

# ***Status of EVIO / LCIO integration***

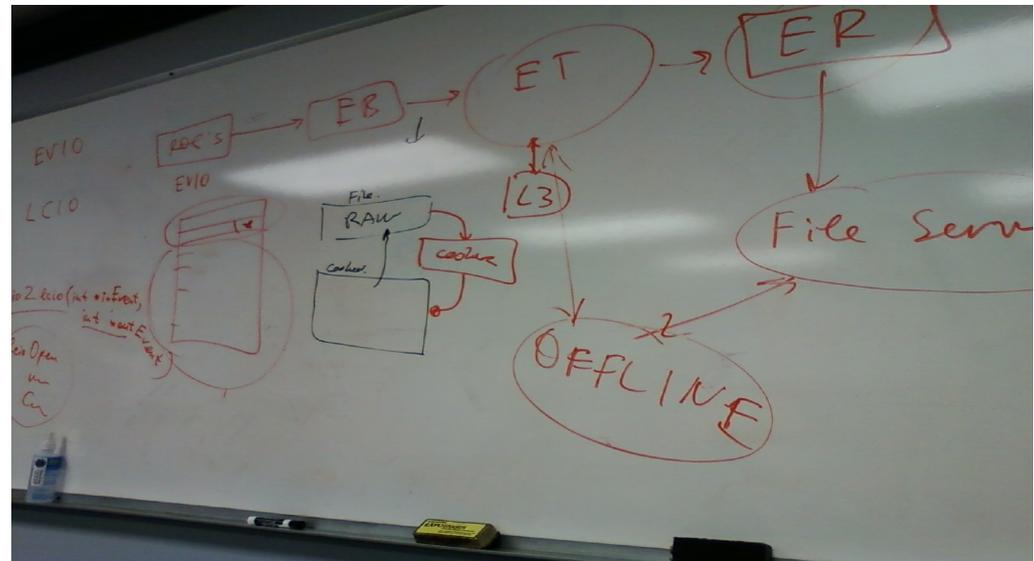


For the 17 Oct 2011  
HPS Software Mtg.  
@ JLAB

By  
Homer NEAL (SLAC)

# EVIO/LCIO Interface

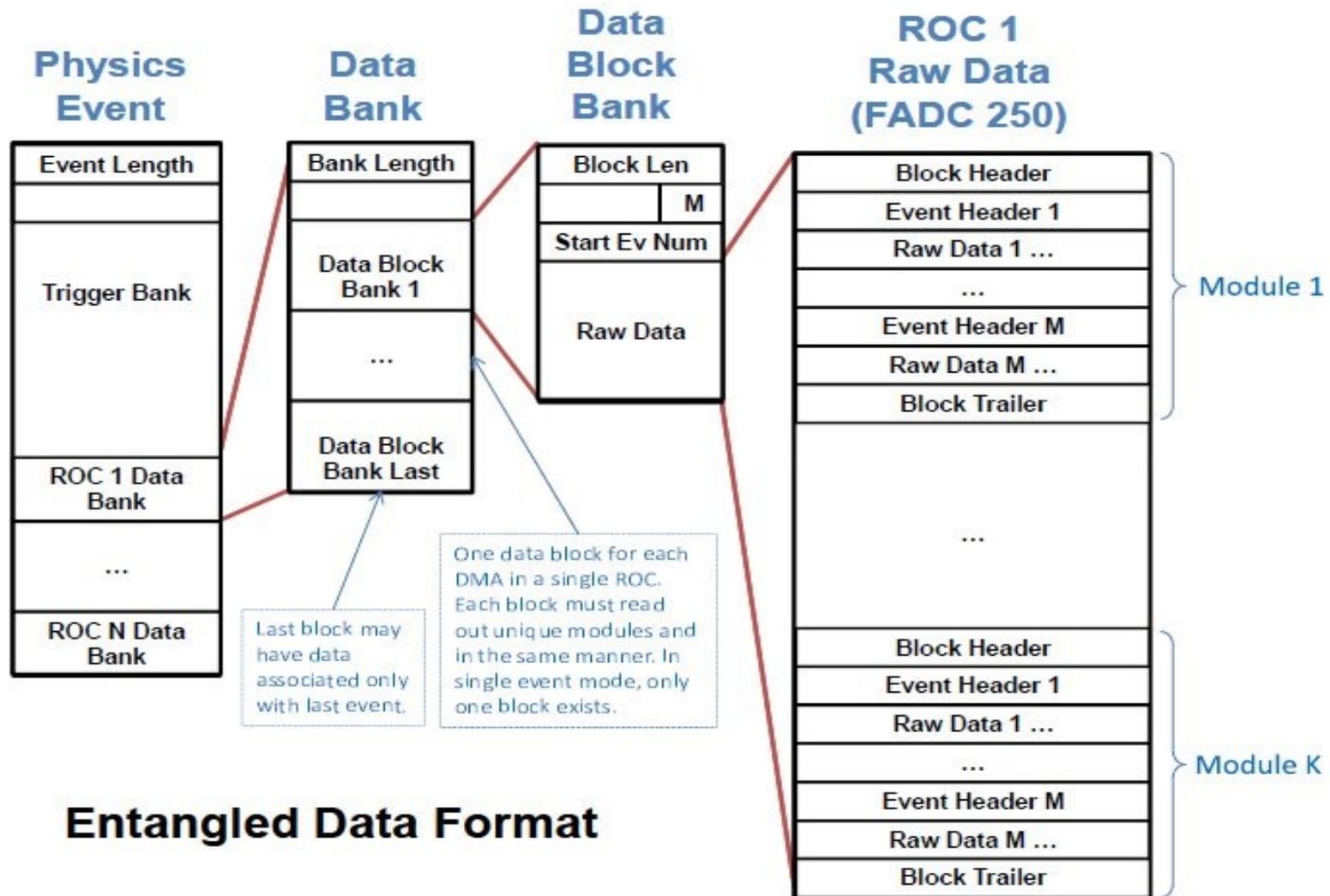
- Raw data will be presented in EVIO format
  - EVIO is the Hall B online existing and trusted format
- Reconstruction will use LCIO format
  - Format used by LCSIM
  - Random access



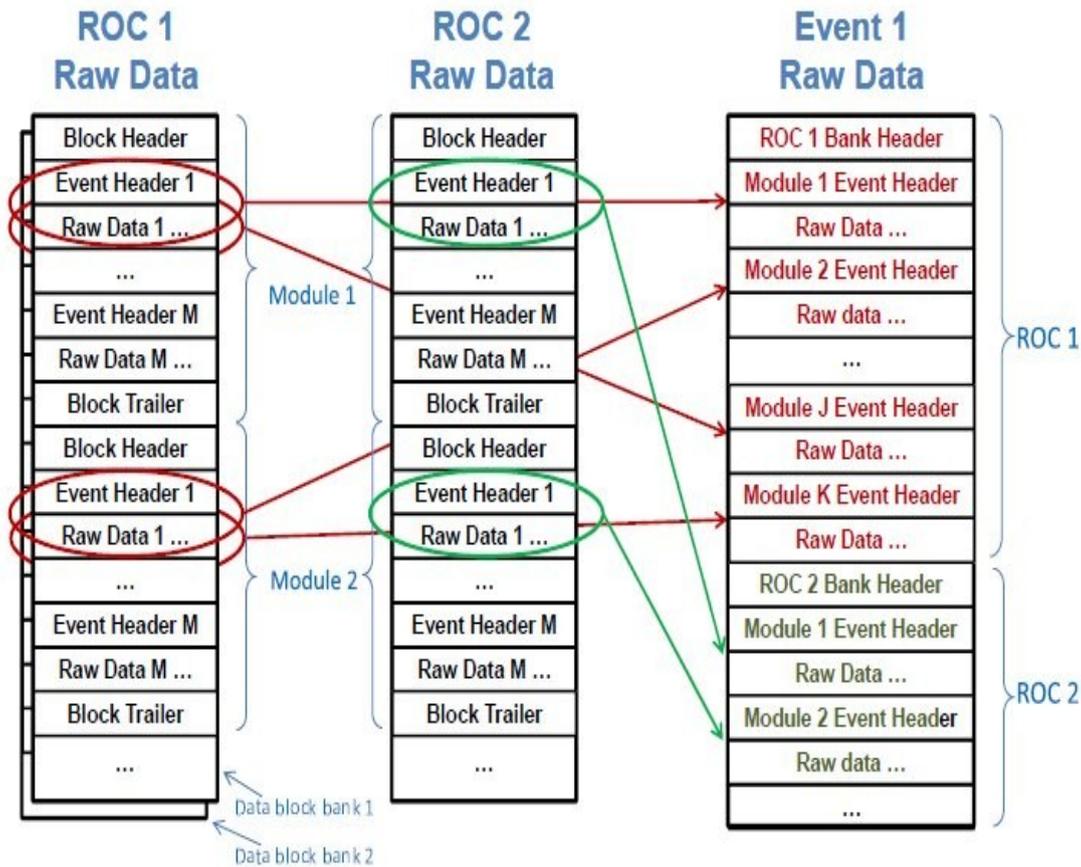
# Conversion – Need to know

- Are the data from all subsystems from a given trigger presented as a single event?
- How to access specific data elements from the EVIO blocks?
- What is contained in the headers?
- metadata

# Entangled Data Format



# Disentangled Data Format



**Entangled To Disentangled FADC 250 Raw Data**

Window Raw Data (4) – raw ADC data samples for the trigger window. The first word identifies the channel number and window width. Multiple continuation words contain two samples each. The earlier sample is stored in the most significant half of the continuation word. Strict time ordering of the samples is maintained in the order of the continuation words. A sample not valid flag may be set for any sample; for example, the last reported sample is tagged as not valid when the window consists of an odd number of samples.

Word 1:

(31) = 1

(30 – 27) = 4

(26 – 23) = channel number (0 – 15)

(22 – 12) = reserved (read as 0)

(11 – 0) = window width (in number of samples)

Words 2 - N:

(31) = 0

(30) = reserved (read as 0)

(29) = sample x not valid

(28 – 16) = ADC sample x (includes overflow bit)

(15 – 14) = reserved (read as 0)

(13) = sample x + 1 not valid

(12 – 0) = ADC sample x + 1 (includes overflow bit)

Window Sum (5) – sum of the raw data samples for the trigger window. Pedestal subtraction may be included.

(31) = 1

(30 – 27) = 5

(26 – 23) = channel number (0 – 15)

(22) = window sum overflow flag

(21 – 0) = window raw data sum

# An Event

## FADC 250

Data Type Values	
0 – block header	7 – pulse integral
1 – block trailer	8 – pulse time
2 – event header	9 – streaming raw data
3 – trigger time	10 – 12 user defined
4 – window raw data	13 – event trailer (debug only)
5 – window sum	14 – data not valid (empty module)
6 – pulse raw data	15 – filler (non-data) word

### Block Header Word Format

Bits	Value	Comment
31	1	This is a type defining word
30 – 27	0	Data type = block header
26 – 22	Slot ID	Set by VME64 backplane
21 – 14	Event #	Number of events in block
13 – 12	Module Type	0=FADC250, etc.
11 – 0	Event block #	Used to align block when building events

### General Data Word Format

31 <sup>st</sup> bit	Bits	Usage
1	30 - 27	4-bit data type (see chart)
1	26 - 0	Data type dependent data payload
0	30 - 0	Data payload using last defined data type

### Block Trailer Word Format

Bits	Value	Comment
31	1	This is a type defining word
30 – 27	1	Data type = block trailer
26 – 22	Slot ID	Set by VME64 backplane
21 – 0	Total # of words in block of events	Number of 32 bit words in block

### Event Header Word Format

Bits	Value	Comment
31	1	This is a type defining word
30 – 27	2	Data type = event header
26 – 22	Slot ID	Set by VME64 backplane
21 – 20	Module type	0=FADC250, etc.
19 – 0	Trigger number	ADC processing chip #

# Examples from Class12 Event Display

```
import cnuphys.jevio.BaseStructureHeader;
import cnuphys.jevio.EventParser;
import cnuphys.jevio.EvioEvent;
import cnuphys.jevio.IEvioListener;
import cnuphys.jevio.IEvioStructure;

/**
 * This is the manager for CLAS specific events. This is where we
 * put the data into convenient arrays.
 * @author heddle
 *
 */
public class EventManager implements IEvioListener {

    //small number check
    private static final double TINY = 1.0e-10;

    /**
     * Maximum number of events we will allow for
     * accumulation
     */
    public static final int MAXACCUMULATIONCOUNT =
10000;

    // singleton
    private static EventManager instance;

    // flag that set set to <code>>true</code> if we are
    accumulating events
    private boolean accumulating = false;

    // list of accumulated events
    private ArrayList<EvioEvent> _accumulatedEvents;
```

```
/**
 * Got a structure from the event source. This is where we look for
 * structures of interest and put
 * them in conveniently accessible arrays.
 * @param evioEvent the actual event.
 * @param the structure received.
 */
@Override
public void gotStructure(EvioEvent evioEvent, IEvioStructure
structure) {

    //grab the structures I'm interested in
    BaseStructureHeader header = structure.getHeader();
    int tag = header.getTag();
    int num = header.getNumber();

    // tag 500 has a lot of DC data
    if (tag == 500) {
        switch (num) {

            case 1:
                _dcGEMCArrayEdep = structure.getDoubleData();
                break;

            case 2:
                _dcGEMCArrayGlobalX =
structure.getDoubleData();
                break;
```

# J[EVI]O Explorer

**Jevio Event Tree**

File View Event

event source /mydat/real/hps/cedExport/data/dvcs\_5\_500.ev event# 1

dictionary num events 148

**Array Data**

```
[01] -2.45804692559e+01
[02] -1.39236555145e+01
[03] 8.09282072794e+00
[04] 8.72249836936e+00
[05] 1.56852878456e+01
[06] 1.66134899844e+01
[07] 2.39065939704e+01
[08] 2.49223000092e+01
```

<Event> len (ints): 13565 tag: 1 num: 0

- BANK of INT32s len (ints): 2 tag: 1 num: 1 datalen (bytes): 4 <#children: 0>
- BANK of BANKs len (ints): 157 tag: 10 num: 0 datalen (bytes): 624 <#children: 1>
  - BANK of BANKs len (ints): 155 tag: 10 num: 200 datalen (bytes): 616 <#children: 7>
    - BANK of DOUBLE64s len (ints): 21 tag: 10 num: 10 datalen (bytes): 80 <#children: 0>
    - BANK of DOUBLE64s len (ints): 21 tag: 10 num: 20 datalen (bytes): 80 <#children: 0>
    - BANK of DOUBLE64s len (ints): 21 tag: 10 num: 30 datalen (bytes): 80 <#children: 0>
    - BANK of DOUBLE64s len (ints): 21 tag: 10 num: 40 datalen (bytes): 80 <#children: 0>
    - BANK of DOUBLE64s len (ints): 21 tag: 10 num: 50 datalen (bytes): 80 <#children: 0>
    - BANK of DOUBLE64s len (ints): 21 tag: 10 num: 60 datalen (bytes): 80 <#children: 0>
    - BANK of DOUBLE64s len (ints): 21 tag: 10 num: 70 datalen (bytes): 80 <#children: 0>
  - BANK of BANKs len (ints): 341 tag: 400 num: 0 datalen (bytes): 1360 <#children: 2>
    - BANK of BANKs len (ints): 31 tag: 400 num: 100 datalen (bytes): 120 <#children: 3>
      - BANK of INT32s len (ints): 9 tag: 400 num: 20 datalen (bytes): 32 <#children: 0>
      - BANK of INT32s len (ints): 9 tag: 400 num: 21 datalen (bytes): 32 <#children: 0>
      - BANK of INT32s len (ints): 9 tag: 400 num: 22 datalen (bytes): 32 <#children: 0>
    - BANK of BANKs len (ints): 307 tag: 400 num: 200 datalen (bytes): 1224 <#children: 17>
      - BANK of DOUBLE64s len (ints): 17 tag: 400 num: 1 datalen (bytes): 64 <#children: 0>
      - BANK of DOUBLE64s len (ints): 17 tag: 400 num: 2 datalen (bytes): 64 <#children: 0>
      - BANK of DOUBLE64s len (ints): 17 tag: 400 num: 3 datalen (bytes): 64 <#children: 0>
      - BANK of DOUBLE64s len (ints): 17 tag: 400 num: 4 datalen (bytes): 64 <#children: 0>
      - BANK of DOUBLE64s len (ints): 17 tag: 400 num: 5 datalen (bytes): 64 <#children: 0>
      - BANK of DOUBLE64s len (ints): 17 tag: 400 num: 6 datalen (bytes): 64 <#children: 0>
      - BANK of DOUBLE64s len (ints): 17 tag: 400 num: 7 datalen (bytes): 64 <#children: 0>
      - BANK of DOUBLE64s len (ints): 17 tag: 400 num: 8 datalen (bytes): 64 <#children: 0>
      - BANK of DOUBLE64s len (ints): 17 tag: 400 num: 9 datalen (bytes): 64 <#children: 0>
      - BANK of DOUBLE64s len (ints): 17 tag: 400 num: 10 datalen (bytes): 64 <#children: 0>
      - BANK of DOUBLE64s len (ints): 17 tag: 400 num: 11 datalen (bytes): 64 <#children: 0>

progress

structure	BANK	tag	400	length	68 bytes
data type	DOUBLE64	number	3	description	???

**Jevio Event Tree**

File View Event

event source /mydat/real/hps/cedExport/data/dvcs\_5\_500.ev event# 1

dictionary num events 148

**Array Data**

```
[01] 1
```

**Tree representation of the EVIO event**

<Event> len (ints): 13565 tag: 1 num: 0

- BANK of INT32s len (ints): 2 tag: 1 num: 1 datalen (bytes): 4 <#children: 0>
- BANK of BANKs len (ints): 157 tag: 10 num: 0 datalen (bytes): 624 <#children: 1>
  - BANK of BANKs len (ints): 155 tag: 10 num: 200 datalen (bytes): 616 <#children: 7>
    - BANK of DOUBLE64s len (ints): 21 tag: 10 num: 10 datalen (bytes): 80 <#children: 0>
    - BANK of DOUBLE64s len (ints): 21 tag: 10 num: 20 datalen (bytes): 80 <#children: 0>
    - BANK of DOUBLE64s len (ints): 21 tag: 10 num: 30 datalen (bytes): 80 <#children: 0>
    - BANK of DOUBLE64s len (ints): 21 tag: 10 num: 40 datalen (bytes): 80 <#children: 0>
    - BANK of DOUBLE64s len (ints): 21 tag: 10 num: 50 datalen (bytes): 80 <#children: 0>
    - BANK of DOUBLE64s len (ints): 21 tag: 10 num: 60 datalen (bytes): 80 <#children: 0>
    - BANK of DOUBLE64s len (ints): 21 tag: 10 num: 70 datalen (bytes): 80 <#children: 0>
  - BANK of BANKs len (ints): 341 tag: 400 num: 0 datalen (bytes): 1360 <#children: 2>
    - BANK of BANKs len (ints): 31 tag: 400 num: 100 datalen (bytes): 120 <#children: 3>
      - BANK of BANKs len (ints): 307 tag: 400 num: 200 datalen (bytes): 1224 <#children: 17>
        - BANK of BANKs len (ints): 146 tag: 50 num: 0 datalen (bytes): 580 <#children: 2>
        - BANK of BANKs len (ints): 8025 tag: 500 num: 0 datalen (bytes): 32096 <#children: 2>
        - BANK of BANKs len (ints): 4211 tag: 300 num: 0 datalen (bytes): 16840 <#children: 2>
        - BANK of BANKs len (ints): 171 tag: 60 num: 0 datalen (bytes): 680 <#children: 2>
        - BANK of BANKs len (ints): 251 tag: 70 num: 0 datalen (bytes): 1000 <#children: 2>
        - BANK of BANKs len (ints): 251 tag: 80 num: 0 datalen (bytes): 1000 <#children: 2>

progress

structure	BANK	tag	1	length	8 bytes
data type	INT32	number	1	description	???

# Summary

- Thanks to Sergey and Maurik we now have a general data description, example code for accessing evio event elements and sample evio data.
- Next step get an EVIO block available in LCIO
  - Then, demonstrate that we can access specific values in the blocks
- Develop set of access primitives for all elements and do consistency tests

# Index to Documentation

The screenshot shows a web browser window displaying a Confluence page. The address bar shows the URL `https://confluence.slac.stanford.edu/display/hpsg/Event+Building+Scheme`. The page title is "Event Building Scheme" and it was added by Homer Neal on Oct 11, 2011. The main content is a text block explaining the event building scheme, including details about data flow from DAQ components to the ET system, and the use of a translator to convert data into a more convenient format. It also provides examples of data formats like `I,3F,6S,7(I,F)` and `I,2F,NS`, and explains how the number of shorts in the end of data is taken from the data itself. The page concludes with "Regards, Sergey".

**Event Building Scheme**  
8 Added by Homer Neal, last edited by Homer Neal on Oct 11, 2011 (view change)

I attached event building scheme as it is presented by JLAB DAQ group at present time. It describes how data will travel through DAQ components up to ET (Event Transport) system. In the beginning of ET we can run some kind of translator to convert all data into more convenient format if needed.

So far our IO package (called EVIO) can handle banks with the same format (all shorts or all ints etc), but new version of EVIO will be able to have mixed-format banks described by user in FORTRAN style, like:  
`I,3F,6S,7(I,F)`  
and even have variable rows, for example format `I,2F,NS` means that the number of shorts in the end of data will be taken from data itself, so for given example you have to write data in following order:  
`125(int), 1.1(float), 2.2(float), 2(int), 22(short), 33(short)` where 2 is the number of following shorts.  
I also attached outdated EVIO manual, new one is not ready.  
...

Regards, Sergey

**eventbuilding.pptx**

The page also includes three diagrams illustrating data flow and formats:

- Entangled Data Format:** A diagram showing data flow from Physics (Calorimeter, Tracker, Muon, DAQ) through Data Bank and Data Block to RDC 1 Raw Data (FADC 256).
- Entangled To Disentangled FADC 256 Raw Data:** A diagram showing data flow from RDC 1 Raw Data through RDC 2 Raw Data to Event 1 Raw Data.
- FADC 256:** A table showing the structure of FADC 256 raw data, including General Data Word Format, Block Trailer Word Format, Block Header Word Format, and Event Header Word Format.