Status of RPC cabling and decoding

L.Bellagamba,
G. Bruni,
G. Cataldi,
G. Chiodini,
M. Corradi,
S. Spagnolo

May 26, 2009

First considerations on the propagation of RPC errors and conditions from the Byte Stream to higher levels

Particular RPC conditions can be detected from the ByteStream:
- data fragments missing (e.g. some hardware was put out of the DAQ chain)
- data truncated
- data transmission errors from CRC checks

At the moment these conditions are monitored during BS decoding but are not propagated to higher levels.

It would be good to propagate these information to HLT / Offline.
Typical application could be:
- HLT: skip RPC data in ROI if flagged as corrupted
- HLT: monitoring errors
- Reconstruction: take into account if some part of the detector was out of the DAQ chain in the calculation of the probability for holes on track

We have to define:
- how/where to keep the information
- what detail and granularity is needed

RDOs are structured as PAD objects, this seems to be the
natural granularity to be used.

So for each PAD we can introduce a condition word that gives, for example
- all OK / ROD missing / RX missing / pad missing / data truncated /
  CM missing / CM CRC error /

There are two options on where to store this condition word:

1) add it to the RDO PAD object

2) create an Error Service that keeps the information (as done for the ID)

Example from the ID (using an Error Server)

1) A LVL2 algorithm asks data to DataProviderTool which has also a method to retrieve errors:

```
StatusCode stat = m_spacePointTool->fillCollections(m_roiEtaMin, ...); //get data
if(stat.isRecoverable()) { // check if any errors occured
 const std::vector<int>* errVect = m_spacePointTool->fillPixelDataErrors(); // get errors
```

2) in BS decoder: if an error is found it is propagated to the BSErrorService:

```
if (headererror != 0){
    sc = StatusCode::RECOVERABLE; // the SC to be returned
    if (headererror & (1 << 3)) // check the type of error
    m_ErrorSvc->addPreambleError(); //<--- send error to the BS error service.
   ...
```

3) The LVL2 DataProviderTool interacts with both BS decoder and BS Error Service:

```
scRod=m_offlineDecoder->fillCollection(robFrag,m_rdoContainer,...);// call the BS decoder
if(scRod.isRecoverable()) //if RECOVERABLE then some errors were observed
  {   n_recov_errors++;
      m_bsErrorSvc->getNumberOfErrors(idx);// get number of errors of type idx ...
       ...
```

Example of use in HLT:

The HLT asks a (vector of) PADs to be decoded to
RpcRawDataProviderTool
which returns SC=Recoverable if the condition word is not “all OK”.
Then the HLT can just skip the PAD or can get the condition word from
(opt .1) the condition word of the pad or (opt. 2) the error service,
to do something more sophisticated.

Example of use in Offline/EF:

The RDOtoPrepData converter gets the condition word for
each pad and propagates it to the corresponding offline collections by
(opt. 1) putting the word in the PrepData or by (opt. 2)
passing the information in an “offline” format to the error service
which then stores it (presumably the same BS ErrorSvc)
Then, for each offline collection, a client can get the information whether
there was some non standard condition in the corresponding PADs
from the (opt. 1) PrepData or from (opt. 2) the Error Service.

Both examples should work similarly in the two implementations.

Note: the BS error Svc (for the case of the ID) is one of the sub-services integrated into the ConditionSummarySvc (our solution for condition data ???)

Which implementation is better ?

Option 1) : adding a condition word to the RDO:
+ simpler
+ all the pad infos are kept together
+ a similar condition word should be added to the PrepData collection
+ can be made persistent
- if parts of the detector are off, the corresponding RDO objects are created just to keep the condition word

Option 2) adding an Error Service:
+ all the error conditions are kept together
+ two vectors of errors have to be kept, one according to pad index and one according to the offline index
- error data are kept internally by the service: does it need to be put in StoreGate or to be made persistent ?

Cabling map in COOL

In COOL i(schema=ATLAS_COOL_RPCDQ; dbname=RPC_DQA)
we need two folders for the cabling map (~180 KB) and the correction file (~20KB):

/RPC/CABLING/MAP_SCHEMA -> two columns of strings (char[255]) and one columns for the CLOB of the cabling map (200 KB)

/RPC/CABLING/MAP_CORR -> two columns of strings (char[255]) and one columns for the CLOB of the correction files (50 KB)

- At the moment the first folder was created and filled with a part of the cabling map in order to perform the first tests.

- Prototype methods to access the map have been prepared by Monica and should be tested in the next few days.

- The aim is to have the structure ready in the release 15.3 in order to perform systematic tests within the new release.

Latest RPC cabling configuration files

Configuration files (likely the final one):
http://chiodini.web.cern.ch/chiodini/rpc_commissioning/LVL1confAtlas.data
latest update:
• Feet sector 12 and 14 configurations according to the real layout and cabling

Many thanks to Michele Bianco

Correction files (likely the final one):
http://chiodini.web.cern.ch/chiodini/rpc_commissioning/LVL1confAtlas.corr
latest update:
• BML7 eta strip correction

Running on real data

Running with the following RPC cabling packages:

1) default RPC cabling package:

MuonSpectrometer/MuonReconstruction/MuonRecExample
MuonSpectrometer/MuonCablings/RPCcabling

2) new RPC cabling package:

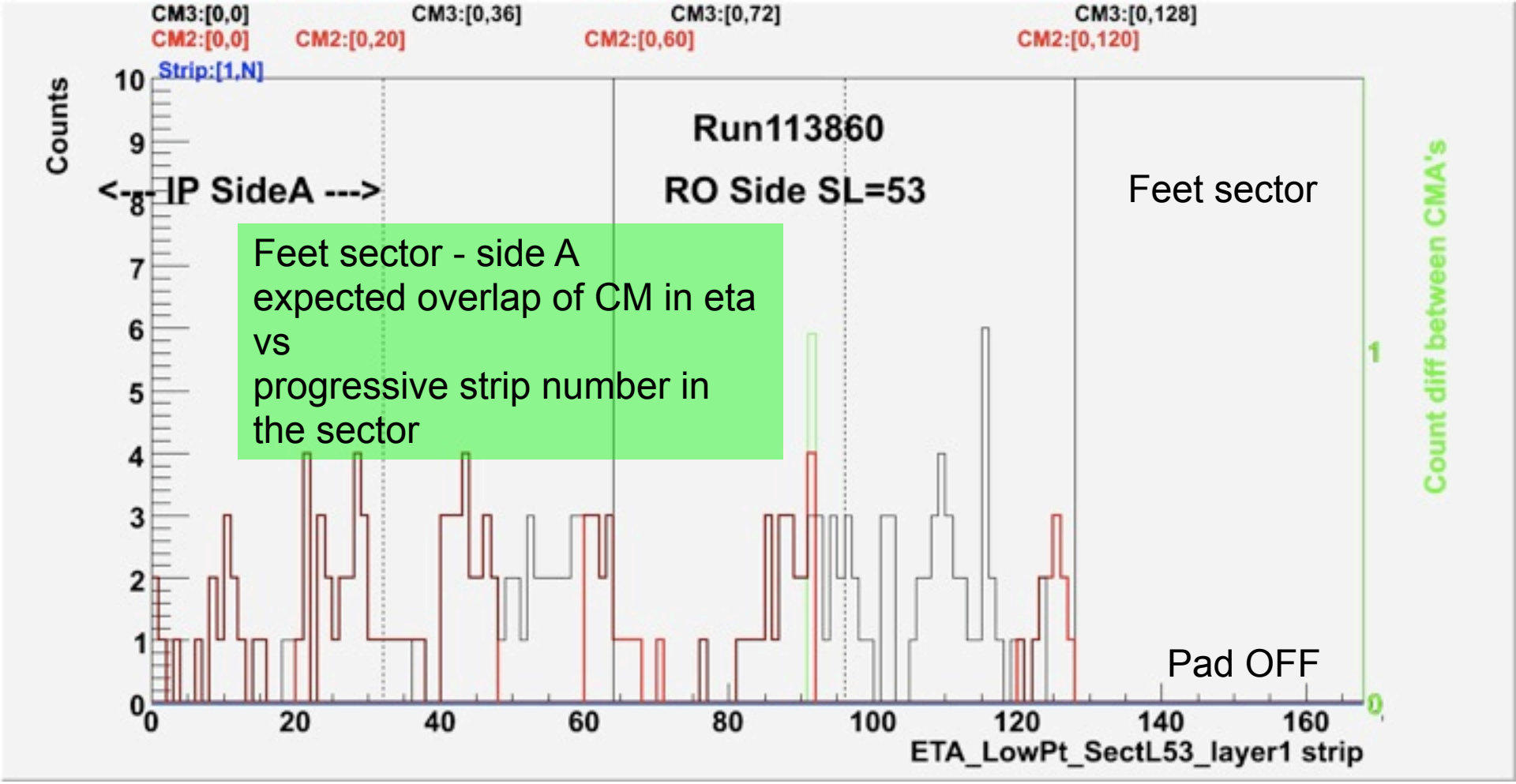
MuonSpectrometer/MuonReconstruction/MuonRecExample
MuonSpectrometer/MuonCablings/MuonCablingServers
MuonSpectrometer/MuonCablings/MuonRPC_Cabling

- equipped with a test algorithm for debugging (now in MuonRPC_Cabling)
- recent fix for unchecked SC
- we should come back to select this svc from the ServerSvc

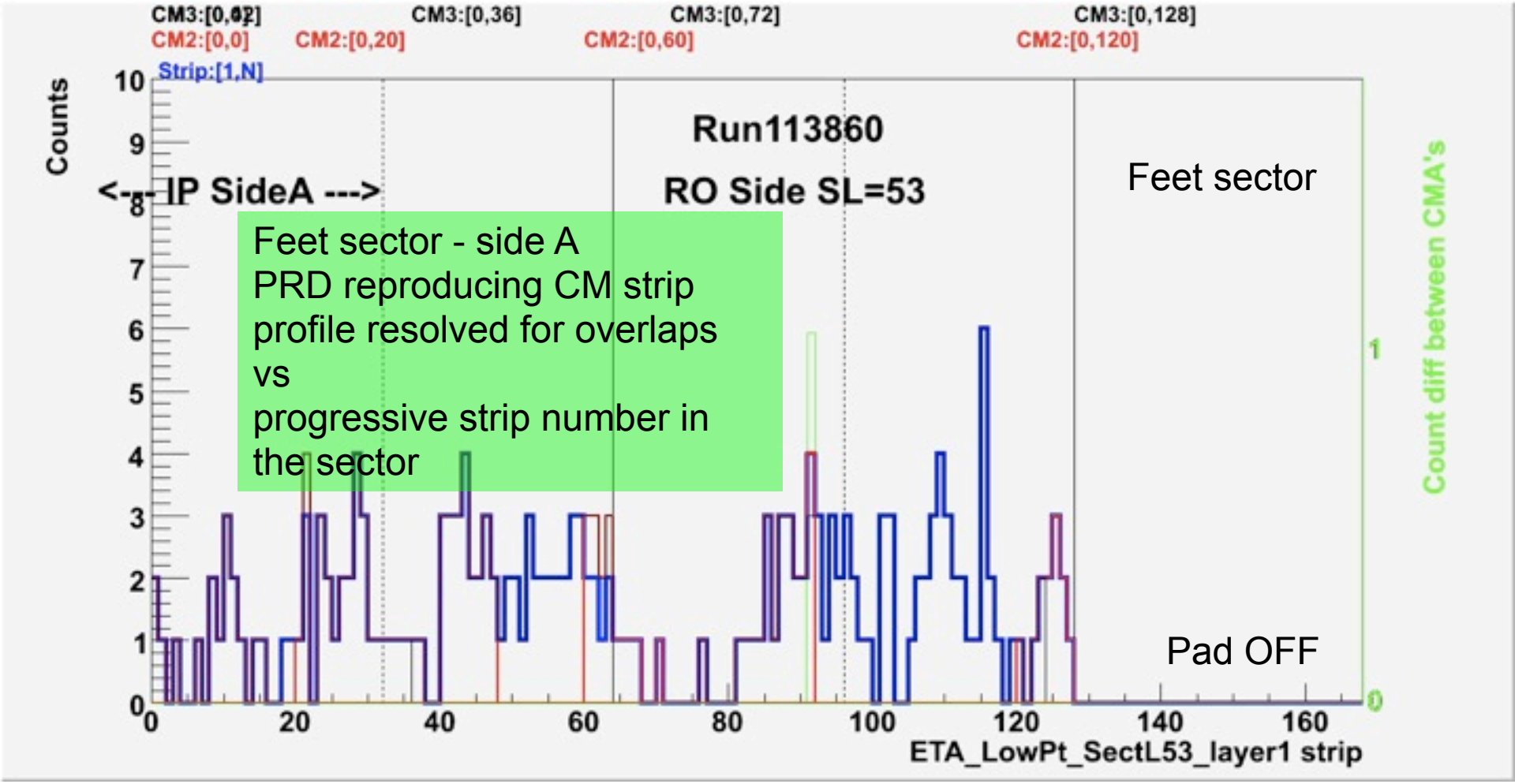
➪ validation of new cabling maps and new cabling package vs old one

The produced prd are the same 1 by 1 !!!

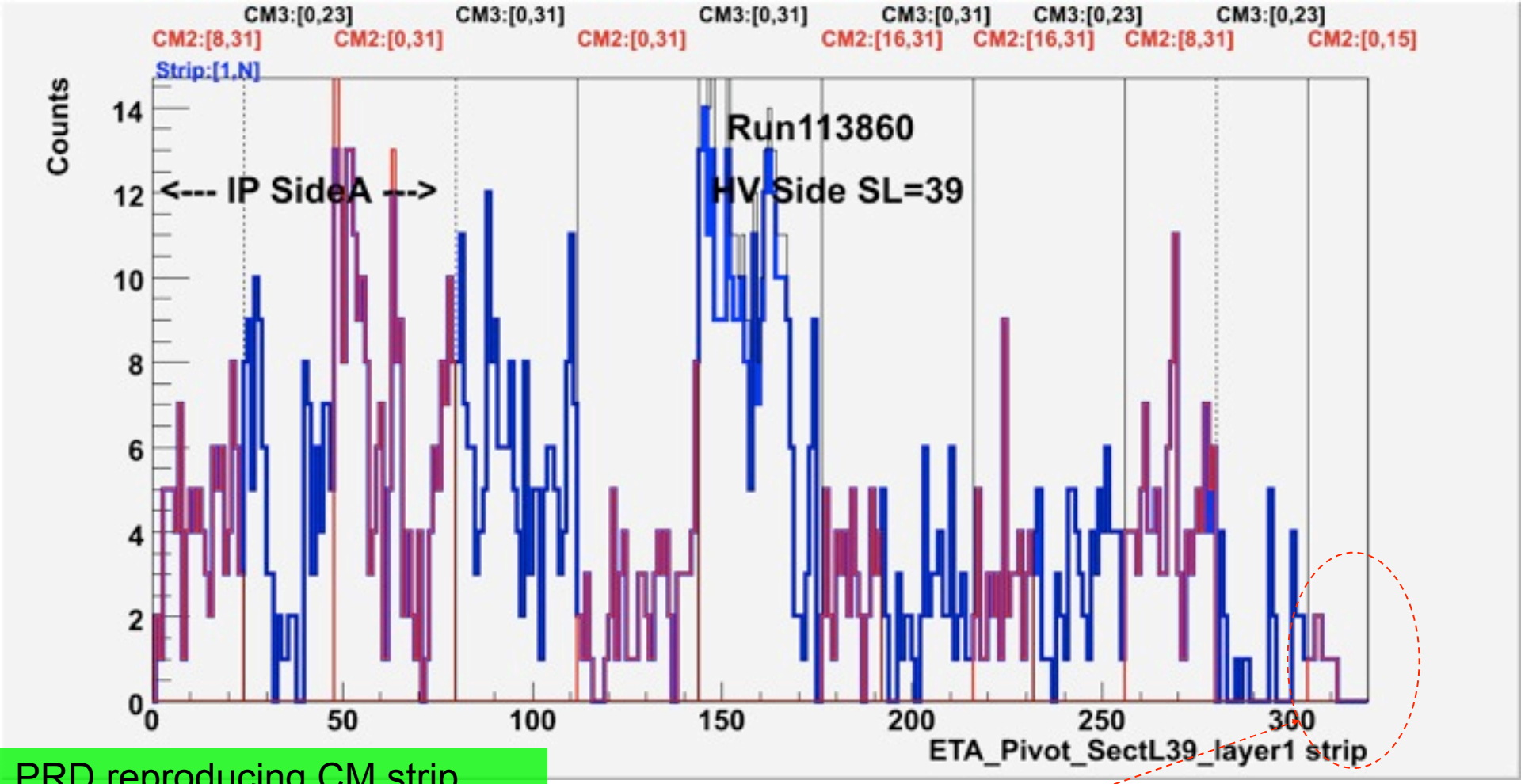
Validation plots: RDO Overlaps



Validation plots: Prep on RDO



Validation plots: Prep on RDO



PRD reproducing CM strip profile resolved for overlaps - RPC prd appear in BML7A/CX

BML7 Sector 5A HV Side