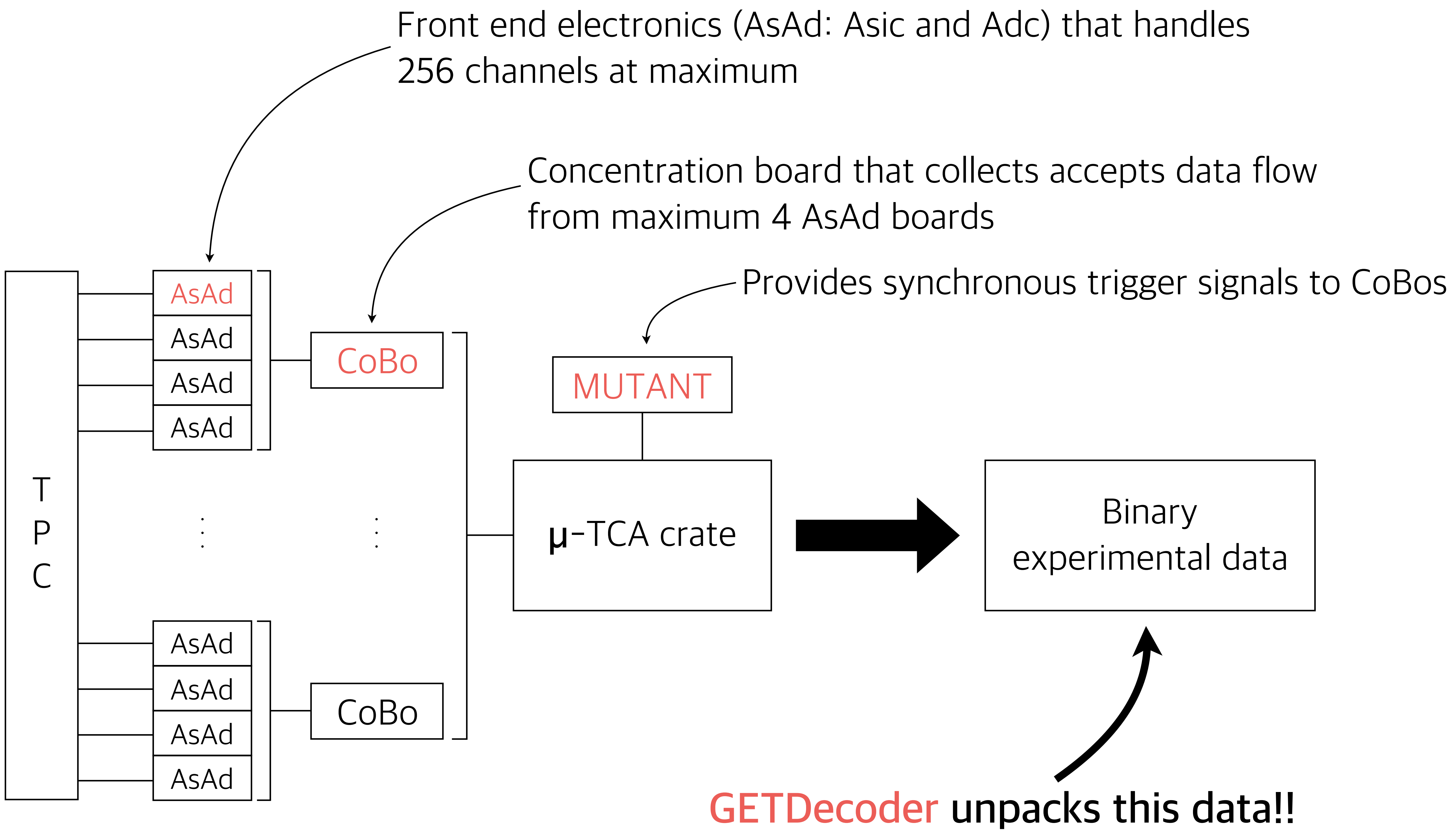
GETDecoder

- Unpacking software for the GET electronics data -

Genie Jhang

GET electronics

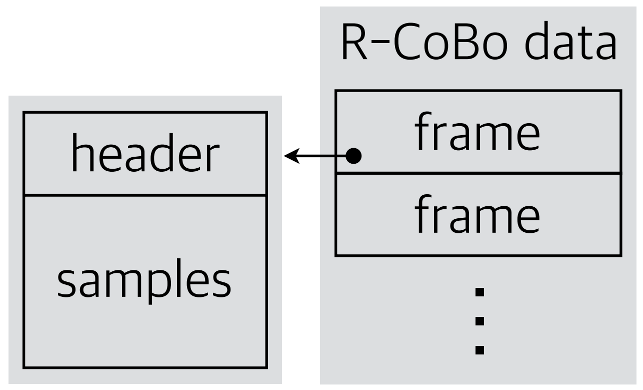

Front end electronics (AsAd: Asic and Adc) that handles 256 channels at maximum
Concentration board that collects accepts data flow from maximum 4 AsAd boards
Provides synchronous trigger signals to CoBos
T P C
AsAd
AsAd
AsAd
AsAd
CoBo
MUTANT
μ-TCA crate
Binary experimental data
AsAd
AsAd
AsAd
AsAd
CoBo

GETDecoder unpacks this data!!

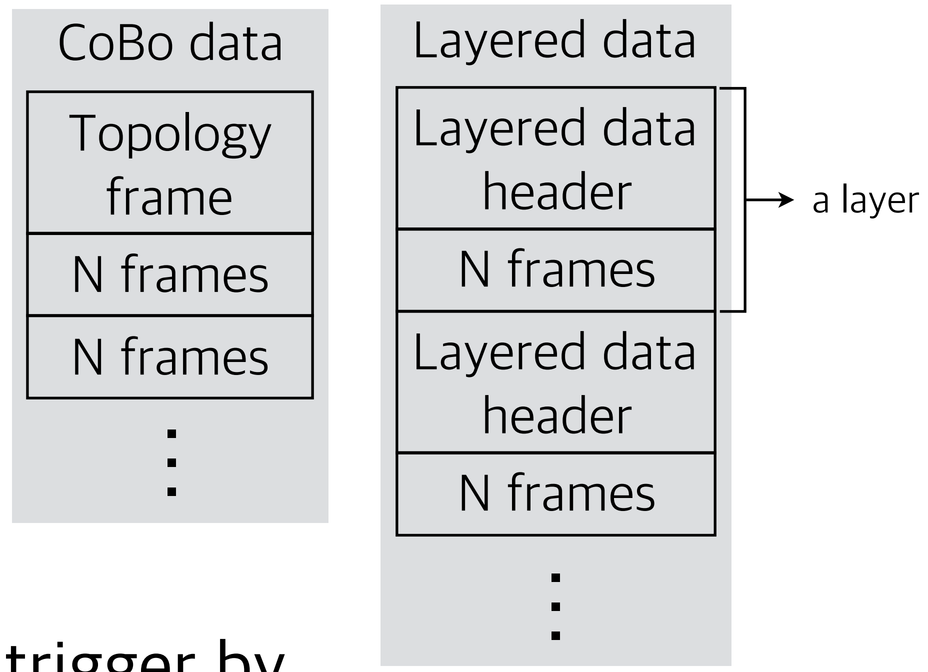
Why GETDecoder

- Bundle software provides the conversion tool to ROOT format.
  - ➡ This takes up about twice the disk space.
  - ➡ One should wait until the conversion is done.
- In case that many CoBos are used to take data, the number of data files are the number of CoBos. To perform any analysis with these data files, one should merge separated files into one data file.
  - ➡ Merging process takes really long time as well as the resulting file takes up the disk space.
  - ➡ Multiple GETDecoder instances can handle multiple files at once.
- GETDecoder provides convenient features not included in the bundle software.
- Easy to use interactively with ROOT prompt.

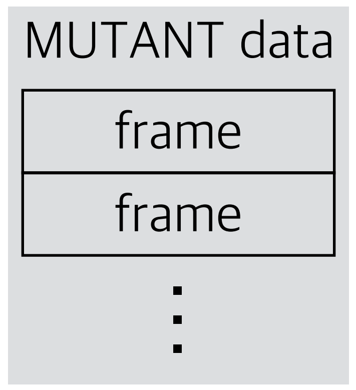
Data types



- R-CoBo data
  - Coming out from the reduced-CoBo.
  - Composed of frames, one of which is the basic component created by a trigger.

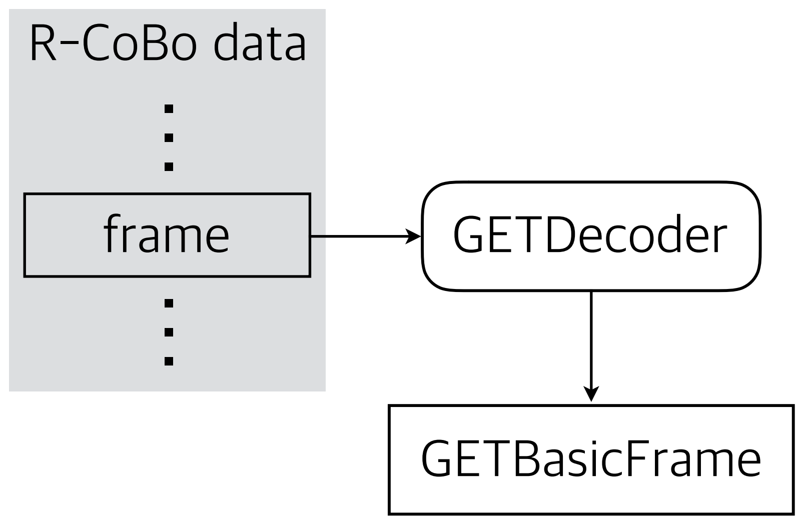


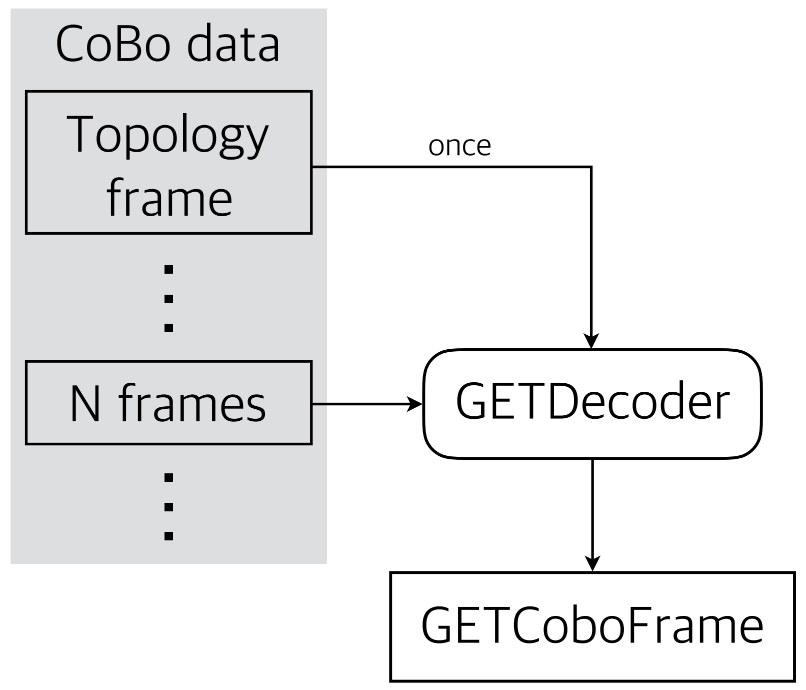
- CoBo data*
  - Coming out from the μ-TCA CoBo.
  - N(1~4) frames are generated at once by a trigger.
- Layered data
  - Merged data of several CoBo data having the same trigger by cobo-frame-merger(bundle)

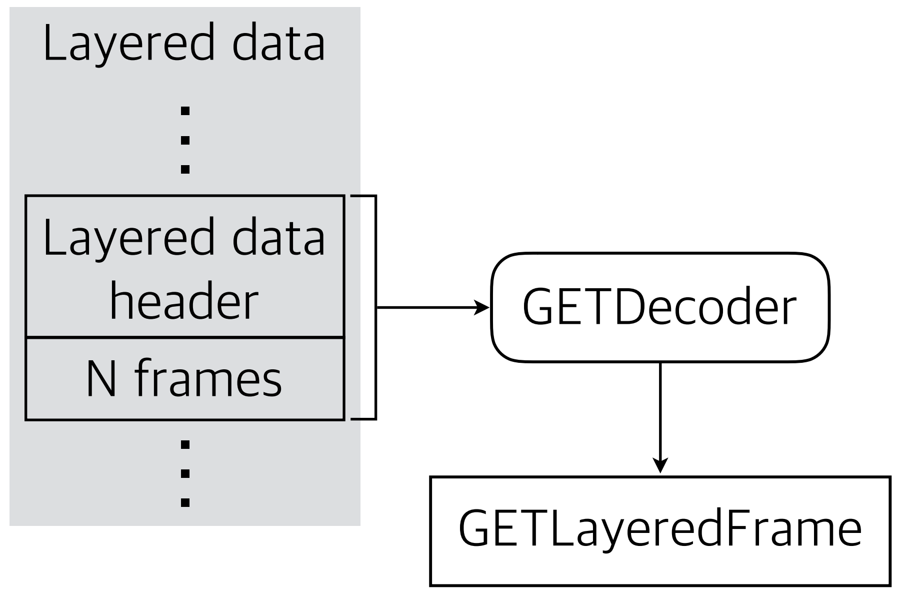


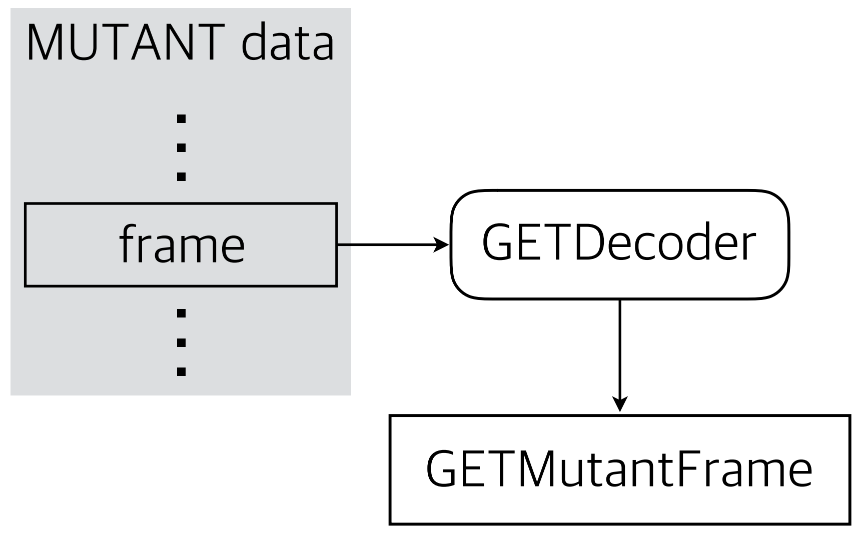
- MUTANT data
  - The data generated by MUTANT module containing trigger and scaler information.

How it works


R-CoBo data
frame
GETDecoder
GETBasicFrame


CoBo data
Topology frame
once
N frames
GETDecoder
GETCoboFrame

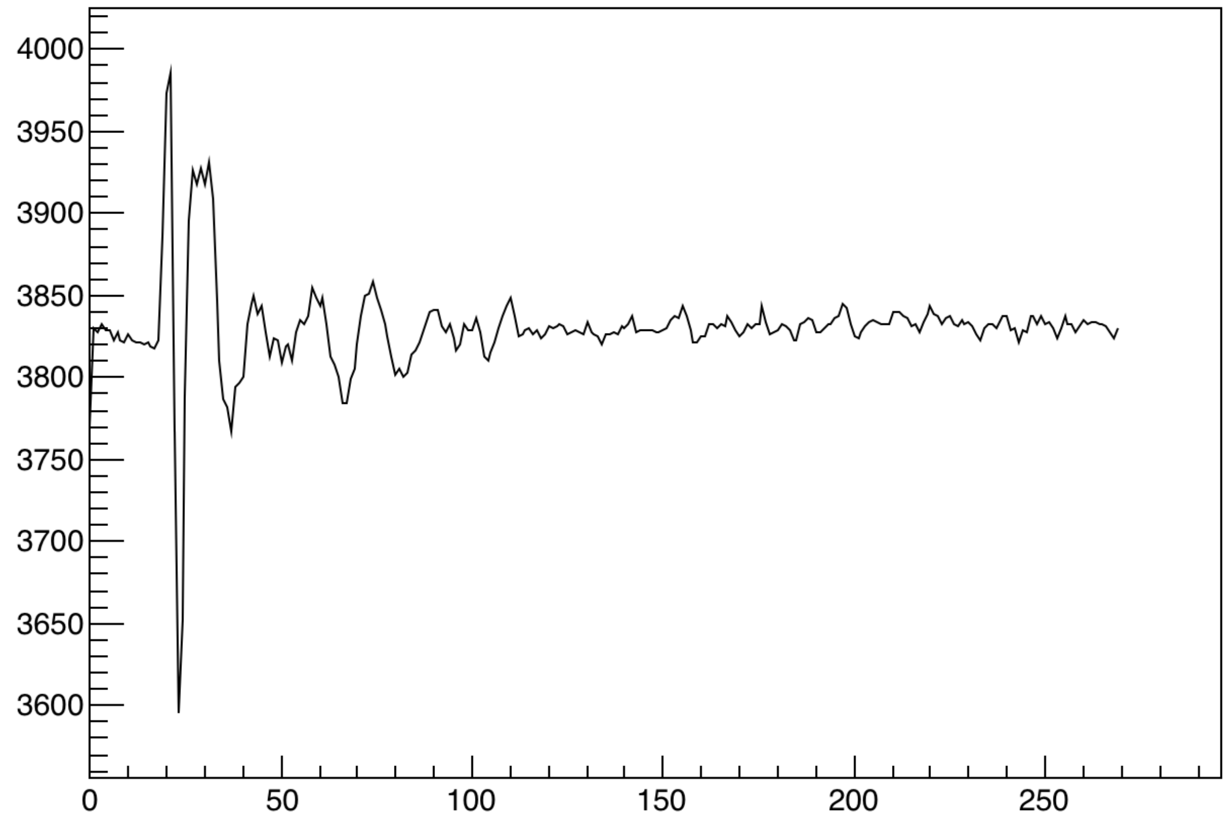

Layered data
Layered data header
N frames
GETDecoder
GETLayeredFrame


MUTANT data
frame
GETDecoder
GETMutantFrame

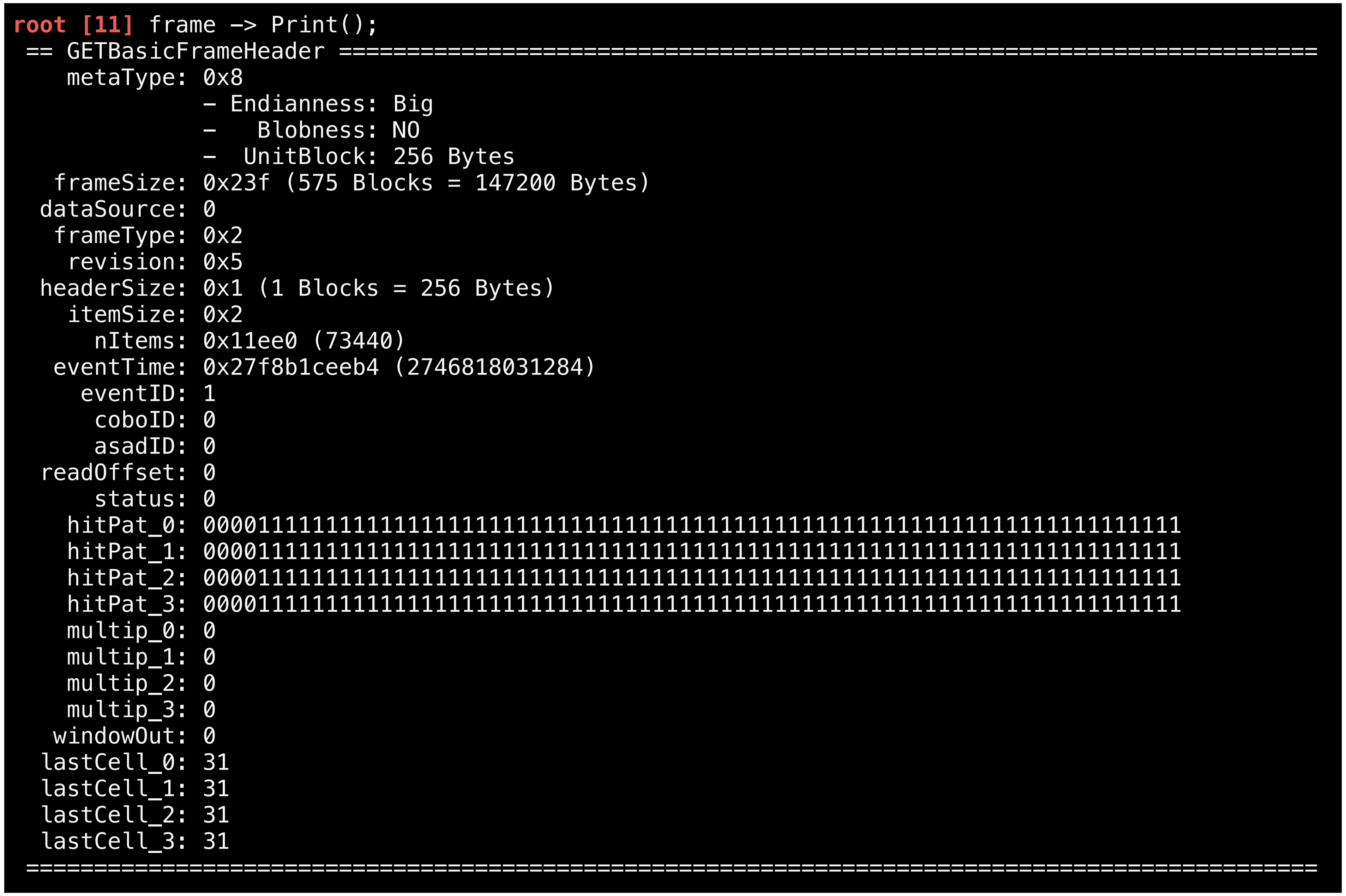
GETCoboFrame example

```
root [0] gSystem -> Load("libGETDecoder.dylib");
root [1] auto decoder = new GETDecoder("~/data.graw");
== [GETFileChecker] File exist: /Users/geniejhang/data.graw
== [GETDecoder] /Users/geniejhang/data.graw is opened!
== [GETDecoder] Frame Type: Cobo frame (Max. 4 frames)
root [2] auto coboFrame = decoder -> GetCoboFrame();
root [3] coboFrame -> GetNumFrames()
(Int_t) 4
root [4] auto frame = coboFrame -> GetFrames()
(GETBasicFrame *) 0x11d003010
root [5] Int_t tb[270];
root [6] Int_t *adc = nullptr;
root [7] std::iota(tb, tb + 270, 0);
root [8] adc = frame -> GetSample(0, 0);
root [9] auto graph = new TGraph(270, tb, adc);
root [10] graph -> Draw("AL");
```

Int_t* GetSample(Int_t agetIdx, Int_t chIdx)

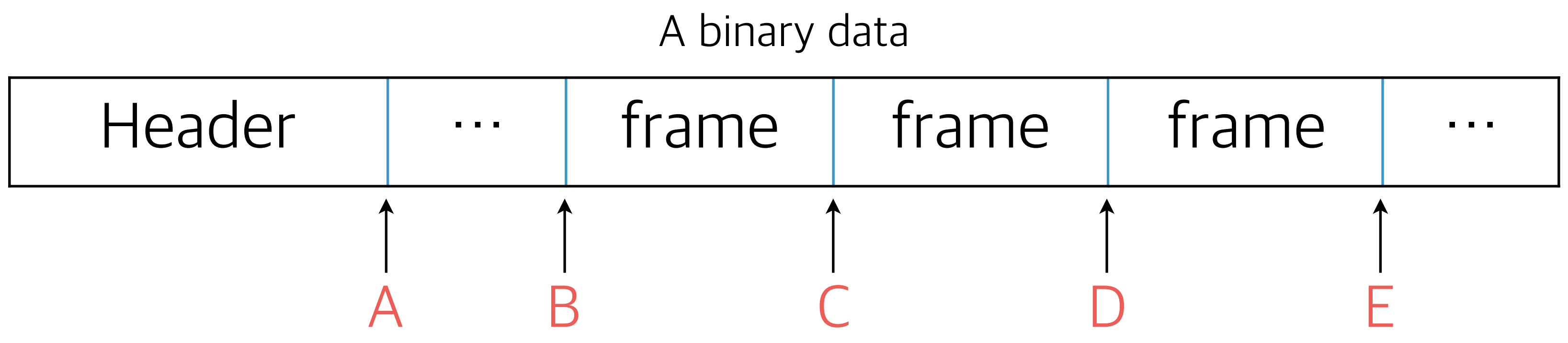


GETCoboFrame example



```
root [11] frame -> Print();
 == GETBasicFrameHeader ========================================================================
    metaType: 0x8
              - Endianness: Big
              -   Blobness: NO
              -  UnitBlock: 256 Bytes
   frameSize: 0x23f (575 Blocks = 147200 Bytes)
  dataSource: 0
   frameType: 0x2
    revision: 0x5
  headerSize: 0x1 (1 Blocks = 256 Bytes)
    itemSize: 0x2
      nItems: 0x11ee0 (73440)
   eventTime: 0x27f8b1ceeb4 (2746818031284)
     eventID: 1
      coboID: 0
      asadID: 0
  readOffset: 0
      status: 0
    hitPat_0: 000011111111111111111111111111111111111111111111111111111111111111111111
    hitPat_1: 000011111111111111111111111111111111111111111111111111111111111111111111
    hitPat_2: 000011111111111111111111111111111111111111111111111111111111111111111111
    hitPat_3: 000011111111111111111111111111111111111111111111111111111111111111111111
    multip_0: 0
    multip_1: 0
    multip_2: 0
    multip_3: 0
   windowOut: 0
  lastCell_0: 31
  lastCell_1: 31
  lastCell_2: 31
  lastCell_3: 31
 ===============================================================================================
```

Metadata



- With the binary file, to read the frame starting from the position C, the process should scan from A to C to look for C.
- GETDecoder automatically records the start byte of each frame (A~E) **in memory** while unpacking.
- It enables us to move to any frame we want instantly.
- This feature is very useful when we submit the batch job. Any batch starting in the middle of the data file runs instantly without scanning the file from the beginning.
- Saving the metadata from the memory to ROOT file format is done with a single line command.

Metadata - example

```
root [0] gSystem -> Load("libGETDecoder.dylib");
root [1] auto decoder = new GETDecoder("~/data.graw");
== [GETFileChecker] File exist: /Users/geniejhang/data.graw
== [GETDecoder] /Users/geniejhang/data.graw is opened!
== [GETDecoder] Frame Type: Cobo frame (Max. 4 frames)
root [2] decoder -> GoToEnd();
== [GETDecoder] End of data list!
root [3] decoder -> SaveMetaData(0);
```

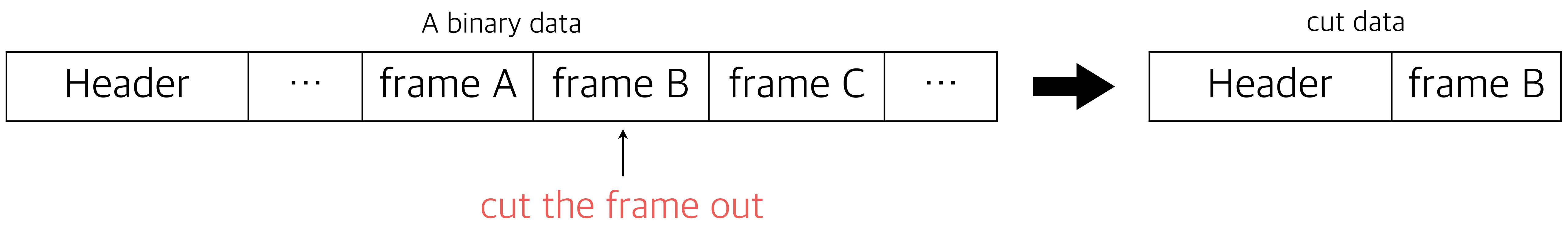
void SaveMetaData(Int_t runNo, TString filename = "", Int_t coboIdx = -1)

void LoadMetaData(TString filename)

- run_0000/metadata/data.graw.meta.root

```
************************************************************************************
*    Row   *    dataID *   eventID * eventTime *    deltaT * startByte *   endByte *
************************************************************************************
*        0 *         0 *         1 *         0 *         0 *        12 *    147212 *
*        1 *         0 *         1 *         0 *         0 *    147212 *    294412 *
*        2 *         0 *         1 *         0 *         0 *    294412 *    441612 *
*        3 *         0 *         1 *         0 *         0 *    441612 *    588812 *
*        4 *         0 *         2 *         0 *         0 *    588812 *    736012 *
*        5 *         0 *         2 *         0 *         0 *    736012 *    883212 *
*        6 *         0 *         2 *         0 *         0 *    883212 *   1030412 *
*        7 *         0 *         2 *         0 *         0 *   1030412 *   1177612 *
*        8 *         0 *         3 *         0 *         0 *   1177612 *   1324812 *
*        9 *         0 *         3 *         0 *         0 *   1324812 *   1472012 *
*       10 *         0 *         3 *         0 *         0 *   1472012 *   1619212 *
*       11 *         0 *         3 *         0 *         0 *   1619212 *   1766412 *
*       12 *         0 *         4 *         0 *         0 *   1766412 *   1913612 *
*       13 *         0 *         4 *         0 *         0 *   1913612 *   2060812 *
*       14 *         0 *         4 *         0 *         0 *   2060812 *   2208012 *
*       15 *         0 *         4 *         0 *         0 *   2208012 *   2355212 *
*       16 *         0 *         5 *         0 *         0 *   2355212 *   2502412 *
*       17 *         0 *         5 *         0 *         0 *   2502412 *   2649612 *
*       18 *         0 *         5 *         0 *         0 *   2649612 *   2796812 *
*       19 *         0 *         5 *         0 *         0 *   2796812 *   2944012 *
************************************************************************************
```

Frame cutter



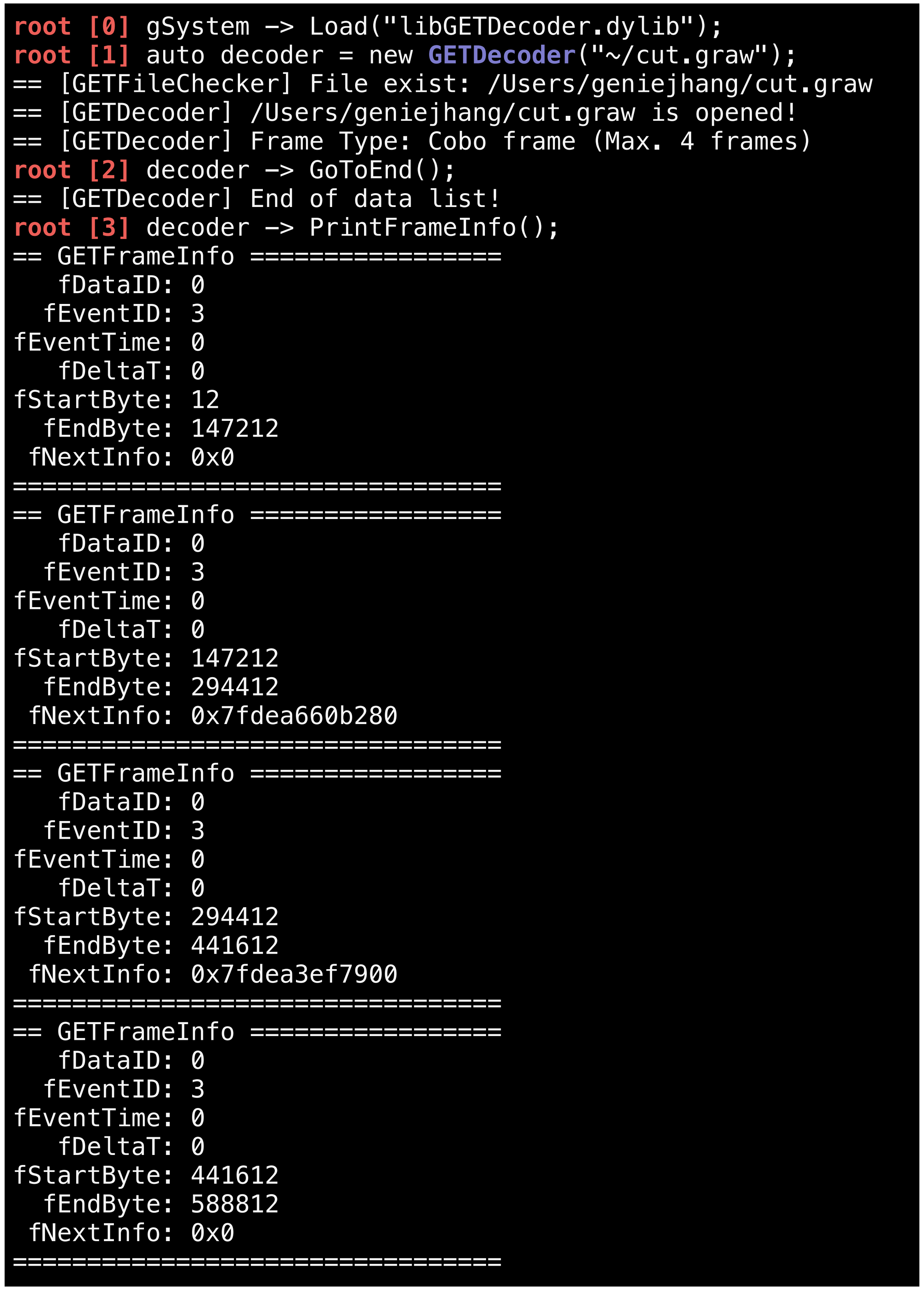
- Creating a new file having the frames cut out under some criteria from the original data file.
- The data structure remains intact. The resulting file consists of header + cut out frames.
- Usage
    1. If the data contains many empty events or events not satisfying the specific criteria, skim out those to make the data smaller.
    2. For the test purpose, only a few events are needed.

Frame cutter - example

```
root [0] gSystem -> Load("libGETDecoder.dylib");
root [1] auto decoder = new GETDecoder("~/data.graw");
== [GETFileChecker] File exist: /Users/geniejhang/data.graw
== [GETDecoder] /Users/geniejhang/data.graw is opened!
== [GETDecoder] Frame Type: Cobo frame (Max. 4 frames)
root [2] decoder -> SetWriteFile(“/Users/geniejhang/cut.graw”);
== [GETFileChecker] File does not exist: /Users/geniejhang/cut.graw
== [GETDecoder] Topology frame is written!
root [3] auto frame = decoder -> GetCoboFrame();
root [4] frame = decoder -> GetCoboFrame();
root [5] frame = decoder -> GetCoboFrame();
root [6] decoder -> WriteFrame();
```

Bool_t SetWriteFile(TString filename, Bool_t overwrite = kFALSE)

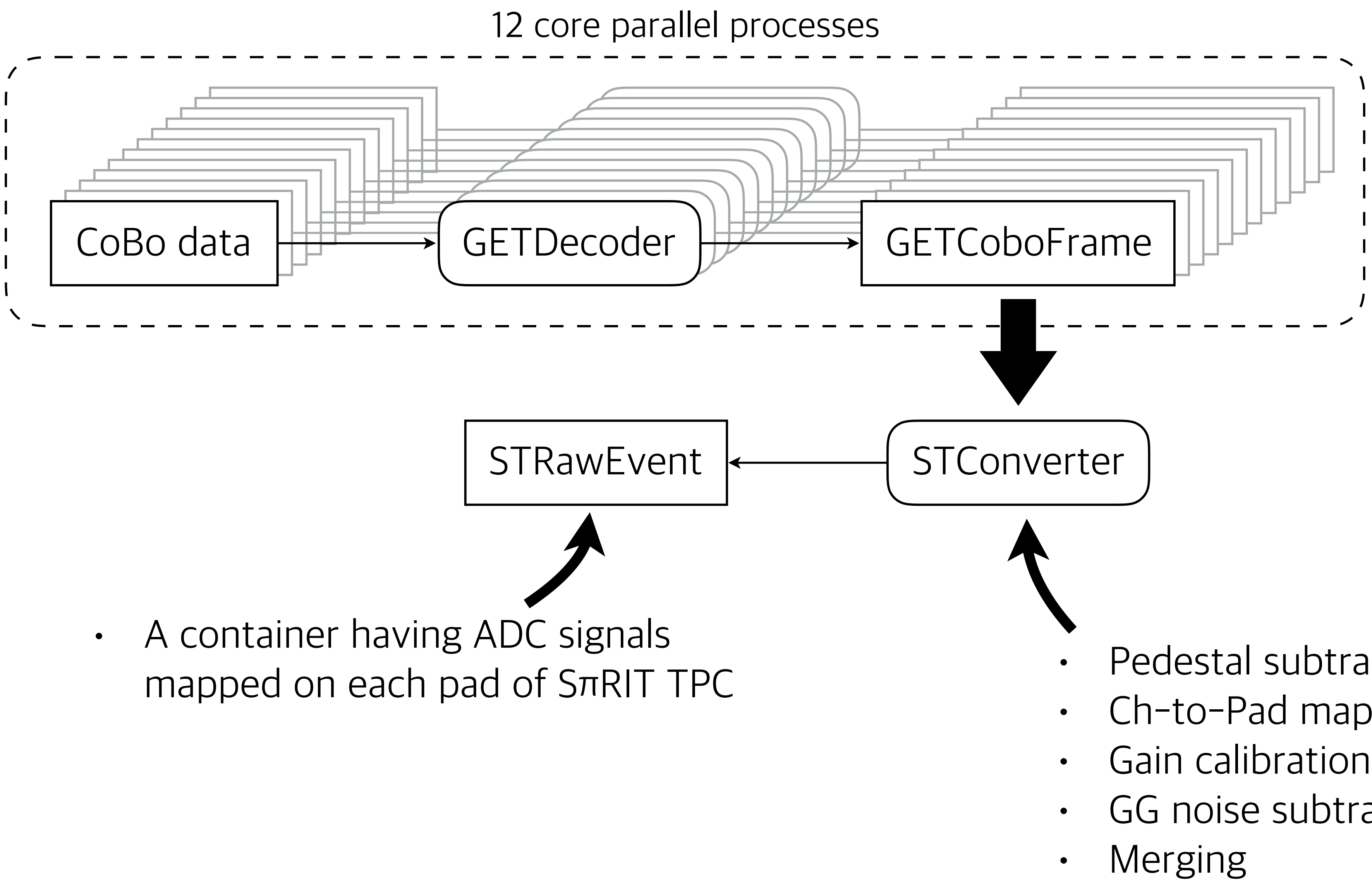
Frame cutter - example



```
root [0] gSystem -> Load("libGETDecoder.dylib");
root [1] auto decoder = new GETDecoder("~/cut.graw");
== [GETFileChecker] File exist: /Users/geniejhang/cut.graw
== [GETDecoder] /Users/geniejhang/cut.graw is opened!
== [GETDecoder] Frame Type: Cobo frame (Max. 4 frames)
root [2] decoder -> GoToEnd();
== [GETDecoder] End of data list!
root [3] decoder -> PrintFrameInfo();
== GETFrameInfo =================
   fDataID: 0
  fEventID: 3
fEventTime: 0
   fDeltaT: 0
fStartByte: 12
  fEndByte: 147212
 fNextInfo: 0x0
=================================
== GETFrameInfo =================
   fDataID: 0
  fEventID: 3
fEventTime: 0
   fDeltaT: 0
fStartByte: 147212
  fEndByte: 294412
 fNextInfo: 0x7fdea660b280
=================================
== GETFrameInfo =================
   fDataID: 0
  fEventID: 3
fEventTime: 0
   fDeltaT: 0
fStartByte: 294412
  fEndByte: 441612
 fNextInfo: 0x7fdea3ef7900
=================================
== GETFrameInfo =================
   fDataID: 0
  fEventID: 3
fEventTime: 0
   fDeltaT: 0
fStartByte: 441612
  fEndByte: 588812
 fNextInfo: 0x0
=================================
```

SpiRITROOT case

- SπRIT TPC uses 48 AsAd boards connected to 12 CoBo boards.
- 1 CoBo data is generated by 1 CoBo board, i.e. 12 CoBo data for each run in total.



Summary

- GETDecoder provides a way to analyze data quantitatively with less efforts.
- GETDecoder especially useful for those who are about to start to test their detectors with GET electronics.
- Because the presentation explains everything you need to unpack the binary data, you can use it right away.
- So, please download and use GETDecoder from GitHub:
  - https://github.com/geniejhang/GETDecoder