Table of Contents

L1Extra Classes

Complete:

Introduction

For the HLT and subsequent offline analysis, it is convenient to deal with physical representations of the hardware trigger objects, rather than the hardware representations themselves. In other words, one would like physically meaningful values of phi, eta, transverse energy, etc., not the integer values used in the trigger hardware. The L1Extra classes, defined in `DataFormats/L1Trigger` and contained in the `l1extra` namespace, provide these physical representations.

Class Definitions

The following classes inherit from `LeafCandidate` (found in `DataFormats/Candidate`). Thus, all the L1 objects are represented by a Lorentz vector, electric charge, and vertex position. One can therefore access directly, for instance, the ϕ, or E_T of each object (via the phi() and et() member functions). The vertex is always set to the origin (0,0,0), and the charge is set to 0 for e/γ, jet, and MET objects; it is only meaningful for muons. There are also accessors for the corresponding hardware objects. The ϕ and η values assigned are the average of the two bin boundaries. So, for η, this is *not* the mechanical center of the η bin. For E_T, the assigned value is the *low* edge of the bin.

For an example of how to access these objects, see L1Trigger/L1ExtraTestAnalyzer. In releases prior to `CMSSW_1_3_0_pre3`, check out tag V01-04-03 to get the .cfi file in the example below.

- `L1EmParticle`
  - Enum describing whether the object is isolated or non-isolated (use the `type()` member function).
  - Corresponding hardware class: `L1GctEmCand` (in `DataFormats/L1CMS.GlobalCaloTrigger`).
- `L1JetParticle`
  - Enum describing whether the object is central, forward, or τ (use the `type()` member function).
  - Corresponding hardware class: `L1GctJetCand` (in `DataFormats/L1CMS.GlobalCaloTrigger`).
- `L1MuonParticle`
  - Status of isolation and MIP bits (use the `isIsolated()` and `isMip()` member functions).
  - Corresponding hardware class: `L1MuGMTCand` (in `DataFormats/L1CMS.GlobalMuonTrigger`).
- `L1EtMissParticle`
  - Even though EtMiss is really only a 2-vector, it is represented by a 4-vector for uniformity. The `etMiss()` member function gives the magnitude of the 2-vector.
  - This class is used to represent both MET and MHT. They are distinguished by the `type()` member function, which returns either `kMET` or `kMHT`. [Not available before `CMSSW_3_1_0`.]
  - Global energy sum, `etTotal()`.
    - Before `CMSSW_3_1_0`, there was no L1 MHT, so `L1EtMissParticle` also had a member function `etHad()` that gave total HT, not HCAL energy. This is now given by `etTotal()` for `type()` = kMHT==.
  - Corresponding hardware classes: `L1GctEtMiss`, `L1GctEtTotal`, `L1GctHtMiss`, and `L1GctEtHad` (see DataFormats/L1CMS.GlobalCaloTrigger/interface). For `type()` = kMET=, `L1GctHtMiss` and `L1GctEtHad` are left empty. Similarly, for `type()` = kMHT=, `L1GctEtMiss` and `L1GctEtTotal` are left empty.
    - Before `CMSSW_3_1_0`, there was no L1 MHT, so `L1GctHtMiss` is not available.
- `L1HFRings`
  - Et sums and bit counts in the four HF rings.

In addition, there is a class that provides lists of objects and object combinations that fired a given trigger:

- `L1ParticleMap`
  - **Deprecated beginning with CMSSW_1_7_0 in favor of GlobalTriggerObjectMapRecord**.
  - Name of trigger (string). This string is not stored; instead, the trigger index, which is stored, is mapped to the name via an enum and a static array of strings.
  - Lists of objects that fired the trigger, either by themselves (for single-object triggers) or in combination with another object (for multi-object triggers). The lists of e/γ, jet, and μ candidates are stored separately.
  - A reference to the global `L1EtMissParticle` object. This reference is null if the global quantities were not used in the trigger.
  - A list of object types (i.e. {e/γ, τ jet, non-τ jet, μ, MET, total E_T, and hadronic E_T}, indexed by an enum) used by the trigger.
  - A list of object *combinations* (e.g. μ-pairs) that fired the trigger. Each combination is given by a vector of indices into the particle lists. The list of object types described in the previous bullet tells which particle list to use for each index. For global objects (MET, total E_T, and hadronic E_T), the index is always 0.
  - Functions for navigating this list of object combinations, returning pointers to the L1Extra objects, of type `L1PhysObjectBase` or its subclasses.
  - The index of each `L1ParticleMap` in the `L1ParticleMapCollection` corresponds to the enum `L1ParticleMap::L1TriggerType` (defined in `DataFormats/L1Trigger/L1ParticleMap.h`).
  - **Example usage:**

```
// Check if the double tau trigger fired.
Handle< L1ParticleMapCollection > mapColl ;
iEvent.getByLabel( particleMapSource_, mapColl ) ;
const L1ParticleMap& doubleTauMap = ( *mapColl )[ L1ParticleMap::kDoubleTau ] ;
bool singleTauFired = doubleTauMap.triggerDecision() ;

// Get the tau candidates that are part of a successful pair.
const L1JetParticleVectorRef& triggeredTaus = doubleTauMap.jetParticles() ;

// Get the successful *pairs* of tau candidates.
const L1IndexComboVector& triggeredTauPairs = indexCombos() ;

// Loop over successful tau pairs.
int pairCounter = 0 ;
for( L1ParticleMap::L1IndexComboVector::const_iterator pairItr = triggeredTauPairs.begin
      pairItr != triggeredTauPairs.end() ;
      ++pairItr )
{
    cout << "Pair #" << pairCounter++ << ": " << endl ;

    // Each pair is a vector of indices into the jetParticles() vector.
    for( L1ParticleMap::L1IndexCombo::const_iterator indexItr = pairItr->begin() ;
          indexItr != pairItr->end() ;
          ++indexItr )
    {
        cout << "   Jet #" << *indexItr << ", ET = " << triggeredTaus[ *indexItr ]->et()
    }
 }
```

How To Produce `L1Extra` Objects

MC truth emulation

For immediate HLT development using CSA06 MC, follow these instructions for running `L1ExtraFromMCTruthProd`, which makes objects of the above classes, based on 4-vectors at generator level. The actual EDProducts that are generated depend on code version and are listed on the respective instructions pages.

Hardware emulation

Beginning with the `CMSSW_1_2_0` series, it is possible to run both calorimeter and muon trigger emulators and produce L1Extra objects from their output. The whole emulation chain is controlled by a single master .cff file, `L1Trigger/L1ExtraFromDigis/data/l1extra.cff`.

Example .cfg file:

```
process TRIG = {
   source = PoolSource {
      untracked vstring fileNames = { 'file:myFile.root' }
      untracked int32 maxEvents = 500
   }

   include "L1Trigger/L1ExtraFromDigis/data/l1extra.cff"
   include "L1Trigger/L1ExtraTestAnalyzer/data/l1extratest.cfi"

   path l1Emulator = {l1emulator,l1extra,l1extratest}
}
```

Along with the hardware objects, the following EDProducts are put into the event. All the collections are disjoint, as in the hardware; any given trigger object appears in only one collection. So, for instance, non-isolated EM candidates *do not* include isolated EM candidates. However, so-called "RelaxedEM" triggers act on the logical OR of the Isolated and NonIsolated collections, which are concatenated by hand. Similarly, "Jet" triggers act on the logical OR of the Forward, Central, and Tau jet collections. By default, only objects for the central bunch crossing (bx = 0) are produced.

- `L1EmParticleCollection` with instance label "Isolated", module "l1extraParticles"
- `L1EmParticleCollection` with instance label "NonIsolated", module "l1extraParticles"
- `L1JetParticleCollection` with instance label "Forward", module "l1extraParticles"
- `L1JetParticleCollection` with instance label "Central", module "l1extraParticles"
- `L1JetParticleCollection` with instance label "Tau", module "l1extraParticles"
- `L1MuonParticleCollection`, module "l1extraParticles"
- `L1EtMissParticleCollection` with instance label "MET", module "l1extraParticles"
  - ♦ Before 31X, no instance label: =L1EtMissParticleCollection, module "l1extraParticles"
  - ♦ Before 18X, no collection: `L1EtMissParticle`, module "l1extraParticles"
- `L1EtMissParticleCollection` with instance label "MHT", module "l1extraParticles" [absent prior to 31X]
- `L1HFRingsCollection`, module "l1extraParticles" [absent prior to 31X]
- `L1ParticleMapCollection` (one `L1ParticleMap` per L1 trigger), module "l1extraParticleMap" [available only before 17X]
- Before 17X, there was also a `L1CMS.GlobalTriggerReadoutRecord` (defined in `DataFormats/L1GlobalTrigger`), which gives the global L1 decision, module "l1extraParticleMap". Beginning in 17X, this object is produced by the GT emulator with module "gtDigis".

`CMSSW_1_7_0`

Beginning with CMSSW_1_7_0, we will no longer produce L1ParticleMaps in standard production. Instead, the GT emulator will be used to evaluate L1 triggers and to produce the GlobalTriggerObjectMapRecord. The

default GT trigger menu implemented in 170 can be seen here.

CMSSW_1_7_0_pre1

See here for the trigger tables implemented temporarily in L1ParticleMapProd.

CMSSW_1_6_0

See here for the trigger tables implemented temporarily in L1ParticleMapProd.

CMSSW_1_6_0_pre8

See here for the trigger tables implemented temporarily in L1ParticleMapProd.

CMSSW_1_6_0_pre5

See here for the trigger tables implemented temporarily in L1ParticleMapProd.

CMSSW_1_6_0_pre4

See here for the trigger tables implemented temporarily in L1ParticleMapProd.

CMSSW_1_3_1_HLTX

See here for the trigger table implemented temporarily in L1ParticleMapProd.

Instructions for CMSSW_1_3_0_pre3

```
cvs co -r V00-01-06 DataFormats/L1CMS.GlobalMuonTrigger
cvs co -r V04-00-02 DataFormats/L1CMS.GlobalTrigger
cvs co -r V01-09-03 DataFormats/L1Trigger
cvs co -r V00-01-05 DataFormats/CMS.L1DTTrackFinder
cvs co -r V00-02-04 L1Trigger/DTTrackFinder
cvs co -r  V00-00-11 L1Trigger/RegionalCaloTrigger
```

Instructions for CMSSW_1_3_0_pre1

This release came out before CMSSW_1_2_2, so the following tags are needed:

```
cvs co -r V00-00-07 L1Trigger/RegionalCaloTrigger
cvs co -r V00-03-03 L1Trigger/CSCTrackFinder
cvs co -r V00-03-00 L1Trigger/L1ExtraFromDigis
cvs co -r V00-03-04 SimCalorimetry/EcalTrigPrimAlgos
cvs co -r V00-05-06 SimCalorimetry/EcalTrigPrimProducers
```

Instructions for CMSSW_1_2_2, CMSSW_1_3_0_pre2, and CMSSW_1_3_0_pre3

```
cvs co -r V00-03-04 SimCalorimetry/EcalTrigPrimAlgos
cvs co -r V00-05-06 SimCalorimetry/EcalTrigPrimProducers
```

The default trigger table is the same as for L1ExtraFromMCTruthProd. See here.

Instructions for CMSSW_1_2_0

To pick up improvements scheduled for CMSSW_1_2_2, check out and compile the following tags:

```
cvs co -r V01-03-02 DataFormats/L1CaloTrigger
```

```
cvs co -r V02-03-05 DataFormats/L1CMS.GlobalCaloTrigger
cvs co -r V00-03-00 SimCalorimetry/EcalTrigPrimAlgos
cvs co -r V00-05-00 SimCalorimetry/EcalTrigPrimProducers
cvs co -r V01-04-03 SimCalorimetry/HcalSimAlgos
cvs co -r V01-02-09 SimCalorimetry/HcalSimProducers
cvs co -r V00-01-04 CalibCalorimetry/CaloTPG
cvs co -r V00-01-04 CalibCalorimetry/HcalTPGAlgos
cvs co -r V01-01-00 CalibFormats/CaloTPG
cvs co -r V00-00-07 L1Trigger/RegionalCaloTrigger
cvs co -r V01-06-07 L1Trigger/CMS.GlobalCaloTrigger
cvs co -r V00-01-07 L1Trigger/L1Scales
cvs co -r V00-00-06 L1Trigger/L1ScalesProducers
cvs co -r V01-02-05 Geometry/HcalTowerAlgo
cvs co -r V00-01-13 L1Trigger/RPCTrigger
cvs co -r V00-03-03 L1Trigger/CSCTrackFinder
cvs co -r V00-03-00 L1Trigger/L1ExtraFromDigis
```

The default trigger table is the same as for `L1ExtraFromMCTruthProd`. See here.

Instructions for `CMSSW_1_2_0_pre2`

The `l1extra.cff` file is, unfortunately, not in this release, so it needs to be checked out:

```
cvs co -r V00-01-01 L1Trigger/L1ExtraFromDigis/data/l1extra.cff
```

In order to use the post-release calorimeter bug fixes (recipe given here), a different version of the .cff file is needed:

```
cvs co -r V00-01-04 L1Trigger/CMS.GlobalMuonTrigger/data/l1muon.cff
cvs co -r V00-01-02 L1Trigger/L1ExtraFromDigis/data/l1extra.cff
```

Also, when writing out events, the `PoolOutputModule` problem documented here can be circumvented by including the following lines in the .cfg file:

```
untracked vstring outputCommands =
{
   "keep *",
   "drop recoCandidatesOwned_*_*_*"
}
```

Known issues:

- Mismatched LUTs: RCT vs. L1Extra.
- Lots of debug output from RCT.
- Currently, no access to hardware muons from L1Extra. To aid developers, some HW bits [isFwd(), isRPC(), detector()] added temporarily to `L1MuonParticle`.
- GT not yet incorporated into L1Extra. `L1ExtraParticleMapProd` evaluates trigger conditions, generates particle <-> trigger maps. Same as `L1ExtraFromMCTruth`; performance should be identical to GT hardware emulator, with the same default trigger table (see here).
- Missing HF towers: generated by TPG but disappear in TPG-RCT interface.
- CSC tracks shifted by one bx, so they are not merged with forward RPC candidates.

Review Status

Editor/Reviewer and date	Comments
Main.wsun - 02 Aug 2006	Page author

Responsible: Main.wsun
Last reviewed by:

This topic: CMSPublic > SWGuideL1Extra
Topic revision: r32 - 2009-03-29 - WernerSun