

StIstDbMaker gets handlers of geometry and calibration Dbs from StarDb.

StIstCalibrationMaker calculates pedestal, rms noise and common mode noise over all time bins. The maker produces two data files, which are used for writing calibration Dbs. The gain calculation for each channel will be available later, and currently we only set the gain to 1.0 for all channel in the StIstCalibrationMaker.

StIstFastSimMaker makes StMcIstHit directly from StarSim output for fast simulation. The StIstSlowSimMaker is not available yet, which will be designed to make raw hits for slow simulation.

StIstQaMaker is designed to generate QA histograms and trees for quick check on IST raw hit and hit levels.

Some basic constants for the IST detector are saved in the StIstConsts.h.

The whole IST offline software distributes in the StRoot directory, and as shown in Fig. 2.

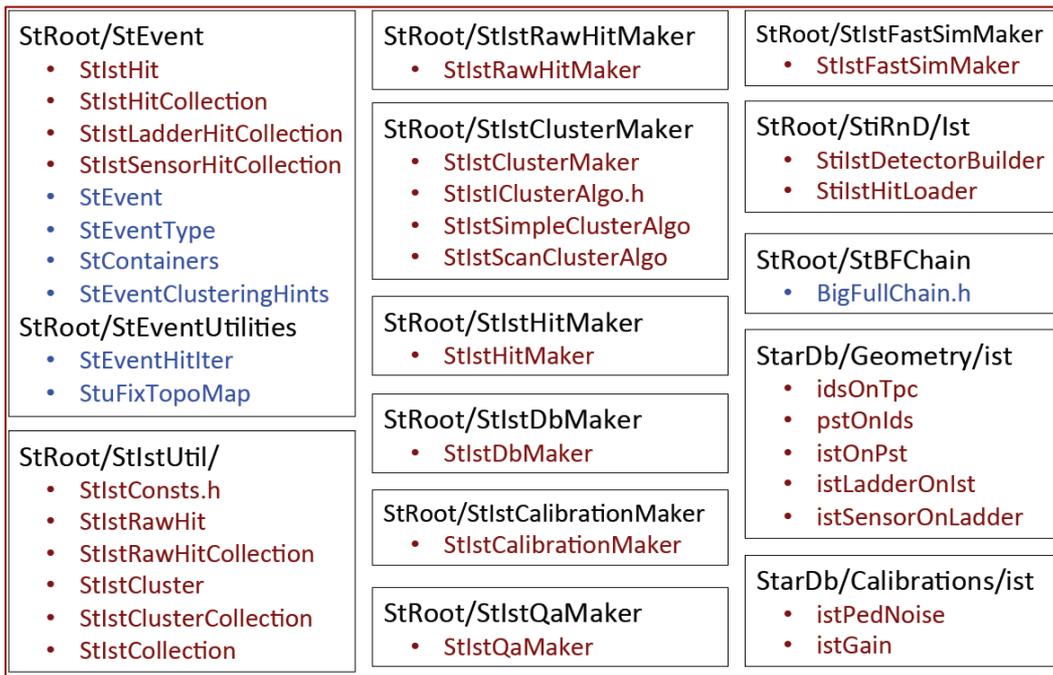


Fig. 2 IST offline software distribution in StRoot

The hit structure/collections and its StIstHitMaker have been submitted to S&C group for review in September. But a few updates have been made in this version. Detailed descriptions for IST raw hit/cluster/hit data structures and makers will be introduced in following chapters.

2. IST raw hit/cluster/hit structures

2.1 StIstRawHit

Each IST raw hit corresponds to a fired silicon pad (channel), and the following data members are defined for the IST raw hit:

```

Int_t    mChannelId;           // channel Id, numbering from 0 to 110591
Float_t  mCharge[kIstNumTimeBins]; // raw ADC value saved in calibration mode;
        // charge value saved in non-calibration mode (pedestal subtracted and CMN corrected)
Float_t  mChargeErr[kIstNumTimeBins]; // charge error in all time bins
UChar_t   mMaxTimeBin;       // the max ADC time bin index of the raw hit
UShort_t  mIdTruth;         // !< for embedding, 0 as background
static UChar_t  mDefaultTimeBin;

```

The IST raw hits are stored in the `StIstRawHitCollection` container sorted by each raw hit geometry ID in ascending order. The raw hit geometry ID is decoded by readout electronics mapping. The ladder ID, sensor ID, column index and row index of the raw hit will be calculated via the saved electronics Id (`mChannelId`) in `StIstRawHit`. The `StIstRawHitCollection` is saved to a temporary dataset, called `StIstCollection` as shown in Fig. 3.

2.2 StIstCluster

Each IST cluster is a group of neighboring raw hits, and following data members are defined for the IST cluster:

```

Int_t      mKey;                //cluster unique label
UChar_t    mLadderId;           //ladder id the cluster belongs to
UChar_t    mSensorId;          //sensor id the cluster belongs to
Float_t    mMeanRow, mMeanColumn; //mean row and mean column
Float_t    mTotCharge;         //charge sum of the cluster
Float_t    mTotChargeErr;      //RMS noise of the cluster
UChar_t    mMaxTimeBin;       //max ADC time bin index
UChar_t    mClusteringType;    //clustering algorithm type
UChar_t    mNRawHits;         //cluster size
UChar_t    mNRawHitsRPhi;     //cluster size in r-phi direction
UChar_t    mNRawHitsZ;       //cluster size in beam direction
UShort_t   mIdTruth;         //!< for embedding, 0 as background
rawHitMap_t  mRawHitMap;      //map container to save raw hits who
contribute to the cluster

```

The IST clusters are stored in the `StIstClusterCollection` containers, and they are saved to the `StIstCollection` as shown in Fig. 3.

2.3 StIstHit

Each IST hit is generated from an IST cluster, and the hit global position is included. The `StIstHit` is inherited from `StHit` class, and the following data members are defined:

```

UChar_t  mMaxTimeBin;           // max charge time bin
Float_t  mChargeErr;           // charge uncertainty
UChar_t  mClusteringType;      //clustering algorithm type
UChar_t  mNRawHits;           // nRawHits: cluster size
UChar_t  mNRawHitsZ;          // cluster size in Z direction
UChar_t  mNRawHitsRPhi;       // cluster size in r-phi direction
Float_t  mLocalPosition[3];    // local position of hit inside the sensor
StDetectorId mDetectorId;     // kIstId

```

The IST hit is stored in the three-level hierarchy containers in top-bottom way: StIstHitCollection→StIstLadderHitCollection→StIstSensorHitCollection. Finally, the StIstHitCollection container is saved to StEvent.

The overall storage scheme of the IST raw hit, cluster and hit containers is shown in Fig. 3.

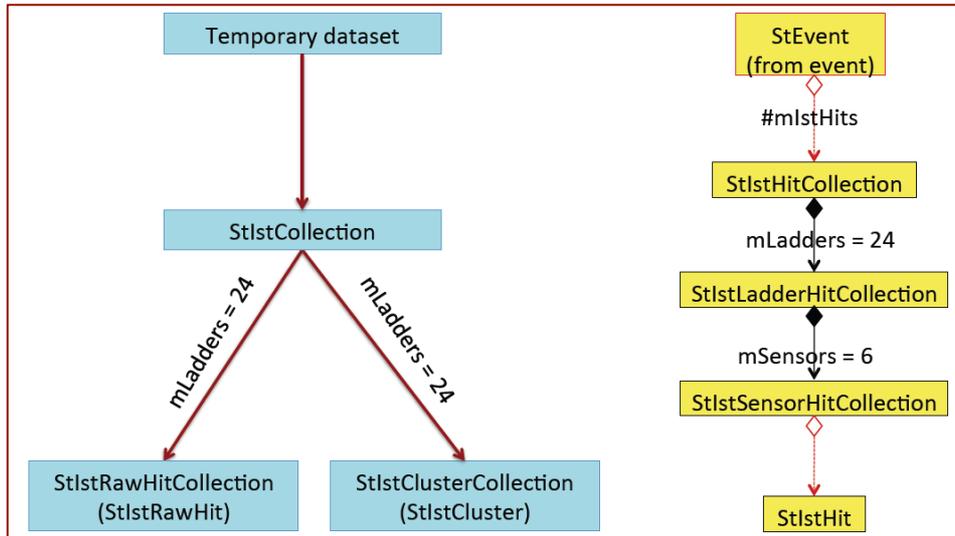


Fig. 3 IST raw hit, cluster and hit containers and hierarchy structure

3. IST raw hit/cluster/hit makers

3.1 StIstRawHitMaker

StIstRawHitMaker un-packs the daq and/or sfs format data and makes IST raw hits. The IST daq(sfs) data contains electronics information, such as **rdo** (ARM Readout Controller board ID, numbering from 1 to 6 per whole IST readout system), **arm** (APV readout module board ID, numbering from 0 to 5 per rdo unit), **apv** (APV chip ID, numbering from 0 to 23 per arm unit), **channel** (readout channels ID, numbering from 0 to 127 per APV chip unit), **adc** (ADC value, range from 0 to 4095) and **timebin** (time bin index, numbering from 0 to 7/15/31). Meanwhile, the broken/noisy channel will be masked out by its strange RMS noise (1000.00, much higher than nominal value). And the bad/strange chip will be masked out by its common mode noise (1000.00, much higher than nominal

value).

The StIstRawHitMaker has two working modes: One is **Calibration mode**, which is designed to deal with pedestal runs data, and in this mode only the raw ADC value with the channel's electronics ID will be stored to the StIstRawHitCollection. The other is **Physics mode**, in which physics data are processed. In the physics mode, the pedestal subtraction and signal-like raw hit decision are executed and the common mode noise correction can be enabled/disabled. In this mode, the calibration Dbs, including istPedNoise table and istGain table, are retrieved from StarDb by the StIstDbMaker.

The ADC to charge transfer is done by multiplying the channel's gain (currently the gain is set to 1.0 for all channels).

3.2 StIstClusterMaker

The StIstClusterMaker does the raw hits clustering ladder by ladder, and its inputs are the raw hit collections (24 ladders). The maker outputs StIstClusterCollection containers filled with StIstCluster elements. The maker executes the raw hits clustering in a sensor area.

An interface, "Int_t setClusterAlgo(StIstIClusterAlgo*)" is left in the StIstClusterMaker to implement different algorithms easily. Here, all algorithm classes should be inherited from a virtual class StIstIClusterAlgo.

The StIstClusterMaker calculates weighted ADC sum/Charge sum, weighted noise, and weighted column/row for each cluster. These parameters are calculated by following formulas:

$$\begin{aligned}
 ADC_{cluster} &= \sum_{i=1}^{N_{pads}} ADC_i \quad , \quad \sigma_{cluster} = \sqrt{\sum_{i=1}^{N_{pads}} \sigma_i^2 / N_{pads}} \\
 x_{cluster} &= \sum_{i=1}^{N_{pads}} x_i \cdot w_i \quad , \quad y_{cluster} = \sum_{i=1}^{N_{pads}} y_i \cdot w_i \\
 w_i &= ADC_i / \sum_{i=1}^{N_{pads}} ADC_i
 \end{aligned}$$

Here the N_{pads} means cluster size. ADC_i and σ_i mean the pedestal subtracted ADC value and RMS noise for the i^{th} raw hit of the current cluster, respectively. x_i and y_i present mean row and mean column, respectively.

Currently, two algorithms are available for the raw hits clustering in this maker.

1) StIstSimpleClusterAlgo

- Read all raw hits per ladder (six sensors) and write into a vector.
- Start from the 1st raw hit, and loop the vector to look for neighboring raw hits (in a sensor area) and do clustering. The found cluster will be filled into a ladder cluster collection.

- A case-by-case splitting algorithm can be enabled/disabled for the found clusters (here only works for cases with cluster size ≤ 4).

2) StIstScanClusterAlgo

- Read all raw hits per ladder (six sensors) and group into vectors (each vector is corresponding to a sensor column).
- Clustering in individual column: loop column vectors and scan each vector sequentially to look for neighboring raw hits and do clustering. Once three continuous raw hits are found and the middle one has the minimum ADC/charge value, then the middle raw hit will be split into two parts weighted by its two neighboring raw hits' ADC/charge, respectively. The found cluster candidate will be filled into the corresponding cluster vector (each cluster vector is corresponding to a sensor column).
- Clustering between columns: loop neighboring columns, and do clustering once two cluster candidates are found with their weighted row index difference ($|\langle \text{row} \rangle_i - \langle \text{row} \rangle_j|$) less than 0.5.
- Fill the found clusters into the ladder cluster collection.

The clustering algorithm is shown as below Fig. 4. The performance comparison with the two algorithms can be found:

http://www4.rcf.bnl.gov/~ypwang/IST_software/weeklyMeeting_Yaping.pdf

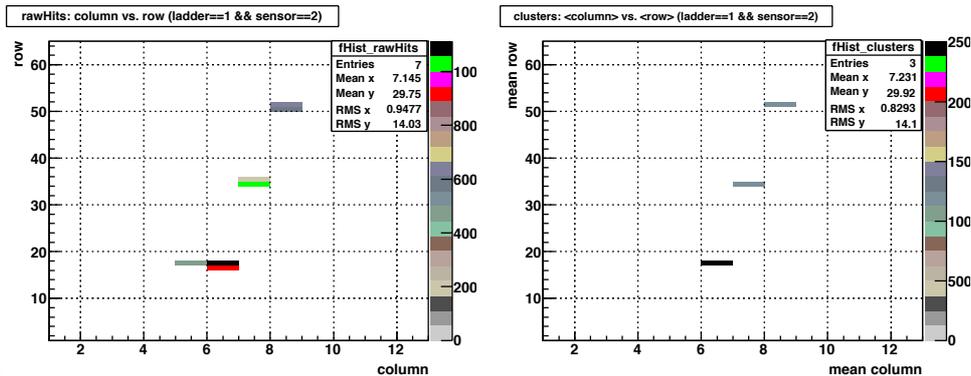


Fig. 4 IST clustering scheme for single event display (Left: Before clustering; Right: After clustering)

3.3 StIstHitMaker

StIstHitMaker produces StIstHit from the StIstCluster information, and calculates hit's global position. The global position is calculated from local x/y/z by rotation and translation according to the geometry Db tables retrieved from StarDb by StIstDbMaker.

4. IST geometry/calibration Db maker (StIstDbMaker)

The StIstDbMaker obtains geometry and calibration Db tables. The geometry tables contain istOnPst, istLadderOnIst, istSensorOnLadder, idsOnTpc and pstOnIds with the Survey_St format. The calibration tables include istGain and istPedNoise with specified formats.

These tables can be checked at: <http://online.star.bnl.gov/dbExplorer/>

5. IST pedestal, noise and gain maker (StIstCalibrationMaker)

The StIstCalibrationMaker calculates pedestal, RMS noise and common mode (CM) noise by histogram method for each channel/chip over all time bins. The calculated values are written to two dataset files. These two files are used to write calibration Dbs. The RMS noise is set to 1000.00 for broken/noisy channel, which is used for channel masking out. And the CM noise is set to 1000.00 for bad/strange chip used for chip masking out. The maker also generates several QA plots.

Currently, the gain calculation maker is not available yet, which will be located in the StIstCalibrationMaker directory. The gain for each channel is set to 1.0 in the current calibration maker.

6. Qa maker (StIstQaMaker)

StIstQaMaker is designed to generate QA histograms and trees for quick check on both raw hit and hit levels. The Fig. 5 shows the hit global position from the QA histograms.

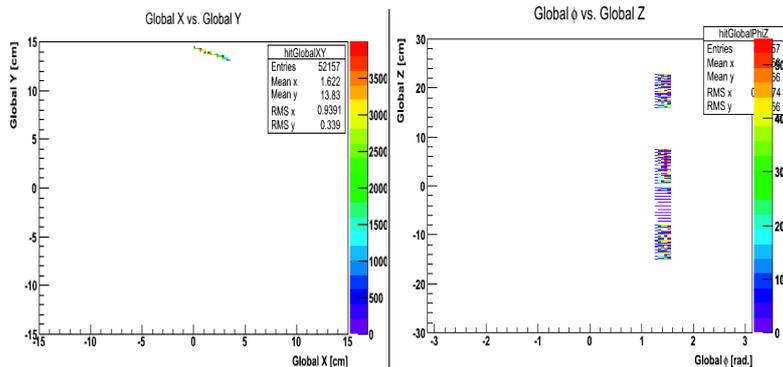


Fig. 5 IST hit global position. (Left: global position in XY view; Right: global position in Z-phi view)

7. Fast simulation maker (StIstFastSimMaker)

The StIstFastSimMaker makes StMcIstHit directly from StarSim output for fast simulation. The StIstDetectorBuilder and StIstHitLoader (located in StRoot/StiRnd/Ist) are updated for tracking. The reconstruction is realized by executing run BFC chain. The Fig. 6 shows the IST hit global positions from simulation results.

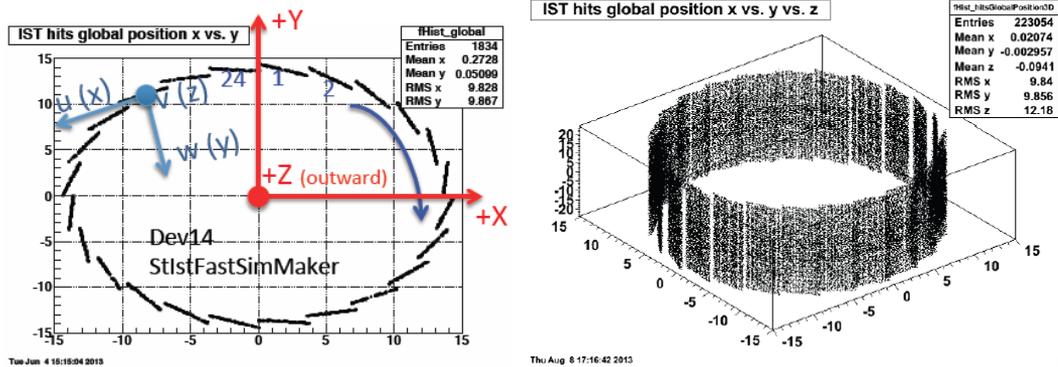


Fig. 6 IST fast simulation (Left: global position in XY view; Right: global position in XYZ view)

8. Codes installation and run instruction

The latest codes are at:

`/star/u/ypwang/disk02/istOfflineChain/istOfflineChain_review`

Codes installation (XXXXX presents your directory):

```
cp -r /star/u/ypwang/disk02/istOfflineChain/istOfflineChain_review/StRoot XXXXX
```

```
cp -r /star/u/ypwang/disk02/istOfflineChain/istOfflineChain_review/StarDb XXXXX
```

```
cp -r /star/u/ypwang/disk02/istOfflineChain/istOfflineChain_review/runIstCalibration.C XXXXX
```

```
cp -r /star/u/ypwang/disk02/istOfflineChain/istOfflineChain_review/runIstOfflineChain.C XXXXX
```

stardev

cons

`root4star -b runIstCalibration.C` (to generate calibration dataset files)

`root4star -b runIstOfflineChain.C` (to run IST offline chain)

Currently, local StarDb provides geometry and calibration Dbs, which have not been written into the STAR DB.

The codes to be reviewed and committed are showing below:

StRoot/StEvent/

- StIstHit
- StIstHitCollection
- StIstLadderHitCollection
- StIstSensorHitCollection

StRoot/StIstUtil/

- StIstConsts.h
- StIstRawHit
- StIstRawHitCollection
- StIstCluster
- StIstClusterCollection
- StIstCollection

StRoot/StIstRawHitMaker/

- StIstRawHitMaker

StRoot/StIstClusterMaker

- StIstClusterMaker
- StIstIClusterAlgo.h
- StIstSimpleClusterAlgo
- StIstScanClusterAlgo

StRoot/StIstHitMaker

- StIstHitMaker

StRoot/StIstDbMaker

- StIstDbMaker

StRoot/StIstCalibrationMaker

- StIstCalibrationMaker

StRoot/StIstQaMaker

- StIstQaMaker

StRoot/StIstFastSimMaker

- StIstFastSimMaker

StRoot/StEvent/

- StIstContainer.h: [lines 162, 220](#)
- StIstContainer.cxx: [lines 146, 204](#)
- StEvent.h: [lines 206, 268-269, 339](#)
- StEvent.cxx: [lines 223, 783-797, 1247-1251, 1367, 1393](#)
- StEventType.h: [lines 278-281](#)
- StEventClusterHints.cxx: [lines 165, 189](#)

StRoot/StEventUtilities/

- StEventHitIter.cxx: [lines 20, 450-525, 750](#)
- StuFixTopoMap.cxx: [lines 126-131](#)

StRoot/StBFChain/

- BigFullChain.h: [following lines are added \(1349-1356\)](#)

```
{ "istDb"          , "istDb" , "" , "tpcDb" , "StIstDbMaker" , "StIstDbMaker" , "Load
and run IstDbMaker" ,          kFALSE} ,
{ "istFastSim"   , "" , "" , "StMcEvent,StEvent" ,
"StIstFastSimMaker" , "StIstFastSimMaker" , "FastIstSimulator" , kFALSE} ,
{ "istUtil"      , "" , "" , "" , "" , "StIstUtil" , "Ist Utilities" , kFALSE} ,
{ "istRaw"       , "" , "" , "istUtil,istDb" , "StIstRawHitMaker" , "StIstRawHit
Maker" , "Ist Raw Hit Maker" , kFALSE} ,
{ "istCluster"   , "" , "" , "istUtil" , "StIstClusterMaker" , "StIstCluster
Maker" , "Ist Cluster Maker" , kFALSE} ,
{ "istHit"       , "" , "" , "istUtil,event,istDb" , "StIstHitMaker" , "StIstHit
Maker" , "Ist Hit Maker" , kFALSE} ,
{ "istQA"        , "" , "" , "istUtil,StEvent"          , "StIstQaMaker" , "StIstQaMa
ker" , "Example of Ist QA" , kFALSE} ,
```