LHCb note 2002-030

# New data model, digitization and reconstruction algorithms for the inner tracker.

Matthew Needham
Physik-Institut der Universität Zürich

April 30, 2002

**Abstract**

The re-implementation of the inner tracker digitization and reconstruction algorithms to conform to the new LHCb software model is discussed. This note is intended to provide an introduction to the new model for the non-expert.

# 1 Introduction

The LHCb collaboration has recently decided to re-implement the various sub-detector data models with the aim of having more homogeneous software across the whole experiment. Special emphasis has been put on developing a model in which access to Monte Carlo truth is very decoupled from the information that will be available from the real detector. The data class's are all generated from XML using the GOD [1] package and are therefore in some sense language independent. This allows:

- Automatic generation of a data dictionary and hence data browsing using the Gaudi introspection service and the Python scripting language [2].
- Automatic generation of streamers to allow inter-conversion between transient and persistent representations of the data.

In the first section of this note the data model and data access is described. This is then followed by a section describing the monitoring tools that have been developed. In the last section the packaging of the code is briefly discussed. An appendix summarizes the purpose of each of the inner tracker algorithms. For a detailed description of the physic's simulation of the inner tracker the reader is refered to [3].

# 2 The Data Model

The structure of the data model is shown graphically in Fig. 1. All the objects are contained in 'keyedContainers'. It should be noted that the objects on the right hand side of the parabola contain no explicit link to Monte Carlo truth information. However, since every ITDigit and every MCITDigit are uniquely identified by a detector channel it is still possible to connect the left and right-hand sides of the parabola. This link is only made in dedicated tools called 'associators' (see Section 3.1 ). Starting from the top left of the parabola the classes are:

- MCHit: Entrance and exit points of a particle on a strip. This information is the output of the Geant3 simulation step. The key is sequentially assigned.

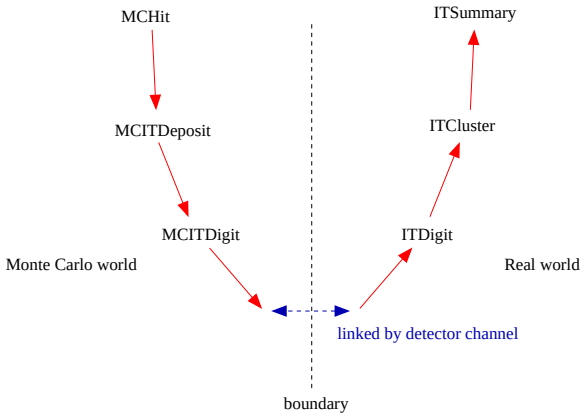


Figure 1: Inner tracker data model.

- MCITDeposit: Charge deposited by a single MCHit on a channel. In this step essentially all of the simulation of the inner tracker response has been simulated. For example: charge deposition, charge sharing and simulation of the readout electronics. The key is sequentially assigned.
- MCITDigit: vector of all MCITDeposits on a single channel. This class simply holds all the MCITDeposits on a single channel in the detector. The detector channel is used as the key.
- ITDigit: digitization as coming from the detector. In the step in going from MCITDigit to an ITDigit the detector noise is simulated. The detector channel is used as the key.
- ITCluster: Cluster of ITDigits. The detector channel is used as the key.
- ITSummary. Summary of inner tracker information in the event.

It should be noted that structure of all the classes from the MCITDeposit level is very homogeneous. For example each of the classes have an access method to the detector channel called 'channelID()'.

Fig. 2 shows the corresponding algorithm model. As can be seen it closely follows the data model — for each data class there is a corresponding algorithm that creates objects of that type and stores them in the transient data-store.

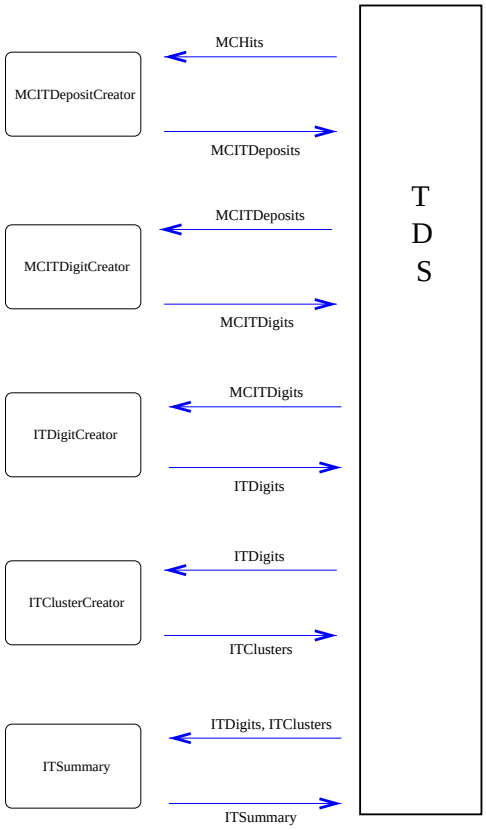


Figure 2: Inner tracker algorithm flow.

## 2.1 Channel Identifier

The detector channel identifier is very important in the new model because it used as the 'key' for many classes. Currently each channel is assigned a bit packed word built from the station, layer, wafer and strip number. As in [4] it is chosen that:

- Stations number from one with increasing station number corresponding to increasing z.
- Layers inside a station number from one with increasing layer number corresponding to increasing z.
- Wafers number from one. Increasing wafer number corresponds to increasing x(y) in the LHCb coordinate system.

- Strips inside a wafer number from one.

The actual implementation of the word is hidden from the user by the ITChannelID class.

## 2.2 Data Access

Objects inside a keyedContainer can either be accessed sequentially using iterators or by using the key. For example:

```
// get ITDigits from store
SmarttDataPtr<ITDigits> digitCont(eventSvc(),
                                  ITDigitLocation::Default)

// sequential access
ITDigits::const_iterator iterDigit = digitCont->begin();
while (iterDigit != digitCont->end()){
  // ...
  iterDigit++;
}

// keyed access
ITChannelId aChan(1,1,1,1);
ITDigit* aDigit = digitCont->object(aChan);
```

However, access by key leads to a substantial CPU overhead and should be avoided where possible. All the inner tracker containers in the store are presorted by the channel identifier. For many algorithms this can make sequential access very attractive from a CPU point of view. In addition, STL provides the following algorithms: lower_bound, upper_bound and equal_range[5]. For a sorted container these provide fast access based on a given criteria using a binary search algorithm. Currently functors have been provided that allow the first two algorithms to be used. Comparison classes have been provided for finding inner tracker data objects based on the station, layer and wafer number. The homogeneous nature of the inner tracker software model allows these classes to be templated for all the inner tracker data objects. For example:

```
// find first cluster in station 2
SmarttDataPtr<ITClusters> clusCont(eventSvc(),
                                   ITClusterLocation::Default);
ITChannelID aChan(2,0,0,0);
  ITClusters::iterator clusIter =
   std::lower_bound(clusCont->begin(),clusCont->end(),aChan,
        ITDataFunctor::compByStation_LB<const ITCluster*>());

// last cluster in station 2
ITClusters::iterator clusIter =
   std::upper_bound(clusCont->begin(),clusCont->end(),aChan,
        ITDataFunctor::compByStation_UB<const ITCluster*>());
```

For completeness a set of functors is also provided that can be used with the STL find_if algorithm. For example:

```
// find first cluster in station 1, layer 2
SmarttDataPtr<ITClusters> clusCont(eventSvc(),
                                   ITClusterLocation::Default);
ITChannelID aChan(1,2,0,0);
ITClusters::iterator clusIter =
   std::find_if(clusCont->begin(),clusCont->end(),
                ITDataFunctor::layerID_eq(aChan));
```

The find_if algorithm works by sequential looping from the first iterator given. Therefore, in many cases it will give slower data access than lower_bound.

# 3 Monitoring Algorithms

For each data class a corresponding monitoring or checking algorithm has been provided that fills histograms. These algorithms have been designed to be as simple as possible. The histograms filled contain information that is known directly to the class. This means for example that that the 'ITClusterChecker' algorithm can be run both on real and simulated data.

Using these algorithms most monitoring requirements are met. However, one important 'use case' that is clearly not met is the monitoring of inner tracker detector resolutions. A separate algorithm — the class 'ITClusterResolution' has been written to perform this task. This algorithm accesses Monte Carlo

truth information via an associator (described below) and therefore **cannot** be run on real data.

## 3.1 Association to Monte Carlo truth

Many rules can be imagined for associating reconstructed inner tracker clusters and digitizations to Monte Carlo truth. It has been chosen to implement the following strategy as the default[1]: a reconstructed object is assigned to the Monte Carlo object that contributes the most charge to it. As an example consider linking an ITCluster to an MCParticle:

- Each ITDigit in the cluster is uniquely linked by the detector channel to a corresponding MCITDigit.
- The MCITDeposit contributing the most charge to the MCITDigit is then found.
- The MITDigit is uniquely linked to a MCHit and hence a MCParticle

The associator makes this sequence of links once and builds a reference table. The code fragments below shows one way to access Monte Carlo truth information [2].

```
// retrieve pointer to associator tool
sc = toolSvc()->retrieveTool( m_nameMCHitAsct, m_hAsct);
if( sc.isFailure() || 0 == m_hAsct) {
  log << MSG::FATAL
      << "Unable to retrieve Associator tool" << endreq;
  return sc;
}

// ....

// use the associator tool to retrieve table
ITCluster2MCHitAsct::DirectType* aTable = m_hAsct->direct();
if (0 == aTable){
```

[1]The individual user is of course free to implement others if he so wishes !
[2]Other simpler ways are under discussion.

```
  MsgStream log(msgSvc(), name());
  log << MSG::WARNING << "Failed to find table" << endreq;
  return StatusCode::FAILURE;
}

ITCluster* aCluster;
// get Hit for a given cluster
ITCluster2MCHitAsct::DirectType::Range range =
    aTable->relations(aCluster);
ITCluster2MCHitAsct::DirectType::iterator iterHit = range.begin();
MCHit* aHit = iterHit->to();
```

More information can be found in the Relations user-guide [6].

# 4 Packaging

The code is packaged as follows:

- The class corresponding to the 'ITChannelID' detector channel can be found in LHCbKernel package.
- All the event class's can be found in the ITEvent package.
- All classes related to the cdf file based geometry can be found in the package ITSicbGeom.
- All the algorithm class's can be found in ITAlgorithms package.
- The associators class's can be found in ITAssociators.

# 5 Summary

The new inner tracker data model has been described in detail. The model as described here has been implemented and tested. It will be used in the forthcoming production for the LHCB-light TDR studies. Clearly this will provide valuable experience of working with the model and will lead to further changes and improvements. One area that is certainly likely to change is the access to Monte Carlo truth as the official way of doing

things is still evolving. An up-to-date version of this note will be kept at "http://mneedham.home.cern.ch/mneedham/itmodel.ps"

# A Algorithms and Tools

**class:** MCITDepositCreator

**Derived from:** Algorithm

**Purpose:** Top Level Algorithm that creates MCITDeposits from MCTrackingHits

**JobOptions Parameters:**

| **Property** | **Description** | **Default** |
|---|---|---|
| meanAmplitude | mean number of electrons deposited | 24000 $e^-$ |
| landauWidth | width of landau | 1400 $e^-$ |
| gaussWidth | sigma of gaussian smearing of Landau | 1700 $e^-$ |
| signalDuration | length of signal | 80 ns |
| decayTime | decay time of signal | 25 ns |
| norm | normalization parameter | 1.93e-7 |
| digiDelay | offset of signal peak | 26 ns |
| tofVector | time of flight vector | — |
| spillVector | spills to digitize | "MC" |
| geometryName | geometry tool to use | "ITGeometry" |
| chargeSharerName | charge sharing tool to use | "ITChargeSharingTool" |

**class:** MCITDigitCreator

**Derived from:** Algorithm

**Purpose:** Top Level Algorithm that creates MCITDigits from MCIDeposits

**JobOptions Parameters:**

**class:** ITDigitCreator

**Derived from:** Algorithm

**Purpose:** Top Level Algorithm that creates ITDigits from MCITDigits

**JobOptions Parameters:**

| Property | Description | Default |
|---|---|---|
| noiseToolName | tool to use to apply noise | "ITNoiseTool" |

**class:** ITClusterCreator

**Derived from:** Algorithm

**Purpose:** Top Level Algorithm that creates ITClusters from ITDigts

**JobOptions Parameters:**

| Property | Description | Default |
|---|---|---|
| threshold | threshold for clustering | 5000 $e^-$ |
| sharingCorr | charge sharring correction | 72.5 |
| maxNtoCorr | max cluster size to apply sharing correction to | 3 |
| geometryName | geometry tool to use | "ITGeometry" |
| errorVec | error estimates | — |

**namespace:** ITGeneral

**Derived from:** —

**Purpose:** namespace for IT related constants

**namespace:** ITDataFunctor

**Derived from:** —

**Purpose:** namespace for IT STL data functors

**class:** IITGeometry

**Derived from:** IAlgTool

**Purpose:** Interface tp tool that creates and owns SICB related geometry objects.

**JobOptions Parameters:**

**class:** ITGeometry

**Derived from:** IITGeometry, AlgTool

**Purpose:** Tool that creates and owns SICB related geometry objects.

**JobOptions Parameters:**

| Property | Description | Default |
|---|---|---|
| newLayout | use updated layout | true |
| newCDF | use new CDF file format | true |
| halfStationClearance | hald station clearance | 5 mm |
| frameClearance | frame clearance | 5 mm |

**class:** IITChargeSharerTool

**Derived from:** IAlgTool

**Purpose:** Interface to Tool that performs charge sharing

**class:** ITChargeSharerTool

**Derived from:** IITChargeSharerTool, AlgTool

**Purpose:** Tool that performs charge sharing

**JobOptions Parameters:**

| Property | Description | Default |
|---|---|---|
| sensitiveWidthFraction | sensitive width of wafer | 0.25 |
| chargeSharingFraction | charge sharing fraction | 0.34 |

**class:** IITNoiseTool

**Derived from:** IAlgTool

**Purpose:** Interface to tool for adding random noise

**JobOptions Parameters:**

**class:** ITNoiseTool

**Derived from:** IITNoiseTool, AlgTool

**Purpose:** Tool for adding random noise

**JobOptions Parameters:**

| Property | Description | Default |
|---|---|---|
| noise | noise level | $1600e^-$ |

**class:** ITCheckAlgorithm

**Derived from:** Algorithm

**Purpose:** Base class for checking algorithms

**JobOptions Parameters:**

**class:** MCITDepositChecker

**Derived from:** ITCheckAlgorithm

**Purpose:** Top Level Algorithm that checks MCITDeposits

**JobOptions Parameters:**

| Property | Description | Default |
|---|---|---|
| geometryName | geometry tool to use | "ITGeometry" |

**class:** MCITDigitChecker

**Derived from:** ITCheckAlgorithm

**Purpose:** Top Level Algorithm that checks MCITDigits

**JobOptions Parameters:**

**class:** ITDigitChecker

**Derived from:** ITCheckAlgorithm

**Purpose:** Top Level Algorithm that checks ITDigits

**JobOptions Parameters:**

**class:** ITClusterChecker

**Derived from:** ITCheckAlgorithm

**Purpose:** Top Level Algorithm that checks output of clustering

**JobOptions Parameters:**

**class:** ITClusterResolution

**Derived from:** ITCheckAlgorithm

**Purpose:** Top Level Algorithm that checks cluster resolutions

**JobOptions Parameters:**

| **Property** | **Description** | **Default** |
|---|---|---|
| geometryName | geometry tool to use | "ITGeometry" |
| associatorName | name of associator to use | "ITCluster2MCHitAsct" |

**class:** ITOccupancy

**Derived from:** ITCheckAlgorithm

**Purpose:** Occupancy histograms

**JobOptions Parameters:**

| Property | Description | Default |
|---|---|---|
| geometryName | geometry tool to use | "ITGeometry" |

**class:** ITCluster2MCHitAsct

**Derived from:** AssociatorWeighted

**Purpose:** Tool to retrieve ITCluster to MCHit relations table

**JobOptions Parameters:**

| Property | Description | Default |
|---|---|---|
| Location | path to retrieve table from | "MC/ITClusters2MCHits" |
| AlgorithmType | algorithm to call to fill table | "ITCluster2MCHitAlg" |
| AlgorithmName | algorithm name for jobOptions | "ITCluster2MCHit" |

**class:** ITCluster2MCParticleAsct

**Derived from:** AssociatorWeighted

**Purpose:** Tool to retrieve ITCluster to MCParticle relations table

**JobOptions Parameters:**

| Property | Description | Default |
|---|---|---|
| Location | path to retrieve table from | "MC/ITClusters2MCParticles" |
| AlgorithmType | algorithm to call to fill table | "ITCluster2MCParticleAlg" |
| AlgorithmName | algorithm name for jobOptions | "ITCluster2MCParticle" |

**class:** ITCluster2MCHitAlg

**Derived from:** Algorithm

**Purpose:** Algorithm to make ITCluster to MCHit relations table

**JobOptions Parameters:**

| Property | Description | Default |
|---|---|---|
| OutputData | path to retrieve table from | "MC/ITClusters2MCHits" |

**class:** ITCluster2MCParticleAlg

**Derived from:** Algorithm

**Purpose:** Algorithm to make ITCluster to MCParticles relations table

**JobOptions Parameters:**

| Property | Description | Default |
|---|---|---|
| OutputData | path to retrieve table from | "MC/ITClusters2MCParticles" |

**namespace:** ITTruthTool

**Derived from:** —

**Purpose:** namespace for associator tools

# References


[1] http://lhcb-comp.web.cern.ch/lhcb-comp/Frameworks/DataDictionary/default.htm.

[2] http://www.python.org.

[3] A. Polouektov *et al.* First results from LHCb inner tracker performance studies using new digitization software. *LHCB-note*, (118), 2001.

[4] M. Merk *et al.* An improved digitization procedure for the Outer Tracker. *LHCB-note*, (55), 2001.

[5] David R. Musser and Atul Saini. *STL Tutorial and Reference Guide.* Addison-Wesley, 1996.

[6] V. Belyaev. Relations: Userguide. *LHCB-note*, (In preparation).