

RawCOPPER data format

June. 23, 2014 (svn rev. 11234)

Satoru Yamada

Revision History of this document

- 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
 - 1st 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

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

- 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

RawCOPPER header/trailer

-> See Sec. 2

COPPER header/trailer

-> See Sec.3

B2link(FEE+HSLB) header/trailer

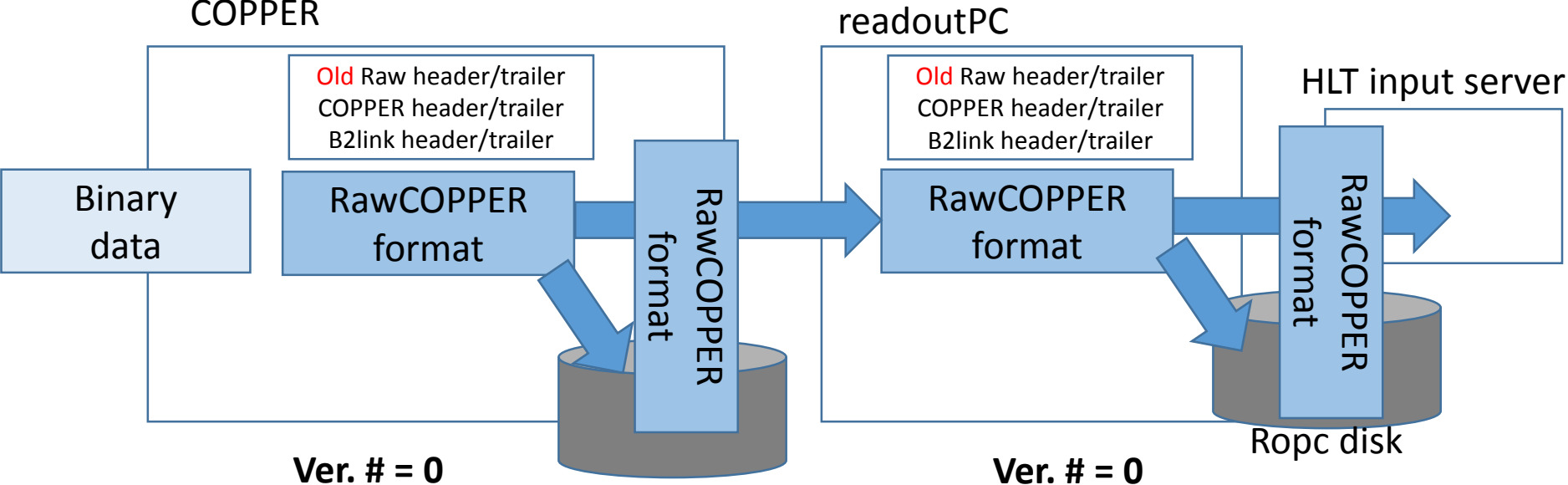
-> See Sec.4

Detector buffer

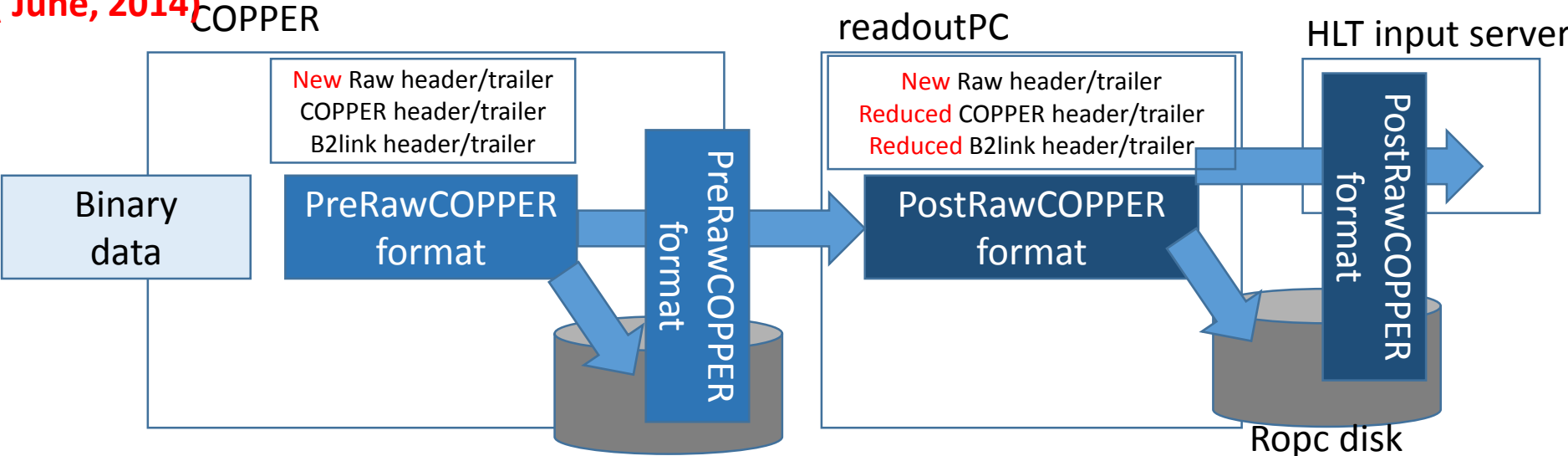
-> Untouched by DAQ

1-1, Online header/trailer reduction

Before rev. 11234(June, 2014)



After rev. 11234(June, 2014)



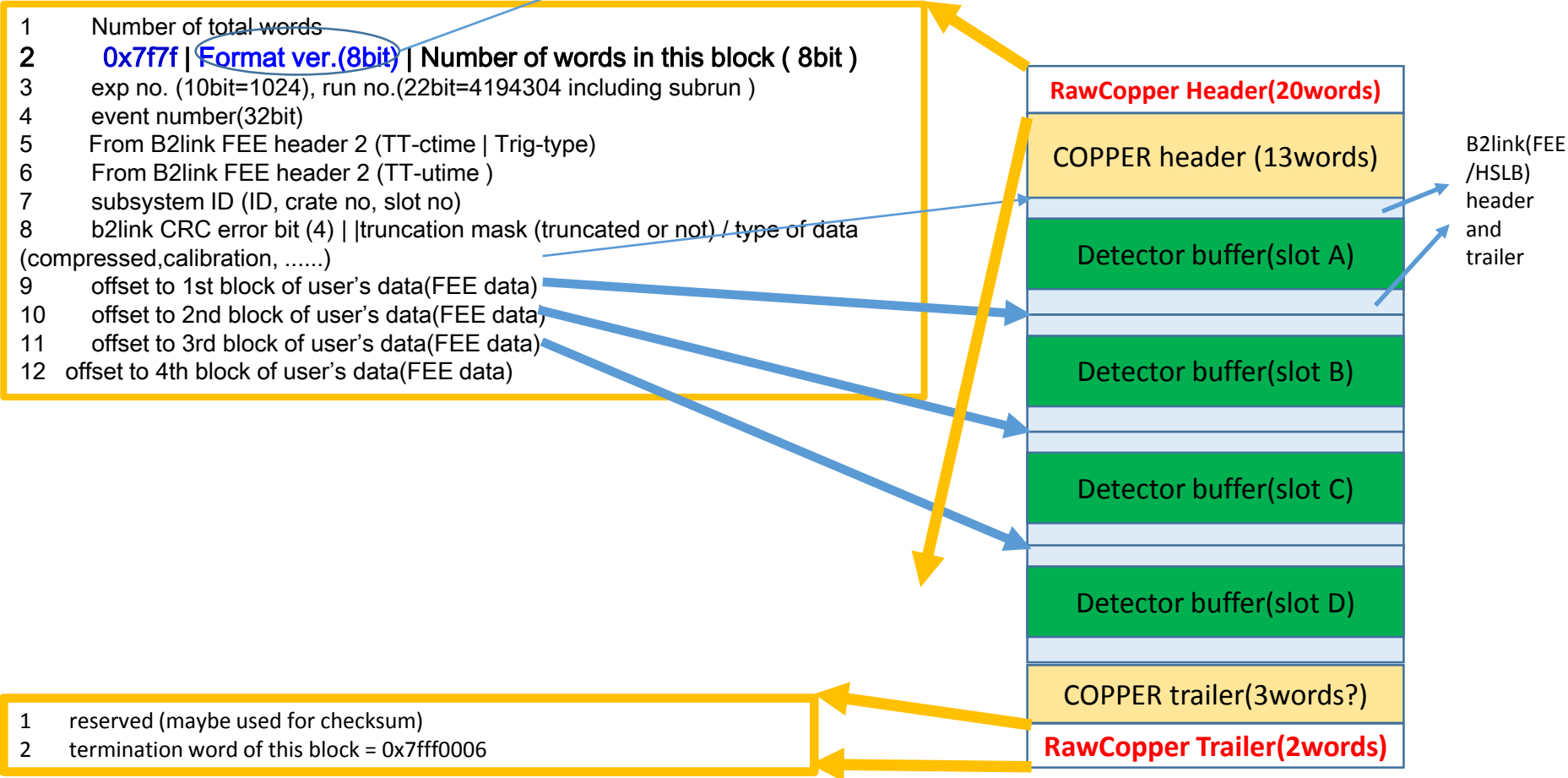
PreRawCOPPER format : Ver # = 1 + 0x80 = 129

PostRawCOPPER format : Ver # = 1

- 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. 1+0x80)

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



2-2, “**RawCOPPER header**” and trailer format in **PostRawCOPPER** format (ver.1)

Same as PreRawCOPPER format

2-3, 32bit Subsystem ID (A.K.A. node ID)

(31-24) Detector ID : 8bit=256 : detector & DAQ nodes
(23-17) CRATE ID : 7bit=128 :
(16-12) SLOT ID : 5bit=32 :
(11-0) N.A. : 12bit (4096) COPPER S/N?

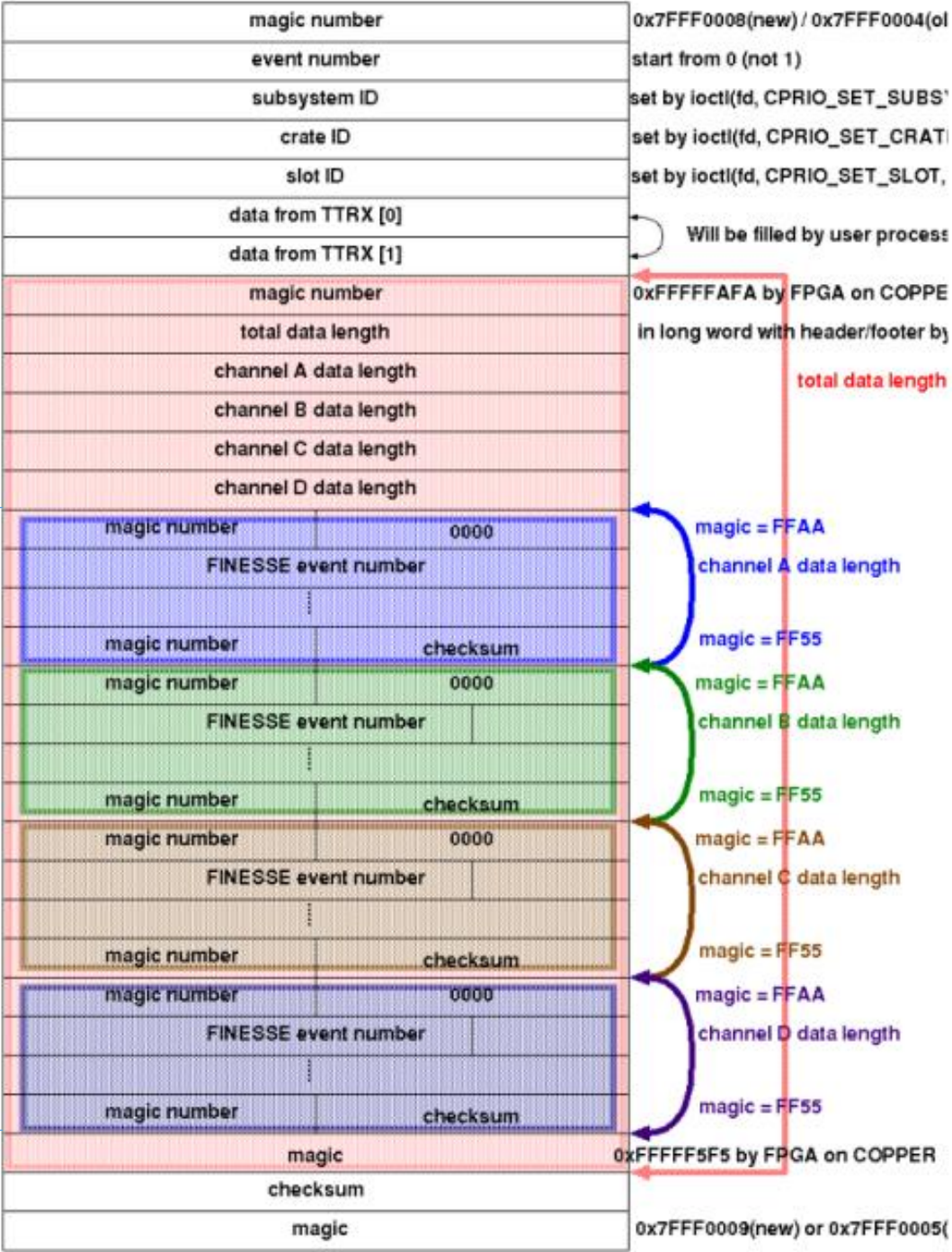
SubsystemID = “TTD ” = 0x54544420 and is reserved by FTSW ID now.

Detector ID (Defined in release/rawdata/dataobjects/include/RawCOPPER.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

3-1, COPPER header and trailer
in **PreRaw**COPPER format (ver. 1 + 0x80)

COPPER header



COPPER Trailer

3-2, COPPER header and trailer in **Post**RawCOPPER format (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. 1+0x80)

From Nakao-san's Belle2link User guide (June 10, 2014):
You can download from 18th B2GM indico page
<http://kds.kek.jp/getFile.py/access?contribId=132&sessionId=28&resId=0&materialId=0&confId=15329>

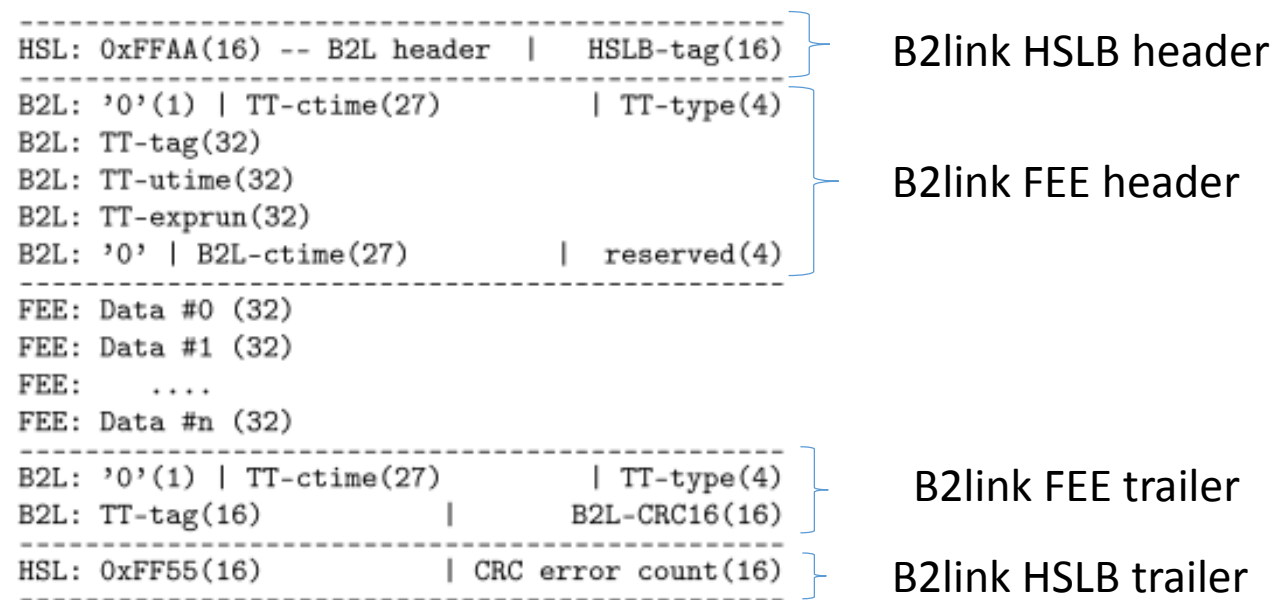


Figure 5: Data format as read out by the COPPER. The header and trailer words labelled with **HSL** are attached by HSLB, the words with **B2L** are attached by the belle2link component, and the words with FEE are those written into the belle2link component by the frontend firmware.

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. 1)

From Nakao-san's Belle2link User guide (June 10, 2014):
You can download from 18th B2GM indico page
<http://kds.kek.jp/getFile.py/access?contribId=132&sessionId=28&resId=0&materialId=0&confId=15329>

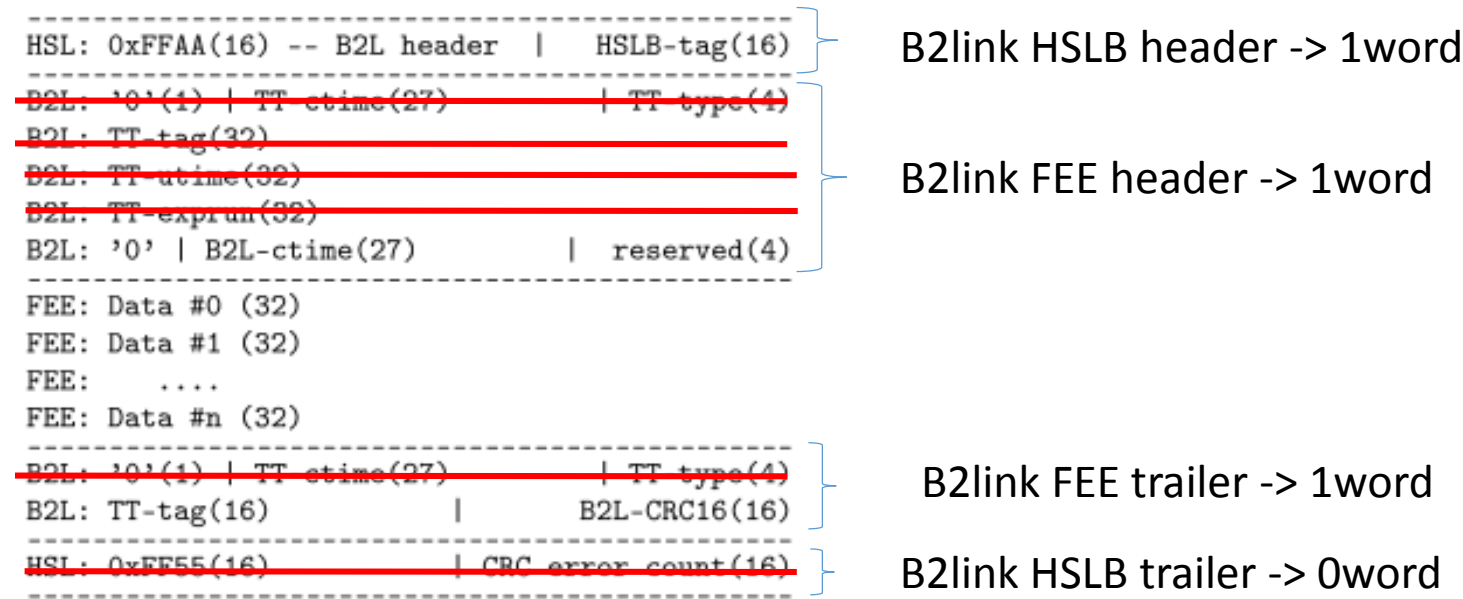


Figure 5: Data format as read out by the COPPER. The header and trailer words labelled with **HSL** are attached by HSLB, the words with **B2L** are attached by the belle2link component, and the words with FEE are those written into the belle2link component by the frontend firmware.

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:

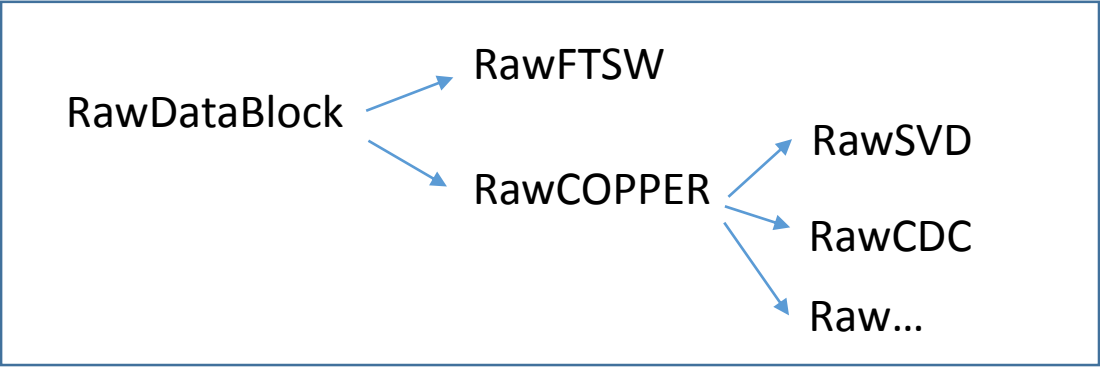
FEE: Data #n (32)

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)

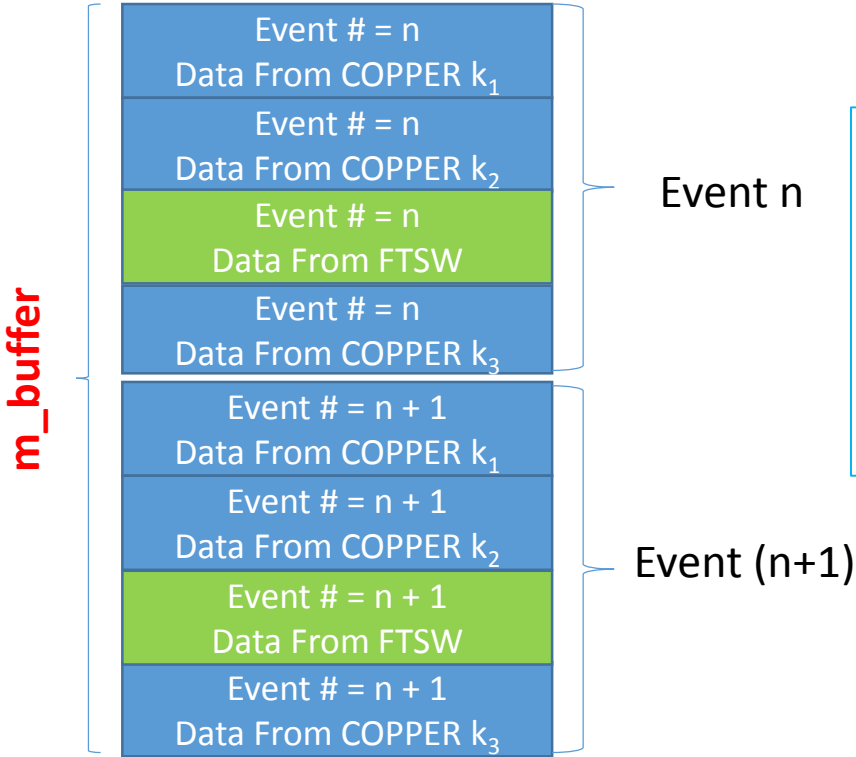


Source code :
<https://belle2.cc.kek.jp/svn/trunk/software/rawdata/dataobjects/>

```
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



In this example,
M_num_nodes = 4
M_num_events = 2.

of data blocks = 4 * 2 = 8

5-2, RawCOPPER class select proper “format class” depending on Pre/Post 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
 - Assign 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

6, Example : how to get information of RawCOPPER header

You can get event # info from RawCOPPER object like this;

```
StoreArray<RawCOPPER> raw_cprarray;
for (int i = 0; i < raw_cprarray.getEntries(); i++) {
    for ( int j = 0; j < raw_cprarray[ i ].GetNumEntries(); j++) {
//      Get Event number
        unsigned int event_no = raw_cprarray[ i ].GetEveNo( j );
//      Get RawCOPPER data block
        int* buf = raw_cprarray[ i ].GetBuffer( j );
//      See contents of a data block (from RawCOPPER header to RawCOPPER trailer)
        for( int k = 0; k < raw_cprarray[ i ].GetBlockNwords( j ); k++ ){
            printf(“%d¥n”, buf[ k ] );
        }
//      Get Detector Buffer (raw data from detector electronics)
        int* buf_slot_a = raw_cprarray[ i ].Get1stDetectorBuffer( j );
        int* buf_slot_b = raw_cprarray[ i ].Get2ndDetectorBuffer( j );
        int* buf_slot_c = raw_cprarray[ i ].Get3rdDetectorBuffer( j );
        int* buf_slot_d = raw_cprarray[ i ].Get4thDetectorBuffer( j );
        int* buf_slot[4]; for( int k = 0; k < 4;k++){ buf_slot[ k ] = raw_cprarray[ i ].GetDetectorBuffer(j,k) }
//      See contents of raw data from detector
        for( int k = 0; j < raw_cprarray[ i ].Get1stDetectorNwords( j ); k++ ){
            printf(“%d¥n”, buf_slot_a[ k ] );
        }
        for( int k = 0; j < raw_cprarray [ i ].Get2ndDetectorNwords( j ); k++ ){
            printf(“%d¥n”, buf_slot_b[ k ] );
        }
        .....
    }
}
```


Test program to read RawCOPPER(RawCDC) data

1, Get dummy data file (data from two CDC FEE boards connected to FINESSE A and C.)
login.cc.kek.jp : ~yamadas/rawdata/[root_output_RawCDC_rev7133.root](#)

2, See contents of the data

```
% cd ${BELLE2_LOCAL_DIR}/daq; svn update
```

```
% cd ${BELLE2_LOCAL_DIR}/daq/rawdata/examples/
```

```
% basf2 ReadStoreTemplate.py -i ./root\_output\_RawCDC\_rev7133.root | less
```

```
[INFO] Steering file: ReadStoreTemplate.py
```

```
>>> basf2 Python environment set
```

```
>>> Framework object created: fw
```



```
==== DataBlock(RawCDC) : Block # 0 : Event # 0 : node ID 0x00000000 : block size 224 bytes
```

```
== Detector Buffer(FINESSE A)
```

```
0x0094c13a 0x91000001
```

```
== Detector Buffer(FINESSE C)
```

```
0x0094c13a 0x91000001
```

```
==== DataBlock(RawCDC) : Block # 1 : Event # 1 : node ID 0x00000000 : block size 224 bytes
```

```
== Detector Buffer(FINESSE A)
```

```
0x0094c23f 0xf1000001
```

```
== Detector Buffer(FINESSE C)
```

```
0x0094c23f 0xf1000001
```

```
==== DataBlock(RawCDC) : Block # 2 : Event # 2 : node ID 0x00000000 : block size 224 bytes
```

```
== Detector Buffer(FINESSE A)
```

```
0x0094c30d 0x69000001
```

```
== Detector Buffer(FINESSE C)
```

```
0x0094c30d 0x69000001
```

```
....
```

In this data,

Detector buffer contains only 2words(=8bytes)
per/FINESSE/event.

Note that block # is a number used by DAQ software
for handling data and not related with **Event #**.