# Fiber Tracker Software Review

Kurt Rinnert

Department of Physics

UNIVERSITY OF LIVERPOOL


## Upgrade Detectors Code Review

CERN, 12.06.2012

# Outline

Det/FTDet

FT/FTDAQ

FT/FTDigitisation

Event/FTEvent

Thoughts on the Event Model

Summary

Det/FTDet

FT/FTDAQ

FT/FTDigitisation

Event/FTEvent

Thoughts on the Event Model

Summary

# Det/FTDet Overview

- Documentation
  - +++ very good overall documentation!
  - +++ the class structure is motivated in the documentation.
  - - the concept of 'net' cell ID was not quite clear.
- Interfaces
  - +++ minimal
  - - parameter passing consistency
  - - typedef name consistency
- Implementation
  - +++ well formatted, easy to read
  - - - potential crash without error message
  - - potential inlining issue
  - - some lengthy functions
  - -? usage of boost::lambda
  - -? some variable names

**Only needs some very minor fixes.**

# Documentation

DeFTLayer.h:

```
/*
 *    SiPM cell numbering:
 *    From 0 to 130, inclusive, always increasing from small to large x,
 *    i.e. from right to left in the figure above. The cell with ID 0(130)
 *    is the right(left) inactive edge of the SiPM. The cell with ID 65 is
 *    the inner inactive gap in the SiPM. Functions are used for converting
 *    between these 'gross' cellIDs and the sensitive ('net') cellIDs.
 */
```

- It was not quite clear to me what a 'net' cell ID is.
- The drawing helps, but the reference to it could be more explicit.
- I think this is a **very minor** issue, though.

# Parameter Passing

DeFTLayer.h:

```
/// Get the u-coordinate of the cell center
double cellUCoordinate(const LHCb::FTChannelID aChan) const;

// ... snip ...

/** Get the FTChannelID of the cell located on the left of the given cell. */
LHCb::FTChannelID nextChannelLeft(const LHCb::FTChannelID& aChan) const;

/** Get the FTChannelID of the cell located on the right of the given cell. */
LHCb::FTChannelID nextChannelRight(const LHCb::FTChannelID& aChan) const;
```

- Once by value, other times by reference...
- ...probably just a typo.
- This is not a performance issue.
- Still I'd recommend to stick to const reference everywhere.

# Typedef Consistency

DeFTDetector.h:

```
typedef std::vector<DeFTStation*> Stations;
typedef std::vector<DeFTBiLayer*> BiLayers;
typedef std::vector<DeFTLayer*> Layers;
```

DeFTLayer.h:

```
typedef std::vector< std::pair<LHCb::FTChannelID, double> > FTPair;
```

This would be more consistent:

```
typedef std::pair<LHCb::FTChannelID, double> FTPair;
typedef std::vector< FTPair > FTPairs;
```

Or even consider a different name reflecting the meaning.

# Wrong Pointer Check

DeFTLayer.cpp:

```
/// Get geometrical limits of the layer
const SolidSubtraction* subtrObject
  = dynamic_cast<const SolidSubtraction*>( this->geometry()->lvolume()->solid() );
const SolidBox*   outerBox = dynamic_cast<const SolidBox*>( subtrObject->coverTop() );
const SolidChild* tmpChild = dynamic_cast<const SolidChild*>( (*subtrObject)[0] );
const SolidCons* innerCons = dynamic_cast<const SolidCons*>( tmpChild->solid() );

if ( 0 == subtrObject || 0 == outerBox || 0 == innerCons ) {
  fatal() << "Can't acquire FT layer geometrical properties. Break ..." << endmsg;
  return StatusCode::FAILURE;
}
```

This code will crash without a message if:

- subtrObject == 0.
- the dynamic cast to SolidChild* fails (unlikely).

# Inlining Issue

DeFTLayer.h:

```
double xAtVerticalBorder(double x0, double y0) const {
  if (std::abs(m_angle)<1.e-4) return x0;
  else {
    double yAtBorder = (y0>0) ? m_layerMaxY : m_layerMinY;
    return x0 + (yAtBorder-y0)*m_tanAngle;
  }
}
```

- This might not get inlined.
- Are you sure you want that anyway?
- Also violates the LHCb coding convention to put non-trivial inline implementations outside the class definition.

# Long Method Implementation

DeFTLayer.cpp:

```
StatusCode DeFTLayer::calculateHits(const Gaudi::XYZPoint& globalPointEntry,
                                    const Gaudi::XYZPoint& globalPointExit,
                                    FTPair& vectChanAndFrac) const
{
  // ... snip ... (240 lines)
}
```

- The implementation is rather lengthy.
- Consider breaking it up?

# Usage of boost::lambda

DeFTDetector.cpp:

```
/// XYZ point -> Layer
const DeFTLayer* DeFTDetector::findLayer(const Gaudi::XYZPoint& aPoint) const {
  if ( !isInside(aPoint) ) { return 0; }
  else {
    Layers::const_iterator iL = std::find_if( m_layers.begin(), m_layers.end(),
                                              bind(&DetectorElement::isInside, _1, aPoint) );
    return (iL != m_layers.end() ? (*iL) : 0);
  }
}
```

- This is not bad *per se*.
- But consider it's a different sub-language of C++.
- And many people can't read it.

# Variable Names

## DeFTLayer.h:

```
double cellUCoordinate(const LHCb::FTChannelID aChan) const;
LHCb::FTChannelID nextChannelLeft(const LHCb::FTChannelID& aChan) const;
LHCb::FTChannelID nextChannelRight(const LHCb::FTChannelID& aChan) const;
DetectorSegment createDetSegment(const LHCb::FTChannelID& aChan, double fracPos) const;
```

## DeFTLayer.cpp:

```
FTChannelID aChan;
```

- I think 'my'-, 'a'- or 'the'-something naming is not a good idea.
- Consider simply 'channel'.
- More on that later...

Det/FTDet

FT/FTDAQ

FT/FTDigitisation

Event/FTEvent

Thoughts on the Event Model

Summary

# FT/FTDAQ Overview

- Documentation
  - ++ OK for interfaces.
  - - - implementation could use more documentation.
- Interfaces
  - ++ straight forward (GaudiAlgorithm)
- Implementation
  - +++ well formatted
  - +++ decoder has clear, straight forward code
  - - - - encoder is hard to understand
  - - -? encoder makes strong assumptions about input
  - -? some variable names
  - -? bank version change not anticipated (it's early, I know)
  - -? maybe the bit shifts etc. should constants defined in a header?

**The encoder could do with an overhaul.**

# Encoder Implementation (Suggestion)

## FTRawBankEncoder.h:

```
class FTRawBankEncoder : public GaudiAlgorithm {

  // ... snip ...

  private:

    /// Reset temporary bank data to empty banks.
    void resetBankData();

    int m_nbBanks;          ///< number of TELL40s
    int m_nbSiPMPerTELL40; ///< number of SiPMs per TELL40

    /** Temporary data structure holding bank data.
     *  Gets reset to empty banks each event.
     *  Dimensions: [m_nbBanks][m_nbSiPMPerTELL40][nClusters]
     */
    std::vector< std::vector < std::vector <unsigned int> > > m_bankData;
};
```

## FTRawBankEncoder.cpp:

- initialize dimensions and data cache in constructor.
- reset data cache at beginning of each event.
- voilà: the encoding becomes a simple loop w/o branches.

Det/FTDet

FT/FTDAQ

FT/FTDigitisation

Event/FTEvent

Thoughts on the Event Model

Summary

# FT/FTDigitisation Overview

- Documentation
  - - - almost none for interfaces.
  - - implementation is a bit better, but needs improvement
- Interfaces
  - ++ mostly straight forward (GaudiAlgorithm)
  - - typedef naming & scope
  - - apparently obsolete remnant functor
  - - naming of functor & header file
- Implementation
  - - - code formatting not very good and inconsistent
  - - variable names

**Documentation & formatting need to be improved.**

# Documentation

```
// from FTClusterCreator.h

    /** @class FTClusterCreator FTClusterCreator.h
     *
     *
     *  @author Eric Cogneras
     *  @date   2012-04-06
     */

// from MCFTDepositCreator.cpp

    //call the calculateHits method
    const DeFTLayer* pL = m_deFT->findLayer(aHit->midPoint());
    std::vector< std::pair<LHCb::FTChannelID, double> > vectCF;
    if (pL) {
      pL->calculateHits(aHit->entry(), aHit->exit(), vectCF);
    }
```

- **all** classes lack class documentation!
- repeating the code in comments does not help.
- comments in the other .cpp files are quite OK.

# Typedef Scope & Naming

MCFTDepositCreator.h:

```
typedef std::vector< std::pair<LHCb::FTChannelID, double> > FTDoublePair;

/** @class MCFTDepositCreator MCFTDepositCreator.h
 *
 *
 *  @author COGNERAS Eric
 *  @date   2012-04-04
 */
class MCFTDepositCreator : public GaudiAlgorithm {
  // ... snip ...
};
```

- the typedef should be in class scope.
- the name is not ideal (see earlier comments on FTPair).

# FTDataFunctor

```
/** @class FTDataFunctor FTDataFunctor.h
 *
 *
 *  @author Eric Cogneras
 *  @date   2012-05-22
 */
namespace FTDataFunctor {
//  // functors
  template <class TYPE1, class TYPE2 = TYPE1 >
    class Less_by_Channel
      : public std::binary_function<TYPE1,TYPE2,bool>
    {
      // ... snip ...
    };
}

struct less_Chan : public std::binary_function<LHCb::MCFTDigit*, LHCb::MCFTDigit*, bool>{
 bool operator()(LHCb::MCFTDigit* x, LHCb::MCFTDigit* y)const {return x->channelID() < y->channelID();}
};
```

- there is no class called FTDataFunctor!
- less_Chan seems to be obsolete (not used anywhere).
- think of a more descriptive name than FTDataFunctor!
- both class names violate LHCb coding conventions.

# Formatting & Variable Names – Code

```
MCFTDigits::const_iterator iterDigit = mcDigitsCont->begin();

while(iterDigit != mcDigitsCont->end()){ // loop over digits

  MCFTDigit* aDigit = *iterDigit;

  if(aDigit->adcCount() > m_adcThreshold){ // ADC count in  digit is over threshold
    std::vector<int> clusterADCDistribution;

    double totalEnergyFromMC = 0;
    std::map< const LHCb::MCParticle*, double> mcContributionMap;

    debug() <<"++ Starts clustering with Channel "<<aDigit->channelID()<<" (ADC = "<<aDigit->adcCount() //...
    MCFTDigits::const_iterator jterDigit = iterDigit;

    do{
      // Add digit to cluster
      MCFTDigit* anotherDigit = *jterDigit;
      debug() <<"+Add to Cluster : "<<anotherDigit->channelID()<<" (ADC = "<<anotherDigit->adcCount() //...
      clusterADCDistribution.push_back(anotherDigit->adcCount());

      // ... snip ...

      ++jterDigit;
    }while((jterDigit != mcDigitsCont->end()) && keepAdding(jterDigit) //...
  }
}
```

# Formatting & Variable Names – Comments

- the use of white space in formatting is not ideal.
- nicely illustrates why 'a'-something names are no good:
  - 'aDigit'...
  - 'anotherDigit'...
  - ...what next? 'yetAnotherDigit'?
- suggestion: 'aDigit' -> 'seedDigit'. And so on, with meaningful names.
- 'jterDigit' is quite inventive...
- ...this tends to work better: 'seedDigitIter'. And so on, with meaningful names.
- there are more examples in other source files, eg. 'vecIter'
  - sure, the code iterates over a vector.
  - but that's obvious and hence the name doesn't carry any further meaning.

Det/FTDet

FT/FTDAQ

FT/FTDigitisation

Event/FTEvent

Thoughts on the Event Model

Summary

# Event/FTEvent Overview

- Documentation
  - - no class is documented. (but they are almost self-documenting).
- XML

  +++ all fine.

**All fine with current conventions.**

Det/FTDet

FT/FTDAQ

FT/FTDigitisation

Event/FTEvent

Thoughts on the Event Model

Summary

# Some Thoughts on the Event Model

- It follows established LHCb conventions, nothing wrong there.
- But maybe these conventions should be revisited for the upgrade:
  - the raw banks contain all information.
  - they are also implicitly associated with the detector geometry.
  - decoding to clusters replicates the information and decouples from the geometry.
  - for the pattern recognition the geometry association has to be reestablished.
- All this consumes memory and CPU resources!
- How about cutting out the middle man?

**No solution here, just food for thought.**

Det/FTDet

FT/FTDAQ

FT/FTDigitisation

Event/FTEvent

Thoughts on the Event Model

Summary

# Summary

- **Det/FTDet**
  - almost perfect!
- **FT/FTDAQ**
  - good, encoder needs a little work, one way or the other.
- **FT/FTDigitisation**
  - mostly needs work on documentation, names and formatting.
- **Event/FTEvent**
  - event model fine with current conventions. . .
  - . . . but think about it!
- **debug() streams**
  - maybe I missed something and the behaviour has changed.
  - otherwise, event scope instances should be protected.

**Overall, it's a good start!**