ^b
6	bigendian	differentiation and zero suppression for integer types only and gzip for the other types ^b
7 - 255	*****not yet implemented*****	
256	littleendian	uncompressed raw values
257	littleendian	gzip
258	littleendian	differential values
259	littleendian	gzip, differential values
260	littleendian	not used
261	littleendian	differentiation and zero suppression for integer types only ^b
262	littleendian	differentiation and zero suppression for integer types only and gzip for the other types ^b
262 - 65535	*****not yet implemented*****	

- a. Each compression scheme has two entries to accommodate the possibility that the reading machine and the writing machine have different “endian-ness”. Big-endian machines typically refer to SUN and Apple(G3/G4) processors and little-endian machines typically refer to Intel, Intel clones and Alpha processors
- b. The zero suppression algorithm codes the data in the following way:
 - [1] Data are differentiate
 - [2] A block of size (nW) is selected and is written as unsigned short (2 bytes)
 - [3] The input data are split in blocks
 - [4] For each block the minimal number of bits (nB) is determined and (nB-1) is written (3 bits for char, 4 for short, 5 for int 6 for long),
 - [5] For each word in the bloc is transform to unsigned by adding $2^{*(nB-1)-1}$ and the useful nB bits are added to the output buffer.
 - [6] Then the next bloc is processed until completion of the input buffer.

Example with block size = 3

- Input vector:
- 82 85 85 81 80 82 84 85 (short)
- Differentiate data: 82 3 0 -4 -1 2 2 1
- Block 1: (82 -3 4) nBits = 8 -> write 8-1 = 7
 82 = 0x52 -> + 7f = 0xd1
 3 = 0x3 -> + 7f = 0x82
 0 = 0x0 -> + 7f = 0x7f
- Bloc 2: (-4 -1 2) nBits = 4 -> write 4-1 = 3
- Bloc 3: (-4 -1 2) nBits = 3 -> write 3-1 = 2,
- output is (hex): 0x0003 2d17 37f8 2963 0025

The following valid vector data types are defined: {type#: name: description};

Table 28: FrVect Data Types

ID	Data Type Name	Data Type
0	FR_VECT_C	CHAR
1	FR_VECT_2S	INT_2S
2	FR_VECT_8R	REAL_8
3	FR_VECT_4R	REAL_4
4	FR_VECT_4S	INT_4S
5	FR_VECT_8S	INT_8S
6	FR_VECT_8C	COMPLEX_8
7	FR_VECT_C16	COMPLEX_16
8	FR_VECT_STRING	STRING
9	FR_VECT_2U	INT_2U
10	FR_VECT_4U	INT_4U
11	FR_VECT_8U	INT_8U
12	FR_VECT_1U	CHAR_U
13 - 255	unimplemented	