# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOTEM Ntuple</td>
<td>1</td>
</tr>
<tr>
<td>Metadata part</td>
<td>1</td>
</tr>
<tr>
<td>Trigger part</td>
<td>1</td>
</tr>
<tr>
<td>Roman Pot part</td>
<td>2</td>
</tr>
<tr>
<td>Digi section (misleading name, this branch refers to clusters)</td>
<td>2</td>
</tr>
<tr>
<td>Pattern-recognition section</td>
<td>3</td>
</tr>
<tr>
<td>Track section</td>
<td>3</td>
</tr>
<tr>
<td>Multitrack section</td>
<td>3</td>
</tr>
<tr>
<td>Single-proton reconstruction section</td>
<td>3</td>
</tr>
<tr>
<td>Proton-pair reconstruction section</td>
<td>4</td>
</tr>
<tr>
<td>T1 part</td>
<td>4</td>
</tr>
<tr>
<td>T2 part</td>
<td>4</td>
</tr>
<tr>
<td>Fetching data from Ntuple</td>
<td>5</td>
</tr>
</tbody>
</table>
TOTEM Ntuple

Description of the TOTEM ntuple structure.

Metadata part

branch: event_info.

```c
struct EventMetaData
{
    unsigned long run_no;  ///< run number in form [run number]*1E4 + [raw-data file index]
    unsigned long event_no;///< event number assigned by CMSSW (RawDataSource), counts from 1
    unsigned long daq_event_number;///< event number assigned by DAQ
    unsigned long long timestamp;///< timestamp of the event (UNIX timestamp), is resolution 1s
    std::vector<unsigned int> optoRx_Id;///< ID of a given OptoRx (the index of the array)
    std::vector<unsigned int> optoRx_BX;///< bunch-crossing number reported by a given OptoRx
    std::vector<unsigned int> optoRx_LV1;///< LV1 as reported by a given OptoRx
};
```

Trigger part

branch: trigger_data.

(the data from LoneG)

```c
struct TriggerData
{
    unsigned char type;///<
    unsigned int event_num;///< incremental counter of triggers accepted by DAQ (thus event counter)
    unsigned int bunch_num;///< the number of bunch(-pair) collided in this event
    unsigned int src_id;///<
    unsigned int orbit_num;///<
    unsigned char revision_num;///<
    unsigned int run_num;///< the run number (without the raw-file index extension)
    unsigned int trigger_num;///< incremental trigger counter
    unsigned int inhibited_triggers_num;///< incremental counter of triggers rejected by DAQ
    unsigned int input_status_bits;///< result of the trigger logic (each bit corresponds to one entry in the trigger menu)
};
```

The meaning of the bits in `input_status_bits` above is defined by the following table:

<table>
<thead>
<tr>
<th>bit</th>
<th>trigger type</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.</td>
<td>RP220_Vert</td>
</tr>
<tr>
<td>1.</td>
<td>RP220_Horiz</td>
</tr>
<tr>
<td>2.</td>
<td>RP220_Cross</td>
</tr>
<tr>
<td>3.</td>
<td>TTBB</td>
</tr>
<tr>
<td>4.</td>
<td>CMS &amp; L1SA</td>
</tr>
<tr>
<td>5.</td>
<td>T2_single arm</td>
</tr>
<tr>
<td>6.</td>
<td>T2</td>
</tr>
<tr>
<td>7.</td>
<td>T2_HighMultiplicity</td>
</tr>
<tr>
<td>8.</td>
<td>T1</td>
</tr>
<tr>
<td>9.</td>
<td>BC0</td>
</tr>
<tr>
<td>10.</td>
<td>T2_LM</td>
</tr>
<tr>
<td>11.</td>
<td>L1SA</td>
</tr>
</tbody>
</table>
Roman Pot part

Below, [RP] stands for RP numerical ID (e.g. 120 for 56-near-top). * RP Numbering Scheme:

```
Sector 45
```

```
Numbering of Roman Pots.
```

Digi section (misleading name, this branch refers to clusters)

branches: digi_rp_[RP].

```
struct RPRootDumpDigiInfo
{
    std::vector<int> numberOfClusters;///< number of clusters in a given plane (indexed from 0 to 9)
    unsigned int numberOfPlanesOn;///< number of planes with at least one cluster
    unsigned int uPlanesOn;///< number of U planes with at least one cluster
    unsigned int vPlanesOn;///< number of V planes with at least one cluster
    std::vector<int> planeId;///< plane ID for a given cluster (array index)
    std::vector<int> clusterSize;///< cluster size of a given cluster
    std::vector<int> centralStrip;///< central strip of a given cluster
};
```
Pattern-recognition section

branches: par_patterns_rp_[RP]. (parallel/road search algorithm)
branches: nonpar_patterns_rp_[RP]. (non-parallel/Hough-transform search algorithm)

Each of the branches has the following structure:

```c++
struct RPRootDumpPatternInfo {
    std::vector<RPRootDumpPattern> u, v; ///< arrays of recognized patterns in u and v projections
    bool fittable;                        ///< whether there is one (and only one) combined u-v pat
};
```

The u and v array elements (linear patterns) are described by:

```c++
struct RPRootDumpPattern {
    double a; ///< slope in rad
    double b; ///< intercept (at the middle of the RP) in mm
    double w; ///< weight
};
```

Track section

branches: track_rp_[RP].

```c++
struct RPRootDumpTrackInfo {
    bool valid;                       ///< whether track fit is valid
    double x, y, z;                   ///< track fit interpolated to the middle of the RP
    double chi2;                      ///< fit chi square
    double chi2ndf;                   ///< fit chi square divided by the number of degrees of freedom
    unsigned int entries;             ///< the number of contributing hits
    double res_x, res_y;              ///< seem not used
    std::vector<int> u_sect, v_sect;  ///< list of active trigger sectors calculated from (strip) data
    int u_sect_no, v_sect_no;         ///< sizes of u_sect and v_sect vectors
};
```

Multitrack section

branches: multi_track_rp_[RP]

Each of the possible track (u-v) combinations is listed in the following array:

```c++
vector<RPRootDumpTrackInfo>
```

Single-proton reconstruction section

branches: rec_prot_[left/right].

```c++
struct RPRootDumpReconstructedProton {
    bool valid;
    double thx, thy, phi, t, tx, ty, xi, x0, y0, chi2, chindf;
};
```
Proton-pair reconstruction section

branch: rec_prot_pair.

```cpp
struct RPRootDumpReconstructedProtonPair {
  bool valid;
  double thxr, thyr, xir, phir;
  double thxl, thyl, xil, phil;
  double x0, y0, z0, chi2, chindf;
  double tr, txr, tyr;
  double tl, txl, tyl;
  double t;
};
```

T1 part

T2 part

```cpp
std::vector<int> Pad_row; // pad row (0..24)
std::vector<int> Pad_col; // pad column (0..63)
std::vector<int> Pad_det; // symbolic id of the detector containing the pad:
                        // 0..9: planes in the Plus Near quarter;
                        // 10..19: planes in the Plus Far quarter
                        // 20..29: planes in the Minus Near quarter
                        // 30..39: planes in the Minus Far quarter

std::vector<int> Strip_row; // strip row (0..255)
std::vector<int> Strip_col; // strip column (0..1)
std::vector<int> Strip_det; // symbolic id of the detector containing the strip, same
std::vector<double> TrkEta_XY; // track eta calculated from the polar angle where the XZ and YZ projection are used.
std::vector<double> TrkZmin_XY; // Z value of the minimum approach distance of the track from the axis.
std::vector<double> TrkRmin_XY; // the corresponding distance.
std::vector<double> TrkAx; // slope of the track projection in the XZ plane
std::vector<double> TrkAy; // slope of the track projection in the YZ plane
std::vector<double> TrkX0; // intercept of the track projection in the XZ plane
std::vector<double> TrkY0; // intercept of the track projection in the YZ plane
std::vector<double> TrkPhi; // phi of the track obtained using the TrkAy and TrkAx.
// For secondaries can be different from the
```
std::vector<int> TrkNumHitInH2; // Number of hits from the quarter MN,
std::vector<int> TrkNumHitInH3; // Number of hits from the quarter MF,
std::vector<double> TrkEta2;   // Eta of the track obtained as an average of the hit eta (assuming
std::vector<double> TrkChiProb; // Track Chi^2 probability
std::vector<double> ProbChi2R_rz; // Track Chi^2 probability for the RZ fit
std::vector<double> Chi2Rreduced_rz; // Reduced Chi2 for the RZ fit
std::vector<double> HitPhi;      // Phi position of all the Hits (degree)
std::vector<double> HitR;        // R position of all the Hits (mm)
std::vector<double> HitType;     // 0-> only pad; 1-> only strip 2->Class 1 Hit (superimposition
std::vector<double> HitNumPad;   // Cluster Pad Size
std::vector<double> HitNumStrip;  // Cluster Strip Size
std::vector<double> TrkEntryX;   // Track X Entry point
std::vector<double> TrkEntryY;   // Track Y Entry point
std::vector<double> TrkEntryZ;   // Track Z Entry point
std::vector<double> TrkExitX;     // Track X Exit point
std::vector<double> TrkExitY;     // Track Y Exit point
std::vector<double> TrkExitZ;     // Track Z Exit point

Warning: to limit the size of the ntuple, it is possible that some field (hit, pad, strip collections) are missing.

**Fetching data from Ntuple**

- **Totem files**

  TFile *totemFile = TFile::Open(totemFileName.c_str()); // opening input files
  TTree *tree_totem = (TTree *) totemFile->Get("TotemNtuple");
  checkAndGetBranch(tree_totem, "trigger_data")→SetAddress(&trigData);
  int totemSize = tree_totem→GetEntries();
  for (int tot_i = 0; tot_i < totemSize; tot_i++) {
    tree_totem→GetEntry(tot_i);
    cout << "Data in trigData " << tot_i << " is equal " << trigData→nameOfTheDataFromTrigData;
  }

- **CMS files**

  TFile *cmsFile = TFile::Open(cmsFinalFileName.c_str());
  TTree *tree_cms = (TTree *) cmsFile→Get("evt");
  MyEvtId *evtcmsUA = nullptr;
  checkAndGetBranch(tree_cms, "evtId")→SetAddress(&evtcmsUA);
  int cmsSize = tree_cms→GetEntries();
  for (unsigned int cms_i = 0; cms_i < cmsSize; cms_i++) {
    tree_cms→GetEntry(cms_i);
    cout << "Data in evtcmsUA " << cms_i << " is equal " << evtcmsUA→nameOfTheDataFromEvtcmsUA;
  }

- **Checking and getting branch (checkAndGetBranch method from example above)**

  TBranch *checkAndGetBranch(TTree *tree, string branchName) {
    TBranch *branch = tree→GetBranch(branchName.c_str());
    if (!branch) {
      string dotBranchName = branchName + ".";
      branch = tree→GetBranch(dotBranchName.c_str());
    }
    if (!branch) {
      tree→Print();
    }
  }
CompNtuple < TOTEM < TWiki

    error(" No data branch " + branchName + " found in input file!");

} return branch;

This topic: TOTEM > CompNtuple
Topic revision: r9 - 2016-09-21 - JakubSebastianBujas

Copyright &© 2008-2019 by the contributing authors. All material on this collaboration platform is the property of the contributing authors.
Ideas, requests, problems regarding TWiki? Send feedback