Frame Library (Fr)
User's Manual

VIR-MAN-LAP-5400-103
Version 8.30
April 20, 2016

Summary:

Introduction
A quick tour of the Library: The examples
The Frame Utilities: FrCopy, FrDump and FrCheck
Reference Part

Library control
Frame Handling
Input File: FrFileI
Output File: FrFileO
File checksum
Error Handling

FrAdcData
FrDetector
FrEvent
FrHistory
FrMsg
FrProcData
FrSerData

FrSimData
FrSimEvent
FrStatData
FrSummary
FrTable
FrVect
FrFilter

The Matlab interface
The ROOT interface
The Octave interface
The Python interface
The Frame Library Installation
Library Changes

Introduction

A frame is a unit of information containing all the information necessary for the understanding of the interferometer behavior over a finite time interval which integrates several samplings. It contains thus not only the sampling performed during the integrated time interval, but also those performed at a frequency smaller than the frame frequency.

To simplify its manipulation, a frame is organized as a set of C structures described by a header holding pointers to additional structures and values of parameters expected to be stable over the integrated time interval: the starting time of the frame, its duration, values produced by the slow monitoring. This header is followed by an arbitrary number of additional structures, each holding the values of a rapidly varying parameter like the main signal, the seismic noise, etc...

This frame structure is a standard which has to be conserved over the various stages of the analysis. Thus Frame history, detector geometry, trigger results, monitoring data, reconstructed data, simulation results just lead to additional structures. It is always possible to add new structures or to drop old ones.

This standard format is the one used by the LIGO and VIRGO Gravitational Wave Detectors. This Document described the software used to manipulate the frames. The definition of the various structures as well as their representation on tape is described in specification document.

The C structures used by The Frame Library

The data are stored in a set of C structures described in the document *Specification of a Common Data Frame Format for Interferometric Gravitational Wave Detector (IGWD)* (VIR-067-08 and LIGO-T970130-v1). The variable names of the C structures are exactly the one given in this document.

A quick tour of the Library: the examples

Many examples are provided with the frame library in the directory src. They have been designed to test the various parts of the library and are good starting points for a new program. The Files are:

- **exampleCompress.c** This program create a frame with all types of vectors, write it with all the compression types available and read it back to check if the frame has not been corrupted.
- **exampleCopyFile.c** a simple copy file program. See the utility FrCopy for more complex file copy.
- **exampleCopyFrame.c** a simple copy file program with some channel selection. See the utility FrCopy for more complex file copy.
- **exampleFileDump.c** dump on the screen a short summary of a frame file content See the FrDump utility for more options.
- **exampleFull.c** This example builds frames with several ADC and many different types of data. Then the frames are written in a file. Finally the tag functionality is tested.
- **exampleMark.c** This function shows the use of random access in a frame file.
- **exampleMultiR.c** Reads the various files produced by exampleMultiW.c This program is useful to search for memory leaks.
- **exampleMultiTOC.c** Reads the various files produced by exampleMultiW.c using random access. This program is used to test random access and to search for memory leaks.
- **exampleMultiW.c** Produces several frames in different files. This program is useful to search for memory leaks.
- **exampleOnline.c** creates a few frames and write them in memory. Different compression type could be used to test the speed.
- **exampleReshape.c** This program shows how to change the frame length using the FrReshape functions..
- **exampleSpeed.c** This program test the in memory read/writing speed for a frame given by the user and the a given compress type.
- **exampleStat.c** Illustrates the use of static data.

If you just want to read data from a frame file:

In the simplest case, to extract data from a frame file, you need to use only three FrameL functions:

```
       iFile = FrFileINew("inputFileName");
```

This function will open the frame file. Usually inputFileName is an "ffl", a list of .gwf file which make the data management much easier.

```
       FrVect* vect = FrFileIGetVectF(iFile,  channel_name, tStart,
tLength);
```

which return a vector for "channel_name" starting at the GPS time tStart and for a duration "tLength" . The start time could be in the middle of a frame and the duration could be over multiple frame or files. The vector is of type single floating point, whatever was the storage on file. The number of elements is vect->nData. The data values could be access by vect->dataF[i].
Then you need to free the space by

```
       FrVectFree(vect);
```

If you want to write data in a frame file, here is a simple program:

```
#include "FrameL.h"

int main() {
 FrFile *oFile;
 FrameH *frame;
 FrProcData *proc;
 double sampleRate;
 long nData,i,j;

  frame = FrameNew("demo"); /*----------------------create a 10s long
frame--*/
  frame->dt = 10;

  sampleRate = 16384;       /*-----add a 16384Hz 32bits float proc channel-
--*/
  nData = sampleRate*frame->dt;
  proc = FrProcDataNew(frame,"Channel_Name",sampleRate, nData, -32);

  /*-------- open output file; compression type 9, 1000 seconds per file --
--*/

  oFile = FrFileONewM("test", 9, "FrFull", 1000);
  if(oFile == NULL) {
    printf("Open file error (%s)\n",FrErrorGetHistory());
    return(0);}

  for(i=0; i<10; i++) {          /*---------------------produce 10 frames -
--*/
      frame->GTimeS = frame->GTimeS + frame->dt;

      for(j=0; j < proc->data->nData; j++) { /*----- update channel
content--*/
           proc->data->dataF[j] = j;}        /*-- change this to your need-
--*/

      if(i < 2) FrameDump(frame, stdout,2); /*------------- just for debug-
--*/

      if(FrameWrite(frame, oFile) != FR_OK) {
        printf("Write error; last error:%s\n",FrErrorGetHistory());
        return(0);}
     }

  FrFileOEnd(oFile); /*------------------------------ close the output
file---*/

  FrameFree(frame);

  return(0);
}
```

The Frame Utilities: FrCopy, FrDump and FrCheck

Three utilities are included in the Frame Package.

To copy a (set of) frame(s): FrCopy

- This program reads frames from one or more input files, merge them if requested and write them to one or more output file, using or not data compression. See the help function bellow for program use.
- The syntax is: FrCopy *options*
  where *option* could be:

-i <input file(s)> If more than one files is given after the keyword -i they will be read in sequence. If several input stream are defined (several -i followed by name(s)), then the frame content will be merged
-o <output file>
-f <first frame: (run # frame #) or (GPS time)> Example: -f 0 20 will start with run 0 frame 20. -f 6544444 will start at GPS time = 6544444. If this option is used, the Table Of Content is mandatory and all frames will be read by increasing time.
-l <last frame: (run # frame #) or (GPS time)>; the -l argument is interpretated as the length if it is a small number (<100000000) instead of the GPS time for the last frame.
-c <compression type> Compression types are -1 (same compression as input file), 0 (no compression), 1 (gzip), 3 (differenciation+gzip), 5 (gzip for float+zero suppress for int), 6 (zero suppress for int). The default is 6.
-a <list of input adc channels>.When this option is used, random access are performed to read only the requested adc channels. Only the Frame header is returned in addition to the adc data. Additional information like the history records is not returned.
-t <list of tag channels> Tag is a channel list with wild cards like: ADC122 ADC5* If a name start by - it is interpreted as an anti-tag
-r <new frame length in second> The reshape option works only with one output file. It assumes that the length of the input frame is a integer number of second. The starting GPS time of the output frame will be a multiple of the frame length. The requested length should be larger than the input frame length.
-decimate <number of sample to average> The decimation is done on all selected channel by doing a simple data averaging.
-d <debug level> (from 0 to 5)
-max <maximum output file length in second>. If this option is used, the output file is split in several file, each of them lasting up to <max> second. The name of these files is no more just <output file> but '<output file>-GPS-maxLength.gwf' (like V-R-730123000-100.gwf if <output file> = 'V-R').
-noTOCts to not write TOC for time series
-noChkSum to not put checksum in the output file.
-h (to get the help)

To dump frames: FrDump

- This program produces a dump of one file, one or more frame or one or more channels.
- The syntax is: FrDump *options*
  where *option* could be:

-i <input file(s)> If more than one files is given after the keyword -i they will be read in sequence. If several input stream are defined (several -i followed by name(s)), then the frame content will be merged
-f <first frame: (run # frame #) or (GPS time)> Example: -f 0 20 will start with run 0 frame 20. -f 6544444 will start at GPS time = 6544444
-l <last frame: (run # frame #) or (GPS time)>
-t <list of tag channels> Tag is a channel list with wild cards like: ADC122 ADC5* If a name start by - it is interpreted as an anti-tag
-d <debug level> (from 0 to 5)
-top <number of ADC in the hit-parade>
-h (to get this help)
If one of the next option is there, we do only a partial frame dump

-adc to dump only the FrAdcData information
-sms to dump only the FrSerData information
-proc to dump only the FrProcData information
-sim to dump only the FrSimData information
-sum to dump only the FrSummary information
-stat to dump only the static information
-raw to dump only the raw data information
-event to dump only the FrEvent and FrSimEvent

To check a frame file: FrCheck

This program check that the frame file could be read successfully. The file checksum are also checked if they are available. In case of success, this program returns zero and set the environment variable FRCHECK_NFRAME to the number of frames in the file. It returns ae error flag in case of error.By default (unless the -t or -s option are used), FrCheck do first a sequentiel read to check the file checksum, then do a random access read to check the frame checksum.

- The syntax is: FrCheck *options*
  where *option* could be:

  -i <input file> Only one file should be used
  -d <debug level> (default 1). 0 will supress all info and error messages.
  -t to scan the file using only the TOC
  -s to scan the file using only sequentiel read(TOC not used)
  -f GPS time of the first frame to scan (default=0) (only used when doing the random access)
  -l GPS time of the last frame to scan (default : 999999999.) (only used when doing the random access).
  -c to check the data compression (if any)
  -h to get the help

Reference Part

Library control	**FrAdcData**	**FrSimEvent**
Frame Handling	**FrDetector**	**FrStatData**
Input File: FrFileI	**FrHistory**	**FrSummary**
Output File: FrFileO	**FrMsg**	**FrTable**
File checksum	**FrProcData**	**FrEvent**
Error Handling	**FrSerData**	**FrVect**
	FrSimData	

Library control

The Frame library do not need any initialization. However, you can change some of the default parameters using the following function or you can access to some information.

FrLibIni

- This function changes the debug level and output file.
- Syntax: **FILE *FrLibIni(char *outFile, FILE *fOut, int dbglvl)**
  - outFile is the output file name provided by the user
  - fOut should be used only if the user wants to send out the debug information on an already opened file. Then he should provide this pointer. In this case, outFile should be NULL**.**
  - dbglvl is the debug level provided by the user. This is used only for internal and technical debug. Usually you can use 0.
    - 0 means no output at all
    - 1 gives a minimal description
    - 2,3 give more information

- The return argument is the pointer to the file opened. If the output file could not be opened the debug information is sent on stdout and the return argument is stdout. In case of severe error due to insufficient space, the return value is NULL and the user should not go further.

FrLibSetLvl

- This function changes the debug level. It could be called at any time.
- Syntax: **void FrLibSetLvl (int dbglvl);** where: dbglvl is the debug level provided by the user with the same meaning as in FrLibIni.

FrLibVersion

- This function returns the Library version as a float.
- Syntax: **float FrLibVersion (FILE *fOut);** It returns the version number (like 3.70). If fOut is not NULL it print also on fOut some debugging information.

FrLibVersionF

- This function returns the full Library version and compile time as a char*.
- Syntax: **char *FrLibVersionF ();** It returns the version number like "Fr Version:6.07 (May 20, 03)(Compiled: May 20 2003 17:45:54)".

Frame Handling

FrameCompress,	FrameMerge	FrameReshape
FrameCopy,	FrameRead,	FrameTagXXX,
FrameDump,	FrameReadN,	FrameUntagXXX,
FrameDumpToBuf	FrameReadRecycle	FrameWrite
FrameExpand,	FrameReadT	FrameWriteToBuf
FrameFree,	FrameReadTAdc,	FrameRemoveUntaggedData
FrameFindVect	FrameReadFromBuf	Back to summary

FrameCompress

- This function compress in memory all the frame vectors.
- Syntax: **FrameCompress (FrameH *frame, int compress, int gzipLvl)** were:
  - frame is the pointer to the frame header provided by the user
  - compress is the type of compression:
    - 1 for gzip,
    - 3 for differentiation and gzip.
    - 5 for differentiation and zeros suppress (only for short)
    - 6 for differentiation and zeros suppress for short and int, gzip for other
    - 7 for differentiation and zeros suppress for short, int and float to integer (not part of the current frame format)
    - 8 for differentiation and zeros suppress for short, int and float. (not part of the current frame format)
    - 255 for user defined compression code (definitely not part of the frame format)

  - gzipLvl is the gzip compression level (provided by the user). 0 is the recommended value.

- In normal use, the compression is done at frame write and the user do not need to take care of it.

FrameCopy

- This function copy a full frame and return the pointer to the new FrameH structure. This means it allocate the memory for the full tree of structures. The input frame is unchanged. It returns NULL in case of error (memory allocation failed).
- Syntax: **FrameH* FrameCopy (FrameH *frame)** were frame is the pointer to the frame header provided by the user

FrameDump

- This function produce a readable dump of the frame content.
- Syntax: **void FrameDump (FrameH *frame, FILE *fp, int debugLevel)** were:
  - frame is the pointer to the frame header provided by the user
  - fp is the file pointer where the debug information will be send (like stdout)
  - debugLevel is the debug level provided by the user. 0 means no output at all, 1 gives a minimal description (<5 lines per frame), 2, 3 give more information

FrameDumpToBuf

- This function produce a readable dump of the frame content.
- Syntax: **char* FrameDumpToBuf (FrameH *frame, int level, cha* buf, long bufsize)** were:
  - frame is the pointer to the frame header provided by the user
  - debugLevel is the debug level provided by the user. 0 means no output at all, 1 gives a minimal description (<5 lines per frame), 2, 3 give more information
  - buf is a buffer provided by the user
  - bufSize is the buffer size in bytes (provided by the user).
- This function returns the pointer to the beginning of the printable area of buf.

FrameExpand

- This function uncompressed all the frame vectors:
- Syntax: **void FrameExpand (FrameH* frame)** where frame is the pointer to the frame header provided by the user
- In normal use the uncompression is done by a FrameRead call or by the channel access.

FrameFree

- This function all the space allocated for a frame.
- Syntax: **void FrameFree (FrameH* frame)** where frame is the pointer to the frame header provided by the user

FrameGetAdcSize

- This function returns the memory used by a all the ADC structures and associated vectors (in bytes)
- Syntax: **FRLONG FrameGetAdcSize (FrameH *frame)**

FrameFindXXX

- Syntax:
  FrDetector *FrameFindDetector(FrameH *frame, char *detNameOrPrefix)
- This function returns the pointer to the detector structure given its name or 2 letters prefix. It return NULL if the detector is not found.Since the detector is still attached to the frame, the user should NOT free it.

FrameFindXXX

- Syntax:
  FrVect* FrameFindVect (FrameH* frame, char* name)
  FrVect* FrameFindAdcVect (FrameH* frame, char* name)
  FrVect* FrameFindProcVect (FrameH* frame, char* name)
  FrVect* FrameFindSimVect (FrameH* frame, char* name)
  FrVect* FrameFindStatVect (FrameH* frame, char* name)
  FrVect* FrameFindSumVect (FrameH* frame, char* name)
- These functions returns the pointer to the vector associated to one channel (Adc, Proc or FrSimData). FrameFindVect look for all kinds of channels. Name is the channel name. It return NULL if the channel is not found.Since the vector is still attached to the frame, the user should NOT free the vector.

FrameMerge

- This function merges the data of two frames.
- Syntax: **FrameH* FrameMerge (FrameH* frame1, FrameH* frame2)** where frame1 and frame2 are the pointers to the frame headers provided by the user. Data from frame2 are added to frame1. All remaining unused structures of frame2 are deleted.
- No check is performed on the time compatibility. If the timing information is available (GTimeS != 0) then the two times are compared and the possible time residual is stored in the adc->timeOffset variables.

FrameNew

- This function create a new frame: the frame header and the FrDetectProc structure. The local time is used to fill the Frame header timing section. A detector (Proc) structure is also added in order to be able to add right away the static data structure.
- Syntax: **FrameH* FrameNew (char *name)** where name is the experiment name.
- This function returns the pointer to the frame header or null in case of problem.
- Remark: the FrameHNew function (syntax: FrameH* **FrameHNew (char *name)**) creates only a frame header and do not fill the timing information.

FrameRead

- This function read the next frame in a file. It returns NULL in case of error or end of file.
- Syntax: **FrameH* FrameRead (FrFile *iFile)** where iFile is the pointers to the input file provided by the user.
- If you want to not uncompress the data at read time see FrFileSet.

FrameReadN

- This function read the frame starting for a given run and frame numbers. This is a random file access and requires frame files version 4 at least. It returns NULL in case of error or if the frame is not in the file or if the table of content is not available.
- Syntax: **FrameH* FrameReadT (FrFile *iFile, int runNumber, int frameNumber)** where iFile is the pointers to the input file provided by the user.</ NULL in case of error or if the frame is not in the file or if the table of content is not available.
- Syntax: **FrameH* FrameReadT (FrFile *iFile, double gtime)** where iFile is the pointers to the input file provided by the user and gtime the request GPS time. The selected frame is the one which start at gtime or which include gtime. If gtime = 0 then the first frame in the file is returned.

FrameReadTAdc, FrameReadTProc, FrameReadTSer, FrameReadTSim

- This function read the frame starting at a given time with only a defined list of Adc, or Proc or Ser or Sim channels. This is a random file access and requires frame files version 4 at least. It returns NULL in case of error or if the frame is not in the file or if the table of content is not available.
- Syntax:
  FrameH* FrameReadTAdc (FrFile *iFile, double gtime, char *name)
  FrameH* FrameReadTProc (FrFile *iFile, double gtime, char *name)
  FrameH* FrameReadTSer (FrFile *iFile, double gtime, char *name)
  FrameH* FrameReadTSim (FrFile *iFile, double gtime, char *name)
  where
    - iFile is the pointers to the input file provided by the user and gtime the request GPS time (several files could be used at the same time).
    - The selected frame is the one which start at gtime or which include gtime. If gtime = 0 then the first frame in the file is returned.
    - The selected frame is the one which start at gtime or which include gtime. If gtime = 0 then the first frame in the file is returned.
    - name is a string containing a list of channel names of different type separated by a space which could include wild card (example "*LSC* *EXC")

FrameReadFromBuf

- Syntax: **FrameH *FrameReadFromBuf (char *buf, long nBytes, unsigned short comp);** where comp is the compression level. if comp = -1, no compression/uncompress is performed.

FrameReshapeNew, Add, End

- This set of function is designed to change the frame size, i.e. to group consecutive frame in a single one.
- Three calls are needed:
    - **FrameH *FrameReshapeNew (FrameH *frame, int nFrame, int position);** This function initialize the reshaping. The new frame size will be nFrame longer than the original frame size. The new frame will start at time offset equal to "position" times the length of the initial frame. This function returns a pointer to the new frame (same has the input frame) or NULL in case of problem.
    - **int FrameReshapeAdd (FrameH *new, FrameH *frame);** This function move the information from the frame "frame" to the frame "new" which should be the output of the function FrameReshapeNew. This function perform a FrameFree of the frame part. It returns 0 in case of success or an error code.
    - **int FrameReshapeEnd (FrameH *new);** This function should be call when all the frame part have been added and before the "new" frame could be used. This function returns 0 in case of success or an error code.

- Remark: The copy utility is a convenient way to change the frame size.
- Example: See the file exampleReshape.c

FrameSetPrefix

- Syntax: **void FrameSetPrefix(FrameH* frame, char* prefix);** h This function set the prefix "prefix" to all channels in this frame.

FrameTagXXX with XXX=Adc, Event, Proc, Ser, Sim, SimEvt, Stat, Sum

- Syntax:

```
void    FrameTag     (FrameH *frame, char *tag);
void    FrameTagEvent(FrameH *frame, char *tag);
void    FrameTagAdc  (FrameH *frame, char *tag);
void    FrameTagProc (FrameH *frame, char *tag);
```

void FrameTagSer (FrameH *frame, char *tag);
void FrameTagSim (FrameH *frame, char *tag);
void FrameTagSimEvt(FrameH *frame, char *tag);
void FrameTagStat (FrameH *frame, char *tag);
void FrameTagSum (FrameH *frame, char *tag);

- These function select all the Adc, Ser, ... which match the names given in the tag string. This string could contain several names, some of them could include "*" and then are interpreted has wild card. After a call to FrameTagXXX all other channels are hidden for the user. If a FrameDump or FrameWrite is performed, only the tagged channels will be dumped or written. For instance the call to:

  void FrameTagAdc(myframe, "Lr* SaDb2")

  will keep from the frame myframe the SaDb2 ADC and all ADC with a name starting with Lr.
  - The function FrameTag call all the other functions. It performed a 'global tag'
  - Remarks:
    - No change of the channel list should be done when the list as been tagged.
    - Wild card "*" could be used anywhere in a name.
    - The tag "*" means select all the channels.
    - The tag "NONE" means no channels are selected.
    - A tag starting by "-" is interpreted as an 'anti-tag': the channel is removed.
    - It is not necessary to call FrameUntag before doing a FrameFree.

FrameUntagXXX with XXX=Adc, Event, Proc, Ser, Sim, SimEvt, Stat, Sum

- Syntax:
  void FrameUntag (FrameH *frame);
  void FrameUntagAdc (FrameH *frame);
  void FrameUntagEvent(FrameH *frame);
  void FrameUntagProc (FrameH *frame);
  void FrameUntagSer (FrameH *frame);
  void FrameUntagSim (FrameH *frame);
  void FrameUntagSimEvt(FrameH *frame);
  void FrameUntagStat (FrameH *frame);
  void FrameUntagSum (FrameH *frame);
- These functions restore the channel linked lists.
- The function FrameUntag call all the other function. It performed a 'global tag'

FrameRemoveUntaggedXXX with XXX=Adc, Event, Proc, Ser, Sim, SimEvt, Stat, Sum

- Syntax:
  void FrameRemoveUntagged (FrameH *frame);
  void FrameRemoveUntagAdc (FrameH *frame);
  void FrameRemoveUntagEvent(FrameH *frame);
  void FrameRemoveUntagProc (FrameH *frame);
  void FrameRemoveUntagSer (FrameH *frame);
  void FrameRemoveUntagSim (FrameH *frame);
  void FrameRemoveUntagSimEvt(FrameH *frame);
  void FrameRemoveUntagStat (FrameH *frame);
  void FrameRemoveUntagSum (FrameH *frame);
- These functions remove (free) all the channels which are not tagged. After a call to this function, the FrameUntag function will have no effect;

FrameWrite

- Syntax: **int FrameWrite (FrameH *frame, FrFile *oFile);**

- where:
    - frame is the pointer to the frame Header to be written
    - oFile is the pointer to the output file.
- This functions returns 0 in case of success or an error code in case of failure.

FrameWriteToBuf

- Syntax: **long FrameWriteToBuf (FrameH *frame, unsigned short comp, char *buf, long nBytes, int computeChecksum));**
- where:
    - frame is the pointer to the frame Header to be written
    - comp gives the compression algorithm used at writing time (see FrFileONew). If comp < -1, the checksum is computed when writing the frame in memory and the compression level used is the absolute value of comp.
    - buf is the buffer which will receive the frame
    - nBytes is the buffer size
    - computeChecksum could have the folowing values :
        - 0: no checksum are computed when writing the frame
        - if the first bit (i.e. like 1 or 3) is set: the file checksum is computed
        - if the second bit is set (i.e like 2 or 3) : the structures checksum are computed.
        - To compute all checksums, use “3”.
- This function returns the number of bytes written or 0 in case of error.

FrAdcData: ADC's data manipulation

FrAdcDataDecimate

- This function reduce the sampling frequency of and ADC by averaging nGroup bins together (in this version, no filter is performed). It increase the number of bits of the appropriate number of nGroup is positive or keep the original number of bits of nGroup is negative. This version works only for short, int, float and double.
- Syntax: **int FrAdcDataDecimate (FrAdcData *adc, int nGroup)**
- This functions returns 0 in case of success and a non zero value in case of error.

FrAdcDataDump

- This function produce a formatted dump of the Adc data. The recommended debug level is 2 (with 0 no output is produced).
- Syntax: **void FrAdcDataDump (FrAdcData *adc, FILE *fp, int debugLvl)** were:
    - adc is the pointer to the FrAdcData structure provided by the user
    - fp is the file pointer where the debug information will be send (like stdout)
    - debugLevel is the debug level provided by the user. 0 means no output at all, 2 gives about 5 lines of information.
- Remark: The DataValid flag is printed according the Virgo use. DataValid = 0 means no problem. The six lower bits are used to describe the following problems:

  1 means non-valid floating point (only used for floating point)
  2 means some data are missing at known position in the vector (see adc->next vector)
  3 means some data are missing at unknown position in the vector
  4 means front end error: hardware parity error (DOL error)
  5 means front end error: too slow DAQ (FIFO full for instance)
  6 means front end error: invalid format

  For example dataValid = 0x14 means FIFO full and missing data at unknown position.

FrAdcDataFind

- This function find an FrAdcData structure in a frame. It returns the pointer to the FrAdcData or null if the structure does not exist.
- Syntax: **FrAdcData* FrAdcDataFind (FrameH* frame, char* name)**

FrAdcDataFree

- This function free all the space allocated for the FrAdcDat structure and the linked structure. This function should be used only if the FrAdcData has been created outside a frame like when using random access (FrAdcDataReadT).
- Syntax: **FrAdcData *FrAdcDataFree (FrAdcData *adc);**

FrAdcDataGetSize

- This function returns the memory used by an ADC structure and the associated vector (in bytes)
- Syntax: **FRLONG FrAdcDataGetSize (FrVect *vect)**

FrAdcDataNew and FrAdcDataNewF

- These functions allocates the space for the data of an ADC, and attach it to the FrameH structure (including the creation of the FrRawData structure if does not yet exist). They just differ by the number of parameters : FrAdcDataNewF perform a full fill of the FrAdcData structure.
- Syntax:
    - **FrAdcData *FrAdcDataNewF (FrameH *frame, char *name, char *comment, unsigned int channelGroup, unsigned int channelNumber, int nBits, float bias, float slope, char *units, double sampleRate, int nData);**
    - **FrAdcData* FrAdcDataNew (FrameH* frame, char* name, double sampleRate, int nData, int nBits)**
- The parameters (provided by the user) are:
    - frame (FrameH*) is the pointer to the root frameH structure. frame = NULL is a valid option (the FrAdcData is created independently of a frame)
    - name (cha*) is the ADC name. This name should be unique within a frame for all ADC structures.
    - comment (char *) is any comment the users need to add to the adc description (could be NULL).
    - channelGroup (int) is the channel group (usually a ,mix of crate number, slot number)
    - channelNumber (int)
    - nBits (int) is the number of bits used to store the information. The word length will be either 1, 2, 4 (or 8) bytes. A negative values means that we store floating point number (nBits = 12 is store as a short, nBits =-32 is stored in a 4 bytes float).
    - bias. (float) Any known pedestal
    - slope (float). The calibration constant.
    - units (char) The calibration unit
    - sampleRate (double) is the sampling frequency in Hz.
    - nData (int) is the number of data for this ADC within a frame
- In case of problem, the function returns NULL.
- Remark: this function only allocate the space. The user has to fill the adc data vector. For example to fill the vector with values coded on 2 bytes:

```
for(i=0; i<adc->data->nData; i++)
    {adc->data->dataS[i] = (the adc value);}
```

- It is possible to allocate multidimension vectors, to support images for instance. In that case, the first dimension must be the time.

FrAdcDataReadT

- Syntax: **FrAdcData *FrAdcDataReadT (FrFile *iFile, char *name, double gtime);**

- This function perform a random read access on the file *iFile for a given GPS time (gtime). I gtime = 0 then the data for the first frame in the file is returned. Only the data for the given Adc are read. Several adc name could be requested in name (name should be separate using space). Names could also include wild card. The function return a pointer to the FrAdcData structure or NULL in case of error (frame not in file, not Table Of Content, malloc failed). It returns also the associated vector but not the associated table.
- After using FrAdcDataRead, the user should free the memory by calling FrAdcDataFree since the FrAdcData structure has been directly extract from a file whitout frame to take care of memory clean up.

FrAdcDataSetDataValid, ..SetFShift, ...SetTOffset

- Syntax:

  void FrAdcDataSetAux (FrAdcData *adc, FrVect *aux);
  void FrAdcDataSetDataValid (FrAdcData *adc, unsigned short dataValid);
  void FrAdcDataSetFShift (FrAdcData *adc, double fShift, float phase);
  void FrAdcDataSetTOffset (FrAdcData *adc, double tOffset);

- These functions set some fields in the FrAdcData structure.

FrDetector

- **void FrDetectorDump (FrDetector *detector, FILE *fp, int debugLvl);** Dump a detector structure.
- **FrDetector *FrDetectorNew (char *name);** Allocate a detector structure
- **void FrDetectorFree (FrDetector *detector);** Free a detector structure and associated data.

FrEvent

- Constructor: **FrEvent *FrEventNew (FrameH *frameH,**
  char *name, char *comment, char *inputs,
  double GTime, float timeBefore, float timeAfter,
  float amplitude, float probability, char *stat,
  FrVect *data, int nParam, ...);
  When nParam is not zero, the event parameters are added right after nParam as a sequence of name[0], value[0], name[1], value[1],... WARNING: the type of the additional parameters needs to be a 'double' even if they are store as 'float'.
  Example of use:
  event = FrEventNew(frame, "Inspiral","MBTA algorithm with 2.5PN templates","V0:Pr_B1_ACq"
  710123123.44, 10., 0.1, 1.e-21, 5.3, "signal/rms", NULL, 3, "m1", 1.4, "m2", 1.4, "chi2" 3.2);
- To copy one event (and the associated data if any, but not the linked list): **FrEvent* FrEventCopy (FrEvent *eventl);**
- Dump: **void FrEventDump (FrEvent *event, FILE *fp, int debugLvl);**
- Destructor: **void FrEventFree (FrEvent*event);**
- Find it in a frame: **FrEvent *FrEventFind(FrameH *frame, char *name, FrEvent *last).** Since there could be more than one event with the same name in one single frame, the "last" parameters is used to make the selection. This function will return the next FrEvent structure following the last structure in the linked list and matching the "name". If last = NULL, the function returns the first event.

- To add an event to a frame: **void FrameAddEvent(FrameH *frame, FrEvent *event).** The event(s) (a single one or a linked list) is added at the end of the event linked list of this frame.
- File random access
- To find all the events within a given time range and with some selection on the event amplitude: **FrEvent *FrEventReadT (FrFile *iFile, char *name, double tStart, double length, double amplitudeMin, double amplitudeMax);**
  This function perform a random read access on the file *iFile. It returns all FrEvent structure (as a linked list) which have a time between tStart and tStart+length and with an amplitude in the [amplitudeMin, amplitudeMax] range. It does NOT returns the associated vector nor the associated tables (this could be added later on using FrEVentReadData). The string name could contain several names and wild cards. The function returns a pointer to the first FrEvent structure of the linked list or NULL in case of error (frame not in file, not Table Of Content, malloc failed).
- To find all the events within a given time range and with some selection on several parameters.: **FrEvent *FrEventReadTF (FrFile *iFile, char *name, double tStart, double length, int readData, int nParam, ...);**
  the additional parameters are: **char* paramName1, double min1, double max1, char* paramName2, ...)** where the paramName* are "amplitude", "timeBefore", "timeAfter" or one of the extra event parameter

  This function perform a random read access on the file *iFile. It returns all FrEvent structure (as a linked list) which have a time between tStart and tStart+length and with the extra parameters in the required range.The associated vector is read if the readData flag is set to 1 (or not read if set to 0). The string name could contain several names and wild cards. The function returns a pointer to the first FrEvent structure of the linked list or NULL in case of error (frame not in file, not Table Of Content, malloc failed).
  Example of use: event = FrEventReadTF(iFile,"Inspiral*",t0,50.,1, 2, "M1", 2., 3., "M2", 1., 3.); will return the linked list of all events with a name starting by Inspiral, with a time in the t0, t0+5à range, with a parameter M1 (and M2) in the range 2.;3. (1.;3.).

- To read the associated vector for one event: **int FrEventReadData (FrFile *iFile, FrEvent *event);**
- **Parameters handling:**
  - Add one more parameter to an event:
    FrEvent *FrEventAddParam (FrEvent *event, char* paramName, double value);
    This function returns NULL in case of error (bad input parameters of malloc failed).
  - Add one vector parameter to an event:
    int FrEventAddVect (FrEvent *event, FrVect* vect, char* newName);
    int FrEventAddVectF (FrEvent *event, FrVect* vect, char* newName);
    This function attach a copy of a vector (cast to a vector of float for ...AddVectF) to an event. If newName is not NULL, the vector name is changed. It returns 0 in case of success.
  - Get the value for one parameter:
    double FrEventGetParam (FrEvent *event, char* paramName);
    This function returns -1. if the parameter could not be found.
  - Get event the parameter id:
    int FrEventGetParamId (FrEvent *event, char* paramName);
    This function returns the parameter number in the list or -1 if the parameter could not be found. The parameter value could then be access at event->parameters[id].
  - To find the pointer to a vector attached to the event:
    FrVect* FrEventFindVect (FrEvent *event, char* vectName);
    This function returns a pointer to the vector or NULL if not found. The returned vector is uncompressed. The user should NOT free the vector.
  - To return a copy of a vector attached to the event:
    FrVect* FrEventGetVect D(FrEvent *event, char* vectName);
    FrVect* FrEventGetVect F(FrEvent *event, char* vectName);
    This function returns a pointer to a copy of type double (..VectD) or float (...VectF) of a vector or NULL if not found. The user MUST free the vector after its use.
- **To save on standalone event in a file:**
  - **int FrEventSaveOnFile(FrEvent *event, FrFile *oFile);** This function creates the needed frame header. If the event is part of a linked list, only this event is saved on file. It returns 0 in case of succes.

Input File: FrFileI

FrFileIDump

- This function dumps a summary (file name starting time and file length) of a file. This could be used to create Frame File List.
- Syntax : **void FrFileIDump (FrFile *iFile, FILE *fp, int debugLvl, char *tag);**
- If debugLvl = 0, then only the available values of start time are printed. To get the real values, you need to set debugLvl to 2. The 'tag' parameter is used to defined a list of channel for which we require some information (for instance, use tag = "*B1*" to get only the list of channel with a name containing 'B1'.
- Example: FrFileIDump (file, stdout, 1, NULL); will dump on the standard output all the file names and time information.

FrFileIEnd: close a input file

- This function close an input file and free all the associated structures.
- Syntax: **void FrFileIEnd (FrFile *iFile);**

FrFileIGetVect, ...GetV, ...VectF, ...VectFN, ...VectD, ...VectDN:

- These functions provide random access for the vector of a single channel (FrAdcData, FrSimData or FrProcData) called 'name'. The starting GPS time is tStart, the vector length in seconds is length.
- Syntax:
  - **FrVect *FrFileIGetVect(FrFile *iFile, char *name, double tStart, double length);**
  - **FrVect *FrFileIGetVectD(FrFile *iFile, char *name, double tStart, double length);**
  - **FrVect *FrFileIGetVectDN(FrFile *iFile, char *name, double tStart, double length);**
  - **FrVect *FrFileIGetVectF(FrFile *iFile, char *name, double tStart, double length);**
  - **FrVect *FrFileIGetVectFN(FrFile *iFile, char *name, double tStart, double length);**
- The returned vector starts at the requested time and last exactly the requested number of second (this is new since version 5).
- The vector is converted to a vector of floats (type=FR_VECT_4R) for ...GetVectF and ...GetVectFN or double (type=FR8VECT_8R) for ...GetVectD or ...GetVectDN
- The fonctions FrFileIGetVectDN and ...VectFN return a normalized vector using the FrAdcData information.
- FrFileIGetV is the old name for FrFileIGetVect.
- If there are missing frames in the request time stretch, the corresponding bins are filled with the vector mean value and an additional vector is returned attached to the field "next" of the main vector. This additional vector has the same size of the main vector but each bin contains the number of frame found for this sample. A zero is therefore used for missing sample. If there are no missing samples the filed "next" is set to "NULL".
  - **FrVect *FrFileIGetVAdc (FrFile *iFile, char *name, double tStart, double length, int group);**
  - **FrVect *FrFileIGetVSim (FrFile *iFile, char *name, double tStart, double length, int group);**
  - **FrVect *FrFileIGetVProc (FrFile *iFile, char *name, double tStart, double length, int group);**

FrFileIGetXXXNames:

- Syntax:

  FrVect *FrFileIGetAdcNames(FrFile *iFile);
  FrVect *FrFileIGetDetectorNames (FrFile *iFile);

FrVect *FrFileIGetEventNames (FrFile *iFile);
FrVect *FrFileIGetProcNames(FrFile *iFile);

FrVect *FrFileIGetSimNames(FrFile *iFile);
FrVect *FrFileIGetSimEventNames (FrFile *iFile);

FrVect *FrFileIGetStatNames (FrFile *iFile);

- These functions extract channel or event names from the Table Of Content. It returns one vector containing the list of names (vector of char *: FR_VECT_STRING) or NULL in case or error.

FrFileIGetFrameInfo:

- Syntax: **FrVect *FrFileIGetFrameInfo (FrFile *iFile, double tStart, double length);**
- This function extracts frame information from the Table Of Content. The tStart and length arguments could be used to specify a time range. It returns a linked list of three vectors (or NULL in case or error):
  - The Frame GPS starting time (vector of double)
  - The frame length (vector of double)
  - The frame data quality (vector of int)
- There is one entry per frame in these vector. The frame are sorted by increasing GPS time.

FrFileIGetChannelList:

- Syntax: **char* FrFileIGetChannelList(FrFile *iFile, int gtime);**
- This function allocates and return a string containing the list of channels contained in a file at a given GPS time and additional meta data.
- The user should take care of the memory free.
- If gtime == 0, the channel list is return for the current file position or for the beginning of the file if no frame has been read

FrFileIGetEventInfo and SimEventInfo:

- Syntax:
  FrVect *FrFileIGetEventInfo (FrFile *iFile, char *tag, double tStart, double length, double amplitudeMin, double amplitudeMax);
  FrVect *FrFileIGetSimEventInfo (FrFile *iFile, char *tag, double tStart, double length, double amplitudeMin, double amplitudeMax);
- These functions extract (simulated) event information from the Table Of Content. The tStart and length arguments could be used to specify a time range as well the minimum and maximum amplitude. The paramter "tag" let you select the events you want. It could contain wilde cards. If tag = "*" then all events are selected. It returns a linked list of two vectors (or NULL in case or error):
  - The event GPS time (vector of double)
  - The event amplitude (vector of float)
- The events are sorted by increasing GPS time.

FrFileINew:

- This function open one or several files.
- Syntax: **FrFile *FrFileINew (char *fileName);** where **fileName** could be:
  - a single file name like "file1.dat"
  - a list of files separeted by space like "file1.dat file2.dat". In that case file1.dat will be first open and when all the frames from this file will by read, it will automatically open the file file2.dat without any special action from the user. It is an easy way to concatenate files. Or to use several files with random access.
  - a **Frame File List.** This is an ASCII file with the file extension ".ffl". which contain either:

        - a plain list of file like
          file1.dat
          file2.dat
          file3.dat
        - a list a file with GPS information for the file start, file length, time of the first event, time of the last event. This is the output of the FrDump tool with the "-d 0" option. So the best way to build the ffl is to issue a command like "FrDUmp -i file*gwf -d 0 > file.ffl". This will give for instance:

```
file1.dat      666000000.000000 11  666000000.200000 666000010.100000
file2.dat      666000011.000000 11  666000011.200000 666000021.100000
file3.dat      666000022.000000 11  666000022.200000 666000032.100000
```

          These full ffl provide efficient random access on a large number of files.
    - Remark: all file name should start by an non digit character.
    - To read frames from a buffer, see the function FrameReadFromBuf
    - If you want to turn off the decompression during frame read you should type after the file opening:

      iFile->compress = 1;

      Then all the vectors will remain compressed.

FrFileINewFd

- This function open an input file for a given file descriptor
- Syntax : **FrFile *FrFileINewFd (FrIO *frfd);**
- This function returns a pointer to the input file or NULL if an error occurs.

FrFileIRewind

- This function rewind to file. The next FrameRead will then return the first frame in the file.
- Syntax : **FrFile *FrFileRewind (FrFile *file);**
- This function returns a pointer to the input file or NULL if an error occurs.

FrFileISetTime

- This function set the file to a given GPS time.The next frame read will be for this GPS time. If called for a time corresponding to a gap in the file, the reading pointer is set to the next existing frame.
- Syntax : **int FrFileSetTime(FrFile *file, double gpsTime);**

FrFileITFirstEvt, FrFileITLastEvt

- This function return the GPS time of the first/last event in the file(s). If more than one file is given, it returns the minimum event time and the maximum event time for all the files. These functions work only for files with table of content. They return a negative time in case of error.
- Syntax :

  double FrFileITFirstEvt (FrFile *iFile);
  double FrFileITLastEvt (FrFile *iFile);

FrFileITStart, FrFileITEnd

- This function return the GPS time of the first/last frame in the file(s). If more than one file is given, it returns the minimum starting time and the maximum end time of all files. They work only for files with table of content. They return a negative time in case of error.
- Syntax :

double FrFileITStart (FrFile *iFile);
double FrFileITEnd (FrFile *iFile);

FrFileITNextFrame

- Syntax : **double FrFileITNextFrame (FrFile *iFile, double gtime);**
- This function return the GPS time of the next frame (ie the frame starting after gtime). It works only for files with table of content. It returns a negative time in case of error.

Output File: FrFileO

FrFileOEnd:

- To close an output file, you need to call:
  int FrFileOEnd (FrFile *file);

FrFileONewXXX

- This function open an output file for a given name. or file descriptor.
- Syntax:

FrFile *FrFileONew (char *fileName, int compress);
FrFile *FrFileONewH (char *fileName, int compress, char *program);
FrFile *FrFileONewM (char *fileName, int compress, char *program, int maxLength);
FrFile *FrFileONewMD (char *fileName, int compress, char *program, int maxLength, char* filePrefix, int dirPeriod);
FrFile *FrFileONewFd (FrIO *frfd, int compress);

- compress gives the compression algorithm used at writing time.
  - -1 to write data without changing the initial compression state
  - 0 for no compression,
  - 1 for gzip (The level of gzip compression could be set by a call to FrFileOSetGzipLevel (file, level) with 0<level<10. The default value is level=1.)
  - 3 for differentiation and gzip.
  - 5 for differentiation and zeros suppress (only for short)
  - 6 for differentiation and zeros suppress for short and gzip for other.
  - 8 for differentiation and zeros suppress for short int and float and gzip for other. (recommended)
- **FrFileONewH** has an extra argument (program) which is string which will be added to the history record at writing time.
- **FrFileONewM** has one more extra parameter: maxLength that define the maximum length for a file in second. When this mawimum is reached, the file is closed and a new one is open. This is convienent to handle large data set. The name of these files is no more just "fileName" but "fileName-GPS-maxLength.gwf " (like V-R-730123000-100.gwf if fileName = "V-R").
- **FrFileONewMD** has two more parameters: filePrefix which defines the file beginning of the file name and dirPeriod which defines the time spam cover by a directory. With these parameters a new file is open each time the GPS time reach a multiple of maxLength and a new folder is created each time the GPS time is a multiple of dirPeriod. If path=”./Test”, maxLength=100, prefix = “Test” and dirPeriod=1000 then file name will be like: ./Test-800000/Test-800000100-100.gwf).
- When an output file has been open, you can suppress the writing of the Table Of Content for the time series (FrAdcData, FrSimData, FrProcDat, FrSerData, Summary) by setting:

```
oFile->noTOCts = FR_YES;
```

FrFileOPutV

- This function write on or more vectors in a file. It automatically create a frame and attach an FrProcData channel that own the vector as data member.
- Syntax:

  int FrFileOPutV (FrFile *oFile, FrVect *vect);

- This function returns 0 in case of succes or an error code.
- The GPS time of the vector (vect->GTime) needs to be properly set if more than one vector is put in the file.

FrFileOSetMsg

- This function sets the name used at writing time for the history record.
- Syntax: **void   FrFileOSetMsg   (FrFile *oFile, char *msg);**
- Remark: if msg = NULL no history message will be added to the file. However, it is advised to always add a history record.

File Checksum

- File checksum are computed when writing a file on disk. They are not computed/checked when the frame is write or if the frame is written in memory.
- To turn on(or off) the checksum computation/check just: iFile->chkSumFlag == FR_YES (or FR_NO); after the file has been open (FrFileIOpen or FrFileOOpen).
- The utility FrCheck could be used to verify the file checksum.
- Checksum are available only since version 4.40

FrFilter

- A filter structure as be set up to hold lilnear filter information to be stroed in file. The following function could be used to manage them. See the FrFilter.h file for more details:
  - **void FrFilterFree(FrFilter* filter);**
  - **FrFilter* FrFilterNew(char* name, double fs, double gain, int ntaps, ...);**
    constructor: the additional parameters are ntaps "a" values followed by ntaps "b" values
  - **void   FrFilterDump(FrFilter *f, FILE *fp, int debugLvl);**
    This function dumps on file fp (like 'stdout') the filters parameters
  - **FrVect*   FrFilterPackToVect(FrFilter *filter);**
    This function creates a vector containing the content of the filter
  - **FrFilter* FrFilterGetFromVect(FrVect *vect);**
    This function creates a filter which was previously packed in a vector
  - **void   FrProcDataAddFilter(FrProcData *proc, FrFilter *Filter);**
    this function pack a filter into a vector and attach it to the proc data. The filter structure is untouched
  - **FrFilter* FrProcDataGetFilter(FrProcData *proc, char *name);**
    This function creates a FrFilter structure according to the parameters of the filter called "name" and attached to the proc data
  - **void   FrStatDataAddFilter(FrStatData *stat, FrFilter *filter);**

    - **FrStatData* FrameAddStatFilter(FrameH* frame, char* detectorName,char* statDataName, unsigned int tStart, unsigned int tEnd, unsigned int version, FrFilter *filter)**
    - **FrFilter* FrameGetStatFilter(FrameH *frame, char *detectorName, char *statDataName, char *filterName, int gpsTime);**

FrHistory

The best way to add an history record is to used the FrFileONewH function which will set the default history record produced at each FrameWrite to the one you want. However, the FrHIstory records could be manipulated using the following functions.

FrHistoryAdd

- This function add an history record. A time stamp is automatically added. The string comment is provided by the user. Its format is free. If frame = NULL the history structure is created but not attached to the frame header. These history records are useful to keep track of the various frame processing. This function returns the pointer to the first History structure or NULL in case of malloc error.
- Syntax: **FrHistory *FrHistoryAdd (FrameH *frame, char *comment);**

FrHistoryFree

- This function free the history records and all attached history.
- Syntax: **void FrHistoryFree (FrHistory *history);**

FrMsg

- An online log message could be added to the frame by using the function:
  FrMsg *FrMsgAdd (FrameH *frame, char *alarm, char *message, unsigned int severity);
  The string message as well as the alarm name are provided by the user. Its format is free. The severity value is provided by the user. frame = NULL is a valid option. This function returns the pointer to the FrMsg structure or NULL in case of malloc error.
- To dump it : **void FrMsgDump (FrMsg *msg, FILE *fp, int debugLvl);**
- Find it in a frame: **FrMsg *FrMsgFind(FrameH *frame, char *alarm, FrMsg *last).** Since there could be more than one FrMsg with the same name in one single frame, the "last' parameters is used to make the selection. This function will return the next FrMsg structure following the last structure in the linked list and matching the "name". If last = NULL, the function returns the first found structure.

FrProcData

- These functions have the same meaning as for the FrAdcData structure:
    - Constructor: **FrProcData *FrProcDataNew (FrameH *frame, char *name, double sampleRate, int nData, int nBits);**
    - Dump: **void FrProcDataDump (FrProcData *procData, FILE *fp, int debugLvl);**
    - Destructor: **void FrProcDataFree (FrProcData *procData);**
    - Find it in a frame: FrProcData ***FrProcDataFind (FrameH *frame, char *name)**
    - File random access (FrProcData and vector): **FrProcData *FrProcDataReadT (FrFile *iFile, char *name, double gtime)**

- To add an history record: **FrHistory *FrProcDataAddHistory(FrProcData *proc, char *comment, int nPrevious, ...)**. This function will add an history record containing the comment "comment" and will copy "nPrevious" previous history record(s) from other FrProcData. The additional parameters are the "nPrevious" FrHistory structures. The usage is the following:

  FrProcDataAddHistory(proc, "FFT(V1:Pr_B1_ACq)", 0)   will add only one history record
  FrProcDataAddHistory(proc, "A+B", 2, procA->history, procB->history) will add one history record and will copy the history records from procA and procB where procA and procB are FrProcData structures

- Parameters handling:
- Add a parameter to an FrProcData structure: **FrProcData *FrProcDataAddParam (FrProcData *proc, char* paramName, double value);** This function returns NULL in case of error (bad input parameters or malloc failed).
- Get parameter value: **double FrProcDataGetParam (FrProcData *proc, char* paramName);**
  This function returns -1. if the parameter could not be found.
- Get parameter id: **int FrProcDataGetParamId (FrProcData *proc, char* paramName);**
  This function returns the parameter number in the list or -1 if the parameter could not be found. The parameter value could then be access at proc->auxParam[id].
- To attach a vector to a procData: **void FrProcDataAttachVect(FrProcData *proc, FrVect *vect);**
  Afte this call the vector still belong to the procData and the user must NOT try to free it.
- To find the vector named "name" attached to one procData: **FrVect* FrProcDataFindVect(FrProcData *proc, char *name);**
  After this call, the vector is still own by the proc data (the user must NOT free it).

FrSerData

- Unless specified, these functions have the same meaning as for the AdcData structure:
  - Constructor: **FrSerData *FrSerDataNew (FrameH *frame, char *smsName, unsigned int serTime, char *data, double sampleRate);** The SerData is defined by a name and a GPS time. Usually the data are all included in the data string. The suggest form is to use a string which is a sequence of names and values (for instance "P1 1.e-6 P2 1.e-7").

  Units and serData:

  If in the FrSerData string, there is the keyword "units" followed by a string like "mbar" then this unit will be stored by the DAQ (instead of the default "Count"), put in the units filed of the FrAdcData channel produced by the trend frame builder and then display by dataDisplay when looking at trend data.
  Notice that once you use the keywords "units", this unit will be assigned to all following data. So, if you have a mix of channels, it is better to group them and put the channels without unit at the beginning of the string.
  As an example, the string
  "K 345 units C T1 21.34 T2 25.3 units mbar P1 1013.1"
  Translate to
  K = 345 Counts
  T1 = 21.34 C
  T2 = 25.3 C
  P1 = 1013.1 mbar

  - Dump: **void FrSerDataDump (FrSerData *serData, FILE *fp, int debugLvl);**
  - Destructor: **void FrSerDataFree (FrSerData *serData);**
  - Find it in a frame: FrSerData ***FrSerDataFind (FrameH *frame, char *name, FrSerData *last).** Since there could be more than one FrSerData with the same name in one single frame, the "last' parameters is used to make the selection. This function will return the next FrSerData

structure following the last structure in the linked list and matching the "name". If last = NULL, the function returns the first found structure.

  - Find the value of one parameter (smsParama): **int FrSerDataGet (FrameH *frameH, char *smsName, char *smsParam, double *value);** It assumes that the data are stored in the data string as names followed by values for the serial data smsName. It returns 0 in case of success.
  - Return directly the parameter value or a default value if not found: **double FrSerDataGetValue (FrameH *frameH, char *smsName, char *smsParam, defaultValue);**
  - File random access: **FrSerData *FrSerDataReadT (FrFile *iFile, char *name, double gtime)**

FrSimData

FrSimData *simData, FILE *fp, int debugLvl);

- Destructor: **void FrSimDataFree (FrSimData *simData);**

- Find it in a frame: FrSimData ***FrSimDataFind (FrameH *frame, char *name)**

- File random access (FrSimData and vector): **FrSimData *FrSimDataReadT (FrFile *iFile, char *name, double gtime)**

- File random access (associated vector for one or more frame): **FrVect *FrFileGetVSim (FrFile *iFile, char *name, double tStart, double length)**

FrSimEvent

- Unless specified, these functions have the same meaning as for the AdcData structure:
  - Constructor: **FrSimEvent *FrSimEventNew (FrameH *frameH,**
    char *name, char *comment, char *inputs,
    double GTime, float timeBefore, float timeAfter,
    float amplitude, FrVect *data, int nParam, ...);
    When nParam is not zero, the event parameters are added right after nParam as a sequence of name[0], value[0], name[1], value[1],...
    Example of use:
    Add one vector parameter to an event:**int FrSimEventAddVect (FrSimEvent *event, FrVect* vect, char* newName);**
    int FrSimEventAddVectF (FrSimEvent *event, FrVect* vect, char* newName);
  - This function attach a copy of a vector (cast to a vector of float for ...AddVectF) to an event. If newName is not NULL, the vector name is changed. It returns 0 in case of success.Get the value for one parameter: **double FrSimEventGetParam (FrSimEvent *event, char* paramName);** This function returns -1. if the parameter could not be found.
  - Get event the parameter id: **int FrSimEventGetParamId (FrSimEvent *event, char* paramName);** This function returns the parameter number in the list or -1 if the parameter could not be found. The parameter value could then be access at event->parameters[id].
  - To find the pointer to a vector attached to the event:
    FrVect* FrSimEventFindVect (FrEvent *event, char* vectName);
    This function returns a pointer to the vector or NULL if not found. The user should NOT free the vector.

- To return a copy of a vector attached to the event:
  FrVect* FrSimEventGetVect D(FrEvent *event, char* vectName);
  FrVect* FrSimEventGetVect F(FrEvent *event, char* vectName);
- This function returns a pointer to a copy of type double (..VectD) or float (...VectF) of a vector or NULL if not found. The user MUST free the vector after its use.
- Dump: **void FrSimEventDump (FrSimEvent *simEvent, FILE *fp, int debugLvl);**
- Destructor: **void FrSimEventFree (FrSimEvent *simEvent);**
- Find it in a frame: FrSimEvent***FrSimEventFind (FrameH *frame, char *name, FrSimEvent *last).** Since there could be more than one FrSimEvent with the same name in one single frame, the "last' parameters is used to make the selection. This function will return the next FrSimEvent structure following the last structure in the linked list and matching the "name". If last = NULL, the function returns the first found structure.
- File random access (FrSimEvent and vector): **FrSimEvent *FrSimEventReadT (FrFile *iFile, char *name, double tStart, double length, double amplitudeMin, double amplitudeMax);**
  This function perform a random read access on the file *iFile. It returns all (as a link list) FrSimEvent structure which have a time between tStart and tStart+length and with an amplitude in the [amplitudeMin, amplitudeMax] range. It returns also the associated vector (if any) but not the associated tables. The string name could contain several names and wild card. The function returns a pointer to the first FrSimEvent structure of the linked list or NULL in case of error (frame not in file, not Table Of Content, malloc failed).
- File random access
- To find all the events within a given time range and with some selection on the event amplitude:
  FrSimEvent *FrSimEventReadT (FrFile *iFile, char *name, double tStart, double length, double amplitudeMin, double amplitudeMax);
  This function perform a random read access on the file *iFile. It returns all FrEvent structure (as a linked list) which have a time between tStart and tStart+length and with an amplitude in the [amplitudeMin, amplitudeMax] range. It does NOT returns the associated vector nor the associated tables. The string name could contain several names and wild cards. The function returns a pointer to the first FrEvent structure of the linked list or NULL in case of error (frame not in file, not Table Of Content, malloc failed).
- To find all the events within a given time range and with some selection on several parameters.:
  FrSimEvent *FrSimEventReadTF (FrFile *iFile, char *name, double tStart, double length, int readData, int nParam, ...);
  the additional parameters are: **char* paramName1, double min1, double max1, char* paramName2, ...)** where the paramName* are "amplitude", "timeBefore", "timeAfter" or one of the extra event parameter
  This function perform a random read access on the file *iFile. It returns all FrEvent structure (as a linked list) which have a time between tStart and tStart+length and with the extra parameters in the required range.The associated vector is read if the readData flag is set to 1 (or not read if set to 0). The string name could contain several names and wild cards. The function returns a pointer to the first FrEvent structure of the linked list or NULL in case of error (frame not in file, not Table Of Content, malloc failed).
  Example of use: event = FrEventReadTF(iFile,"Inspiral*",t0,50.,1, 2, "M1", 2., 3., "M2", 1., 3.); will return the linked list of all events with a name starting by Inspiral, with a time in the t0, t0+5à range, with a parameter M1 (and M2) in the range 2.;3. (1.;3.).

FrStatData

A static data is a structure which may stay valid for more than one frame. FrStatData is versioned and is tied to particular detectors and can change from epoch to epochIt is written on file only once. These data stay as long as they are valid compare to the frame time boundary, or as long there is not a new bloc of data with the same name but with a highest version number. In the case of long frames there could be several static data with the same name if they have different starting times which cover the frame duration. The FrStatData do not need to be confined to the time range of the frame in which it is written.

FrStatDataAdd

- This function adds a static data bloc.
  void FrStatDataAdd (FrDetector *detector, FrStatData *sData);

- See also:
  - **int FrameAddStatData(FrameH* frame, char* detectorName, FrStatData *stat);**
    This function attached a static data to a detector structure belonging to this frame.
    If no detector exists for this name, a new one is created.
    If name is NULL, it is attached to the first detector or to a new detector called "Default" is there is no detector structure.
    Any new detector structure is attached to the frame->detectProc list.
  - **FrStatData* FrameAddStatVector(FrameH* frame, char* detectorName, char* statDataName, unsigned int tStart, unsigned int tEnd, unsigned int version, FrVect* vect);**
    This function attached a vector to a static data to a detector structure belonging to this frame.
    If no detector exists for this name, a new one is created.
    If name is NULL, it is attached to the first detector or to a new detector called "Default" is there is no detector structure.
    Any new detector structure is attached to the frame->detectProc list.
  - **int FrDetectorAddStatData(FrDetector* detector, FrStatData *stat);**
    This function attached a static data to a detector structure. The user must NOT free the static data since it will then belong to the detector.
  - **void FrStatDataAddVect(FrStatData *stat, FrVect *vect);**
    This function attached a vector to a static data structure. The user must NOT free the vector since it will then belong to the static data.

FrStatDataDump

- To dump the static data content on the FILE 'fp' (useful values of debugLevel are 1, 2, or 3):
  void FrStatDataDump (FrStatData *sData, FILE *fp, int debugLevel);

FrStatDataFind

- This function finds a static data bloc.
  FrStatData *FrStatDataFind (FrDetector *detector, char *name, unsigned int timeNow);
  timeNow is the time for which we want the static data. If timeNow = 0 then the first static data with that name is return.
- See also:
  - **FrVect* FrameGetStatVect(FrameH *frame, char *detectorName, char *statDataName, char *vectorName, int gpsTime);**
    This function return a copy vector named "vectorName" attached to the static data named "statDataName".
    The user must take car of the vector free to avoid memory leak.
  - **FrStatData *FrameFindStatData(FrameH *frame, char *detectorName, char *statDataName, int gpsTime);**
    This function return the pointer to the static data named "statDataName" and attached to a frame.
    The user must NOT free the structuer after using it.
  - **FrStatData* FrDetectorFindStatData(FrDetector *det, char *statDataName, int gpsTime);**
    This function return the pointer to the static data named "statDataName" and attached to a detector.
    The user must NOT free the structuer after using it.

FrStatDataFree

- This function free the static data bloc including the vectors and all attached static data.
  void FrStatDataFree (FrStatData *sData);

FrStatDataFreeOne

- This function free the static data bloc including the vectors. It returns the pointer to the next bloc in the linked list.
  FrStatData *FrStatDataFree (FrStatData *sData);

FrStatDataNew

- This function creates a new static data bloc.
  FrStatData *FrStatDataNew (char *name, char *comment, char *represent, unsigned int tStart, unsigned int tEnd, unsigned int version, FrVect *data, FrTable *table);
  where:
  - name is the name of this bloc of static data.
  - comment is some user information
  - tStart is the starting time (GPS) of validity for this bloc
  - tEnd is the end time (GPS) of validity for this bloc (tEnd = 0 means no end)
  - version is the static data version number provided by the user
  - data is the data bloc (like a vector) provided by a user.

  * To attach a static bloc to a frame you should attach it to one detector structure.

FrStatDataReadT

- To extract on static data block from a file using a random access read.
  FrStatData *FrStatDataReadT (FrFile *iFile, char *staticDataName, double gpsTime);

  Example: Suppose you have

stat_data_1	which is valid in [t1,t1+T1)
stat_data_2	which is valid in [t2,t2+T2)

  where t2>= t1 + T1.

  Then if your frame file is for time [t0,t0+T0) and you ask for the stat_data at time t in this frame file with t0<= t< t0 + T0 you will get:

  stat_data_1 , starting at time t1 , if t1<= t< t1 + T1
  stat_data_2 , starting at time t2 , if t2<= t< t2 + T2
  nothing otherwise.

  Note: The vector should not be time series, or if it is the case, the full vector is returned, ignoring it is a time series, (i.e. independently of the requested start time and effective start time). Therefore FrStatData should not be used to store time series to avoid confusion.

FrStatDataTouch

- When you update the content of a static data bloc you should tell the system by calling:
  void FrStatDataTouch (FrStatData *sData);

FrSummary

- Unless specified, these functions have the same meaning as for the AdcData structure:
  - Constructor: **FrSummary *FrSummaryNew (FrameH *frame, char *name, char *comment, char *test, FrVect *moments, FrTable *table);**
  - Dump: **void FrSummaryDump (FrSummary *summary, FILE *fp, int debugLvl);**
  - Destructor: **void FrSummaryFree (FrSummary *summary);**
  - Find it in a frame: FrSummary ***FrSummaryFind (FrameH *frame, char *name).**
  - File random access (FrSimEvent and vector): **FrSummary *FrSummaryReadT (FrFile *iFile, char *name, double tStart, double length);** This function perform a random read access on the file *iFile. It returns all (as a link list) FrSummary structure which have a time between tStart and tStart+length. It returns also the associated vector (if any) but not the associated tables. The string name could contain several names and wild card. The function returns a pointer to the first FrSummary structure of the linked list or NULL in case of error (frame not in file, not Table Of Content, malloc failed).

FrTable

- Complex tables could be created to stored different types of object. However, tables are not efficient for small numbers of values where a simple string encoding or a plain vector is more efficient.
  - Constructor: **FrTable *FrTableNew (char *name, char *comment, int nRow, int nColumn, ...);**
  - Constructor: **FrTable *FrTableCopy (FrTable *table);**
  - Constructor: **void FrTableExpand (FrTable *table);**
  - Dump: **void FrTableDump (FrTable *table, FILE *fp, int debugLvl);**
  - Access on column: **FrVect* FrTableGetCol (FrTable *table, char *colName);**
  - Destructor: **void FrTableFree (FrTable *table);**

FrVect: Vectors handling

FrVectNew

- This function create a multi dimension vector.
- Syntax: **struct FrVect *FrVectNew (int type, int nDim, ...);**
- The parameters (provided by the user) are:
  - type the type of data stored. It could be one of the following value:
    FR_VECT_C, /* vector of char */
    FR_VECT_2S, /* vector of signed short */
    FR_VECT_4S, /* vector of signed int */
    FR_VECT_8S, /* vector of signed long */
    FR_VECT_1U, /* vector of unsigned char */
    FR_VECT_2U, /* vector of unsigned short */
    FR_VECT_4U, /* vector of unsigned int */
    FR_VECT_8U, /* vector of unsigned long */
    FR_VECT_8R, /* vector of double */
    FR_VECT_4R, /* vector of float */
    FR_VECT_8C, /* vector of complex float (2 words per number)*/
    FR_VECT_16C, /* vector of complex double (2 words per number)*/
    FR_VECT_STRING; /* vector of string *
    FR_VECT_2U, /* vector of unsigned short */
    FR_VECT_8H, /* half complex vectors (float) (FFTW order) */ (not part of the frame format; convert to 8C when writing to file)
    FR_VECT_16H, /* half complex vectors (double) (FFTW order) */ (not part of the frame format; convert to 16C when writing to file)
  - nDim the number of dimension (1 for a vector, 3 for a "movie": a set of images,...). If one of the dimension is the time, it must be the first dimension to be properly handled when chaning the duration of a frame.
  - nx[0] The number of element for each dimension (0 is a valid value)

    - dx[0] The step size for each dimension
    - unitX[0] The unit for each dimension .
    - Then, additional parameters for multi dimension vectors: nx[1], dx[1], unitX[1], nx[2],...
- This function return NULL in case of problem (not enough memory). After creation, all the different type of pointer in the FrVect structure point to the same data area. The names of these pointers are:

```
char *data;
short *dataS;
int *dataI;
long *dataL;
float *dataF;
double *dataD;
unsigned char *dataU;
unsigned short *dataUS;
unsigned int *dataUI;
unsigned long *dataUL;
```

- Remark: by default, the vector space is initialized to zero. To bypass this iinitialization, put a minus sign in front of the type argument.

FrVectNewTS

- This function creates a one dimension time serie vector. Like for an FrAdcData, the vector type is set according the number of bit (integer for positive values, float or double for negative value)
- Syntax: **FrVect *FrVectNewTS (char *name, double sampleRate, int nData, int nBits)**
- The parameters (provided by the user) are:

name the name of the vector
sampleRate: sampling frequency
nData The number of elements (0 is a valid value)
nBits: number of bits.

FrVectNew1D

- This function creates a one dimension vector.
- Syntax: **FrVect *FrVectNew1D (char *name, int type, int nData, double dx, char *unitX, char *unitY)**
- The parameters (provided by the user) are:

name the name of the vector
type the type of data stored (see FrVectNew).
nData The number of elements (0 is a valid value)
dx The step size
unitX The step unit (NULL is a valid value) .
unitY The content unit (NULL is a valid value) .

FrVectFree

- This function free a vector and its memory allocated space
- Syntax: **void FrVectFree (struct FrVect *vect)**

FrVectCompress

- This function compress a vector.
- Syntax: **void FrVectCompress (FrVect *vect, int compress, int gzipLvl)** were:

  - vect is the vector provided by the user
  - compresses the type of compression:
    - 6 for differentiation and zeros suppress for short and gzip for other
    - 7 for differentiation and zeros suppress for short, int and float to integer (not part of the current frame format)
    - 8 for differentiation and zeros suppress for short, int and float. (not part of the current frame format)
    - 255 for user defined compression code (definitely not part of the frame format)
  - gzipLvl is the gzip compression level (provided by the user). 0 is the recommended value.
- In normal use, the compression is done at frame write and the user do not need to take care of it.

FrVectCopy

- This function duplicates a vector and its data:
- Syntax: **FrVect *FrVectCopy (FrVect *in)**
- This function returns NULL in case of problem (not enough memory).

FrVectCopyToF, FrVectCopyToD, FrVectCopyToI, FrVectCopyToS

- Syntax:
  FrVect * = FrVectCopyToF(FrVect *vect, double scale, char* newName);
  FrVect * = FrVectCopyToD(FrVect *vect, double scale, char* newName);
  FrVect * = FrVectCopyToI(FrVect *vect, double scale, char* newName);
  FrVect * = FrVectCopyToS(FrVect *vect, double scale, char* newName);
- These functions create a new vector of type float (FrVectCopyToF), double (FrVectCopyToD), int (FrVectCopyToI) or short (FrVectCopyToS). The data are copy using the scale factor 'scale' and casted to the proper type. The new vector will have the same name as the original one except if a newName is provided (value non NULL).
- These functions return NULL in case of error (malloc failed, no input vector).
- Supported input types: all types except complex. Syntax: **FrVect * = FrVectCopyTo(FrVect *vect, double scale, FrVect *copy);**
- This function copy the data from vector vect to the vector copy using the scale factor 'scale'and casted to the vector copy type.
- This function returns NULL in case of error (malloc failed, no input vectors).
- Supported types: all types for vect except complex: float, double, int and short for copy.<

FrVectDump

- To dump a vector in a readable format:
- Syntax: **void FrVectDump (FrVect *vect, FILE *fp, int debugLvl)** were
  - vect is the vector provided by the user
  - fp is the file pointer where the debug information will be send.
  - dbglvl is the debug level provided by the user. 0 means no output at all, 1 gives a minimal description (<5 lines per frame), 3 give you a full vector dump.
- Example: FrVectDump (vect, stdout, 1) will dump the vector information on the standard output.

FrVectDecimate

- This function decimates the vector data by averaging nGroup values together if nGroup is positive. If nGroup is negaive, a pure decimation (no averaging) of -nGroup is performed. The result is put in the vector outVect. (outVect could be the input vector). If outVect = NULL, the result is put in the input vector. The size of outVect should be nGroup time smaller than the size of vect.
- Syntax: **FrVect * FrVectDecimate (FrVect *vect, int nGroup, FrVect *outVect)**
- Examples:
  - FrVectDecimate (vect, 2, NULL) will average two by two the vector content of vect.(0, 1, 2, 3, 4, 5...) -> (0.5, 2.5, 4.5...)

- FrVectDecimate (vect, -2, NULL) will take one values out of two input values.(0, 1, 2, 3, 4, 5...) -> (1, 3, 5...)

FrVectDecimateMin, FrVectDecimateMax

- These functions decimate the vector data by taking the minimum (maximum) values over nGroup values. The result is put in the input vector and the memory allocated is schrinked.
- Syntax: **FrVect * FrVectDecimateMin (FrVect *vect, int nGroup)**
- Syntax: **FrVect * FrVectDecimateMax (FrVect *vect, int nGroup)**
- Examples:
  - FrVectDecimateMin (vect, 2) will transform (0, 1, 2, 3, 4, 5) to (0, 2, 4)
  - FrVectDecimateMax (vect, 2) will transform (0, 1, 2, 3, 4, 5) to (1, 3, 5)

FrVectExpand

- This function uncompressed a vector:
- Syntax: **void FrVectExpand (FrVect *vect)** where vect is the vector provided by the user
- In normal use the uncompressed is done by a FrameRead call or by the channel access.

FrVectExtend

- Syntax: **void FrVectExtend (FrVect *vect, int nTimes, FrVect *outVect, char *newName)**
- This function extend the data from the vector vect by duplicate nTimes each values and returns the extended vector (outVect).
- The result is put in the vector outVect. The size of outVect should be nTime larger than the size of vect. values.
- If outVect is NULL, the output vector is created and named "newName" or as the original vector is newName = NULL.

FrVectFillX

- This function add one value at the end of the vector. The vector size is automatically increased.
- Syntax:
  - **int FrVectFillC (FrVect *vect, char value);**
  - **int FrVectFillD (FrVect *vect, double value);**
  - **int FrVectFillF (FrVect *vect, float value);**
  - **int FrVectFillI (FrVect *vect, int value);**
  - **int FrVectFillS (FrVect *vect, short value);**
- This function returns 0 in case of success or a non zero value in case of problem (not enough memory).

FrVectFindQ

- For a vector of string, this function returns the index of the string which match the parameter "name" or a negative value if not found.
- Syntax: **int FrVectFindQ (FrVect *vect, char *name)**

FrVectGetIndex

- Syntax: **FRLONG FrVectGetIndex(FrVect *vect, double x)**
- This function returns the bin index for a a given 'x' abcisse.
- It returns
  - -1 if vect == NULL
  - -2 if vect->dx[0] == 0
  - -3 if x is lower than the vector start (vect->startX[0])
  - -4 is x is larger than the vector end.

FrVectGetTotSize

- This function returns the total memory used by a FrVect structure (in bytes)
- Syntax: **FRLONG FrVectGetSize (FrVect *vect)**

FrVectGetValueI

- This function returns the bin content for a vector at index 'i' or zero if 'i' is outside the vector boundaries.
- Syntax: **double FrVectGetValueI (FrVect *vect, FRULONG i)**
- This function replace the obsolete function FrVectGetV.

FrVectGetValueGPS

- This function returns the bin content for a vector asusming that the x axis is GPS time. The overall GPS value is also subtracted.
- Syntax: **double FrVectGetValueGPS (FrVect *vect, double gps)**

FrVectGetValueX

- This function returns the bin content for a vector at position 'x' or zero if 'x' is outside the vector boundaries. The vector->startX is subtracted before getting the value.
- Syntax: **double FrVectGetValueX (FrVect *vect, double x)**

FrVectHtoC

- This function convert an half complex vector to a regular vector.
- Syntax: **int FrVectHtoC (FrVect *vect)**
- This function returns 0 in case of success or a non zero value in case of problem (not enough memory).

FrVectIsValid

- This function check that all floating points contained in a vector are valid IEEE floating point numbers.
- Syntax: **itn FrVectIsValid (FrVect *vect)** where vect is the vector provided by the user
- This function returns 0 if the vector does not contains NaN or INF numbers or if the vector contain only integers. It returns a non zero value (the index of the first bad value +1) in the other cases.
- remark: -0.(minus zero) is a valid floating point for FrVectIsValid.

FrVectLoad

- This function read from file a vector which has been saved with the function FrVectSave. It returns NULL in case of error.
- Syntax: **FrVect* FrVectLoad(char *fileName)**

FrVectMean

- This function return the vector mean value or 0 in case of problem.
- Syntax: **double FrVectMean(FrVect *vect)**

FrVectMinMax

- This function computes the min and max value of the input vector vect. It returns 1 in case of failure or 0 in case of success.
- Syntax: **int FrVectMinMax(FrVect *vect, double *min, double *max)**

FrVectRMS

- This function return the vector RMS value or -1 in case of problem.
- Syntax: **double FrVectRMS(FrVect *vect)**

FrVectResample

- Syntax: **FrVect *FrVectResample(FrVect *vect, int nDataNew, FrVect *outVect, char* newName)**
- This function resample the dData data from the vector vect to nDataNew values. It returns the resampled vector. The result is put in the vector outVect that must have the right size (but could have diffrent type). If outVect is NULL, the output vector is created and named "newName" or as the original vector is newName = NULL.

FrVectSave

- This function write on file a vector. It returns 0 in case of success. The vector could be read back using the FrVectLoad function.
- Syntax: **int FrVectSave(FrVect *vect, char *fileName)**
- If fileName == NULL, the output file name is "vectorName_vectorGPStime.vect"

FrVectSetName

- This function set or reset the vector name
- Syntax: **void FrVectSetName(FrVect *vect, char *name)**

FrVectSetUnitX

- This function set or reset the vector first dimension name
- Syntax: **void FrVectSetUnitX(FrVect *vect, char *unitX)**

FrVectSetUnitY

- This function set or reset the vector 'Y' dimension name (bin content)
- Syntax: **void FrVectSetUnitY(FrVect *vect, char *name)**

FrVectToAudio

- This function convert an vector to an audio file (format ".au" with 16 bits dynamic). The ".au" extension is added to the output filename. The sound is automatically adjust to use the full dynamic. The parameter "option" is unseed for the time being.
- Syntax: **void FrVectToAudio(FrVect *vect, char *fileName, char *option)**

FrVectZoomIn

- Syntax: **int FrVectZoomIn(FrVect *vect, double start, double length)**
- This function change the vector boundaries. After this call, the vector start at "start" and sas a length "length". The new vector boundaries could only be within the original boundaries. The unit used is the vector x unit.
- This function works only for one dimension vector.
- It returns 0 in case of success or an error code.

FrVectZoomInI

- Syntax: **int FrVectZoomInI(FrVect *vect, int iFirst, int nBin)**
- Same as FrVectZoomIn except that the arguments are bin numbers.

FrVectZoomOut

- Syntax: **int FrVectZoomOut(FrVect *vect)**
- This function cancel the effect of any previous FrVectZoomIn call. It returns 0 in case of success or an error code.

Frame Library Error Handling

Several errors may occurs during the code execution. A typical one is the failure of the memory allocation. In this case, the functions return NULL. But when the error occurs, a default handler is called. This handler is the following:

```
/*---------------------------------------------- FrErrorDefHandler---*/
void FrErrorDefHandler(level,lastMessage)
int level;
char *lastMessage;
/*-------------------------------------------------------------------*/
/* default handler for the FrameLib error. */
/* input parameters: */
/* lastMessage: the string which contain the last generated message */
/* level: 2 = warning, */
/* 3 = fatal error: requested action could not be completed*/
/*-------------------------------------------------------------------*/
{
if(FrDebugLvl > 0)
{fprintf(FrFOut,"%s",lastMessage);
fprintf(stderr,"%s",lastMessage);}
return;}
```

If the debug level (dbglvl) set by the call to FrLibIni has a value > 0 this handler print debug information on stderr and on the debug output file. This handler could be changed by the user at the initialization by calling the function:
void FrErrorSetHandler (void (*handler) (int, char *));
At any time the user can get the history of the errors (recorded in one string) by using the function:

char *FrError (0," ","get history");

The Frame Library Installation

Copyright and Licensing Agreement:

This is a reprint of the copyright and licensing agrement of the Frame Library:

Installing the library and associated tools

A compressed (gzip) tar file is available at http://wwwlapp.in2p3.fr/virgo/FrameL.

To uncompress it you should type:

```
tar xvf vXrYY.tar.gz
```

Then use the simple scripts:

- One script (makegcc) available in the mgr directory build the library (including a shared library). It uses the GNU (gcc) compiler. The binary is placed in a directory named by the system type (like SunOS or OSF1) in order to work in a multi platform environment. Remark: On Mac OS X you need to use the makeMacOS script instead of the standard makegcc script.
- To compile the examples/test program, use the script maketest, after using the script makegcc.
- To compile on Alpha, using the alpha compiler, use the script makealpha.

If you run on a non standard system, you may want to change the low level I/O function calls. By default the Unix function call are used. To use the standard C FILE library you should compile the code using the option -DFRIOCFILE. To do more specific changes to the I/O you just need to change the FrIO.c file which group all those function call.

To use a user defined compression code (compression = 255) the functions FrVectUComp and FrVectUExpand should be provided by the user and the library should be compiled with the option -DFR_USER_COMPRESSION.

To use long long types you can compile the library with the -D FR_LONG_LONG option.

For any questions about this software, please contact Benoit Mours (mours@lapp.in2p3.fr) or Didier Verkindt (verkindt@lapp.in2p3.fr).

Install it with the autotools command:

Supposing your Fr packages is available in /somewere /Fr/v8r18 and that $UNAME point to the name of a system like Linux-x86_64-SL5:

1- cd /somewere /Fr/v8r18

2- autoreconf --install
3- ./configure --prefix=/somewere/Fr/v8r18/$UNAME
4- make (build results are stored on the src directory)
5- make install (All the build results will be copied into: /somewere /Fr/v8r18/$UNAME)
6- make distclean (to remove all intermediate build products)

Computer requirement for the Frame Library

The Frame software requests that the computer is at least a 32 bits computer. The Frame software writes the data in their original size and format. When reading the data on a different hardware, the frame library performed the byte swapping if needed (big-endian versus little-endian). It also expends or truncates the INT_8 variables if one machine has only 32 bits integer. The floating point variables are assumed to be always in IEEE format. The frame software (and installation scripts) has been tested on the following platforms:

- Alpha
- Linux
- Sun Solaris
- HP-UX
- Power PC under LynxOS
- Cygnus (GNU under Windows)

The Frame Library is ANSI-C code with POSIX compliance.

Test procedure

Once the library and the example have been installed, you can test it by running these examples. The prefix example has been replace by Fr. So to run the exampleFull, go in your machine sub directory and run FrFull. The first obvious thing to check is that the example run completely without crashing. Then some of the examples run in loop (like FrMark, FrMultiR, FrMultiW). They more designed to search for memory leak and it would be a good idea to check that the program size stay constant. Most of these tests created an frame file called test.dat. Each time this file is created, it is a good idea to run the utility FrDump with debug 1 and 2 and 3 on these file to check that the file content looks right. The suggested test sequence is:

- FrFull No arguments are needed. This test program produce a file called test.dat with different type of channel. Try "FrDump -i test.dat -d 3" to check if the file can be read.
- FrMark No arguments are needed. This program loop many time on the filecall test.dat. Check that the program size is stable (no memory leak)
- FrStat No arguments are needed. This test program produces a file (called test.dat) with static data. Check that you can read the file with FrDump.
- FrMark No arguments are needed. It will use the test.dat file produced with static data.
- FrOnline No arguments are needed. This program write frame in memory. It could be used to search for memory leaks
- FrMultiW No arguments are needed. This program creates 10 differents files which will be used by FrMultiR
- FrMultiR No arguments are needed. This program open and close many files. Usefull for memory leak.
- FrCompress No arguments are needed. This program test the compression algorithms. It should end with the message "Compression test OK"
- FrSpeed You should provide a file name and compress level. This program is used to estimate the reading/writeg speed.

The Matlab interface

Introduction

Matlab is a popular numeric computation and visualization Software. Since the Frame library is a plain C software, the connection between frame files and Matlab is easy to set. In the FrameLib package there is a matlab directory which contains:

- two MEX-file: frextract.c and frgetvect.c
- one script to compile the MEX-file: mymex
- three M-file to illustrate the use of the MEX-file:
    - exampleGetAdc.m Shows how to extract the Adc data from a frame file (using frextract), to plot a time series and it's FFT.
    - exampleGetVect.m Shows how to get a vector from a frame file with random access (using frgetvect), to plot a time series and it's FFT.
    - exampleAudio.m Shows how to produce and audio file from a frame file.

The purpose of this interface is to provide a direct path to extract data from a frame.

Matlab interface setting installation

The first operation to set the MEX-file is to compile it. This is done using the script mymex (just type ./mymex from the matlab directory).

Using frextract:

The frextract function could be called with the following arguments:

- Input arguments:
    1. file name(s). This could be a single file, a list of file separeted by space or a frame file list (ffl)

    1. ADC or PROCdata name (do not add extra space around the name)

    1. (optional) file index of the first frame used (default = first frame in the file)

    1. (optional) number of frame (default = 1 frames)
- Returned Matlab data:
    1. ADC or PROC data (time series)

    1. (optional) x axis values relative to the first data point. This is usual time but it is frequency in the case of a frequency serie.

    1. (optional) frequency values in the case of time series (double) (usefull for FFT's)

    1. (optional) GPS starting time (in second.nanosec)

    1. (optional) starting time as a string

    1. (optional) ADC or PROC comment as a string

    1. (optional) ADC or PROC unit as a string

    1. (optional) additional information: it is a 9 words vector which content the variables: crate #, channel #, nBits, bias, slope, sampleRate, timeOffset(S.N), fShift, overRange (or the equivalent for proc data). All these values are stored as double

Using frgetvect or frgetvectN

The frgetvect function performs a random access in the frame file using the table of content. The function frgetvectN is doing the same think but the returned vector is normalized in the case of an ADC channel (using the “slope” parameter). This function is much faster than frextract when working with large file. This function could be called with the following arguments:

- Input arguments:

  1) file name(s). This could be a single file, a list of file separeted by space or a frame file list (ffl)

  2) channel name (it could be an ADC, SIM or PROC channel)

  3) (optional) starting GPS time(default= first frame in the file)

  4) (optional) vector length in second (default = 1 second)

  5) (optional) debug level (default=0; -1=no output; > 1 more out.))

- Returned Matlab data:

  1) ADC or SIM or PROC data stored as double

  2) (optional) x axis values relative to the first data point. This is usual time but it could be frequency in the case of a frequency serie (double)

  3) (optional) frequency values in the case of time series (double)

  4) (optional) GPS starting time (in second, stored in double)

  5) (optional) starting time as a string

  6) (optional) vector unitX as a string

  7) (optional) vector unitY as a string

  All values are stored as double

Using other Frame tools with Matlab:

Do not forget also than you can run any Frame Utility program from Matlab by using the shell escape command ! For instance:

```
! FrDump -i ../data/test.dat
```

will call the program FrDump with the argument ran.dat.

Remark: if you call frgetvect and an exception is thrown (mexErrMsgIdAndTxt, etc), it is supposed to print its error message and dump you back to the commandline, but instead, it aborts, exiting matlab completely. To fix that, add "-fexceptions" to the build options in mgr/makegcc.

The ROOT interface

Introduction

ROOT is a powerful interactive environment developed at CERN (http://root.cern.ch). Among its various tools, It provide a very nice interactive C/C++ interpreter and detailed histograms capability. In the root directory of the Frame Library you will find a few scripts and macro to use the frames in the ROOT environment.

Frame library installation for ROOT

Assuming that you have already installed ROOT on your computer, you need first to build a special shared library. To do that, just adapt the build script to your system. You need at least to change the path to the ROOT directory and you may need to change some of the compilation flags... Once this is done, you need to update the PATH and LD_LIBRARY_PATH to include the FrameLib binary directory (named by your system). Then if you start root from the Fr root sub directory, it will execute the FrLogon.C which load everything you need.

Using the Frame Library in ROOT

Once ROOT is properly started, any Frame Library function is available as a ROOT command. Then 2 ROOT macro have been provided to build histograms out of the FrAdcData and the frame vector (FrVect). Just look at the three macro example to see what you can do. The FrVect vectors play a key role in these interactive analysis and more complex programs have been developed to provide direct interface to FFT and signal processing. See the Frv package (see http://wwwlapp.in2p3.fr/virgo/FrameL) and the Vega package (http://wwwlapp.in2p3.fr/virgo/vega). The test.dat file used by the exampleAscii.C and exampleAdc.C macros could be generated by running the FrFull example.

ROOT macros availabe:

- FrVP; This macros convert one or more vector to one histogram
  - TH1F* FrVP(FrVect *vect, char *draw = NULL, int color = 1, double xStart = 0., double xEnd = 0., double scale = 1.) This macros plot a single vector. Draw could take the value NULL to just build the histogram, "DRAW" to build and draw it or "SAME" to build and draw it on top of an existing histogram.
  - TH1F* FrVP(FrVect *vect1, FrVect *vect2 = NULL, double scale2 = 1., FrVect *vect3 = NULL, double scale3 = 1., FrVect *vect4 = NULL, double scale4 = 1.) This macros plot up to four vectors. The vectors 2 to 4 could be rescaled.
- FrAP: To plot and Adc channel:
  - TH1F* FrAP(FrAdcData *myadc, char *draw = NULL) This macro plot an ADC channel.
- FrCP: To plot a channel giving a file name and channel name(s):
  - TH1F* FrCP(char *fileName, double tStart = 0., double len = 2., char *channel1, char *channel2 = NULL, double scale2 = 1., char *channel3 = NULL, double scale3 = 1., char *channel4 = NULL, double scale4 = 1.)
  - TH1F* FrCP(char *fileName, char *channel1, char *channel2 = NULL, double scale2 = 1., char *channel3 = NULL, double scale3 = 1., char *channel4 = NULL, double scale4 = 1.)

The Octave interface

Introduction

GNU Octave www.octave.org is a high-level language, primarily intended for numerical computations. The interface frame to octave contains two routines [loadadc, loadproc] for loading ADC and PROC data from a given frame file into the Octave context. It has great similarities with the interface to Matlab previously described.

How it works?

Here is a description of what input variables should be provided to the loading interface and what output variables are available to the user:

LOADADC: Download an ADC signal in the Octave workspace from a given frame file.
Usage: [adc,fs,valid,t0,timegps,unit,slope,bias] = loadadc (fileName,[adcName[,nFrames[,first]]])
Input parameters:

- fileName: the name of the frame file
- adcName: [opt] the name of the ADC signal to be extr. [if missing: send a dump of fileName]
- nFrames: [opt] the number of frames to be extr. [if missing: send a dump of adcName frames]
- first: [opt] number of the first frame to be extr. [default=first frame avail.]

Output parameters:

- adc: the ADC signal
- fs: the sampling frequency
- valid: an index specifying whether the data are OK or not
- t0:   the GPS time associated to the first bin in the first extracted frame
- timegps: [string] same thing but human readable format
- unit: physical units of the signal
- slope: slope coef. used to calibrate the signal X
- bias: bias coef. used to calibrate the signal X

LOADPROC: Download an PROC signal in the Octave workspace from a given frame file.
Usage: [proc,fs,t0,timegps] = loadproc (fileName,[procName[,nFrames[,first]]])
Input parameters:

- fileName: the name of the frame file
- procName: [opt] the name of the PROC signal to be extr. [if missing: send a dump of fileName]
- nFrames: [opt] the number of frames to be extr. [if missing: send a dump of procName frames]
- first: [opt] number of the first frame to be extr. [default=first frame avail.]

Output parameters:

- proc: the PROC signal
- fs: the sampling frequency
- t0:   the GPS time associated to the first bin in the first extracted frame
- timegps: [string] same thing but human readable format

SAVEADC: Write an ADC signal from the Octave workspace to a given frame file.
Usage: status=saveadc(fileName,signalName,data[,fs,[t0]])
Input parameters:

- fileName: the name of the output frame file
- signalName: name of the ADC signal to be written
- data: input data (column vector of double)
- fs: sampling frequency
- t0: GPS time associated to the first bin of the first output frame

Output parameters:

- status: report about the writing operation.

SAVEPROC: Write an PROC signal from the Octave workspace to a given frame file.
Usage: status=saveproc(fileName,signalName,data[,fs,[t0]])
Input parameters:

- fileName: the name of the output frame file
- signalName: name of the PROC signal to be written
- data: input data (column vector of double)
- fs: sampling frequency
- t0: GPS time associated to the first bin of the first output frame

Output parameters:

- status: report about the writing operation.

Note that this description is also available online, by typing ``**help loadadc**'' or ``**help loadproc**'' at the octave prompt.

Test and getting started

A test script **plotframe.m i**s also part of the package. It uses the test framefile [test.dat] in the directory /data of the Frame Lib distribution. The script produces a plot of the first 1024 data points of the ADC signal 'Adc0', computes and plots its spectrum. This may be used as a start for learning how the interface works.

For any question about the Octave interface, please contact Eric Chassande-Mottin (ecm at obs-nice.fr)

The Python interface

This is the equivalent of the Matlab frgetvect interface.

To build the Python interface, look at the README file in the Python subdirectory.

Library Changes

From Version 2.37 to Version 3.10

- Several structures, structure's element names and vector types have changed. The new names follow the new Specification document. Some function names have been changed according these new names. The function names changes are:
    - - FrSmsxxx -> FrSerXXX
    - - FrRecXXX->FrProcXXX
    - - FrameDumpBuf -> FrameDumpToBuf
- In addition, the ASCII option for output file has been suppressed. The corresponding argument in the function FrOFileNew is now used for the frame data compression. Several static data with the same name but different time range can now be present in the same frame.

Files written with version 2.37 can be read by version 3.10.

From Version 3.10 to Version 3.20

- Add vector compression and fix bugs in FrVectCopy. Old files (from 2.3x) could still be read with the version 3.20.

From Version 3.20 to Version 3.30

- Change Utime to Gtime according to the specification LIGO-T970130-05.
- Add the variable Uleaps in the FrameH structure.
- Add the handling of FrSummary and FrTrigData structures.
- Fix bugs when writting compressed frames.

From Version 3.30 to Version 3.40

- Add the variable detector in the FrStatData structure.
- Change one parameter in the FrStatDataAdd function call (replace root by detector).
- Compute the number of bytes for the EndOfFile structure.
- Fix bugs in GPS time versus UTC time.
- Fix bugs in Floating point conversion with compression.
- Put I/O call in a separate file (there is one more file to compile so check your script).
- Improve the utility programs.

All files written with version 2.37 and higher can be read by version 3.40.

From Version 3.40 to Version 3.42

- Fix bug in FrFileIClose: the Static Data structures were not free.
- Improve the logic for static data update(data with timeEnd = 0 where not properly handled).

All files written with version 2.37 and higher can be read by version 3.42.

From Version 3.42 to Version 3.50

- Support multiple input file in FrFileINew.
- Remove some warning when reading old files
- Suppress the need to call FrLibIni
- Add the functions: FrameCopy, FrameHCopy, FrAdcDataCopy, FrameSelectAdc.
- Add a Matlab section.
- Update the examples

All files written with version 2.37 and higher can be read by version 3.50.

From Version 3.50 to Version 3.60 (March 22, 1998)

- Add the functions: FrDataFind.
- Fix the ADC sampleRate in exampleOnline.c and exampleMultiW.c
- Update the FrCopy FrDump and Frexpand utilities.
- Fix the bug in the directory creation in the makecc script.
- Fix bug in FrReadVQ (wrong malloc size).
- Add in situs framewrite

All files written with version 2.37 and higher can be read by version 3.60.

From Version 3.60 to Version 3.70 (Sep 16, 1998)

- Add the functions:
    - FrameWriteToBuf put a frame in one single buffer
    - FrameReadToBuf read a frame from a buffer
    - FrLibVersion return the FrameLib version number
- Fix bugs in:
    - FrameCopy : (uninitialized variable which in some case return the previous frame)

    - FrAdcDataNew : the 'adc' with floating point values where not properly created
    - FrVectWrite: write next vector if available
    - FrFile->Header fix the size from 32 to 40
    - FrReadLong and FrReadVL : logic for bytes swapping in the Pentium case.
    - FrameRead : in the case of reading unknown record
    - GPS time convention and associated print statement
    - FrSENew : the number of structure element of the dictionary was sometime wrong.
    - One variable name (localTime) in FrameH dictionary.
- Suppress the additional arguments in throvements in the case of Linux or DEC Alpha. See the FrIO.c file for details.
- Add a protection in FrVectNew if the function is called with strange arguments
- Add includes (unistd.h) in FrIO.h
- Change YES and NO global variables by FR_YES and FR_NO (internal variables)
- Add a parameters in the example FrDumpFile
- Print dictionary warning only if debugLevel > 1.
- Specify O_BINARY type for file open (needed for Windows NT)
- FrVectCompress: put a protection on gzipLevel (if set to 0 on Sun, the program crashed).
- Fix the format of a few print statements
- Update all the examples to use the latest functions and remove some unused variables

All files written with version 2.37 and higher can be read by version 3.70.

From Version 3.70 to Version 3.71 (October 6, 1998)

- Code cleaning in the FrCopy and FrDump utilities.
- Fix a bug in FrAdcDataNew when creating ADC?s with floating point values.
- Add a protection for bad arguments in FrAdcDataFind and FrSerDataFind in

All files written with version 2.37 and higher can be read by version 3.71.

From Version 3.71 to Version 3.72 (October 9, 1998)

- Use 315964811 to convert UTC to GPS time for old files instead of 315964810.

All files written with version 2.37 and higher can be read by version 3.72.

From Version 3.72 to Version 3.73 (November 11, 1998)

- FrVectCompress; add new compression scheme (zeros suppress for short) and try to optimize the code
- Add FrVectZComp and FrVectZExpand
- FrVectDiff: return the differentiate result (the original input is now unchanged)
- FrVectExpand: cleanup the logic
- FrVectWrite: To not copy the vector before compressing it
- FrVectDump: printf update

All files written with version 2.37 and higher can be read by version 3.73.

From Version 3.73 to Version 3.74 (April 2, 1999)

- FrAdcDataNew: Put adc name in the vector
- FrameDumpToBuf: Protect the case when the temporary file open failed.
- FrameWriteToBuf fix a bug to get the size including the EndOfFile record
- FrDicFree: Reset the file->SH pointer in order to be able to call directly FrDicFree
- FrFileINew and FrFileONew: Add protection for missing file name
- FrSimDataNew: Put data name in the vector and allow more type of storage
- FrSerDataGet fix bug to avoid name confusion with longer name
- FrProcDataNew: Change calling sequence to be compatible with Adc and Sim data.
- In FrVectCompress: compress only if stay in initial size and return the compress vector, the original is unchanged
- FrVectExpand: Trap error for unknown compression flag
- FrVectNew: do not fill the vector name
- FrVectNewTS: New function

- FrVectNew1D: New function
- FrVectWrite: do not compress if compress flag = -1
- FrIO.h: remove include to unistd.h
- exampleDumpFile.c: Add protection for missing file name
- frextarct.c: fix memory leaks, api description and printf statement for GPS starting time.
- Update the utilities FrDUmp.c FrCopy.c and FrExpand.c
- Clean up the FrameL.h to be compatible with ROOT/Vega
- Script: use only gcc and add the option -fPIC

All files written with version 2.37 and higher can be read by version 3.74.

From Version 3.73 to Version 3.75 (April 29, 1999)

- FrFileONew: restore the possibility to have fileName == NULL
- FrDicDump: fix a bug in the loop (this function is used only for debug)

All files written with version 2.37 and higher can be read by version 3.75.

From Version 3.75 to Version 3.80 (May 17, 1999)

- Change the FrIO.c code to support the standard C FILE.
- Add random frame access. This is an option which is under evaluation.

All files written with version 2.37 and higher can be read by version 3.80.

From Version 3.80 to Version 3.81 (June 4, 1999)

- Fix a bug in random frame access for multiple file open and close
- FrFileINew and FrFileONew: Removed protection for missing file name (added in 3.74)

All files written with version 2.37 and higher can be read by version 3.81.

From Version 3.81 to Version 3.82 (June 9, 1999)

- Add FrFileMarkFree to fix a memory leak when using the file marks.
- All files written with version 2.37 and higher can be read by version 3.82.

From Version 3.82 to Version 3.83 (June 15, 1999)

- Move the FrIO structure definition from FrIO.c to FrIO.h.

All files written with version 2.37 and higher can be read by version 3.83.

From Version 3.83 to Version 3.84 (August 23, 1999)

- FrVectWrite: Fix Memory leak when using compress.
- FrVectZComp: Add protection for buffer overflow

All files written with version 2.37 and higher can be read by version 3r84.

From Version 3.84 to Version 3.85 (September 11, 1999)

- FrTrigDataWrite and FrSummaryWrite: Fix bug to write link list
- struct FrVect: Add temporary variables (startX, frame, ?)
- Add define for GPS data and fix bug in time setting in the examples

All files written with version 2.37 and higher can be read by version 3r85.

From Version 3.85 to Version 4.00 (May 1, 2000)

- Change from frame format from version B to C. This generate many changes in the code. File written with FrameLib version 3.40 and above can still be read.
- Simplify the FrFileINew end FrFileONew API: the user do not need any more to pas a buffer. If he wants to directly in agreement with the new frame spec.
- Suppress the direct write in memory functionality
- Add miscellaneous feature, protections and fix various bugs:
- Add Gtime in FrVect
- Add the possibility to specify the history message produce at write time
- FrProcDataFind, SImDataFind: test if name == NULL
- FrAdcDataNew : remove the vect->name
- Fix bugs in compression flags logic.
- FrTrigDataWrite and FrSummaryWrite: Fix bug to write link list
- Reduce the use of long

All files written with version 3.40 and higher can be read with version 4.00

From Version 4.00 to Version 4.01 (May 15, 2000)

- Fix format of FrameL.h to be Root/Vega compatible.
- Add the functions: FrAdcDataNewF, FrAdcDataDecimate, FrameTagAdc, FrameUntagAdc, FrameMerge
- Remove the function FrameSelectAdc.
- Fix bug in FrameDumpToBuf.
- Change logic fr="#000000">From Version 4.02 to Version 4.03 (June 2, 2000)
  - Fix two bugs for FrStatData which showed up for static data copy and multiple write in file.

All files written with version 3.40 and higher can be read with version 4.02

From Version 4.03 to Version 4.10 (October 11, 2000)

- Random access: Add a random access for a vector over several frames: FrameGetV. Add the function FrProcDataReadT, FrSerDataReadT, FrSimDataReadT. Add random access by run/frame number: function FrTOCFrameFindN. Change FrameReadT to work if not TOC available, Fix several bugs in random access. Add FrFileITStart(FrFile *iFile) FrFileITEnd (FrFile *iFile) functions. Add read function to frame Header FrameHReadN and FrameHReadT.
- improve debug for frame reading
- Define types for structures and reorganize FrameL.h
- Tag: Add Tag/untag function for Proc,Sim,Trig and Summary data. Add the function FrameTag, AntiTag is done by a word starting by -. Improve the wild cards for tag. Change the internal logic for Tag/Untag/find using a new structure (FrBasic). Protect Tag function for tag == NULL
- Add a function FrameNew (frameH with time + detectProc structures)
- Add a function FrFileONewH and FrFileONewFdH to define the program name in the history record.
- Add direct vector access: FrAdcDataGetV, FrProcDataGetV, FrSimDataGetV,
- Add a function FrVectGetV to access and convert a vector element.
- Table: Add FrTableExpand and the function FrTableGetCol. Fix a memory leak in FrTableFree
- To speed up frame creation, read and write: do not set to zero the vectors element when creating a new one.
- Update the examples and utilities to use the random access tools.
- FrMerge now do a full frame merge and not only the raw data.
- Fix bug in static data write (if on static data was changed it was not written on tape)
- Fix bugs for the FR_VECT_STRING vectors in several places (FrVectFree and New, Read, Write, Copy)
- Fix a memory leak in FrVectCompData
- Fix a bug in FrTrigDataNew: Copy timeBefore/timeAfter.
- Fix a bug in FrAdcDataNew to store nBit as an unsigned int.
- Fix the matlab interface.
- Fix a mismatch with the Frame spec for long: Write long as 8 bytes even on a 4 bytes computer (fix bug in ReadLong)

All files written with version 3.40 and higher can be read with version 4.10

From Version 4.10 to Version 4.11 (October 23, 2000)

- Change the way to write NULL string: now we write at least the '\0' character.
- Output files (oFile) and Table Of Content: add on option to not write the TOC for the time series (ADC, sim, ...).
- FrCopy.c : Add program name in history record, Add the option noTOCts to not write the TOC for the time series.
- Fix a bug in the zero supress compression algorithm in the case of consecutive values = -32768. (Update the test program exampleCompress.c)
- Fix bug in FrVectDump for vector of string with the string = NULL:
- Remove extra printf in FrLibVersion

All files written with version 3.40 and higher can be read with version 4.11

From Version 4.11 to Version 4.20 (December 7, 2000)

- Add buffer in FrIO. This speed up the disk read by a factor 2 to 3.
- Add new functions for random access: FrTrigDataReadT, FrSimEventReadT, FrameReadTAdc and check that all random access function skip FrSH and FrSE records.
- Add a new function FrameReadRecycle to speed up read of frame with lot of summary information.
- Update the following test program to test these new functions: exampleMark.c, exampleDumpFile.c
- Upgrade FrCopy to use the new random access tools and to allow it to uncompress already compressed files.
- Add decimation for float in FrAdcDataDecimate .
- Upgrade FrAdcDataReadT to work with wild card in the Adc name.
- Reorganize FrVectCompData to fix some bugs in the flags selection. (compression 5 was crashing for 4 bytes integers, could not uncompress a file already compressed).
- Add a test version of lossy compression for float.
- fix memory leak in FrSimEventXXX functions.
- fix a bug in FrSpeed in the speed computation.
- fix a bug in FrProcDataGetV.
- Add an Fr prefix to the zlib includes
- Fix some printf.

All files written with version 3.40 and higher can be read with version 4.20

From Version 4.20 to Version 4.21 (December 8, 2000)

- Fix a bug in FrAdcDataReadT (the wrong adc was read)
- Did some changes in the float to in compression.

All files written with version 3.40 and higher can be read with version 4.21

From Version 4.21 to Version 4.22 (December 12, 2000)

- Fix a bug in FrTOCFrameFindT which prevent to extract frame for files with only one frame.
- Add the convention for random access that if the requested time is 0, you get the first frame in the file.

All files written with version 3.40 and higher can be read with version 4.22

From Version 4.22 to Version 4.23 (Jan 16, 2001)

- Fix bug in FrFileIGetV to get the proper frame is TStart = 0.
- Fix a printf bug in FrLibVersion.
- Fix memory leak when using taewrite of the pointer relocation logic to speed up read/write for frames with many channels.
- Sort by alphabetic order the channels in the table of content.
- Fix a memory leak which was observed when using random access with table of content on many files.
- Add scripts for autoconf GNU tools (thanks to Duncan Brown).

All files written with version 3.40 and higher can be read with version 4.30

From Version 4.30 to Version 4.31 (Feb. 27, 2001)

- Fix bugs in the handling of static data. The static data were not properly read/write since version 4.30.
- Update the example exampleSpeed.c to compute more things.
- Update the root macro to add more option for the plots of one vector

All files written with version 3.40 and higher can be read with version 4.31

From Version 4.31 to Version 4.40 (July 11, 2001)

Thanks to Isidoro Ferrante, Martin Hewitson, Julien Ramonet, Andrei Sazonov, Peter Shawhan, Didier Verkindt and Andrea Vicere for finding and reporting bugs.

- Major rewrite of random access functions (most of them). Some new random access function added. Now random access is supported when using multiple files with usually wild card as well. The random access work properly now in case of missing frames. The definition of the search time window for FrTrigDataReadT as been update to be consistent with the other random access function. Fix a weakness in the random access. The GPS time is defined as double and sometime if we use the integer starting time converted to double we got the previous frame due to round off error.

- Add Frame File List option for the FrFileIOpen for efficient random access in multiple files.
- Add one more parameter to the functions FrTrigDataFind, FrSerDataFind, FrSimEventFind to be able to access structure with the same name in the same frame. Warning: code using these functions need to add this extra parameter which should be set to NULL to work as previously.
- Change the function FrAdcDataFree to free now the full linked list
- Improving some dump, debug and printout statements, especially in FrAdcDataDump, FrVectDump (with large debug level, you could have a full vector dump), and FrVectStat (add management of doubles) and a wrong comment about compression in FrCopy. Improve the table of content dump: add the number of frames containing the channel and allow the use of tag to control the channel list. To do that, the function to tag data have been reorganized.
- Add a new FrVect type: FR_VECT_H8 and FR_VECT_H16 for FFTW half complex vectors
- Add the type FR_VECT_8C as specify in the frame spec. The miss typed FR_VECT_C8 type is still valid.
- Fix a bug in FrTOCtsMark: The Adc group ID and channel ID were swapped. This means that all files written with previous version of the library have this information swap in the table of content. If the file is copy, the table of content is rebuild with the right information.
- In case of unknown compression flag, free the working space in FrVectExpand to avoid memory leak.
- Add the frame reshaping function (FrameReshapeXXX) to change the frame length.
- Add more feature to FrCopy like decimation and change of the frame length.
- Update FrAdcDataDecimate and FrVectIsValid to handle double precision floating points and to fix a bug in the update of nx[0].
- Add a test when writing data to check for duplicate channels.
- Fix a bug in FrFileORealloc (the memory allocation failure was not properly trapped).
- Fix bug in TOC writing on PC. The TOC of files writen on PC with version < v4r40 are not usable. To regenerate the TOC, just copy the file with the latest FrCopy utility.
- Add zero suppress compression options for 4 bytes integers and floating points numbers
- Add the prefix "Fr" to all gzip functions to avoid name conflicts.
- Add the function FrVectDecimate and FrStrUTC
- Add the computation of checksum when writing file.
- Update some example programs.
- Update the compilation script to work on cygnus also. The makecc script has been renamed makegcc to be more consistent and do not compile anymore the example. To compile them use the script maketest.

All files written with version 3.40 and higher can be read with version 4.40

From Version 4.40 to Version 4.41 (July 31, 2001)

Thanks to Sam Finn, Frederique Marion and Peter Shawhan for finding and reporting problems and bugs.

- Fix the checksum computation on SGI (FrChkSum function).
- Fix a bug in FrCopy when using input list from standard input (STDINLIST option)
- Fix FrADCDataReadT, FrProcDataReadT and FrSimDataReadT to avoid memory leak if there is no frame for the requested time.
- Fix bug in FrameReshapeNew to properly handle the case with non zero position.
- FrFileIGetVType: increase round off margin to deal with vector with a slight offset compared to the frame start time.
- Minor update of the TOC dump
- Initialize to zero any new vector, except the one created by FrAdcDataNew. (This initialization was suppress at version v4r10).
- Add the function FrVectMinMax.

All files written with version 3.40 and higher can be read with version 4.41

From Version 4.41 to Version 4.42 (Sept 19, 2001)

- Remove the to the FrCHkSum function (declare the first argument of the FrChkSum function as char * instead of signed char *) to be able to work with the current version of root in order to bypass a bug in CINT.
- Change FrVectIsValid to flag sub normal floating point numbers as invalid floating points.

All files written with version 3.40 and higher can be read with version 4.42

From Version 4.42 to Version 4.43 (Oct 15, 2001)

- Fix the FrVectIsValid function to not mark zero floating point value as invalid floting point.
- Fix a Bug in FrTOCSetPos to be able to do random access on file larger than 2 GBytes.

All files written with version 3.40 and higher can be read with version 4.43

From Version 4.43 to Version 4.44 (Nov 14, 2001)

- Fix a Bug in the zero suppress compression algorithm for floating point (compression flag = 8).

All files written with version 3.40 and higher can be read with version 4.44

From Version 4.44 to Version 4.50 (March 13, 2002)

Thanks to Damir Buskulic, Eric Chassande-Mottin, Ed Daw, Martin Hewitson, Frederique Marion Alain Masserot, Peter Shawhan and Didier Verkindt for suggestions, finding and reporting problems and bugs.

- FrVectIsValid: rewrite the code to fix a bug when checking double and to not trig on -0.
- FrVectDecimate: update the value of out->size = in->size/nGroup.
- FrVectStat: minor changes in the output format (replace 4 by %5).
- FrVectDump: for complex numbers: add protection for invalid complex number and provide full dump. Dump also GPS time if available.
- FrMerge: if no info timing for one frame, use the values from the other one.
- Fix a memory leak when using random access on multiples files (avoid the double reading of FrSH structures).
- FrTagMatch: Fix a bug in the case of 'multiples' antitag. For instance if the tag was "-adc1 -adc2", only "-adc1" was taken into account.
- Add a protection when performing random acces on file with channels missing for a few frames.
- FrChkSum: Select char or signed char according to the compiler definition.
- FrCopy: Uncompress data if the option decimate is used. Check that the option -a is requested after the input file declaration.
- FrVectRead/Write fix a bug to fullfil the frame spec in the case of uncompressed vector on littleendian machine. Warning: due to this change, uncompress data produced on littleendian machines will be unreadable with previous library version.
- FrFileINext and FrDum: add a protection on invalid file when reading multiples files.
- Add function FrMsgDump, FrameReadTSer, FrameReadTSim, FrameReadTProc, FrameReadTChnl.
- Add an Octavia directory (code from Eric Chassande-Mottin).
- Fix names in the gnu install scripts.

All files written with version 3.40 and higher can be read with version 4.50

From Version 4.50 to Version 4.51 (March 18, 2002)

Thanks to Andrei Sazonov, Peter Shawhan and Didier Verkindt for suggestions, finding and reporting problems and bugs.

- FrVectWrite fix a bug to fullfil the frame spec in the case of uncompressed vector on littleendian machine. Warning: due to this change, uncompress data produced on littleendian machines will be unreadable with previous library version.
- Add the function FrFileIChannelList.
- For FFL, paths to files from the list are now interpreted as relative to the path of the list file, not relative to the current directory.

From Version 4.51 to Version 4.52 (March 20, 2002)

- FrVectDecimate: Free the unused space at the end
- Fix a format in FrFileIChannelList.

All files written with version 3.40 and higher can be read with version 4.52

From Version 4.52 to Version 4.53 (May 2, 2002)

Thanks to Damir Buskulic, Jean-Marie Teuler and Didier Verkindt for suggestions, finding and reporting problems and bugs.

- FrReshapeAdd: Do not add images, just move them
- FrVectIsValid: Fix a bug introduced since v4r50 (in case of error, it returns now the index of the first invalid vector element)
- FrIO.h: add a protection for multiple includes.
- Add channel list (FrCList) functions for AdcData and SerData
- FrAdcDataDecimate: use a double for local sum (to avoid problem foir large float values)
- FrCopy: Use FrameReadTChannel instead of FrameReadTAdc
- Add the functions FrTOCFrameFindNext and FrTOCFrameFindNextT
- Fix a memory leak in FrExtract

All files written with version 3.40 and higher can be read with version 4.53

From Version 4.53 to Version 5.00 (Augst 12, 2002)

Many changes have been made, most of them to support the frame format version 5.

A few function calls have been changed, basically to add extra parameters.

- FrSerDataNew: add the sampleRate parameter.
- Rename FrTOCFrameFindNextT to FrFileITNextFrame.
- Rename FrTrigData to FrEvent.
- Change the calling sequence of FrEventNew and FrSimEventNew (GtimeN,S have been replace by a double and event parameters could be directly added).
- FrEventReadT and FrSimEventReadT : add two extra parameters to select event on the amplitude if needed.
- Change the return argument for FrHistoryFree and FrStatDataFree to be like the other Free function.
- FrFileIGetV return now a vector with the requested boundaries.

New functions have been added, especially to provide Table Of Content data access:

- FrFileIGetXXXNames and GetXXXInfo provide information from the TOC.
- FrFiltITFirstEvt and FrFileITLastEvt provide the time of the first and last event in the file.
- FrMsgFind find a FrMsg in a frame.
- FrVectNewL let you create vector with a long long argument type for nData.
- FrameGetLocalTime return the localTime offset for on Adc channel.
- FrProcDataAddHistory let you add history to an FrProcData channel.
- FrVectZoomIn and FrVectZoomOut to change a vector boundaries

New feature have been added:

- When doing random access check that we read the right vector.
- The name of the history record is now the frame name.
- Split the time boundary for frame and for events in the Frame File List.
- Add file statistic in FrDump (the debug flag level have slightly changed).
- Change FrIO.h for HPSS include when needed.
- FrCopy do not change the compression type by default now.
- FrAdcDataDecimate: realloc space at the end to free unused space.
- FrFileIOpen: add a protection for multiple calls on a file already closed
- FrFileIChannelList: print also the min and max value for each channel.
- upgrade FrCheck to check sequentiel reads and check individual frame checksums.

Several bugs have been fixed:

- The type and users vector in FrameH are now read, written to file and free when they are used.
- Fix a bug in FrVectDecimate (wrong realloc).
- In case of random access with some channels that are not in all the frames, the following channels were not read.
- FrPutLong was not properly working on some computers.
- FrFileINew: fix a bug when the ffl file was not present.
- FrCopy: when the requested starting time was after the end on file, all the file were copied.
- Complete the zlib prefix list in Frzconf.h.
- Add a protection when doing frameReshape on compressed frames.

Your compiler should flag all the change you need to do on your software to convert it to version 5. The most likely changes will to replace FrTrigData by FrEvent, update the new and find functions, remove the use of localTime from the frame header and fix some printf when Files written with version 3.40 and higher can be read with version 5.00

From Version 5.00 to Version 6.00 (Augst 14, 2002)

Since some LIGO software (the version of FrameCPpec to version 6. Files written with FrameLib version 3.40 and higher can be read with version 5.00. The 'version 5' frames produced by FrameCPP could also be read by this new version of FrameLib, but only in a sequential mode (no random access using the TOC). The changes made between v5.00 and v6.00 are due to this modification.

From Version 6.00 to Version 6.01 (September 9, 2002)

Thanks to Damir Buskulic, Eric Chassande-Mottin, and Didier Verkindt for suggestions, finding and reporting problems and bugs.

- Fix a bug which prevent to performed random access read with FrEvent for version4 frames.
- FrFileIChannelList: fix a bug in the min/max values.
- Add functions : FrameRemoveUntaggedData and FrEventAddParam
- Describes Virgo dataValid values in FrAdcDataDump
- FrameReshape: protect it when using tagged frames and set dataValid flag for ADC if part of the ADC is missing
- Update OCTAVE interface:

  Bug fix in loadadc/loadproc
  New functions saveadc/saveproc
  Update Makefile

Most files written with version 3.40 and higher can be read with version 6.01

From Version 6.01 to Version 6.02 (October 22, 2002)

Thanks to Damir Buskulic, Eric Chassande-Mottin, Frederique Marion and Soumya Mohanty for suggestions, finding and reporting problems and bugs.

- Add a protection agains NULL vector in FrVectCompress and FrVectExpandF
- Change FrameH->run from unsigned int to signed int.
- FrVectCompData: fix a bug for gzip compression: the output spce was slightly to short and in a very few cases, this was creating problems.
- Fix dictionnary for FrProcData->history, and for arrays in FrTOC (INT_4U * changed to *INT_4U for instance).
- Commented out the test compression method 7 (not part of the specevent when performing a random access using FrEventReadT (and similar functions).
- Update the Matlab interface frextract to work also with FrProcData
- Update the data test files (and the exampleFull.c program to produce more channels)
- Remove the FrProcData->sampleRate variable.

Most files written with version 3.40 and higher can be read with version 6.02

From Version 6.02 to Version 6.03 (December 20, 2002)

Thanks to Damir Buskulic, Frederique Marion, Bernd Machenschalk, Jean-Marie Teuler, Gabriele Vedovato and Didier Verkindt for suggestions, finding and reporting problems and bugs.

- Add paramaters handling functions for FrProcData (FrProcDataAddParam,...)
- Fix a bug in FrProcDataRead: the GTimeN field was not properly used when computing the GPS time.
- FrVectMinMax: add a protection against invalid floating point numbers
- Read unsigned short instead of signed short for the length of 'char'. It provides the full length for 'char' and prevents segmentation fault when reading unvalid frames. This is transparent for regular frames.

- Add various consitency checks to protect against corrupted files when reading them. This allows to recovert some frame file version 4 with when doing random access on some missing channel.
- Fix a bug in FrFileIGetAdcNames to prevent to read the TOC for most of the files and when reading data randomly
- FrCopy: fix a bug in a malloc which was generating crashes on Linux
- Minor change in FrVectDump to avoid the access of missing vector element for short vectors and to print the position of the maximum value.
- Include the licensing agreement.

Most files written with version 3.40 and higher can be read with version 6.03

From Version 6.03 to Version 6.04 (February 11, 2003)

Thanks to Elena Cuoco, Bernd Machenschalk, Frederique Marion, Szabolcs Marka, Ed Maros, Michele Punturo, Peter Shawhan, Gabriele Vedovato, Didier Verkindt and John Whelan for suggestions, finding and reporting problems and bugs.

- Fix a bug introduced in v6r03 which prevent to access FrProcData in random access on some files.
- Fix a bug in FrVectZCompI (failing to compress a vector if the 4 bytes vector includes the value -1)
- Fix a bug in FrVectCompData: With the compression type 8, If a vector could not be differentiate, the vector type was corrupted.
- Fix a bug in the checksum computation on Sun since v6r00 (in FrEndOfFileWrite).
- In FrChannelList: Do not compute min/max. This fix a memory leak.
- FrVectDump: improve the full debug for a few rare cases (like 8U)
- Minor debug messages fixes: (FrEndOfFileRead: do not print checksum if it is not computed, FrFileIOpen,...)
- Protect FrIO.h against multiple includes
- Add setting function for FrAdcData (FrAdcDataSetAux, ...DataValid, ...SetFShift, ...SetTOffset)
- Add the function FrStatDataReadT (for random access) and FrStatDataDump.
- Upgrade FrVectDecimate to provide a true decimation (without averaging) if nGroup < 0.
- Upgrade the event handling:
  - FrEventNew and FrSimEventNew: add a warning for the multiple paramters type (they should be double).
  - Add the FrameAddEvent, FrEventCopy, FrEventReadData and FrEventReadTF functions and the equivalent function for FrSimEvent.

Most files written with version 3.40 and higher can be read with version 6.04

From Version 6.04 to Version 6.05 (February 25, 2003)

Thanks to Bernd Machenschalk, Frederique Marion and Didier Verkindt for suggestions, finding and reporting problems and bugs.

- Fix a bug in FrameAddEvent (when adding a linked list of events, only the first one was added, the other were lost).
- Fix a bug in FrChannelList: (in case of FrProcData, their was a segmentation fault).
- Fix a bug in FrFileIGetV (for too large vectors, a failed malloc was not protected and was producing a segmentation fault)

Most files written with version 3.40 and higher can be read with version 6.05

From Version 6.05 to Version 6.06 (March 17, 2003)

Thanks to Franco Carbognani, Frederique Marion, Ed Maros and Gabriele Vedovato for suggestions, finding and reporting problems and bugs.

- Change FrameAddEvent to add the events in direct order instead of reverse order in the linked list.
- Make shure that bytes 18 to 25 of the file header are properly filed on all machines.
- Fix a bug in FrDump (it was crashing on some platforms like Linux when more that one file was given as input argument).
- Small changes the build and environment scripts for root

Most files written with version 3.40 and higher can be read with version 6.06

From Version 6.06 to Version 6.07 (May 24, 2003)

Thanks to Bruce Allen, Fabrice Beauville, Frederique Marion, Gabriele Vedovato and Didier Verkindt for suggestions, finding and reporting problems and bugs.

- FrEventAdd: Fix a Bug in (crash when adding the first event).
- FrameMerge: fix memory leak if the frame have the same detector structure.
- FrAdcDataNewF: set the vector vertical axis to the units parameter.
- Fix a bug in the wildcards: "Adc*" was not selecting the name "Adc". (FrTagMatch1 function).
- Improve the file error messages (open/read/write) using the errno information.
- FrDump, FrCopy and FrCHeck: send error messages to stderr instead of stdout.
- FrCheck: change the convention of the returned value of FrCheck. The old returned value (number of frame) was usually wrong because of the limitation of the exit function to 255.
- Add the functions: FrLibGetVersion, FrFileOPutV, FrVectReadZoom.
- Update the exampleEvent.c file for more checks.

Most files written with version 3.40 and higher can be read with version 6.07

From Version 6.07 to Version 6.08 (August 28, 2003)

Thanks to Bruce, Allen, Marie-Anne Bizouard, Jolien Creighton, Vladimir Dergachev, Sam Finn, Gianluca Maria Guidi, Alain Masserot, Andre Merzky, Peter Shawhan, Andrea Vicere for suggestions, finding and reporting problems and bugs.

- Bug fixes:
    - FrEventReadData: fix a bug (the data were not found).
    - FrameDumpToBuf: Full rewrite to fix a bug: the internal buffer size was not properly defined and the function did not work for large dump.
    - Fix a bug in FrFileHeader: The header was not correct (bit 18 to 25) for file produced on Alpha computer with version v6r06 and v6r07. This bug had no effect on data themself, but checksum error have been improperly reported and the TOC may be corrupted for thoses files.
    - FrAdcDataDecimate: Fix a bug in memory reallocation at the end of the function.
    - FrTOCRead: add protection to trap more malloc failures.
    - FrFileINewFd: add the cleanup of allocated structure in case of error during file openning.
    - Fix a bug in the latitude, longitude and arm azimuth translation between frame version 4 to frame version 6.
    - FrRead: change an unsigned long to a long to be able to read read errors.
    - FrSimEventReadData: improve it to avoid confusion when many simulated event are generated.

- Add the functions:
  - FrVectReshapeAdd, FrVectReshapeNew, FrVectReshapeEnd
  - FrFileONewM
- FrSimEventAdd: add the event at the end of the linked list instead of the beginning.
- FrVectDump: dump hexadecimal values in the case of invalid floating points numbers
- FrFileIGetVxxx: add error messages in case of read error
- FrDump: add the option "-c -1" to leave the data compressed.
- FrCheck: add the "-c" option.
- FrCopy: add the "-max" option to split output files and handle large data sets
- Add the makeMacOS script in the mgr directory. This script is needed to build the shared library on MacOS.
- Matlab macros: add an extra parameters to control de debug level and fix the string returning the time for time series.
- Copy the data test file.
- cmt/requirements file: change "UNAME" to "Fr_tag"
- Add the Memwatch include (protected by #ifdef MEMWATCH)
- Introduce the possibility to use FFTW malloc and free functions. Use the mgr/makefft script to build the corresponding library.

Most files written with version 3.40 and higher can be read with version 6.08

From Version 6.08 to Version 6.09 (October 01, 2003)

Thanks to Marie-Anne Bizouard, Frederique Marion and Eric Chassande-Mottin for suggestions, finding and reporting problems and bugs.

- Bug fixes:
- The wild card selection "tag*" was not selecting the name "tag".
- FrCListBldAdc, FrCListBldSer: a bug which was producing crash on Linux when channel names had too different length.
- FrFileIGetFrameInfo: there was a segmentation fault if the information requested was not starting at the file beginning.
- Update octave/saveadc and saveproc to be compatible with octave-2.1.50.

Most files written with version 3.40 and higher can be read with version 6.09

From Version 6.09 to Version 6.10 (December 11, 2003)

Thanks to Frederique Marion, Alain Masserot, Andrea Vicere, Didier Verkindt for suggestions, finding and reporting problems and bugs.

  - Bug fixes:
  - FrTagMatch: The selection of a tag and antitag (like "Pr*" and "-Pr_B5*") was order dependant. This is no more the case.
  - FrVectMergeT: this was producing crash when doing an FrDump with multiples files with events.
  - FrTagNew: copy the tag string in the internal structure.
  - FrVectConcat: the frame reshaping in FrCopy was crashing if the frames were longer than one second.
  - Fix a memory leak in TOC reading when a random access was perfomered after a sequential read.
  - FrCopy: Fix a bug when using the reshaping option with frame havin a lenght different than one second.
  - FrDump: Fix error message for missing frame with debug level 1. The quoted time was the end of the missing segment instead of the begining.
  - Do not open the frame file when doing FrFileINew. The file open is only performed in the FrameRead. This was a problem when opening an ffl having the first file of the list incorrect. (function touched: FrameRead, FrFileINew, FrFileINewFd)

- FrFileIChannelList: rename it to FrFileIGetChannelList and the second (up to now) unsed parameters is now used to define the GPS time to get the list (usefull for ffl).
- FrAdcDataNew: preset the "slope" parameter to 1. instead of 0.
- Change the format of timeBefore and timeAfter for the FrEvent dump functions.
- Move the FrVect definition from FrameL.h to FrVect.h
- FrDump: Change de default debug level from 2 to 1
- FrEventReadData and FrSimEventReadData: read the associated vectors ONLY for the first event if it is a linked list.
- Add the functions:
- FrFileISetTime to set the file position
- FrVectDecimateMin: to decimate a vector keeping the minimal values
- FrVectDecimateMax : to decimate a vector keeping the maximal values
- FrVectSetName: to set or reset the vector name
- FrVectSetUnitX0: to set or reset the first dimension name
- FrVectSetUnitX1: to set or reset the second dimension name
- FrVectSetUnitY: to set or rest the 'y' dimension (bin content)
- FrVectGetSize/FrAdcDataGetSize/FrameGetAdcSize: to get the memory used by a vector/ADC/frame.
- FrVectSave/FrVectLoad function to save on file and reload a vector.
- FrVectGetIndex, FrVectGetValueI, FrVectGetValueX

Most files written with version 3.40 and higher can be read with version 6.10

From Version 6.10 to Version 6.11 (January 20, 2004)

Thanks to Frederique Marion, Alain Masserot, Didier Verkindt for suggestions, finding and reporting problems and bugs.

- Bug fixes:
- FrameReadFromBuf: fix a bug introdcued in v6r10 (the function was returning NULL in all cases)
- Fix the name of the FrVectSetUnitX function (was improperlly named FrVectSetUnitX0 in v6r10)
- Add FrVectSave in FrameL.h
- Add the functions:
- FrEventAddVec: to add a vector to an event
- FrEventAddVecF: to add a vector (cast to floats) to an event
- FrVectCopyPartI, FrVectCopyPartX,
- FrVectCopyTo, FrVectCopyToD, FrVectCopyToF, FrVectCopyToI, FrVectCopyToS
- FrVectZoomInI: to zoom a vector using integer arguments (bin number instead of x value)

Most files written with version 3.40 and higher can be read with version 6.11

From Version 6.11 to Version 6.12 (March 03, 2004)

Thanks to Duncan Brown, Damir Buskulic, Frederique Marion, Peter Shawhan, Didier Verkindt for suggestions, finding and reporting problems and bugs.

- Bug fixes:
- FrFileBreakName: reduce the size allocated to each file when using an ffl. (it was 2kBytes per file instead of 100 bytes/files).
- FrEventAddVect and FrEventAddVectF: multiples bugs; the wrong vector was used and startX was corrupted.
- Matlab frgetvect and frextract: add support for complex numbers
- Add the functions:
- FrEventFindVect, FrEventGetVectD, FrEventGetVectF
- FrSimEventFindVect, FrSimEventGetVectD, FrSimEventGetVectF
- FrSimEventAddVect, FrSimEventAddVectF
- Add an option to convert hermitian vector to regular vector at write time.

- Add a fix to return the localTime for Virgo channel with wrong names
- Change the name of a few function from xxxGetV to xxxFindV to show that they return only a pointer to a vector and not a fresh copy that the user has to remove after using it. The olde names will remain available in the library (not in the .h file) for a few months for backward compatibility:
- FrameGetV becomes FrameFindVect
- FrAdcDataGetV becomes FrameFindAdcVect
- FrProcDataGetV becomes FrameFindProcVect
- FrSummaryGetV becomes FrameFindSumVect
- FrStatDataGetV becomes FrameFindStatVect
- FrSimDataGetV becomes FrameFindSimVect

Most files written with version 3.40 and higher can be read with version 6.12

From Version 6.12 to Version 6.13 (May 23, 2004)

Thanks to Stuart Anderson, Duncan Brown, Tom Kobialka, Alain Masserot, Joe Romano, Peter Shawhan, Patrick Sutton, Daisuke Tatsumi, Didier Verkindt for suggestions, finding and reporting problems and bugs.

- Bug fixes:
- FrFileITNextFrame (the function was not jumping over missing frames between files)
- FrDump: Add one more space between vector values (float and double) when doing a full ASCII dump.
- Update the maketest script which was not working on RedHat 9 machine.
- FrFileINewM: add protection when the output directory is not available.
- FrVectMergeT: WHen frames were not in increasing GPS time, the function was doing to an infinit loop (This was observer in FrDump for instance)

- In the case of single file (not ffl) FrFileINew do again the file open (like up to v6r09).
- Add the tag/untag feature for StaticData
- Update the Matlab 'help'
- Add the functions:
    - FrFileIGetVect, FrFileIGetVectF, FrFileIGetVectFN, FrFileIGetVectD, FrFileIGetVectDN to retrive from file a vector already converted to Float or Double and Normalized (in the case of ADC's).
    - FrVectSetMissingValues to set the vector values in the case of missing frames for vectors returned by FrFileIGetVect
    - FrVectExtend and FrVectResample to change the vector length.

Most files written with version 3.40 and higher can be read with version 6.13

From Version 6.13 to Version 6.14 (August 5, 2004)

Thanks to Stuart Anderson, Bruce Allen, Fabrice Beauville, Thierry Bouedo, Kipp Cannon, Martin Hewitson, Bernd Machenschalk, Sam Finn for suggestions, finding and reporting problems and bugs.

- Bug fixes in:
- FrVectExtend: propagate time GPS time information (this was not done).
- FrVectConcat: works now with frames of different length. Segement fault were observed when using FrFileIGetVXXX functions with frames of different length in the same files or FFL.
- FrDetectorMerge: to support merge of frames with static data attached to the same detector structure in both frames.
- FrVectReadNext: in random access, not all vectors attached to one FrAdc, FrProcData, FrSimData, FrEvent, FrSimEventADC structures where properly read. Two read were usually needed.
- FrEventReadData and FrSimEventReadData: work now with linked list of event as input.
- Add the functions FrFileIGetSerNames. Now FrDump -d 1 returns also the list of serData

- Change FrIOOpenR to be able to read file from the standard input (if the file name is "STDIN"). When input file are not open with "rb" instead of just "r" to avoid some problems on Windows.
- FrDump -d 2 print now the maximum frame size in a file.
- Add the functions:
    - FrProcDataAttachVect, FrProcDataFindVect, FrameAddStatDat, FrameAddStatData, FrameAddStatVector, FrStatDataAddVect, FrameGetStatVect, FrameFindStatData, FrDetectorFindStatData, FrDetectorAddStatData, FrameFindDetector
- Add the FrFilter.c anf .h files and update the compilation scripts.
- Update the matlab frextract.c frgetvect.c functions.
- Remove the fcntl.h include from all files and insert it only in FrIO.c (portability on Debian)

Most files written with version 3.40 and higher can be read with version 6.14

From Version 6.14 to Version 6.15 (September 10, 2004)

Thanks to Kipp Cannon, Eric Chassande-Mottin, Sam Finn for suggestions, finding and reporting problems and bugs.

- Bug fixes in:
- FrVectConcat: change the rounding procedure when computing the number of bin for a vector. The last sampled were missing since v6r13 when using vector random access (FrFileIGetVXXX or frgetvect functions).
- Fix the octave interfaces.

Most files written with version 3.40 and higher can be read with version 6.15

From Version 6.15 to Version 6.16 (February 14, 2005)

Thanks to Stuart Anderson, Fabrice Beauville, Thierry Bouedo, Franco Carbognani, Shourov Chatterji, Gianluca Guidi, Alain Masserot , Peter Shawhan, Gabriele Vedovato, Andrea Viceré, John Whelan for suggestions, finding and reporting problems and bugs.

- Bug fixes in:
- FrameRemoveUntaggedData: There was a memory leak if all channels where removed.
- Add a protection in FrEndOfFrameRead. There was a segmentation fault doing a full frame read after vector random access.
- FrVectConcat: Remove a meaningless error message when accessing vector ending before the end of a frame.
- Remove the "cd -" at the end of the compilation script that were not properly handled on Solaris.
- Fix a bug in FrFileIGetV that prevent to read FrProcData of type complex. This was affecting the matlab frgetvect function.
- New features:
- FrDump: gives now the overall compression factor when used with the option -c -1
- FrameTag:/Untag support now the tag for FrSimEvent. (Add also the FrameTagSimEvt and FrameUntagSimEvt functions)
- FrAdcDataDump: dump also the auxiliary vector if available
- FOr the Frame File List (.ffl files) skip files that have un undefined start time since they are unreadable
- Add the function FrSerDataGetValue, FrVectMean, FrProcDataNewV.
- Add the functions FrFileIGetEventSNRV, FrFileIGetSimEventSNRV and modify the function FrameReadTChnl to return voctors containing the events position.
- Add the function FrGetValueGPS and in FrVectGetValueX: subtract also the local vector offset (startX)
- FrCopy: the -l argument is interpretated as the length if it is a small number instead of the GPS time for the last frame.
- Minor change in EventDump functions.
- Remove the configure.tar file since it was not up to date.

- Change from double to float the type of the "value" paramter in FrEventAddParam since it is only a float that is saved on file.
- Update the cmt/requirement file.
- The matlab frgetvect function handles the start time and vector duration as double and not integer.
- Add the frextract.m help file
- Add the exampleComplex.c file
- Warning: any code that access directly the fields of FrSimEvent structure need to be recompiled since the order of the fields has been changed.

Most files written with version 3.40 and higher can be read with version 6.16

From Version 6.16 to Version 6.17 (February 21, 2005)

Thanks to Gianluca Guidi, and Andrea Viceré for suggestions, finding and reporting problems and bugs.

- Fix a bug introduce in v6r16: if a ffl contains only the file name, the file were rejected.
- Allow the non alignement on GPS for multiple output files.

Most files written with version 3.40 and higher can be read with version 6.17

From Version 6.17 to Version 6.18 (March 23, 2005)

Thanks to Stuart Anderson, Eric Chassande-Mottin, Martin Hewitson, Jean Jacquemier, Peter Kalmus, Didier Verkindt for suggestions, finding and report problems and bugs.

- Remove a debuging statement from the matlab frgetvect function
- Fix a bug in FrFileIGetChannelList: the size of an internal buffer was not properly incremented.
- Set an incremental instance number for the FrStatData structure
- Add the FrCListGetElement function
- Update the octave Makefile
- Add a script to compile the shared library for root under CYGWIN

Most files written with version 3.40 and higher can be read with version 6.18

From Version 6.18 to Version 6.19 (May 14, 2005)

Thanks to Eric Chassande-Mottin, Bernd Machenschalk and Frederique Marion for suggestions, finding and report problems and bugs.

- ffl files could now contain other ffl files.
- FrCheck work now with ffl.
- Fix a bug in FrAdcDataReadT() and FrProcDataReadT: If the GTime argument was set to 0, only the first call was working fine.
- Fix a bug in FrSimEventAddVect et FrEventAddVect: the newName argument was not use to set the name of the vector attached to the event but we changing the name of the initial vector. Remark: FrEventAddVectF was working fine
- FrFileISetTime: if call for a time corresponding to a gap in the file, the reading pointer is set to the next existing frame instead of the beginning of the file.
- Update the octave Makefile
- Add the FrameDumpTopADC and the -top option in FrDump

Most files written with version 3.40 and higher can be read with version 6.19

From Version 6.19 to Version 6.20 (Mar 15, 2006)

Thanks to Fabien Cavalier, Eric Chassande-Mottin, Jolien Creighton, Nick Fotopoulos, Gianluca Guidi, Bernd Machenschalk, Frederique Marion, Didier Verkindt and Andrea Viceré for suggestions, finding and report problems and bugs.

- Fix the incomplete handeling of the Static data structure in case of multiple detectors. Changes have been made in several places like FrameRead, FrameReadT, FrameAddStatData. FrameReadRecycled has been simplify it by using a straight FrameFree and FrameRead. FrStatDataCopy now add the stat data to the detector linked list
- FrSerDataFind: fix a bug in the format of the dataQuality flag when getting the SerData for the FrameH.
- FrTagNew: fix the logic to be able to split words even if this is a negative number.
- FrDump: add an error message if the file could not be opened.
- FrameDump: adjust the format for the time dump.
- FrSerDataNew: Add a protection against too long string (> 65535); Do the same in the string write function (FrPutSChar).
- FrFileISetTime: fix a bug that prevents to read in random access the first frame of a set of files if multiple files names were used instead of an ffl. This may speed up some data access using the FrFileIGetV.
- FrFileOPutV: fix a bug when using with file open by FrFileONewM (seg. fault).
- FrVectSave: bug fix in the case of no file name was provided.
- FrVectConcat: Fix the function to avoid to introduce a biais in the starting time. This was visible in the FrFileIGetVxx functions.
- Update the default leap seacond.
- Add the first version of the Python interface
- Add the FrVectRMS, FrVectToAudio, FrEventSaveOnFile, FrameSetPrefix, FrFileIGetTimeOfNextEvent functions.
- Update char *FrFileIGetChannelList and FrameReadTChnl to return the full list of events
- Extend the frame taging option to FrMsg messages.
- Optimize the memory used by the internal "taging" functions.

Most files written with version 3.40 and higher can be read with version 6.20

From Version 6.20 to Version 6.21 (Jan 23, 2007)

Thanks to Franco Carbognani, Nick Fotopoulos, Martin Hewitson, Ed Maros, Alain Masserot, Keith Thorne, Didier Verkindt, Igor Yakushin and others for suggestions, finding and report problems and bugs.

- Bugs fix:
  - FrVectConcat: introduced in v6r20 that produce missing sampling when the start time was not aligned on the frame start. This was visible in the FrFileIGetVxx functions.
  - Fix a bug that prevent to write a frame with GPS time= 0; when using multiple file output
  - In FrameNew: the default GPS time was off by the GPS to TAI number
  - In the tag functions (the –xx tag was not working since v6r20)
- FrSerDataNew: Add a protection if data == NULL
- Add the support for compression flag 9 (try the best compression flag 6 and 8)
- Put back the support for compression flag = 1 (gzip only)
- Add first support for segment list, including in FrCopy.
- Add support for missing channel prefix (“V1:”);
  - Update functions FrameFindProcData; FrameAdcDataFind, FrSimDataFind, FrFileIGetVType
  - Add FrLibSetFlipPrefix, FrStrFlipPrefix, FrameFindBasic, FrSerDataFind functions
- Update the Phyton interface

- In the script makegcc, fix the shared library compilation to avoid problem on Solaris
- In FrFilter.h: remove the #include <math.h> that generate trouble on HPSS in Bologna.
- Update the matlab interface (frgetvect and frextarct.c and .m) to handle longer file name, better error message and improve the documentation

Most files written with version 3.40 and higher can be read with version 6.21

From Version 6.21 to Version 6.22 (Feb 4, 2007)

Thanks to Frédérique Marion and others for suggestions, finding and report problems and bugs.

- Bugs fix in :
  - FrVectNew and FrVectCopy that could create segmentation fault on some computer when using multidimension vectors.
  - FrVectGetValueX and FrVectGetValueGPS the vect->startX[0] value was subtracted twice.
  - Segements management.
- FrVectGetIndex: add some protection to avoid rounding problems
- Implement forward compatibility for format version 7:
  - FrEvent and FrSimEvent: change the type of the param variable from float to double
  - FrSerData and FrSimData: change the type of sampleRate from float to double
  - Add the function FrLibSetVersion to write frame with the version 7 format.
- Add the function FrVectFixNAN

From Version 6.22 to Version 6.23 (April 26, 2007)

- Improve the backward compatibility for the channel prefix option
- When files open with FrFileONewM get the actuall file length if the file is shorter
- Fix a bug in FrTagFreeBugs

From Version 6.23 to Version 6.24 (May 2, 2007)

- Fix a bug in the Frame resizing functions

From Version 6.24 to Version 6.25 (May 28, 2007)

Thanks to Frédérique Marion and Alain Masserot for suggestions, finding and report problems and bugs.

- Shift by half a bin the return value for the FrVectGetIndex and GetGPS function
- Fix a bug in the FrFileIGetVect function that prevents the return of a vector in some cases. The function FrTOCFramexx functions have been updated.

From Version 6.25 to Version 6.26 (Mar 14, 2009)

Thanks to Stuart Anderson, Jolien Creighton, Carlos Filipe Da Silva Costa, Ben Johnson, Edward Maros, Alain Masserot, Didier Verkindt and John Zweizig for suggestions, finding and report problems and bugs.

- Fix a bug in FrReadVL: There was a memory corruption on 32-bit machine when FrameL was build with -DFR_LONG_LONG when reading vectors of long written with the opposite endian.
- Fix a bug that prevents to write the frameH->user vector.
- Fix a bug in FrFileIAddSegList which was not working if the second input argument (FrSegList *segList) was NULL.
- Fix a memory leak when reading static data

- Update the default value for FRGPSDELTA (now 15) since one leap second as been added on Jan 2009.
- Change the default value of FRIOBSIZE (file FrIO.h) from 2kBytes to 16kBytes to help speeding up the reading on some platform
- Implement loop-unrolling technique that can improve FrCheck performance by ~50% (done by Ben).
- When doing a frame merge (FrameMerge function) the resulting frame takes now the largest value of frameH->UleapS in case one of the frames has not capture the leap second change.
- Change the type of some test files in the examples from ".dat' to ".gwf"
- Fix a bug in FrameWriteToBuf: if there was not enough space in the buffer to write the TOC, the return flag was OK despite the error. However, nothing was written outside the too short buffer.
- Add the possibility to do Frame tagging by class of signals (ADC, PROC, …)
- Fix a bug in the tagging function when using "anti-tag": If a frame contains channel A et B the selection "* -A" was not selecting the channel B.
- Add support for reading frame v8. But frame are still written with format 6 by default.

Most files written with version 3.40 and up to format v8 can be read with version 6.26

From Version 6.26 to Version 8.00 (Mar 14, 2009)

This v8r00 is equivalent as v6r26 except that the default frame format version is v8 instead of v6. This means that this version write frame format v8 but can read also from format version 6.

From Version 8.00 to Version 8.01 (Apr 16, 2009)

Thanks to Jordi Burguet-Castell, Jolien Creighton, Alain Masserot and Didier Verkindt for suggestions, finding and report problems and bugs.

- Fix a bug in FrFileIGetChannelList which was no more returning the channel list on frame file with format version 8.
- Fix a bug in the tagging function "Adc0 Adc1 Adc2" only the last Adc was selected
- Change the default setting of the read pointer at the end of the call to FrTOCReadFull to 40 bytes after the beginning of the file in all case to be shure that file checksum will be properly computed (there was a recent case of LIGO M10 frames not compatible with this logic).
- Add the option to write the file in multiple folders: FrFileONewMD function.
- In the FrDetectorNew function, fill the detector channel prefix by 'V1' if the detector name is 'Virgo'.

Most files written with version 3.40 and up to format v8 can be read with version 8.01

From Version 8.01 to Version 8.02 (May 4, 2009)

Thanks to Frédérique Marion, Ed Maros and Didier Verkindt for suggestions, finding and report problems and bugs.

- Fix a bug in the TOC writing/reading when no events were present. This was generated wrong error message about checksum test that failed when reading a frame format version 8 files build with FrameCPP using the TOC.
- Fix a memory leak when calling the functions FrFileIGetEventInfo, FrFileIGetSimEventInfo, when no events were present in the requested time range.
- Enable the proper checksum computation if using random access for vector. Assuming the structure checksum is turned on (iFile->chkSumFrFlag = FR_YES after opening the file "iFile"), then the FrFileIGetVxx functions return a vector only if the checksum is OK. This feature is only valid with frame file written with format 8.
- Add the function FrGPS2UTC
- Update the list of ULeapS in FrStrGPS/FrGPS2UTC
- Add the Matlab function frgetvectN to return a normalized ADC vector.

- Update the test file in data/test.gwf

From Version 8.02 to Version 8.03 (May 6, 2009)

There were no changes since the intended changes where actually not included in the distributed version. Thanks to Adam Mercer for spooting it!

From Version 8.03 to Version 8.04 (May 6, 2009)

- Fix a bug in the TOC reading introduced in v8r02: the frame file version 6 and lower were no more readable with v8r02.
- Fix one more bug in the "tagging" function: in case of multiple "antitags", only the last one was used. For example if the tag "* -P* -E*" was request, only "* -E*" was actually applied.

Most files written with version 3.40 and up to format v8 can be read with version 8.04

From Version 8.04 to Version 8.05 (May 19, 2009)

Thanks to Peter Shawhan and Didier Verkindt for suggestions, finding and report problems and bugs.

- Add the functions :
  - FrHistoryCopy to copy a full linked list of FrHistory
  - FrProcDataCopy a procData
  - FrameCopyPart to copy part of a frame (with a zoom) and to resize frame to a shorter duration
  - FrVectFixVirgoImage to fix old Virgo images
- Fix a bug in FrEventCopy and FrSimEventCopy which was visible only if more than one FrSimEvent was present in the linked list.
- Improve the FrVectZExpand code write and speed (expension part of the zero suppress compression mode).
- Update the FrVectDump for 3d vectors
- FrVectConcat: rewrite it to support images/movies
- FrVectZoomIn to work with 3d vectors (movies)

Most files written with version 3.40 and up to format v8 can be read with version 8.05

From Version 8.05 to Version 8.06 (June 23, 2009)

Thanks to Adam Mercer, Loic Rolland and John Zweizig for suggestions, finding and report problems and bugs.

- Add the support to read vector with zero suppress algortithm for 8 bytes words.
- Add files for the autotools build.
- Fix a bug in FrVectSave which was introduced in v6r26. The vectors were no more readable.

Most files written with version 3.40 and up to format v8 can be read with version 8.06

From Version 8.06 to Version 8.07 (June 27, 2009)

Thanks to Adam Mercer and Greg Mendell for suggestions, finding and reporting problems and bugs.

- Fix a bug in the IO routine which impact random access for the last elements of the files only in case of short file, with limited number of channel. This bug was probably visible only after v6r26 when the size of the internal buffer (FRIOBSIZE) was increased.
- Remove some internal limitation which was preventing the read of files with more than 65536 FrAdcData channels. The FrAdcDataFree function has been upgraded to avoid recursive call.
- Fix a type definition in FrVectExpendL which was generating compilation error messages.
- Add the option to compute checksum when writing a frame in a buffer (function FrameWriteToBuf)

Most files written with version 3.40 and up to format v8 can be read with version 8.07

From Version 8.07 to Version 8.08 (June 30, 2009)

Thanks to Adam Mercer for suggestions, finding and reporting problems and bugs.

- Remove the example file: examplePutV.c

Most files written with version 3.40 and up to format v8 can be read with version 8.08

From Version 8.08 to Version 8.09 (October 11, 2009)

Thanks to Alain Masserot, Didier Verkindt and John Zweizig for suggestions, finding and reporting problems and bugs.

- Fix a buf that was creating a memory leak when resizing several small frame to a large one with channels present only for part of the small frame and not the first one.
- Fix the file checksum flag value which was incoretly set to "90" when no file checksum was request. This result in missleading FrCheck error messages
- Fix a buf in FrFileOEnd which induce segmentation fault if a file was open with the option to limit the number of frames per file and then closed before any frame was written
- Add protection in FrSimEventAddParam, FrEventAddParam and FrProcDataAddParam to avoid trying to add more parameters than the maximum allowed number (65335).
- Improve the FrSerDataGet (and therfore FrSerDataGetValue) to look for the channel in the single sample AdcData if the channel is not found as SerData. This may happens now in Virgo because of the frame resizing.
- Rewrite the FrameReshape function to avoid memory leak if a channel is not always present.
- Update FrVectConcat to handle "available data" vectors
- Update the FrameCopyPart function to not copy vector wihout available data (this was used for going from 10s. long frame to 1 s. long frame for instance)
- Supress the channel position in the FrDump with option –d 2

From Version 8.09 to Version 8.10 (December 20, 2009)

Thanks to Alain Masserot for suggestions, finding and reporting problems and bugs.

- FrEventCopy and FrSimEventCopy: fix a bug which prevents to copy the event parameters.
- FrameCopyPart: change the test on event to avoid duplicated event if the event is at the border of two frames
- Fix a bug introduce in Fr/v8r09 in FrFileOEnd which prevent to write frame in a local buffer (line 7471)
- Protect the function FrCksumGnu against buffer of size 0.
- FrameCopyPart: zoom on the procdata if the type is 0 or1, instead of just 1
- FrVectZoomIn: fix it to zoom when the number of point is less than 1 on average
- FrSerDataGet: fix a bug introduce in v8r09 when looking for AdcChannel if a ser data was converted to an ADC (for channel name larger than 18 characters)
- Add the function FrameRemoveDuplicatedADC
- Upgrade FrameDumpTopADC to list the Proc channels in addition to the ADC channels.

From Version 8.10 to Version 8.11 (Febraury 20, 2010)

Thanks to Adam Mercer and Alain Masserot for suggestions, finding and reporting problems and bugs.

- Update the root/build script to use rootcint instead of cint. The file src/FrRootLinkDef.h has been added since it was needed by rootcint.

- Fix a problem introduce in bv8r07 to use checksum in FrameWriteToBuf. If the compression was asked to be -1, this was interpreted as compression 1 plus checksum. Now it is back to “do not change compression state”.
- Patches some tools (configure.ac, debian…) to fix build in some environments.

From Version 8.11 to Version 8.13 (November 14, 2010)

Thanks to Franco Carbognani, Adam Mercer for suggestions, finding and reporting problems and bugs.

- Include the sorting by GPS time of ffl. Before that, if an ffl was not correctly sorted, it was unusable (v8r11p1)
- Add a protection in FrameSetPrexif when an ADC has not data(v8r11p2)
- Upgrade of the exampleCompression code (v8r11p2)
- fix FrVectConcat for vector with GTimeS=0 (v8r11p3)
- Detect (and protect against) corrupted file by checking the end of file record (v8r12). Before that, especially since v8r08, trying to open an incomplete file generate a seg fault
- Fix the detection of end of file which was not properly working for very short files (v8r12p1)
- Update the autconf tools (v8r13)

From Version 8.13 to Version 8.14 (April 15, 2011)

Thanks to Thomas Adams, Duncan Brown, Franco Carbognani, Alain Masserot, Frederique Marion, Emmanuel Pacaud, Loic Rolland, Patrick Sutton and Didier Verkindt for suggestions, finding and reporting problems and bugs.

- Fix FrCopy and FrDump: When doing the copy or dump for a time range starting at a time without a frame available for the start of this time range, nothing was opied or returned, even if there was some frame later on. Now, it start with the first available frame in the requested range.
- FrCopy: update the information message to display only the GPS time (run/frame numbers are now removed)
- FrDump: if the debuglevel is 5 or more and a time range has been given, now, only the corresponding part of the vector is dumped.
- Update FrFileIDumpT to use fix floating point number and remove trailling zeros". This is to make ssure that long file (1.e6 seconds or more) are properly described and to produce ligther ffl files.
- FrIO: FrIORead: add a check if the nuber of bytes read is zero. This was produced crashes on some file system when the file does not really exist but the file open was successful.
- Add in the src directory the application FrFileMerge which merges two frame files into one file, filling the holes in the first one with the frames present in the second one.
- FrVectIsValid accept now the denormalized number. The function FrVectIsValidStrict(FrVect *vect) has been added and is equivalent to the old FrVectIsValid function. The FrVectIsValid now create if needed and fill the auxiliary vector with the -1 value for the bins with a nan or inf floating pointing number.
- Fix a bug in FrVectConcat which affect the the FrFileIGetVect function (and frgetVect): if the requested time start before the frame file, the returned vector start with one bin late.
- Change FrameReshapeEnd to not reshape the frequency domain spectrum
- Fix a bug in FrSegListIntersect. The intersection of two segment lists was not properly computed in some cases.
- Minor printf format change in FrCheck exampleFull.c, to make shure that it will work if GPS>999999999 (Local Time: Sep14,11-03:46:24)
- Add the function FrProcData* FrProcDataNewVT(FrameH* frame, FrVect *vect, int type)
- Add the function FrameH* FrameReshape(FrameH* frame, FrameH** workSpace, int newFrameLength);
- Put back the matlab/mymex script wich was lost in the previous versions.
- Change the makegcc script to compile directly with the option -fPIC
- Update the requirement file to use the version of root which is available on this platform. Do not make the root part if root is not available.

From Version 8.14 to Version 8.15 (May 30, 2011)

Thanks to Alberto Colla, Franco Carbognani, Adam Mercer and Didier Verkindt for help, suggestions, finding and reporting problems and bugs.

- Improve the documentation for the FrStatData part to try to avoid confusion when recovering FrStatData information.
- Add a protection in FrVectConcat. When the FrFileIGetVect function was call with some unisuall parameters (non integer start time close to an integer vaule), the program crashed.
- Add a protection in FrameStat to avoid crash if an ADC structure has no associated vector (which is anyway unusual).

From Version 8.15 to Version 8.15p1 (September 19, 2011)

Thanks Franco Carbognani, and Bas Swinkels for help, suggestions, finding and reporting problems and bugs.

- Recover matlab mex file from v8r13 to use static library instead of dynamic; Put back the FrDump/Copy/Check alias in the requirement file.

From Version 8.15p1 to Version 8.15p2 (September 19, 2011)

- Add the functions FrameAddAdc, FrameAddProc, FrameAddSer, FrameAddSum, FrameAddSimData. Used them in the corresponding "New" functions (FrameAdcNew,…). These functions works even if tagging is used. Fix FrameAddEvent to work also when tagging is used.
- Add a protection in FrameReshapeConvertSer if the requested frame is too short (less than 1 second).

From Version 8.15p2 to Version 8.15p3 (December 15, 2011)

Thanks to Franco Carbognani and Adam Mercer help, suggestions, finding and reporting problems and bugs.

- Add Adam patches: don-t-package-.la-files, use-default-autoconf-location-by-default, package-in-fhs-locations

From Version 8.15p3 to Version 8.15p4 (December 22, 2011)

Thanks to Didier Verkindt for help, suggestions, finding and reporting problems and bugs.

- Minor bugs fix:
  - FrameReshapeConvertSer: Remove the limit to 500 FrSerData values when converting FrSerData to FrAdcData.
  - FrVectExtend: Fill the ULeapS field when creating a new vector.
  - FrAdcDataCopy: The AdcData structure was not correctly added to the linked list if a tag on the structure was applyed on the frame at the same time.

From Version 8.15p4 to Version 8.15p5 (January 2, 2012)

- Bugs fix in:
  - FrVectCopyTo: to properly handle the rounding when going from integer to floating point.
  - FrVectGetValueI: was returning a bad value for unsgined integer larger than half the maximum value.
- FrameWrite: When using the feature of automatic file reopening when a given number of frames is written, the file was closed only when the next frame was witten. Now the file is closed when the last needed frame is written, without waiting for the next frame. The file renaming (removal of the suffix "NOT_YET_CLOSED") is now also done when the file is really closed. Before it was done after the last frame write, but before the TOC write, a little bit too early.

From Version 8.15p5 to Version 8.15p6 (January 5, 2012)

Thanks to Alain Masserot for help, suggestions, finding and reporting problems and bugs.

- Bugs fix in:
    - FrVectConcat: if an input vector has already some missing samples reported in the auxiliary vector, then there was a crash. This could be produced in the case of change of the frame size.
    - FrVectCopyTo: add a check to expend a compressed vector before doing the copy.

From Version 8.15p6 to Version 8.15p7 (January 15, 2012)

Thanks to Frédérique Marion for help, suggestions, finding and reporting problems and bugs.

- FrameAddEvent: go back to the logic used before v8r15p2: the event is added at the end of the linked list and not the beginning. This was creating some problem since some applications (like the clustering of Mbta) were reliying of this order.

From Version 8.15p7 to Version 8.16 (June 5, 2012)

Thanks to Franco Carbognani, Gergely Debreczeni, Adam Mercer and Loic Rollandfor help, suggestions, finding and reporting problems and bugs.

- For for pkgconfig file: include the path to the math library in src/libframe.pc.in.
- UPdate the requirement file to use VirgoPolicy/v2r5 (for SL6 support)
- Update the LEAP seconds values for the upcoming change of Jul 1 2012.
- Add some more documentation about reading/writing frames

From Version 8.16 to Version 8.16p1 (July 23, 2012)

Thanks to Nicolas Leroy for help, suggestions, finding and reporting problems and bugs.

- Add protections in FrameCopyPart to avoid segmentation fault when working on frame having FrProc or FrAdc channels without associated vectors.
- Add a protection in FrFileONewM to set fLength to a large number if zero or a negative value is provided.

From Version 8.16p1 to Version 8.17 (August 29, 2012)

Thanks to Franco Carbognani, Pacaud Emmanuel, Peter Shawhan for help, suggestions, finding and reporting problems and bugs.

- Add the C programs FrChannels, FrameDataDump, FrDiff, FrTrend and the Matlab programs mkframe, mkframe_simevt from LIGOtools
- FrAdcDataDump: print the frequency shift and associated phase if the values are different from zero
- FrProcDataNew: set type to 1 (time serie) by default.

From Version 8.17p1 to Version 8.17p2 (October 14, 2012)

Thanks to Franco Carbognani and Adam Mercier for help, suggestions, finding and reporting problems and bugs.

- Apply a set of patches to various autotools files.

From Version 8.17p2 to Version 8.17p3 (February 2, 2013)

Thanks to Didier Verkindt for help, suggestions, finding and reporting problems and bugs.

- Add a protection in FrameCopyPart to avoid a crash when resizing a frame with a FrProdData channel without associated vector.

From Version 8.17p3 to Version 8.18 (February 10, 2013)

Thanks to Didier Verkindt for help, suggestions, finding and reporting problems and bugs.

- Fix a bug: for vectors with an auxiliary vector telling if only part of the vector was fill, the functions of the type FrFileGetVect were not returning properly this auxiliary vector, generating also a memory leak in the case of incomplete vector.
- Add the functions FrGetLeapS, FrGetCurrentGPS, FrameLatency, FrSimEventSetParam.
- Use the function FrGetCurrentGPS set the the current GPS time when creating a frameHeader or various history records.
- Add some support to read frame cache files.
- Whange a cast from FRLONG to FRULONG to try to fix a compilation error on Solaris.

From Version 8.18 to Version 8.19 (May 2, 2013)

Thanks to Franco Carbognani, Duncan Macleod, Frédérique Marion, Alain Masserot and Chris Pankow for help, suggestions, finding and reporting problems and bugs.

- Fix a bug: in FrSimDataNew: if no frame header was passed, there was a segmentation fault when trying to attach the structure to a non existing frame.
- Add a protection in FrameReshapeConvertSer: there was a crash when a SerData had the same name of an existing Adc.Now, the sampling frequency is added at the end of the SerData to avoid name conflicts.
- When converting a serData to and AdcData (when incrinsing the frame duration), the unitY (if available) is now propagated to the FrAdcData vector (fix in FrameReshapeConvertSer).
- Remove the special logic for "V1:" channel tag. The V1 prefix was not mandatory to select/remove a channel. Now it is. The old logic could be set by calling the function: FrLibSetFlipPrefix("V1:");
- Update the Python wrapper using the latest version from http://www.lsc-group.phys.uwm.edu/cgit/lalsuite/tree/pylal/src/Fr.c (written by Nick Fotopolous).
- Move to VirgoPolicy v2r6.
- Move to root/v5r34p5 and fix the requirement file to work without root with the new version of CMT.

From Version 8.19 to Version 8.19.1 (May 24, 2013)

Thanks to Adam Mercer, Dan Moraru and Emmanuel Pacaud for help, suggestions, finding and reporting problems and bugs.

- Apply few minors patches for LIGO distribution.
- Fix a typo in FrCheck comments.
- Move to root/v5r34p050.
- Update the requirement file to avoid linking FrDump, FrCheck and FrCopy with the root libraries

From Version 8.19.1 to Version 8.20 (September 25, 2013)

Thanks to Jeff Kline, Gerrit Kühn , Adam Mercer, Florent Robinet, Gabriele Vedovato for help, suggestions, finding and reporting problems and bugs.

- Remove the FrHeader string.
- Fix a too short memory allocation creating overflow in FrameDataDump.c

- Bug fix: the function FrFileIGetVectX was crashing if there were data gaps due to the improper construction of the auxiliary vector telling the position of the missing samples (bug introduced in v8r18 while trying to fix another problem).
- Update FrFileINew to open frame stored in memory by passing just a file name.
- Update FrFileOEnd and FrFileOReopen to trap write error when closing a file and to avoid removing the NOT_YET_CLOSED suffix in case of incomplete file.
- When converting a serData to and AdcData (when incrinsing the frame duration), the unitY (if available) is now propagated to the FrAdcData vector only if the slope is 1, which means the vector is already calibrated (fix in FrameReshapeConvertSer).
- Fix a bug in FrFileIGetVectDN and FrFileIGetVectFN : the bias was subtracted instead of added.

From Version 8.20 to Version 8.21 (April 24, 2014)

Thanks to Frederique Marion, Adam Mercer and Leo Singer for help, suggestions, finding and reporting problems and bugs.

- Fix a memory leak in FrameReshapeConvertSer
- Fix the prototypes of functions that take no arguments from () to (void) to avoid problem with icc compiler
- Improve error message with FrCheck (EOF reached and name of the file in error, useful for ffl).
- ensure libframe.la isn't packaged

From Version 8.21 to Version 8.22 (September 26, 2014)

Thanks to Philip Charlton and Frédérique Marion, for help, suggestions, finding and reporting problems and bugs.

- Matlab frgetvect interface: renable the test on the debug level to remove a compiler warning noticed by Philip. Turn the warning message which describes the request to a regular info message. This message could be now suppress with debugLvl set to a negative value.
- Turn documentation file (this file) from .htlm type to .docx
- FrVectConcat: Remove an offset correction which creates vector misalignment in some cases.
- Fix FrVectDiff, FrVectNew, FrVectNewL to make them thread safe.
- FrEventFree and FrEventWrite: remove recursvive calls with the linked list to avoid stack overflow with large number of events.
- Add the support for multiple files in memory (FrFileIInMemoryNew and FrFileIInMemoryAddBuffer functions)

From Version 8.22 to Version 8.23 (December 7, 2014)

Thanks to Didier Verkindt, for help, suggestions, finding and reporting problems and bugs.

- FrameWriteToBuf : add a parameter to tell to compute the checksum
- FrCopy: fix nBytes info message for files larger than 2GB
- FrameRead: remove file name from a read error message since it is not be available when reading from buffer which was generating seg. fault
- Add -g compilation option to the requirement file
- Fix FrVectCopyToD, F, S, I to work with 2 or 3 d vectors. Add the FrVectCopyToType function.
- FrFileIGetVxx functions: if the vector is not a time series, just return the raw vector in the requested time range instead or rebuilding a continuous time series. If more than one vector is present, return them as a linked list.
- FrSegListRead: Add a protection against segment list files which could not be open to avoid a seg. fault

From Version 8.23 to Version 8.24 (February 24, 2015)

Thanks to Didier Verkindt, for help, suggestions, finding and reporting problems and bugs.

- Fix bug: when reading vectors from multiple in-memory files and calling the FrFileIGetVectXX read function with decreasing time, the read were not successful.
- Add support for FR_VECT_4U when computing vector statistics
- Remove variables set but not used (catch by gcc v4.8.1)

From Version 8.24 to Version 8.25 (April 12, 2015)

Thanks to Carlos Filipe Da Silva Costa and Alain Masserot, for help, suggestions, finding and reporting problems and bugs.

- FrameReshape bug fix: if the first provided short frames was the last part of the long frame, then the vectors were not properly placed and extended.
- FrameCopyPart : add a protection to not copy ADC/ProcData with incomplete vector;
- Improve MacOs support: Add root/build_MacOs; and fi typo in mgr/makeMacOS
- Change the EVENT_SNR_ALL and SIM_ EVENT_SNR_ALL sampling rate from 20kHz to 1kHz.
- FrEventFindVect: make sure the returned vector is uncompressed.
- FrSimEventReadTF: exclude the upper bound of the range when searching the event

From Version 8.25 to Version 8.26 (May 17, 2015)

- Introduce leap second for July 1st 2015
- FrVectConcat: Add protections when extending vector with one or less sample like a 20 seconds vector with a sampling rate of 1/200

From Version 8.26 to Version 8.27 (September 13, 2015)

Thanks to Alain Masserot and Didier Verkindt, for help, suggestions, finding and reporting problems and bugs.

- F Bug fix: remove the check on FrProcData->tRange to check that the vector is a time series or not. Due to this check, the vectors returned by the function FrFileIGetVectXX were not properly rebuilt on some LIGO data.
- Update the FrGPS2UTC function to use the leap seconds from FrGetLeapS instead of duplicating the table which, by the way, was not completely updated last time.

From Version 8.27 to Version 8.28 (December 11, 2015)

Thanks to Florent Robinet and Didier Verkindt, for help, suggestions, finding and reporting problems and bugs.

- Fix a bug in FrFileIGetVectXX (actually in FrVectConcat) that prevented to return vectors with only on bin. A longer vector was returned.
- Fix a bug in FrVectZoomInI which was not setting the proper protection against wrong argument for multiple dimension vectors
- Fix a bug in FrVectZoomIn which was not working properly for multi dimension vectors
- FrVectNew and FrVectNewL: Remove a test on vect->nData < 0 never used since nData is an unsigned variable.
- Requirements file: add few lines to enrich paths with the Fr applications and libraries
- Add FrVectMin and FrVectMax functions

From Version 8.28 to Version 8.29 (January 2, 2016)

- Fix Fix a memory leak when using the FrameRemoveUntaggedProc and FrameRemoveUntaggedEvent functions.

From Version 8.29 to Version 8.30 (April 20, 2016)

Thanks to Adam Mercier, Yousuke Itoh and Didier Verkindt, for help, suggestions, finding and reporting problems and bugs.

- Add FRVECTFREE
- FrTOCFrameFindT: add a protection if called with a negative time, which is then convert to zero i.e. first frame
- Update autotools and configure scripts
- Add proper definition and cast for the FrDicAddS function and calls