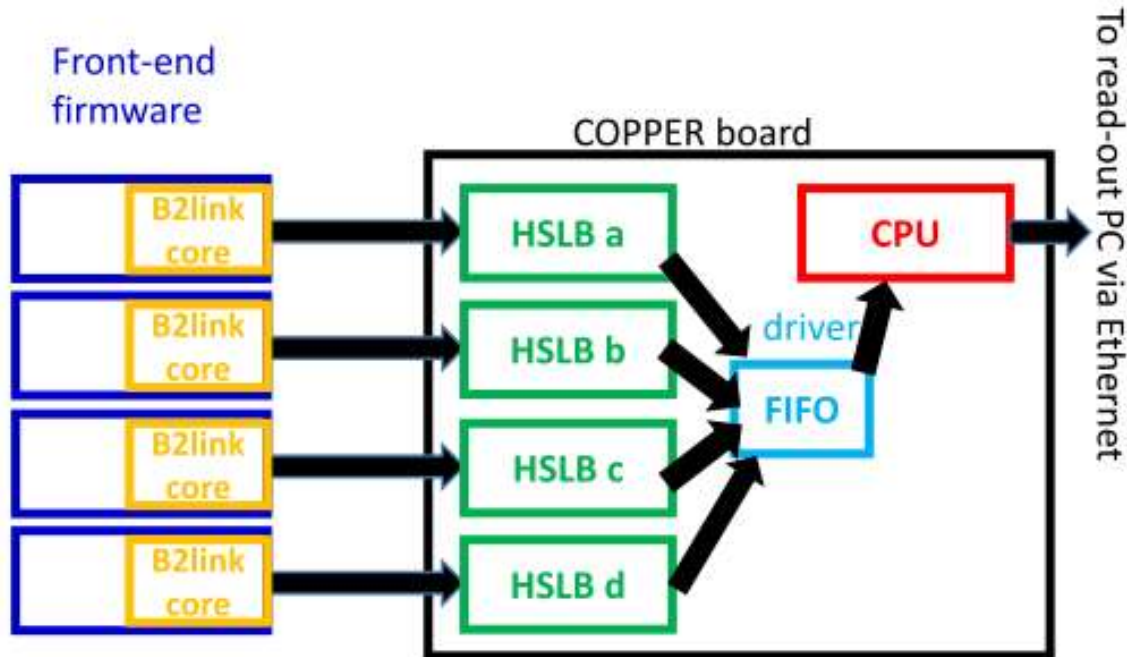# RawCOPPER data format

Apr. 21, 2015   (svn rev. 17269)

Satoru Yamada

1, Overview of RawCOPPER format (one data block from a COPPER board)

RawCOPPER header/trailer   -> See Sec. 2
COPPER header/trailer   -> See Sec.3
B2link(FEE+HSLB) header/trailer -> See Sec.4
Detector buffer   -> Untouched by DAQ

- **RawCOPPER header**
  - **COPPER header**
    - **B2link HSLB header (slot A FINNESSE)**
      - **B2link FEE header(slot A FINNESSE)**
        - **Data contents(Detector buffer) (slot A FINNESSE)**
      - **B2link FEE trailer (slot A FINNESSE)**
    - **B2link HSLB trailer (slot A FINNESSE)**
    - **B2link HSLB header (slot B FINNESSE)**
      - **B2link FEE header(slot B FINNESSE)**
        - **Data contents(Detector buffer) (slot B FINNESSE)**
      - **B2link FEE trailer (slot B FINNESSE)**
    - **B2link HSLB trailer (slot B FINNESSE)**
    - **B2link HSLB header (slot C FINNESSE)**
      - **B2link FEE header(slot C FINNESSE)**
        - **Data contents(Detector buffer) (slot C FINNESSE)**
      - **B2link FEE trailer (slot C FINNESSE)**
    - **B2link HSLB trailer (slot C FINNESSE)**
    - **B2link HSLB header (slot D FINNESSE)**
      - **B2link FEE header(slot D FINNESSE)**
        - **Data contents(Detector buffer) (slot D FINNESSE)**
      - **B2link FEE trailer (slot D FINNESSE)**
    - **B2link HSLB trailer (slot D FINNESSE)**
  - **COPPER trailer**
- **RawCOPPER trailer**

HSLB : High speed link board

1-1, Online header/trailer reduction

Before rev. 11234
(June, 2014)

COPPER

readoutPC

HLT input server

Old Raw header/trailer
COPPER header/trailer
B2link header/trailer

Old Raw header/trailer
COPPER header/trailer
B2link header/trailer

Binary data

RawCOPPER format

RawCOPPER format

RawCOPPER format

RawCOPPER format

Ropc disk

Ver. # = 0

Ver. # = 0

After the update

After rev. 11234
(June, 2014)

COPPER

readoutPC

HLT input server

New Raw header/trailer
COPPER header/trailer
B2link header/trailer

New Raw header/trailer
Reduced COPPER header/trailer
Reduced B2link header/trailer

Binary data

PreRawCOPPER format

PreRawCOPPER format

PostRawCOPPER format

PostRawCOPPER format

Ropc disk

PreRawCOPPER format : Ver # = 2 + 0x80 = 129

PostRawCOPPER format : Ver # = 2

3

- PreRawCOPPER format
  - If you store data by COPPER CPU, then output data will be in Pre(reduction)RawCOPPER format.
- PostRawCOPPER format
  - Store the data downstream from readout PC, the output data will be in Post(reduction)RawCOPPERFormat

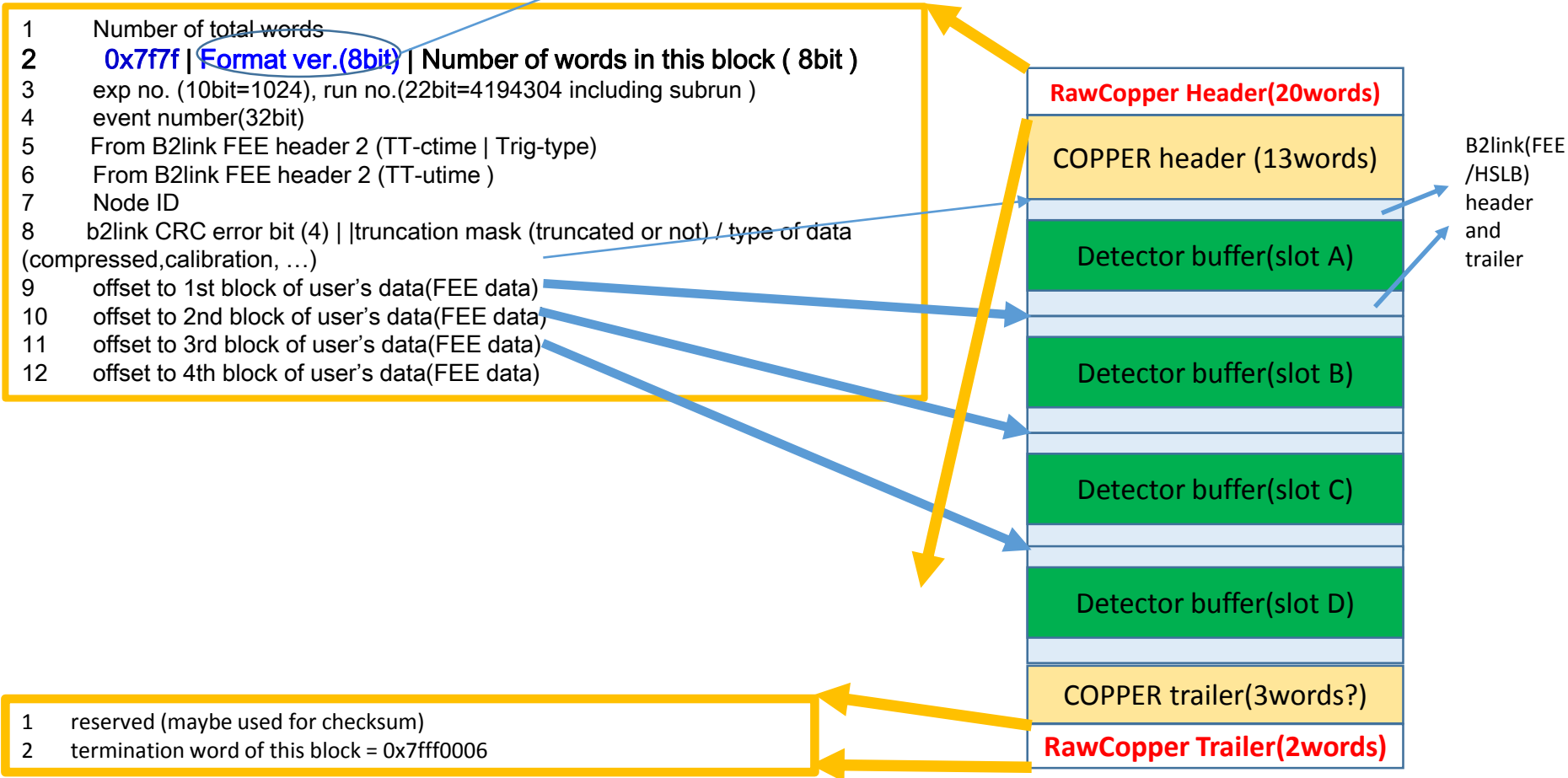# 2-1, "RawCOPPER header/trailer" format in PreRawCOPPER format (ver. 2+0x80)

* Same as ver.1

Use this version number to distinguish
Different data format.
Ver.0 : to 2014. June(including DESY test)
ver.1 : from June.2014
Ver.2 : from Apr. 2015 (rev.17269)

1	Number of total words
2	0x7f7f	Format ver.(8bit)	Number of words in this block (8bit)
3	exp no. (10bit=1024), run no.(22bit=4194304 including subrun)
4	event number(32bit)
5	From B2link FEE header 2 (TT-ctime | Trig-type)
6	From B2link FEE header 2 (TT-utime)
7	Node ID
8	b2link CRC error bit (4)		truncation mask (truncated or not) / type of data (compressed,calibration, …)
9	offset to 1st block of user's data(FEE data)
10	offset to 2nd block of user's data(FEE data)
11	offset to 3rd block of user's data(FEE data)
12	offset to 4th block of user's data(FEE data)

RawCopper Header(20words)

COPPER header (13words)

B2link(FEE /HSLB) header and trailer

Detector buffer(slot A)

Detector buffer(slot B)

Detector buffer(slot C)

Detector buffer(slot D)

COPPER trailer(3words?)

RawCopper Trailer(2words)

1	reserved (maybe used for checksum)
2	termination word of this block = 0x7fff0006

2-2, "RawCOPPER header" and trailer format in PostRawCOPPER format (ver.0x02)

Same as ver.1

Same as PreRawCOPPER format

2-3, tentative format of 32bit node ID

Format :

> (31-24) Detector ID : 8bit=256 : detector ID
> (9-0)   lower bits of COPPER ID : 10bit (1024)

Detector ID :

Detector ID (Defined in rawdata/dataobjects/include/RawCOPPERFormat.h)
- #define SVD_ID   0x01000000 // tentative
- #define CDC_ID   0x02000000 // tentative
- #define BPID_ID 0x03000000 // tentative
- #define EPID_ID 0x04000000 // tentative
- #define BECL_ID 0x05000000 // tentative
- #define EECL_ID 0x06000000 // tentative
- #define BKLM_ID 0x07000000 // tentative
- #define EKLM_ID 0x08000000 // tentative

- #define TRGDATA_ID 0x10000000 // tentative
  - #define CDCTRGDATA_ID   0x11000000 // tentative
  - #define ECLTRGDATA_ID   0x12000000 // tentative
  - #define TOPTRGDATA_ID   0x13000000 // tentative
  - #define KLMTRGDATA_ID   0x14000000 // tentative
  - #define GDLTRGDATA_ID   0x15000000 // tentative

Full COPPER ID :

> ➢ Except for TRG, full COPPER ID can be reconstructed by "(Detector ID >> 24) * 1000 + COPEPR ID(12bit) "
>   e.g. NodeID = 0x0600000a   -> COPPER ID = cpr6010
>   NodeID = 0x0100000a   -> COPPER ID = cpr1010
> ➢ For TRG data, full COPPER ID can be reconstructed by " 9000 + COPEPR ID(12bit) "
>
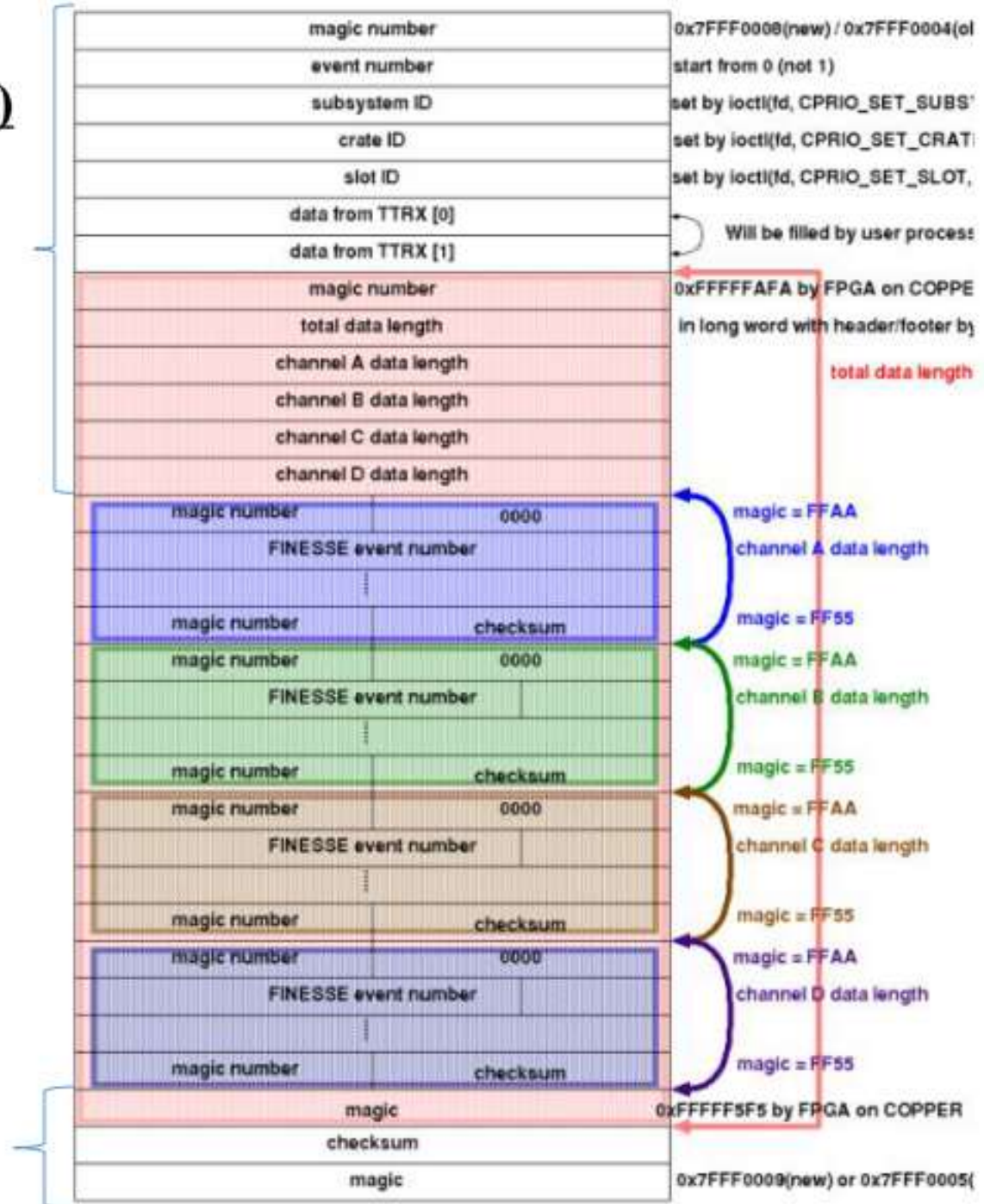> A label of COPPER ID will be attached on the front of a COPPER board

Node ID = "TTD " = 0x54544420 and "FTSW" are reserved for VME CPU and FTSW now.

3-1, COPPER header and trailer
in PreRawCOPPER format (ver. $2 + 0x80$)

* Same as ver.1

COPPER header

COPPER Trailer

magic number	0x7FFF0008(new) / 0x7FFF0004(ol
event number	start from 0 (not 1)
subsystem ID	set by ioctl(fd, CPRIO_SET_SUBS'
crate ID	set by ioctl(fd, CPRIO_SET_CRAT
slot ID	set by ioctl(fd, CPRIO_SET_SLOT,
data from TTRX [0]	
data from TTRX [1]	Will be filled by user proces:
magic number	0xFFFFFAFA by FPGA on COPPE
total data length	in long word with header/footer b:
channel A data length	total data length
channel B data length	
channel C data length	
channel D data length	
magic number 0000	magic = FFAA
FINESSE event number	channel A data length
magic number checksum	magic = FF55
magic number 0000	magic = FFAA
FINESSE event number	channel B data length
magic number checksum	magic = FF55
magic number 0000	magic = FFAA
FINESSE event number	channel C data length
magic number checksum	magic = FF55
magic number 0000	magic = FFAA
FINESSE event number	channel D data length
magic number checksum	magic = FF55
magic	0xFFFFF5F5 by FPGA on COPPER
checksum	
magic	0x7FFF0009(new) or 0x7FFF0005(

3-2, COPPER header and trailer in PostRawCOPPER format (ver. 0x02)

* Same as ver.1

No COPEPR header and trailer in Post
reduction rawcopper format.

4-1, B2link FEE header/Trailer, B2link HSLB header/Trailer in PreRawCOPPERFormat (ver. 0x02 + 0x80)

* Same as ver.1

From Nakao-san's Belle2link User guide (June 10, 2014):

You can download from 18 th B2GM indico page

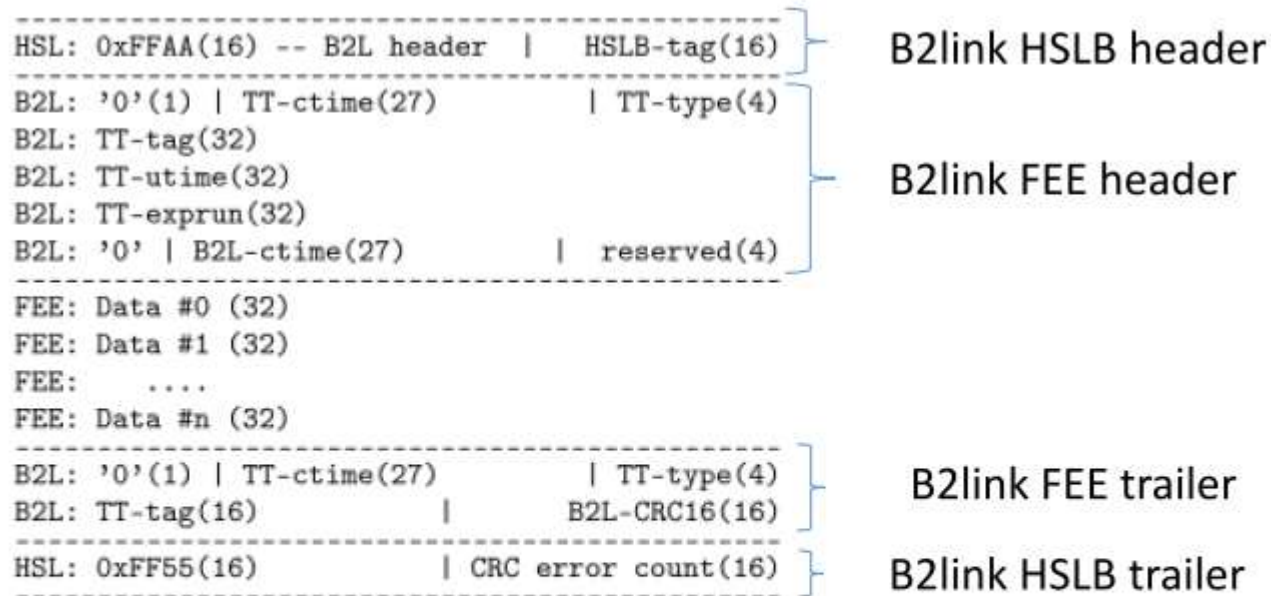http://kds.kek.jp/getFile.py/access?contribId=132&sessionId=28&resId=0&materialId=0&confI d=15329

```
HSL: 0xFFAA(16) -- B2L header  |   HSLB-tag(16)          B2link HSLB header
----------------------------------------------------
B2L: '0'(1) | TT-ctime(27)      | TT-type(4)
B2L: TT-tag(32)
B2L: TT-utime(32)                                        B2link FEE header
B2L: TT-exprun(32)
B2L: '0' | B2L-ctime(27)        |   reserved(4)
----------------------------------------------------
FEE: Data #0 (32)
FEE: Data #1 (32)
FEE:    ....
FEE: Data #n (32)
----------------------------------------------------
B2L: '0'(1) | TT-ctime(27)       | TT-type(4)
B2L: TT-tag(16)                  |   B2L-CRC16(16)        B2link FEE trailer
----------------------------------------------------
HSL: 0xFF55(16)                  | CRC error count(16)    B2link HSLB trailer
----------------------------------------------------
```
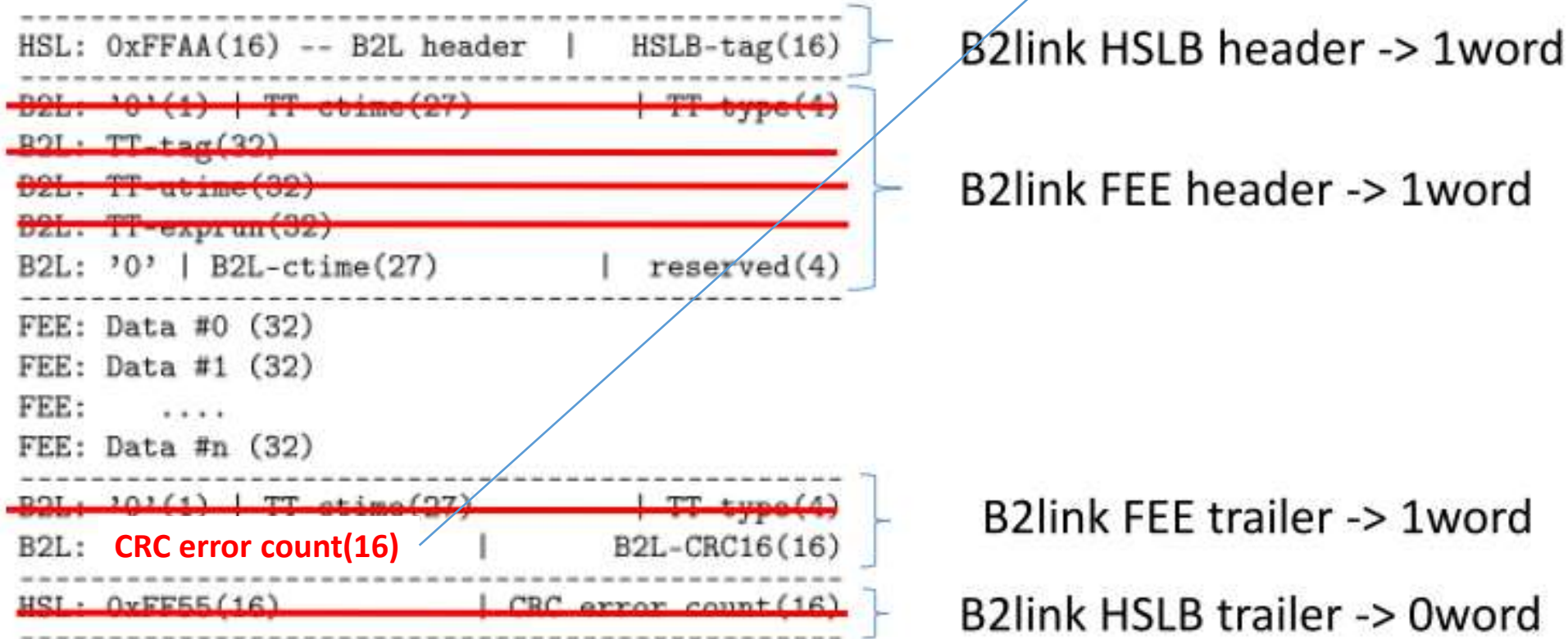
Figure 5: Data format as read out by the COPPER. The header and trailer words labelled with **HSL** are attached by HSLB, the words with **B2L** are attached by the belle2link component, and the words with FEE are those written into the belle2link component by the frontend firmware.

NOTICE :

To produce this format, the b2tt core used in

the FEE firmware should be the latest.

Please see Nakao-san's following e-mails :

[b2link_ml:0143] Belle2link version 0.01 - SVN update

And

[b2link_ml:0144] Re: Belle2link version 0.01 - SVN update .

4-2, B2link FEE header/Trailer, B2link HSLB header/Trailer in PostRawCOPPERFormat (ver. 0x02)

* Modified from ver.1

```
HSL: 0xFFAA(16) -- B2L header  |  HSLB-tag(16)
```
B2link HSLB header -> 1word

```
B2L: '0'(1) | TT-ctime(27)        | TT-type(4)
B2L: TT-tag(32)
B2L: TT-utime(32)
B2L: TT-exprun(32)
B2L: '0' | B2L-ctime(27)          |  reserved(4)
```
B2link FEE header -> 1word

```
FEE: Data #0 (32)
FEE: Data #1 (32)
FEE:     ....
FEE: Data #n (32)
```

```
B2L: '0'(1) | TT-ctime(27)        | TT-type(4)
B2L: CRC error count(16)   |    B2L-CRC16(16)
```
B2link FEE trailer -> 1word

```
HSL: 0xFF55(16)            |  CRC error count(16)
```
B2link HSLB trailer -> 0word

Figure 5: Data format as read out by the COPPER. The header and trailer words labelled with **HSL** are attached by HSLB, the words with **B2L** are attached by the belle2link component, and the words with FEE are those written into the belle2link component by the frontend firmware.

4-3, Older B2link header/trailer formats

At DESY test in January of 2014

From Nakao-san's B2GM slides:
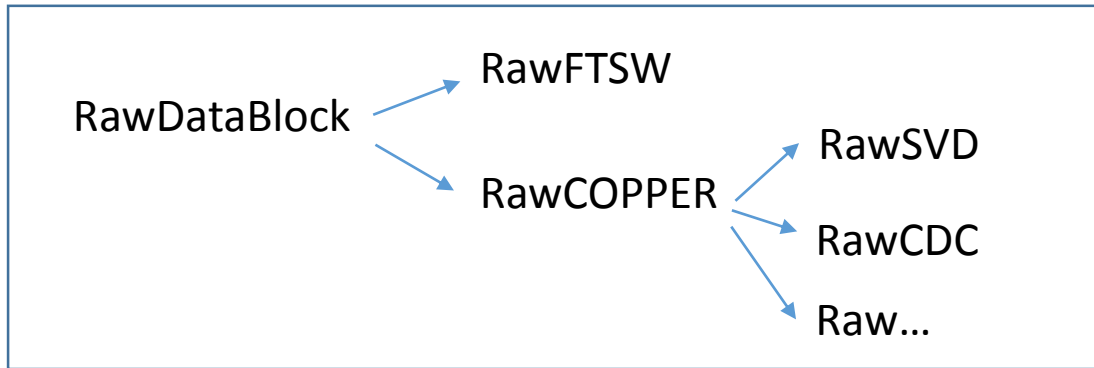http://kds.kek.jp/getFile.py/access?contribId=143&sessionId=38&resId=0&materialId=slides&confId=13911

Data format (Final?)

The format used at the telescope test

```
-------------------------------------------------------
HSL: 0xFFAA(16) --- B2L header  |       HSLB-tag(16)
-------------------------------------------------------
B2L: '0'(1) | TT-ctime(27)              | TT-type(4)
B2L: TT-tag(32)
B2L: TT-utime(32)
B2L: TT-exprun(32)
B2L: '0' | B2L-ctime(27)          |      debug-flag(4)
-------------------------------------------------------
FEE: Data #0 (32)
FEE: Data #1 (32)
FEE:    ....
FEE: Data #n (32)
-------------------------------------------------------
B2L: TT-tag(16)                   |  B2L-checksum(16)
-------------------------------------------------------
HSL: 0xFF55(16)                   | HSLB checksum(16)
-------------------------------------------------------
```

● tag (event number) and utime to be increased to 32-bit (done),
   HSLB-checksum, B2L-checksum to be added

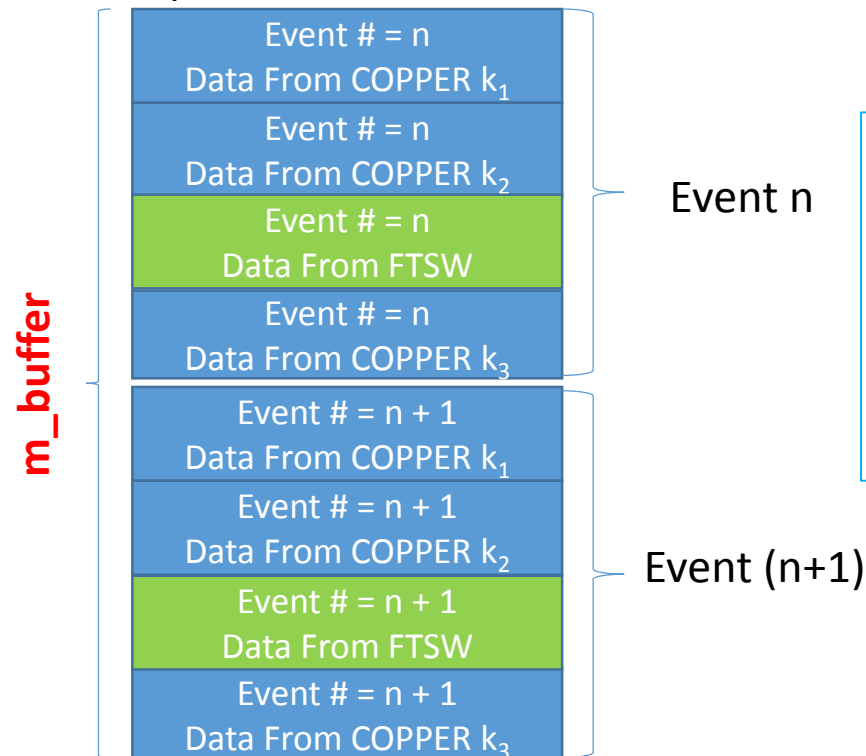# 5-1, RawDataBlock object (to handle Raw data from COPPER board)

RawDataBlock → RawFTSW

RawDataBlock → RawCOPPER → RawSVD

RawCOPPER → RawCDC

RawCOPPER → Raw…

Source code :

https://belle2.cc.kek.jp/svn/trunk/software/rawdata/dataobjects/

RawDataBlock{

    methods to access data;

    int m_num_nodes; // # of nodes

    int m_num_events;// # of events

    int* m_buffer; -> buffer for data

}

Example of data structure

m_buffer

| Event # = n Data From COPPER k_1 |
| Event # = n Data From COPPER k_2 |
| Event # = n Data From FTSW |
| Event # = n Data From COPPER k_3 |

Event n

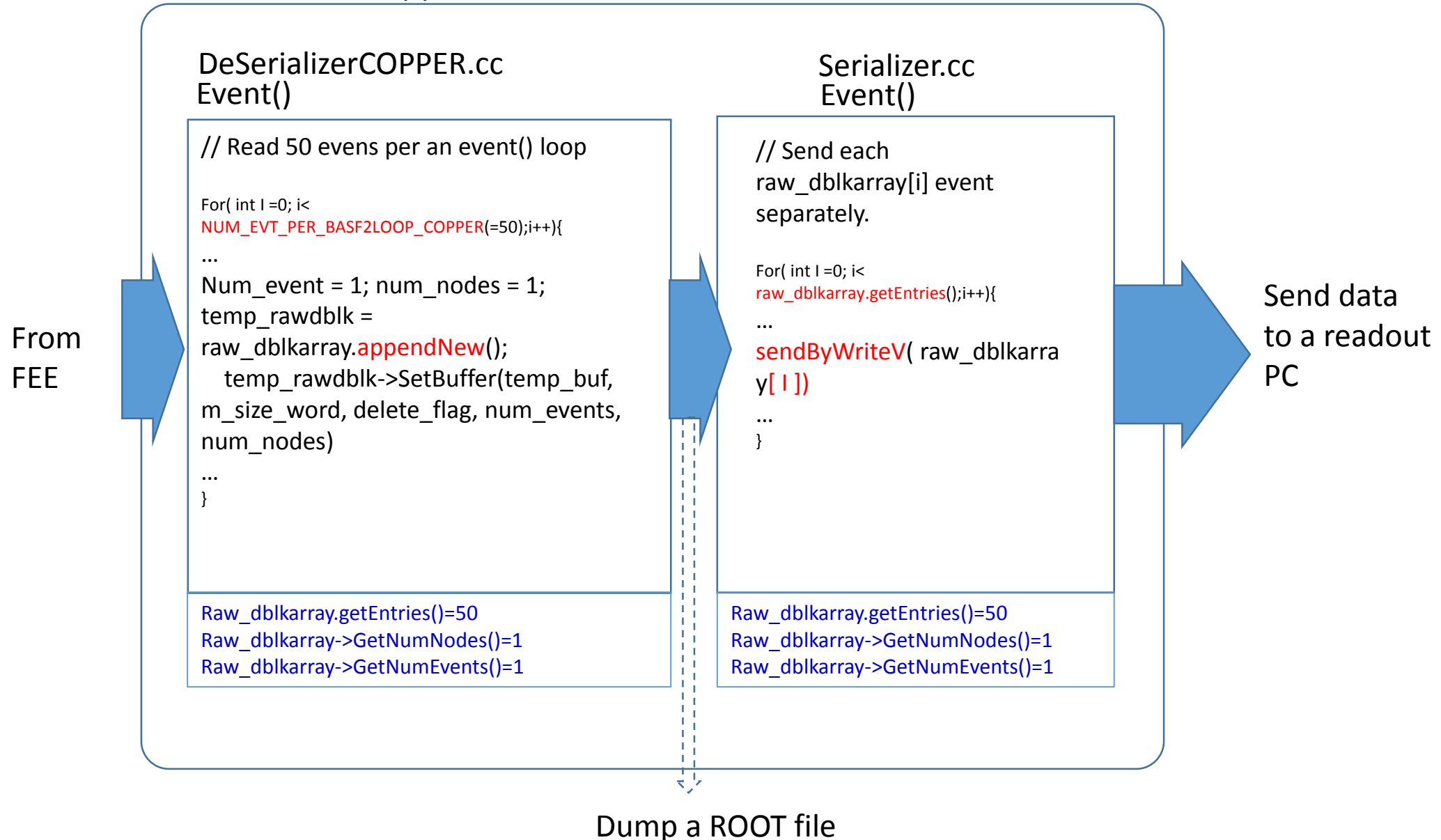| Event # = n + 1 Data From COPPER k_1 |
| Event # = n + 1 Data From COPPER k_2 |
| Event # = n + 1 Data From FTSW |
| Event # = n + 1 Data From COPPER k_3 |

Event (n+1)

In this example,

M_num_nodes = 4

M_num_events = 2.

# of data blocks = 4 * 2 = 8
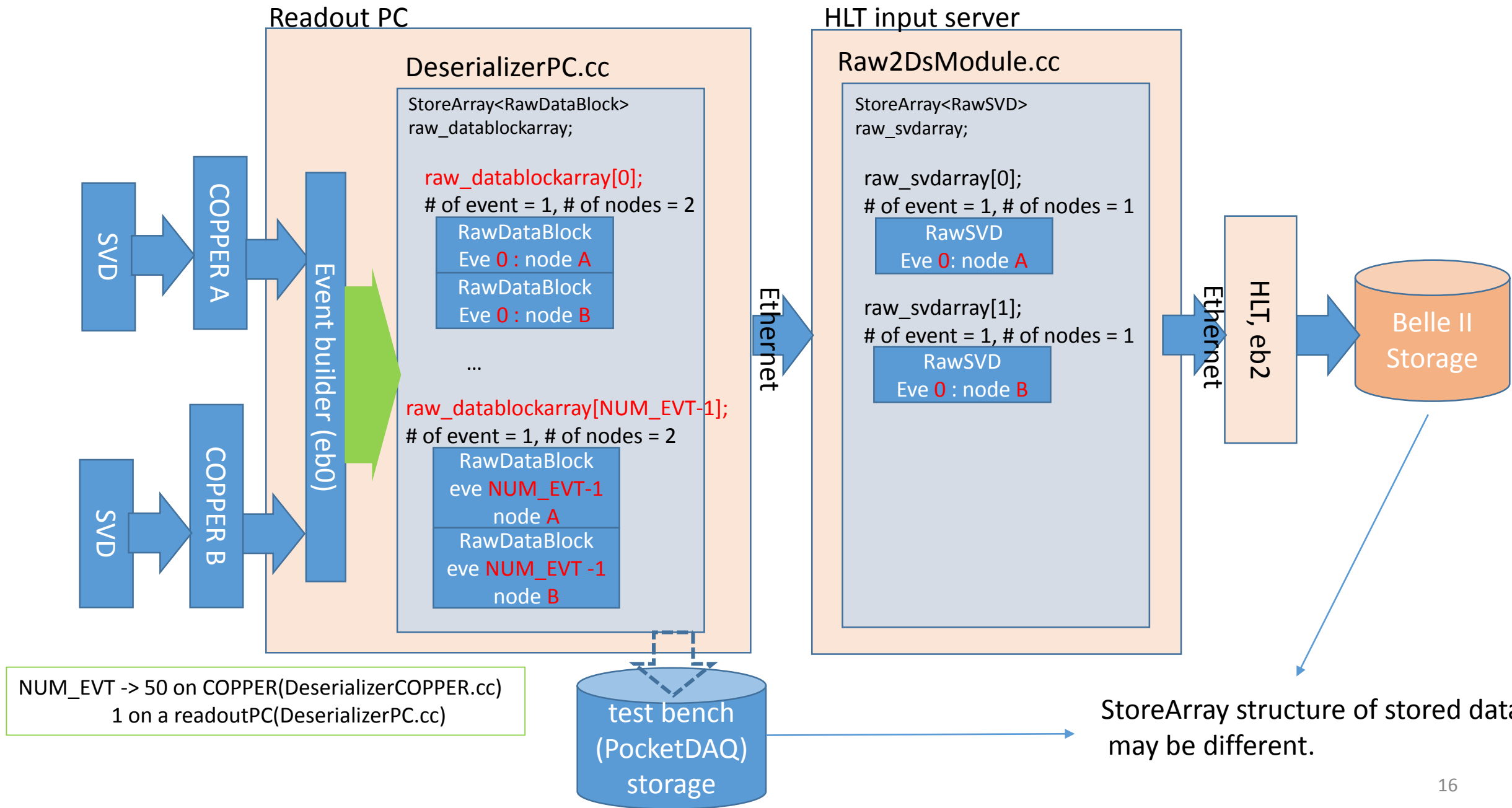
13

Example of Data handling on COPPER (as of rev.12453)

RecvSendCOPPER.py

DeSerializerCOPPER.cc
Event()

Serializer.cc
Event()

From
FEE

```
// Read 50 evens per an event() loop

For( int I =0; i<
NUM_EVT_PER_BASF2LOOP_COPPER(=50);i++){
...
Num_event = 1; num_nodes = 1;
temp_rawdblk =
raw_dblkarray.appendNew();
    temp_rawdblk->SetBuffer(temp_buf,
m_size_word, delete_flag, num_events,
num_nodes)
...
}
```

Raw_dblkarray.getEntries()=50
Raw_dblkarray->GetNumNodes()=1
Raw_dblkarray->GetNumEvents()=1

```
// Send each
raw_dblkarray[i] event
separately.

For( int I =0; i<
raw_dblkarray.getEntries();i++){
...
sendByWriteV( raw_dblkarra
y[ I ])
...
}
```

Raw_dblkarray.getEntries()=50
Raw_dblkarray->GetNumNodes()=1
Raw_dblkarray->GetNumEvents()=1

Send data
to a readout
PC

Dump a ROOT file

14

Example of Data handling on a readout PC (as of rev.12453)

RecvSendCOPPER.py

COPPER

COPPER

⋮

COPPER

Eb0(partial event building daemon)

DeserializerPC.cc
Event()

```
// Read 1 evens per an event() loop

For( int I =0; i< NUM_EVT_PER_BASF2LOOP_PC (=1);i++){
...
raw_datablkarray.appendNew();
...
}
```

Raw_dblkarray.getEntries()=1
Raw_dblkarray->GetNumNodes()= # of COPPERs
Raw_dblkarray->GetNumEvents()=1

Serializer.cc
Event()

```
For( int I =0; i<
raw_dblkarray.getEntries();i++){
...
sendByWriteV( raw_dblkarra
y[ I ])
...
}
```

Raw_dblkarray.getEntries()=1
Raw_dblkarray->GetNumNodes()=# of
COPPERs
Raw_dblkarray->GetNumEvents()=1

Send data to eb1tx (sender daemon)

Dump a ROOT file

Example : # of events and nodes in one RawDataBock(Detector) object and StoreArray :

StoreArray structure of stored data may be different.

NUM_EVT -> 50 on COPPER(DeserializerCOPPER.cc)
1 on a readoutPC(DeserializerPC.cc)

16

Example : how to store rawdata in RawDetector object by DAQ program

- ## Full Belle II DAQ
  - HLT(High Levele Trigger) receives serialized binary data from readout PCs and stores them in RawDetector Class. This RawDetector objects will be stored in storage.
    - Module : daq/rfarm/event/modules/src/Raw2DsModule.cc

- ## Pocket DAQ
  - DATA are stored as a RawDataBlock object
  - Example program can be used to convert RawDataBlock/COPPER to RawDetector objects
    - Module : daq/rawdata/modules/src/Convert2RawDet.cc

2-2, Rawdata Unpacker for new and old data formats

Data taken at the DESY beam test(old format) can be read with the latest rawdata package
-> by checking data ver. In header.

New RawCOPPER class

➢ No change in style of the member functions -> No effect on derived class
➢ Does not have a format information in itself
  ➢ Format class contains format information
    ➢ RawCOPPERformat.cc -> the latest format
      ➢ RawHeader.cc
    ➢ RawCOPPERformat_v0.cc -> an old format
      ➢ RawHeader_v0.cc
  ➢ Assgin a format class to m_access in CheckVersionSetBuffer()
  ➢ Use m_access to access buffer contents

```
inline int RawCOPPER::GetExpNo(int n)
{
 CheckVersionSetBuffer();
  return m_access->GetExpNo(n);
}

inline int RawCOPPER::GetRunNo(int n)
{
 CheckVersionSetBuffer();
  return m_access->GetRunNo(n);
}
```

Notice :
• RawCOPPER class supports both formats for a while (0.5-1 year after the format becomes stable?).
• In that case, the latest RawCOPPER class cannot be used to read old format
• Of course, you can use old rawdata repository to read old format
• For ver.0 format, use rawdata repository before 11228

18

Revision History of this document(1)

- Jan.5, 2014 rev. 8376 : Add definition of tentative subsysID format
- Dec. 16, 2013 rev.7974 :
  - Add B2linkFEE header format
  - Add comments about handling StoreArray when unpacking Raw*** data.
- Oct.21, 2013 :rev.7133
  - Add instruction about Rawdata unpacking program
- Oct. 18, 2013 :rev. 7095
  - 1 st draft
- Jun. 23, 2014 : rev. 11234
  - Online (header/trailer) reduction scheme on readout PC is introduced
    - RawHeader format is changed
    - COPPER header/trailer format is changed
  - Nakao-san updated B2LFEE/HSLB header/trailer format
    - See [b2link_ml:0144] Re: Belle2link version 0.01 - SVN update
- Aug. 23, 2014: rev. 12453
  - Add a description of how RawDataBLock objects are handled by the actual DAQ program.
- Sep. 26, 2014 : rev.13065
  - Add TRG ID definition (0x09000000)
- Oct. 24, 2014 : rev. 13460
  - Add a slide about # of events and nodes in one RawDataBock(Detector) object and StoreArray and how to store raw data in RawDetector object by DAQ program (p.16. p.17)
- Jan. 23, 2015: rev. 15030
  - Modify TRGDATA_ID and add ***TRGDATA_ID for trigger from sub-detectors
- Mar.3, 2015 : rev. 15988
  - Use "node ID" instead of subsystem ID.

Revision History of this document(2)

- Apr. 21, 2015 rev. **17269** : Description about a new data format (ver.2)
-

Backup

4-2, B2link FEE header/Trailer, B2link HSLB header/Trailer in PostRawCOPPERFormat (ver. 0x01)


```
HSL: 0xFFAA(16) -- B2L header  |  HSLB-tag(16)        B2link HSLB header -> 1word

B2L: '0'(1) | TT-ctime(27)              | TT-type(4)
B2L: TT-tag(32)
B2L: TT-utime(32)                                      B2link FEE header -> 1word
B2L: TT-exprun(32)
B2L: '0' | B2L-ctime(27)        |  reserved(4)

FEE: Data #0 (32)
FEE: Data #1 (32)
FEE:     ....
FEE: Data #n (32)

B2L: '0'(1) | TT-ctime(27)              | TT-type(4)
B2L: TT-tag(16)                |  B2L-CRC16(16)        B2link FEE trailer -> 1word

HSL: 0xFF55(16)                | CRC error count(16)   B2link HSLB trailer -> 0word
```

Figure 5: Data format as read out by the COPPER. The header and trailer words labelled with **HSL** are attached by HSLB, the words with **B2L** are attached by the belle2link component, and the words with FEE are those written into the belle2link component by the frontend firmware.

23