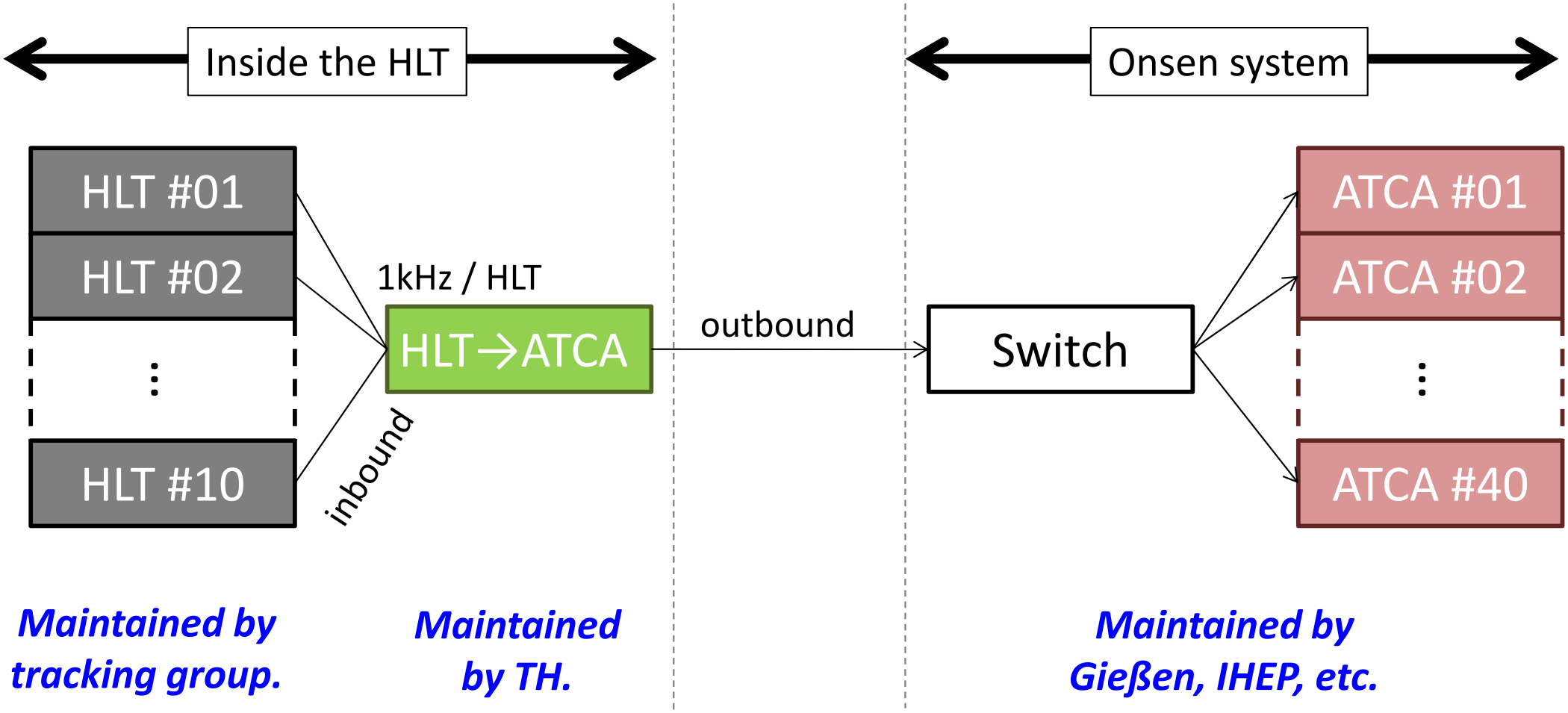
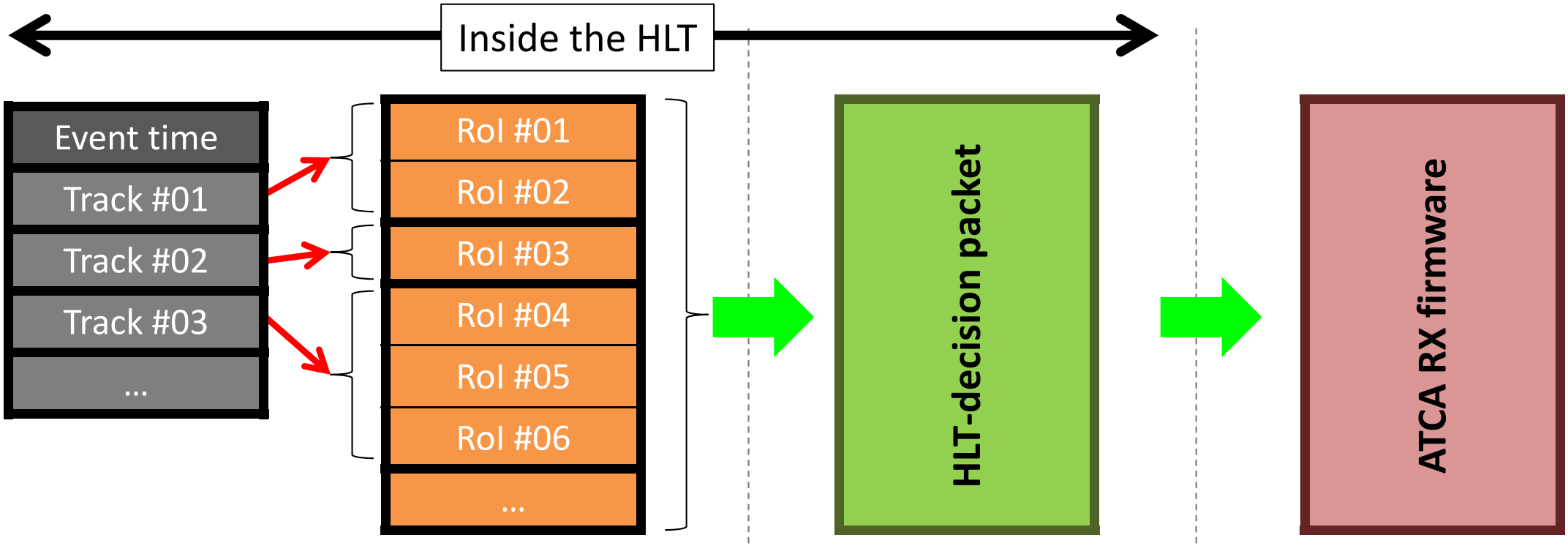
HLT → ATCA

Takeo Higuchi (University of Tokyo)

Sketch of HLT → ATCA (Hardware Link)



Sketch of HLT → ATCA (Software Link)



A basf2 module running on the HLT converts each track information in the event to the relevant RoI(s) on the PXD.

List of RoIs are packed into an HLT-decision packet by another basf2 module on the HLT.

Maintained by tracking group.

Maintained by TH.

Maintained by Gießen, IHEP, etc.

Boundary Conditions

- **PXD cell configuration**
  - (40 half-ladders) x (768 rows) x (250 cols)
- **HLT reduces L1 rate of ~ 30kHz to 10kHz.**
  - Latest assumption as of .
- **Track multiplicity including junk is ~200.**
  - This is a possible worst case anticipated by M.Heck-san.

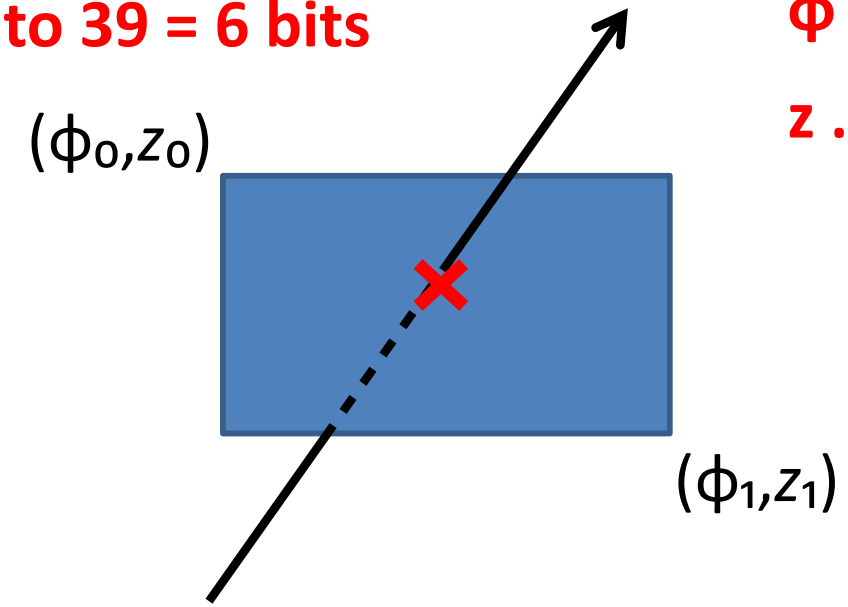
RoI

- **We have agreed that RoI will take a rectangular shape.**
  - Size of each RoI would depend on the helix error matrix.

Half ladder ID … 0 to 39 = 6 bits

ϕ … 0 to 249 = 8 bits

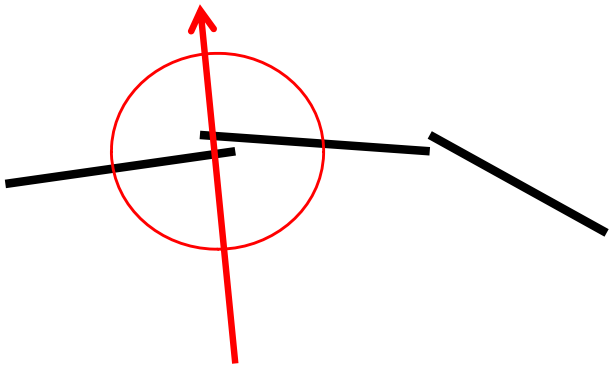
z … 0 to 767 = 10 bits



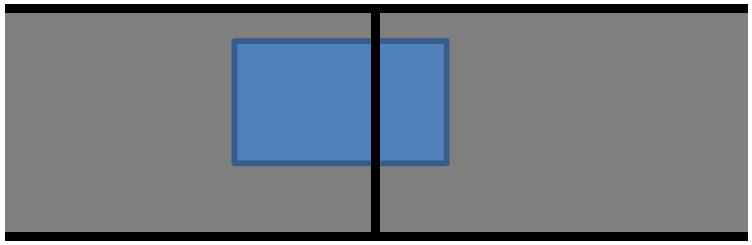
6 + 2x(8+10) = 42 bits per RoI → 64 bits per RoI if rounded to powers of 2.

RoI Across Ladders

- **Case (A)**



- **Case (B)**

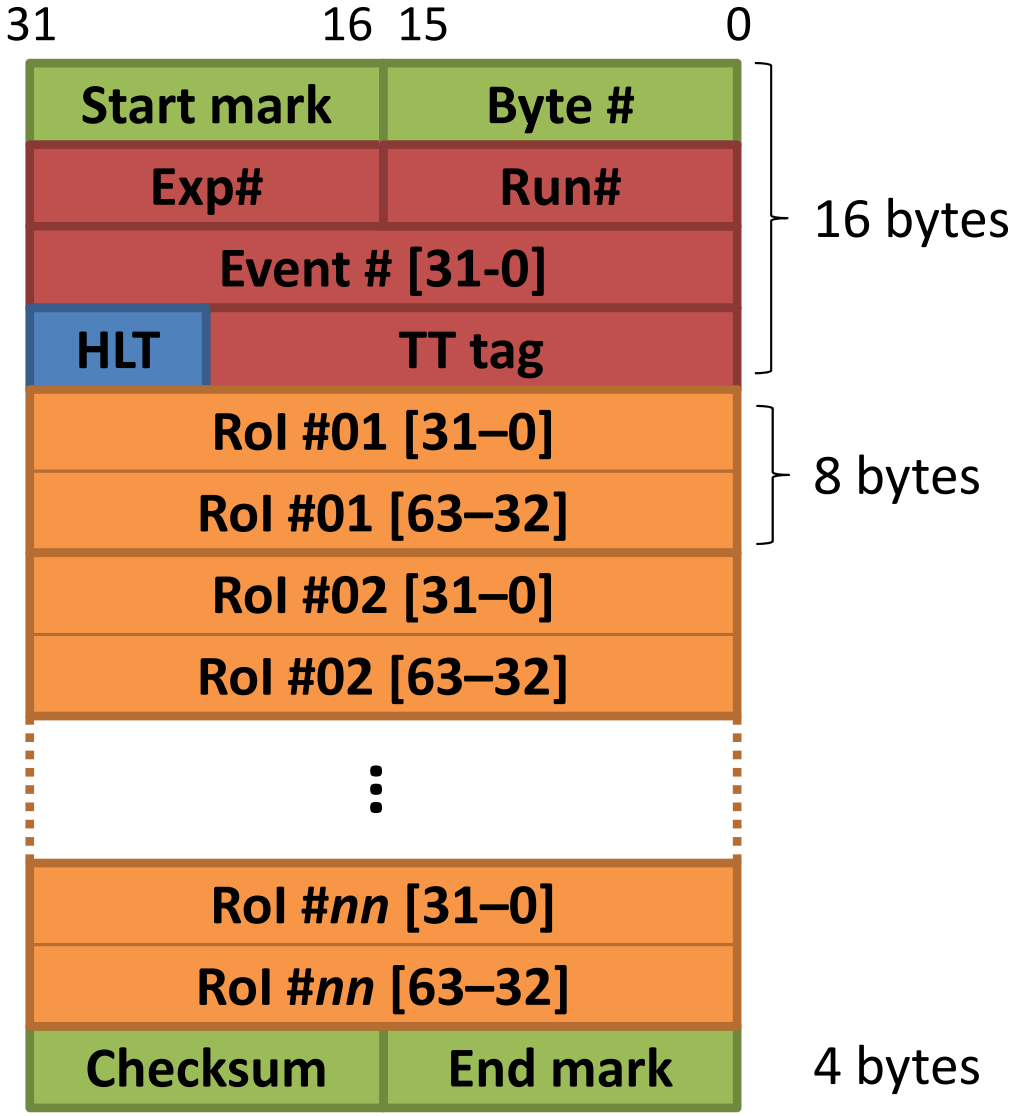


- **Case (C)**
  - Combination of (A) + (B).

Need to survey for the fraction of the across-ladder RoI. But **let's say, conservatively, all tracks have 4 associated RoIs**.

NOTE: every track has at least 2 associated PXD RoIs (layer-1 and layer-2).

HLT-Decision Packet Structure (Ver.0.0)



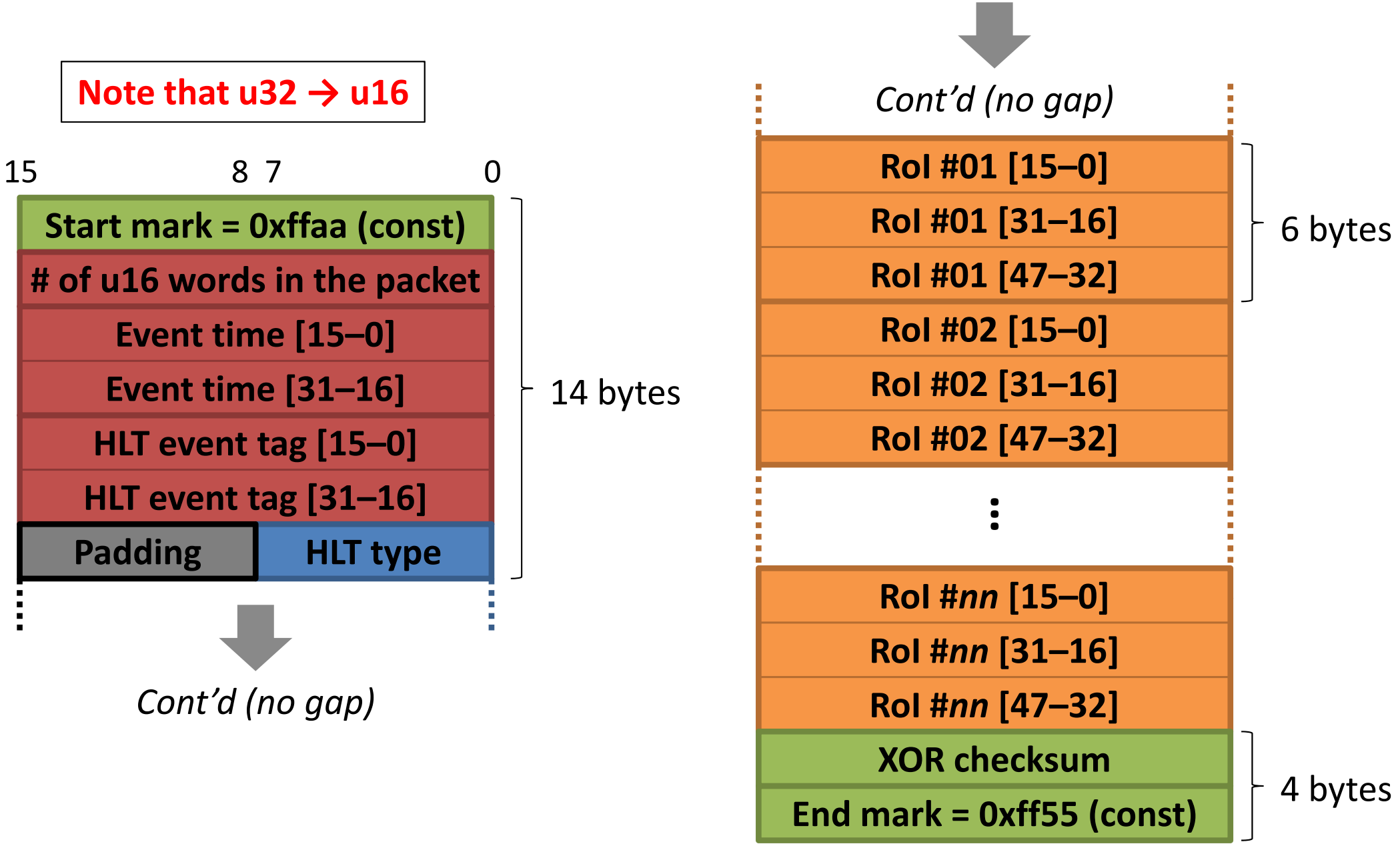
Revision is suggested by M. Nakao-san, S. Lange-san, *et al* in the last VXD-DAQ workshop (Nov. 20th, 2012).

- Replace a set of (exp#, run#, event#, and TT-tag) with **HLT event tag** (32 bit).
- Put **event time** (32 bit) in the header part provided by TT distribution system.
- Put **HLT type** (8 bit) in the header part.
- Using 64 bits (8 bytes) is just a waste of bandwidth for the 42-bit RoI; **pack each RoI information into 48 bits (6 bytes)**.

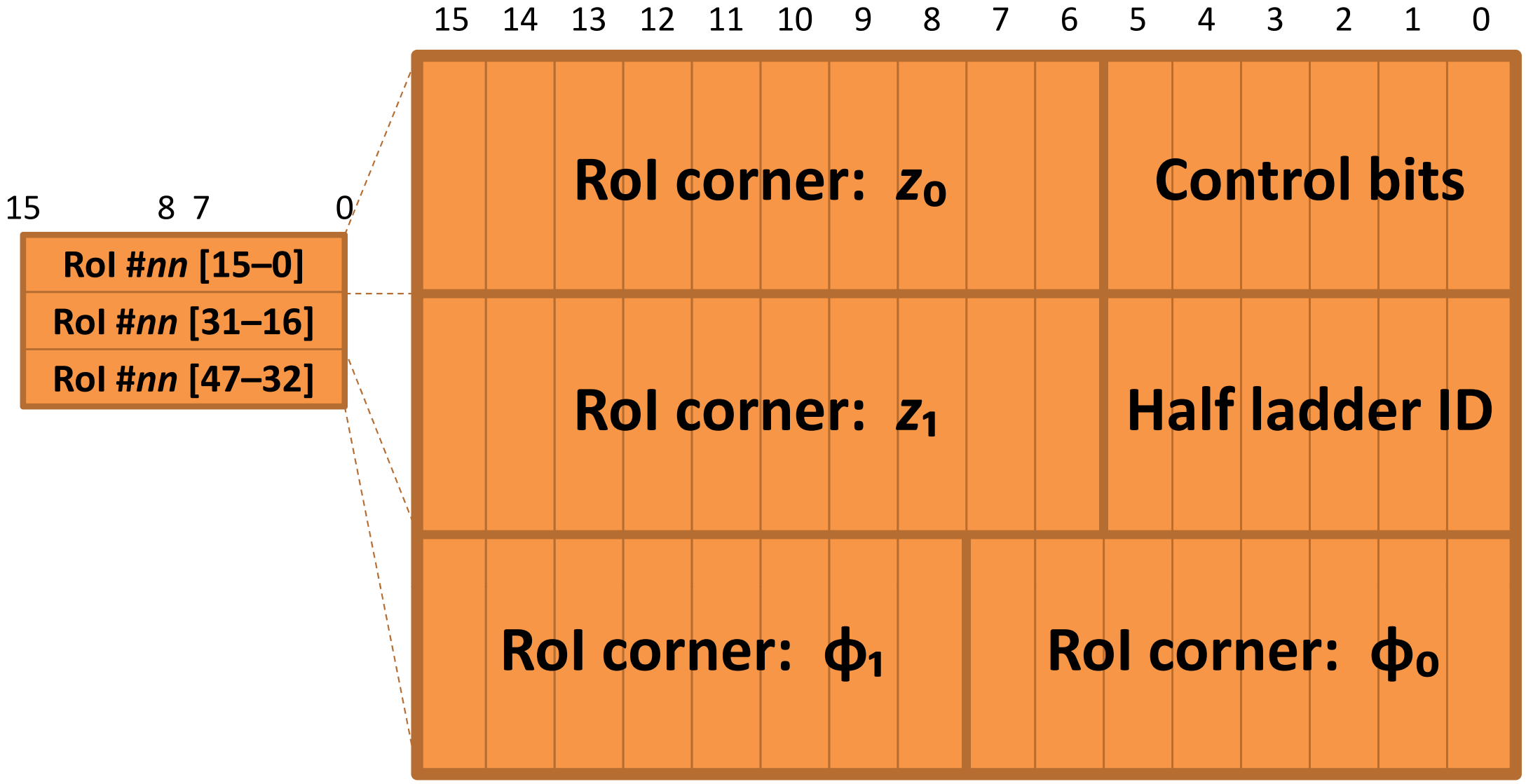
Other comments

- Event number starts from 1.
- Sufficiently large MTU is needed if we use the UDP for the communication.

HLT-Decision Packet Structure (Ver.0.5)



RoI-Block Structure (Ver.0.6) [1]



RoI-Block Structure (Ver.0.6) [2]

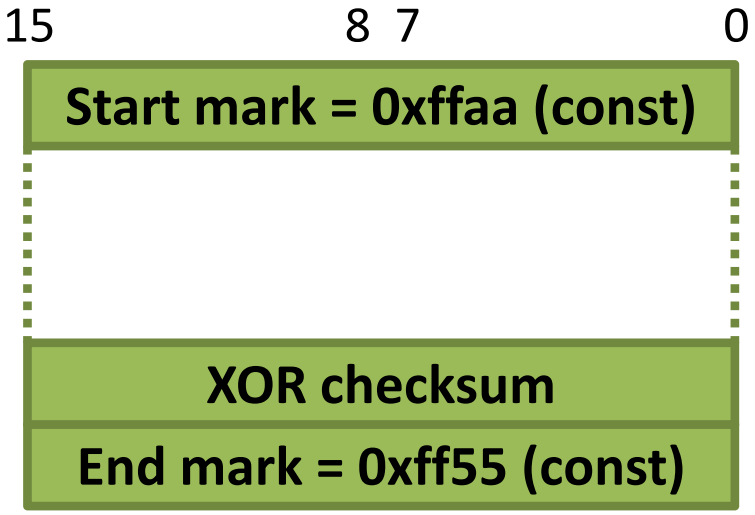
- **Order of (ϕ_0,z_0) and (ϕ_1,z_1)**
  - A routine to generate the HLT-decision packet assures that $\phi_0 \leq \phi_1$ and $z_0 \leq z_1$, which may be helpful for the Onsen firmware.

- **Error handling**
  - If the provided half ladder ID is out of 0…39, the RoI block is filled with three of 0xffff irrespectively of the ϕ and z values .
  - If the half ladder ID is within the range but the provided ϕ is out of 0…249 or the z is out of 0…767, the RoI-block is filled with three of 0xffff as well except the half ladder ID is properly set.

- **Usage of control bits is undefined.**

XOR Checksum (Ver.0.5)

- **Checksum definition**
  - If you take a XOR of all words from the start mark to end mark including the checksum itself, the XOR will become 0x0000.



Software Development (Ver0.5) [1]

- **A class to point the RoI corner**

```
class pxd_point_t;

pxd_point_t::pxd_point_t(void);
pxd_point_t::~pxd_point_t(void);

const bool pxd_point_t::setPHI(const unsigned short phi);
const bool pxd_point_t::setZ  (const unsigned short z);

const unsigned short pxd_point_t::getPHI(void) const;
const unsigned short pxd_point_t::getZ  (void) const;
```

Source codes are found at KEKCC:~higuchit/wrk/belle2/daq/src/roi

Software Development (Ver0.5) [2]

- A class to form the RoI rectangular from 2 points

```
class roi_rectangular_t;

roi_rectangular_t::roi_rectangular_t(void);
roi_rectangular_t::~roi_rectangular_t(void);

const bool roi_rectangular_t::setCorners(
  const unsigned short phi0, const unsigned short z0,
  const unsigned short phi1, const unsigned short z1);
const bool roi_rectangular_t::setCorners(
  const class pxd_point_t c0,
  const class pxd_point_t c1);

const unsigned short getPHI0(void) const;
const unsigned short getZ0  (void) const;
const unsigned short getPHI1(void) const;
const unsigned short getZ1  (void) const;
```

Software Development (Ver0.5) [3]

- **A class for the RoI-block container**

```
class roi_block_t;

roi_block_t::roi_block_t(void);
roi_block_t::~roi_block_t(void);
```

Software Development (Ver0.5) [4]

- A class to build a single RoI block from a rectangular

```
class packROI;

packROI::packROI(void);
packROI::~packROI(void);

const bool packROI::setROI(
  const bool b0, const bool b1, const bool b2,
  const bool b3, const bool b4, const bool b5,
  const unsigned char hlid, const class roi_rectangular_t roi_rectangular);

const roi_block_t packROI::getROIBlock(void) const;
```

Software Development (Ver0.5) [5]

- A class to unpack a single RoI block (for debug)

```
class unpackROI;

unpackROI::unpackROI(void);
unpackROI::~unpackROI(void);

void unpackROI::setROIPacket(const roi_block_t block);

const bool                    unpackROI::getB0(void) const;
const bool                    unpackROI::getB1(void) const;
const bool                    unpackROI::getB2(void) const;
const bool                    unpackROI::getB3(void) const;
const bool                    unpackROI::getB4(void) const;
const bool                    unpackROI::getB5(void) const;
const unsigned char           unpackROI::getHLID(void) const;
const class roi_rectangular_t unpackROI::getROIRectangular(void) const;
```

Software Development (Ver0.5) [6]

- A class to make an HLT-decision packet

```
class packHLTDecision;

packHLTDecision::packHLTDecision(void);
packHLTDecision::~packHLTDecision(void);

void packHLTDecision::setHLTOutput(
  const unsigned long event_time,
  const unsigned long hlt_event_tag,
  const unsigned char hlt_type,
  const std::vector<packROI> rois);

unsigned short * const packHLTDecision::getHLTDecisionBody(void) const;
const size_t           packHLTDecision::getHLTDecisionU16Count(void) const;
```

Software Development (Ver0.5) [7]

- A class to unpack the HLT-decision packet (for debug)

```
class unpackHLTDecision;

unpackHLTDecision::unpackHLTDecision(void);
unpackHLTDecision::~unpackHLTDecision(void);

const bool unpackHLTDecision::setHLTDecisionBody(
  const unsigned short *hlt_decision_body,
  const size_t hlt_decision_u16_count);

const unsigned long unpackHLTDecision::getEventTime(void)   const;
const unsigned long unpackHLTDecision::getHLTEventTag(void) const;
const unsigned long unpackHLTDecision::getHLTType(void)     const;
const size_t        unpackHLTDecision::getROICount(void)    const;
const unpackROI     unpackHLTDecision::getROI(const int i)  const;

void unpackHLTDecision::dumpHLTDecisionBody(void) const;
```

Software Development (Ver0.5) [8-1]

- **User code example (2-RoI case)**

```
/* user.cc */


#include <stdio.h>

#include <vector>
#include "ROIPacket.h"


using namespace std;
using namespace Belle2;
using namespace Belle2::daq;


int main(int c, char *v[])
{
  const unsigned long event_time    = 0x12345678;
  const unsigned long hlt_event_tag = 0x3456789a;
  const unsigned char hlt_type = 0x61;
```

Software Development (Ver0.5) [8-2]

```
struct roi_rectangular_t roi_rectangular[2];
roi_rectangular[0].setCorners( 12,  34,  56,  78);
roi_rectangular[1].setCorners( 21,  43,  65,  87);

class packROI rois[2];
rois[0].setROI( 1, 1, 1, 0, 0, 0, 19, roi_rectangular[0] );
rois[1].setROI( 0, 0, 0, 1, 1, 1, 20, roi_rectangular[1] );

std::vector<packROI> vec_roi;
vec_roi.push_back(rois[0]);
vec_roi.push_back(rois[1]);

class packHLTDecision event;
event.setHLTOutput( event_time, hlt_event_tag, hlt_type, vec_roi );
unsigned short * const hlt_decision_body      = event.getHLTDecisionBody();
const size_t           hlt_decision_u16_count = event.getHLTDecisionU16Count();
```

Software Development (Ver0.5) [8-3]

```
  class unpackHLTDecision unpacker;
  unpacker.setHLTDecisionBody( hlt_decision_body, hlt_decision_u16_count );
  unpacker.dumpHLTDecisionBody();
  printf("event time    = 0x%08x¥n", unpacker.getEventTime());
  printf("HLT event tag = 0x%08x¥n", unpacker.getHLTEventTag());
  printf("HLT type      = 0x%02x¥n", unpacker.getHLTType());

  size_t roi_count = unpacker.getROICount();
  printf("ROI count     = %d¥n",     roi_count);
  for( int i=0; i<roi_count; i++ ){
    printf("  [%d] <%c%c%c%c%c%c> %02d: (%03d,%03d)-(%03d,%03d)¥n",
      i,
      unpacker.getROI(i).getB0() ? 'T' : 'F',
      unpacker.getROI(i).getB1() ? 'T' : 'F',
      unpacker.getROI(i).getB2() ? 'T' : 'F',
      unpacker.getROI(i).getB3() ? 'T' : 'F',
      unpacker.getROI(i).getB4() ? 'T' : 'F',
      unpacker.getROI(i).getB5() ? 'T' : 'F',
      unpacker.getROI(i).getHLID(),
      unpacker.getROI(i).getROIRectangular().getPHI0(),
      unpacker.getROI(i).getROIRectangular().getZ0(),
      unpacker.getROI(i).getROIRectangular().getPHI1(),
      unpacker.getROI(i).getROIRectangular().getZ1()
    );
  }

  return 0;
}
```

Software Development (Ver0.5) [9-1]

- **Outputs (2-RoI case)**

```
higuchit@ccw06% ./ROIPacket_mini
00: 0xffaa 0x000f 0x5678 0x1234 0x789a 0x3456 0x0061 0x0887
08: 0x1393 0x380c 0x0af8 0x15d4 0x4115 0x7530 0xff55
event time    = 0x12345678
HLT event tag = 0x3456789a
HLT type      = 0x61
ROI count     = 2
  [0] <TTTFFF> 19: (012,034)-(056,078)
  [1] <FFFTTT> 20: (021,043)-(065,087)
```

Software Development (Ver0.5) [9-2]

- **Outputs (8-RoI case)**

```
higuchit@ccw06% ./ROIPacket_full
00: 0xffaa 0x0021 0x5678 0x1234 0x789a 0x3456 0x0061 0x08aa
08: 0x1393 0x380c 0x0aea 0x15d4 0x4115 0x002a 0x0057 0x0200
16: 0x002a 0x0057 0x0200 0x002a 0x0057 0x0200 0x002a 0x0057
24: 0x0200 0xffc0 0xffd8 0xffff 0xffc0 0xffff 0xffff 0x7506
32: 0xff55
event time    = 0x12345678
HLT event tag = 0x3456789a
HLT type      = 0x61
ROI count     = 8
  [0] <FTFTFT> 19: (012,034)-(056,078)
  [1] <FTFTFT> 20: (021,043)-(065,087)
  [2] <FTFTFT> 23: (000,000)-(002,001)
  [3] <FTFTFT> 23: (000,000)-(002,001)
  [4] <FTFTFT> 23: (000,000)-(002,001)
  [5] <FTFTFT> 23: (000,000)-(002,001)
  [6] <FFFFFF> 24: (65535,65535)-(65535,65535)
  [7] <FFFFFF> 63: (65535,65535)-(65535,65535)
```

Any variations of specifying the RoI-rectangular corners (*i.e.*: any of $(z_0,z_1)=(0,2)$ and $(\phi_0,\phi_1)=(0,2)$) are sorted to (0,0)–(2,1) as the RoI-rectangular shape.

← bad ϕ or z

← bad half ladder ID

Expected Outbound Throughput

- **Packet size**
  - HLT-fired case:
    18 (header+footer) + 4800 (200xRoIs) bytes → 4812 bytes.
  - HLT-silent case:
    18 (header+footer) bytes.

- **Expected throughput**
  - 10kHz x 4812bytes + 20kHz x 18bytes = 46.6MB.

HLT fired ***HLT silent***

It was 61.6MB if we used 8 bytes per RoI.

Summary

- **A realistic “HLT-decision-packet generator” has been developed with incorporating all/most suggestions raised up in the last VXD-DAQ workshop (Nov, 2012).**

- **The generator has been demonstrated with calling it from an example user codes.**