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

Document Type Technical Note	LIGO-T970130-B- E: VIRGO-SPE-LAP-5400-102	12 Oct 97

Specification of a Common Data Frame Format for Interferometric Gravitational Wave Detectors (IGWD)

LIGO Data Group
VIRGO Data Acquisition Group

VIRGO CNRS/INFN
Via Livornese
1291-56010 San Piero a Grado, Pisa, Italy
Phone 39.50.880.352
Fax 39.50.880.350
E-mail: virgo@virgoa1.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 20B-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.pg.infn.it/virgodoc

Table of Contents

List of Tables

List of Figures

SPECIFICATION REVISION HISTORY

Revision	**Authority**	**Pages Affected**	Item(s) Affected
A	Initial release	-	-
01	Pending release as Rev. B	14	Table 8, added row 16 (overRange).
		15	Table 9, last row, delete FrDetector* element of structure.
		19	Table 18, row 5, change variable name to *type.*
		20	Table 18, rows 11, 12, interchange variable names (unitY, unitX).
		20	Table 18, footnote a, change {class#...} to {type#...}.
		22	Figure 3, renumber 3rd bytes of various elements to reflect changes described above.
02	Pending release as Rev. B	5	First version of new format changed to 3
		7	Table 4, edited description of Byte 5 (row 2)
		19	Table 17, delete row 10, FrStatData* element of structure
		19	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.
		19	Table 18, change class of variable *type* to INT_2U
		19	Table 18, introduce new element, class INT_4U, variable name *nBytes*. Needed to support data compression.
		19	Table 18, footnote a becomes footnote b.
		20	Table 18, added footnote a; footnote a becomes footnote b.
03	Pending release as Rev. B	5	Table 2, Added time standard acronym definitions.
		6	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.
		6	Table 3, correct C/C++ Data Type entry for INT_4U. int_U -> unsigned int.
		20	Table 18, exchange the Descriptor & Comments fields for the entries unitX and unitY (comments are reversed)
		20	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.

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 $6.^{d}0$. The integer number of leap seconds, N_S, between TAI and UTC in the present epoch is defined by the relationship: $\mathrm{TAI} - \mathrm{UTC} = N_S \cdot 1^{s}.000$.

4 IGWD FRAME STRUCTURE

This document specifies Frame Format Version Number 3, 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
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)

a. 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.

b. 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.

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.

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:

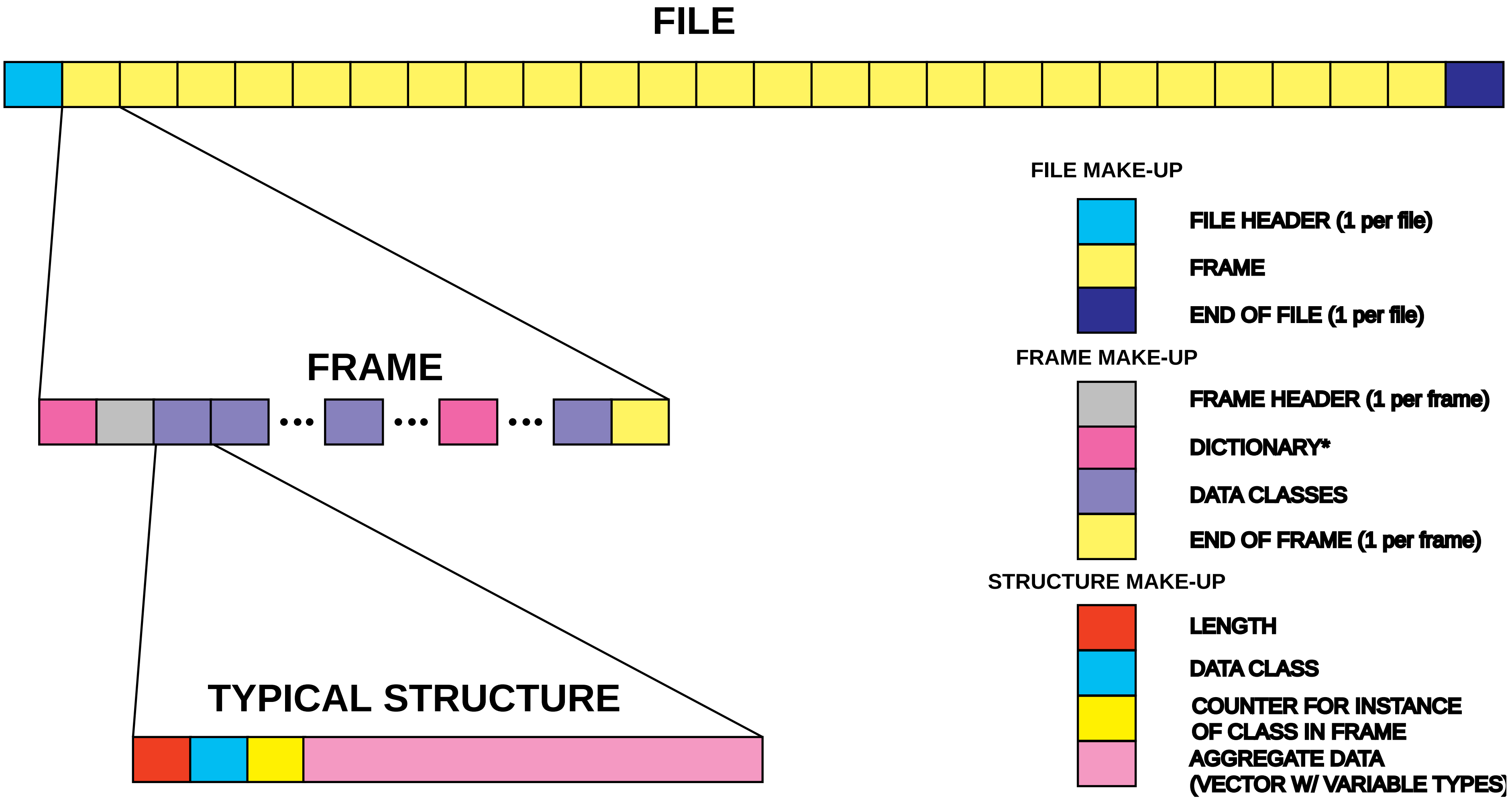
File:{FileHeader, Frame_1, Frame_2,..., Frame_m,... Frame_N, EndOfFile}

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.
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 π = 3.14159265358979323 84.......
30 - 37	IEEE double precision floating point representation of π = 3.14159265358979323 84.......
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 preceeds header for first frame of file;
2. Dictionary is built up incrementally as addititional structures are incorporated into frame
3. It is valid for entire file (persistent)

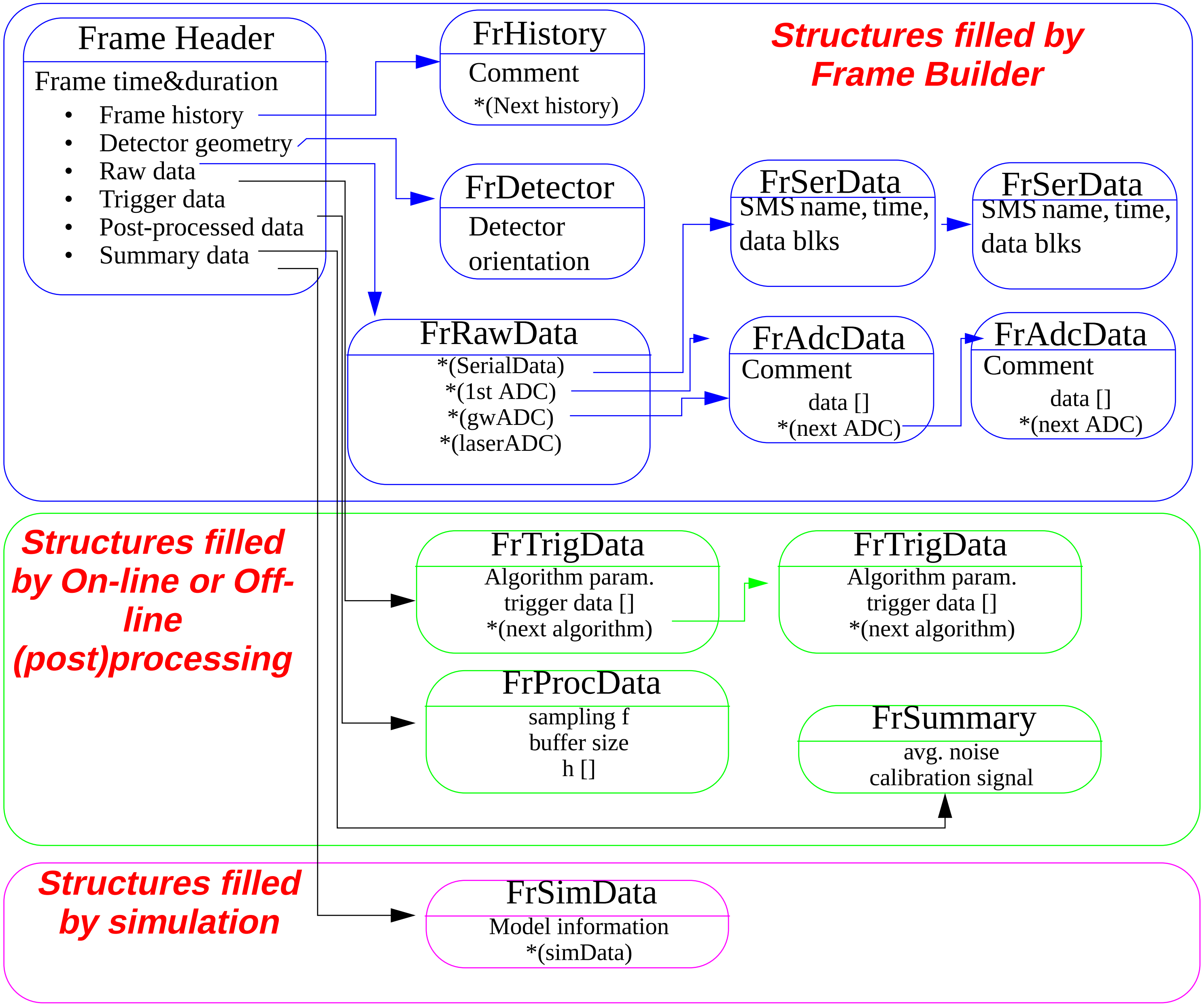


Figure 2 Schematic representation of frame structures and relative pointers

b. Frame:{DictionaryStructure, FrameHeader, DictionaryStructure, Structure_1, Structure_2,..., 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 frame 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.). 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.

b.1 Frame Structure Header -- FrSH

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

Table 5: Frame Structure Header Data

Data class	*Variable Name*	*Descriptor & Comments*
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
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. It contains the following data:

Table 6: Frame Structure Element Data

Data class	*Variable Name*	*Descriptor & Comments*
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
STRING	name	Name of an element of the structure being described by dictionary. 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 7: Frame Header Structure Definition

Data class	*Variable Name*	*Descriptor & Comments*
INT_4U	length	Byte length of this structure, including byte count of this variable
INT_2U	class	Structure class for FrameH (defined at run time)
INT_2U	instance	Counter for occurrence of class FrameH structure within the current frame
STRING	name	Name of project or other experiment description (e.g., GEO; LIGO; VIRGO; TAMA;...)
INT_4U	run	Run number
INT_4U	frame	Frame number, monotonically increasing until end of run
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].
INT_4S	localTime	Local seasonal time - UTC in seconds [integer multiple of 1800^s].
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.

Table 7: Frame Header Structure Definition

Data class	*Variable Name*	*Descriptor & Comments*
(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
(FrSummary *) PTR_STRUCT	summaryData	Identifier for the first statistical summary data
(FrVect *) PTR_STRUCT	auxData	Identifier for the first auxiliary data

b.4 ADC Data -- FrAdcData

This is a structure containing the following data:

Table 8: ADC Data Structure Definition

Data class	*Variable Name*	*Descriptor & Comments*
INT_4U	length	Byte length of this structure, including byte count of this variable
INT_2U	class	Structure class for FrADCData (defined at run time)
INT_2U	instance	Counter for occurrence of class FrADCData structure within the current frame
STRING	name	Channel name
STRING	comment	Comment
INT_4U	crate	Crate number containing ADC
INT_4U	channel	Channel number
INT_4U	nBits	Number of bits in A/D output
REAL_4	bias	DC bias on channel (Volts @ ADC_counts = 0)

Table 8: ADC Data Structure Definition

Data class	*Variable Name*	*Descriptor & Comments*
REAL_4	slope	ADC calibration: input units/count
STRING	units	ADC calibration: input units for slope
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 - fNyquist@sampleRate)
INT_2U	overRange	Data valid flag: overRange = 0 -> ADC data valid; overRange!= 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.

b.5 Frame Detector Data -- FrDetector

This is a structure containing the following data:

Table 9: Detector Data Structure Definition

Data class	*Variable Name*	*Descriptor & Comments*
INT_4U	length	Byte length of this structure, including byte count of this variable
INT_2U	class	Structure class for FrDetector (defined at run time)
INT_2U	instance	Counter for occurrence of class FrDetector structure within the current frame
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

Table 9: Detector Data Structure Definition

Data class	*Variable Name*	*Descriptor & Comments*
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	arrmXazimuth	Orientation of X arm, measured in radians CCW from East.
REAL_4	armYazimuth	Orientation of Y arm, measured in radians CCW from East.
REAL_4	armLength	Length of interferometer arm, measured between arm cavity mirror surfaces, in meters
(FrVect *) PTR_STRUCT	more	Identifier for user-provided (presently undefined) structure for additional detector data.

b.6 End of Frame Data -- FrEndOfFrame

This is a structure containing the following data:

Table 10: End of Frame Data Structure Definition

Data class	*Variable Name*	*Descriptor & Comments*
INT_4U	length	Byte length of this structure, including byte count of this variable
INT_2U	class	Structure class for FrEndOfFrame (defined at run time)
INT_2U	instance	Counter for occurrence of class FrEndOfFrame structure within the current frame
INT_4U	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.7 Frame Message Log Data -- FrMsg

This is a structure containing the following data:

Table 11: Message Log Data Structure Definition

Data class	*Variable Name*	*Descriptor & Comments*
INT_4U	length	Byte length of this structure, including byte count of this variable
INT_2U	class	Structure class for FrMsg (defined at run time)
INT_2U	instance	Counter for occurrence of class FrMsg structure within the current frame
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.8 Frame History Data -- FrHistory

This is a structure containing the following data:

Table 12: Frame History Structure Definition

Data class	*Variable Name*	*Descriptor & Comments*
INT_4U	length	Byte length of this structure, including byte count of this variable
INT_2U	class	Structure class for FrHistory (defined at run time)
INT_2U	instance	Counter for occurrence of class FrHistory structure within the current frame
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 Frame Raw Data -- FrRawData

This is a structure containing the following data:

Table 13: Frame Raw Data Structure Definition

Data class	*Variable Name*	*Descriptor & Comments*
INT_4U	length	Byte length of this structure, including byte count of this variable
INT_2U	class	Structure class for FrRawData (defined at run time)
INT_2U	instance	Counter for occurrence of class FrRawData structure within the current frame
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.
(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.10 Post-processed Data -- FrProcData

This is a structure containing the following data:

Table 14: Post-Processed Data Structure Definition

Data class	*Variable Name*	*Descriptor & Comments*
INT_4U	length	Byte length of this structure, including byte count of this variable
INT_2U	class	Structure class for FrProcData (defined at run time)
INT_2U	instance	Counter for occurrence of class FrProcData structure within the current frame
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 - fNyquist@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.
(FrProcData *) PTR_STRUCT	next	Identifier for next ADC structure in the linked list.

b.11 Frame Simulated Data -- FrSimData

This is a structure containing the following data:

Table 15: Frame Simulated Data Structure Definition

Data class	*Variable Name*	*Descriptor & Comments*
INT_4U	length	Byte length of this structure, including byte count of this variable
INT_2U	class	Structure class for FrSimData (defined at run time)
INT_2U	instance	Counter for occurrence of class FrSimData structure within the current frame
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 seed MC input data for simulation.
(FrSimData *) PTR_STRUCT	next	Identifier for next simulated data structure in the linked list.

b.12 Frame Serial Data -- FrSerData

This is a structure containing the following data:

Table 16: Frame Serial Data Structure Definition

Data class	*Variable Name*	*Descriptor & Comments*
INT_4U	length	Byte length of this structure, including byte count of this variable
INT_2U	class	Structure class for FrSerData (defined at run time)
INT_2U	instance	Counter for occurrence of class FrSerData structure within the current frame
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.
(FrVect *) PTR_STRUCT	more	Identifier for the additional user-defined data structures.
(FrSerData *) PTR_STRUCT	next	Identifier for next serial data structure in the linked list.

b.13 Static Data -- FrStatData

This is a structure containing the following data:

Table 17: Static Data Structure Definition[a]

Data class	*Variable Name*	*Descriptor & Comments*
INT_4U	length	Byte length of this structure, including byte count of this variable
INT_2U	class	Structure class for FrStatData (defined at run time)
INT_2U	instance	Counter for occurrence of class FrStatData structure within the current frame
STRING	name	Static data name
STRING	comment	Comment
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 number of times plus one for which this particular static data structure has been calculated or defined.
(FrVect *) PTR_STRUCT	detector	Identifier for the detector this static data is associated with.
(FrVect *) PTR_STRUCT	data	Identifier for vector of data.

a. A file may not 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.

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, Table17, applies to each detector structure separately.

b.14 Frame Vector Data -- FrVect

This is a structure containing the following data:

Table 18: Frame Vector Data Structure Definition

Data class	*Variable Name*	*Descriptor & Comments*
INT_4U	length	Byte length of this structure, including byte count of this variable
INT_2U	class	Structure class for FrVect (defined at run time)
INT_2U	instance	Counter for occurrence of class FrVect structure within the current frame
STRING	name	Channel name
INT_2U	compress	Compression algorithm number[a]
INT_2U	type	Vector class[b] (see footnote at end of this table)
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;
*STRING	unitX	nDim strings containing scale factors in ASCII; "unit per step size along each coordinate"
STRING	unitY	String describing how to interpret the value of each element
(FrVect *) PTR_STRUCT	next	Identifier for additional data.

[a] The following are supported compression algorithms: {type#:description}; (0: uncompressed raw values); (1:gzip);(2:differential values);(3:gzip, differential values);(4 - 256: unimplemented); (257:gzip, byte-swapped);(258:differential values, byte-swapped);(259:gzip, differential values, byte-swapped); (260 - 65535: unimplemented). Every compression scheme has a corresponding entry for the case in which the compressing machine used a byte-swapped convention relative to the uncompressing machine.

[b] The following valid vector data types are defined: {type#: name: description};
(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_S: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)

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

b.15 Frame Trigger Data -- FrTrigData

This is a structure containing the following data:

Table 19: Frame Trigger Data Structure Definition

Data class	*Variable Name*	*Descriptor & Comments*
INT_4U	length	Byte length of this structure, including byte count of this variable
INT_2U	class	Structure class for FrTrigData (defined at run time)
INT_2U	instance	Counter for occurrence of class FrTrigData structure within the current frame
STRING	name	Name of trigger.
STRING	comment	Descriptor of trigger.
STRING	inputs	Input channels and filter parameters to trigger process.
INT_4U	GTimeS	Event occurrence time, Integer seconds since beginning of frame.
INT_4U	GTimeN	Event occurrence time residual, integer nanoseconds.
INT_4U	bvalue	Boolean (T = 1, F = 0) value returned by trigger.
REAL_4	rvalue	Continuous value 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.
(FrTrigData *) PTR_STRUCT	next	Identifier for another trigger.

b.16 Frame Summary Data -- FrSummary

This is a structure containing the following data:

Table 20: Frame Summary Data Structure Definition

Data class	*Variable Name*	*Descriptor & Comments*
INT_4U	length	Byte length of this structure, including byte count of this variable
INT_2U	class	Structure class for FrSummary (defined at run time)
INT_2U	instance	Counter for occurrence of class FrSummary structure within the current frame
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
(FrSummary *) PTR_STRUCT	next	Identifier for other summary.

c. End of File Structure - FrEndOfFile

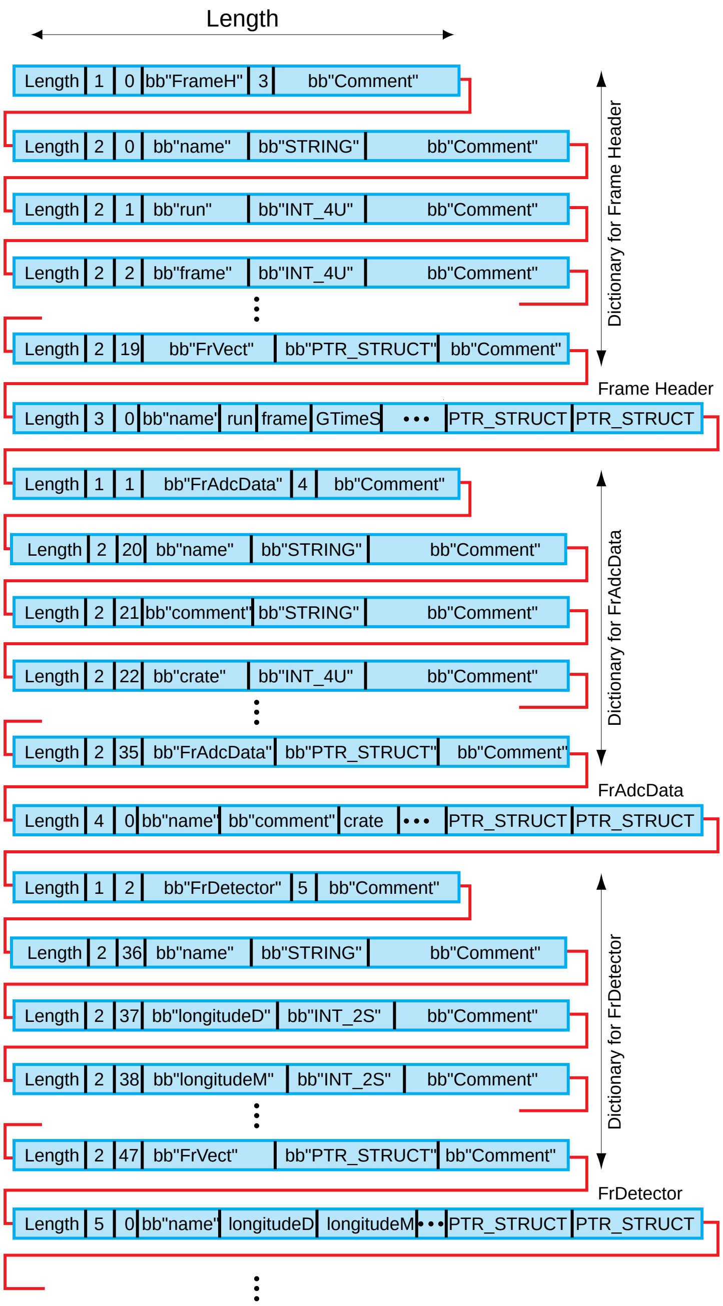
There will be a structure to indicate end of file. In addition, beyond, this structure, there will be hardware-specific End of File marker for media.This is a structure containing the following data:

Table 21: End of File Data Structure Definition

Data class	*Variable Name*	*Descriptor & Comments*
INT_4U	length	Byte length of this structure, including byte count of this variable
INT_2U	class	Structure class for FrEndOfFile (defined at run time)
INT_2U	instance	Counter for occurrence of class FrEndOfFile structure within the current frame
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.

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


Length
Length | 1 | 0 | bb"FrameH" | 3 | bb"Comment"
Length | 2 | 0 | bb"name" | bb"STRING" | bb"Comment"
Length | 2 | 1 | bb"run" | bb"INT_4U" | bb"Comment"
Length | 2 | 2 | bb"frame" | bb"INT_4U" | bb"Comment"
Length | 2 | 19 | bb"FrVect" | bb"PTR_STRUCT" | bb"Comment"
Dictionary for Frame Header
Frame Header
Length | 3 | 0 | bb"name" | run | frame | GTimeS | • • • | PTR_STRUCT | PTR_STRUCT
Length | 1 | 1 | bb"FrAdcData" | 4 | bb"Comment"
Length | 2 | 20 | bb"name" | bb"STRING" | bb"Comment"
Length | 2 | 21 | bb"comment" | bb"STRING" | bb"Comment"
Length | 2 | 22 | bb"crate" | bb"INT_4U" | bb"Comment"
Length | 2 | 35 | bb"FrAdcData" | bb"PTR_STRUCT" | bb"Comment"
Dictionary for FrAdcData
FrAdcData
Length | 4 | 0 | bb"name" | bb"comment" | crate | • • • | PTR_STRUCT | PTR_STRUCT
Length | 1 | 2 | bb"FrDetector" | 5 | bb"Comment"
Length | 2 | 36 | bb"name" | bb"STRING" | bb"Comment"
Length | 2 | 37 | bb"longitudeD" | bb"INT_2S" | bb"Comment"
Length | 2 | 38 | bb"longitudeM" | bb"INT_2S" | bb"Comment"
Length | 2 | 47 | bb"FrVect" | bb"PTR_STRUCT" | bb"Comment"
Dictionary for FrDetector
FrDetector
Length | 5 | 0 | bb"name" | longitudeD | longitudeM | • • • | PTR_STRUCT | PTR_STRUCT

* 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 rest 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.