Table of Contents

Class Description

LHCbID is a class that provides a single generic channel identifier throughout LHCb. It is mainly used by the (new) track model to uniquely identify the measurements that were used to make the track. Given a list of LHCbIDs and the corresponding measurements (that can be regenerated trivially from the RawBuffer), it should be possible to reproduce the results of the track fit.

The main requirement when designing this class was that it should be possible to construct from (and to get back) the individual sub-detector channel IDs (VeloChannelID, STChannelID, OTChannelID, RichSmartID, MuonTileID). The detector specific part is coded on 28 bits, leaving 4 bits to identify the sub-detector. The detector specific part is further sub-divided into a "Specific" part that stores the information necessary to rebuild the channel ID, and a detector dependent number of "Spare" bits that contain additional detector specific information (for example the chosen solution for the Outer Tracker left-right ambuguity)

LHCbID is currently not implemented for MuonTileID. This is mainly due to packaging problems (introduction of a circular dependency between LHCbKernel and MuonKernel packages) which could not be easily solved due to the rather complex dependency structure of the MuonKernel code. The review should give suggestions on how to resolve this issue.

Documentation

- Doxygen
- LHCb-2008-047 (For STChannelID usage)

-- MarcoCattaneo - 15 Jul 2005

Comments

Due to these 'Spare' bits, the comparison between LHCbID has to be done with some care. Method should be implemented to compare e.g. OT IDs ignoring the right-left ambiguity. For ST and Velo, where the spare bits contains the cluster size, the comparison may not be just equality, probably one wants to match if there is 'some overlap'... -- OlivierCallot - 26 Jul 2005

Comparing LHCbIDs raises I think the question as to how much the LHCbID should know about the individual sub-detector IDs. I would suggest that the LHCbID class should know as little as possible about the individual IDs. Currently some knowledge is required during construction but this should be minimised, and whenever possible the individual IDs should provide methods that a) presents their internal information in a format suitable for creation as an LHCbID, and b) Allow easy construction from LHCbID. Similarily for comparisions, I do not think it is the task of the LHCbID to know how to compare, for example two OTChannelIDs. This task should be delegated to the OTChannelID class itself. One could imagine implementing comparision operators for LHCbID, which first could test if the detector types are the same, and then if so delegate the detailed comparision to the appropriate sub-detector channel ID class. Something along the lines of :-

```
LHCbID::operator==(const LHCbID& id)
{
   bool OK = this->detectorType() == id.detectorType();
   if ( OK )
   {
     if      ( id.detectorType() == otType ) OK = ( id.otID() == this->otID() );
     else if ( ..... )
   }
   return OK;
}
```

If you want a more "fluzzy" comparision, as described above for OT, then again I see this being more a method of OTChannelID, or maybe even some helper tool. I do not really see this as a task for the LHCbID class itself.

-- ChrisRJones - 22 Aug 2005

I think you missed a bit my point on the comparison between LHCbIDs. The LHCbID contains 3 parts: Detector type, channelID AND 'spareBits' information. The comparison can be based only on the first two (your proposal), but then the comparison of two clusters will ignore the size (which is in the spareBits), which may be the wanted feature but is probably not when comparing L1 and HLT tracks for example. That's why I was suggesting a more complex match when the spareBits are used. In fact, I would also suggest to change the name 'spareBits' and implement methods 'size' for Velo/ST and 'RLChoice' for OT, to be clearer. This would made the code simpler, and faster. -- OlivierCallot - 23 Aug 2005

To my tastes this is getting slightly messy. It introduces what are essentially detector specific methods into a class which I thought is supposed to be detector independant. Is there no way this infomation can be got from the OTChannelIDs themselves (Apologies, I am probably demonstrating extreme ignorance of the OT event model here...) -- ChrisRJones - 23 Aug 2005

The point of these 'spareBits' is just information NOT in the channelID itself. If not, why add them ? One use is (Velo, ST) to have an ID for cluster, not strips, i.e. adding a size. Another use (OT) is to indicate if the track was on the right or the left of the wire, and is then track specific. In the first case a comparison should take into account the size, and process it correctly (but this is not clear: Same size or overlap ?), for the OT I guess the comparison should ignore the RL ambiguity information. BTW, the class is full of SD specific information, as this is mainly methods to create and access channel ID according to the detector ! -- OlivierCallot - 23 Aug 2005

Looking at the class and reading your comments it seems to me that there is one point which needs clarifying - Is the LHCbID class **really** an LHCb wide ID, or just an ID class used by the tracking code to aid its work. These are not the same thing. Are there really use cases which involve using the MUON, CALO and RICH in these things (Maybe muon track refits for the MUON hits, but I struggle with the others) ? For me, I have always really viewed it more as the latter than the former. Would this be a fair comment ? -- ChrisRJones - 25 Aug 2005

I agree that it should be clarified if the LHCbID class is a general LHCb class, or it is a utility class for Tracking. If the latter I think to have there RICH and CALO is confusing. For the MUON one can argue it could be used for track refitting with MUON information. In any case I think it is important to have easy contruction of DetectorID from the LHCbID whatever detector it applies to. I noticed that a contructor exist for LHCbID from ITChannelID but then the accessor is called isST(). I think they should be consistent. -- Main.gcorti - 14 Sep 2005

Constructors from int and conversion to int

Another missing feature is the conversion to/from int, available for most/all channel ID, and useful when using this LHCbID in a Linker relation. -- OlivierCallot - 26 Jul 2005

These methods/constructors could easy be added. However, I am aware that there is a difference of opinion as to whether such methods are a good idea (Personally, I can see both sides of the argument). Perhaps it is time for a definitive discussion as part of the overall event model review on the question as to whether a channel ID is or is not an int ? -- ChrisRJones - 22 Aug 2005

Muon Packaging Problems

Looking at the MuonKernel code the solution seems quite simple - Perhaps I am missing the problem ?

The problem appears to be that the class the MuonTileID class is defined in MuonKernel package, unlike the other sub-detector channel ID classes which are all defined in LHCbKernel. MuonTileID could be moved to LHCbKernel, and as far as I can tell just looking at the code this would also require moving "MuonKernel/MuonBase.h" and "MuonKernel/MuonLayout.h" into LHCbKernel. Maybe with some more thought from the Muon Group this could be avoided, but it doesn't seem to much of a problem, particularly as it was also done for other sub-detectors, such as RichSmartID which required a few enum files to also be moved from RichKernel into LHCbKernel -- ChrisRJones - 22 Aug 2005

Does it require MuonLayout.h to be movedto LHCbKernel? It looks to me that it uses the IMuonLayout interface (implemented by MuonLayout). There is an implementation of MuonTileID that includes also MuonStationLayout.h and MuonSystemLayout.h: are those really necessary? Could MuonTileID depend only on IMuonLayout and MuonBase? To me it look like it should be or am I missing something? By looking at the code there is a forward declaration of MuonTileID in MuonTileID.h... Some tidying up of the code would be nice. -- GloriaCorti - 14 Sep 2005

"Magic Numbers"

A minor comment, but there are some magic numbers scattered throughout the class (Hardcoded numbers without any "obvious" meaning). It would help make the class clearer if these could be either removed (for example derived from more fundamental numbers, such as the number of bits for each field) or at least clearly documented so readers of the code can see how they where arrived at. -- ChrisRJones - 22 Aug 2005

This topic: LHCb > LHCbID
Topic revision: r9 - 2012-02-15 - MarcoCattaneo