CMSSW/ DataFormats/ EcalRecHit/ src/ EcalRecHit.cc

```
001 #include "DataFormats/EcalRecHit/interface/EcalRecHit.h"
002 #include "DataFormats/EcalDetId/interface/EBDetId.h"
003 #include "DataFormats/EcalDetId/interface/EEDetId.h"
004 #include "DataFormats/EcalDetId/interface/ESDetId.h"
005 #include "FWCore/MessageLogger/interface/MessageLogger.h"
006 #include <cassert>
007 #include <math.h>
008
009 #include <iostream>
010
011 EcalRecHit::EcalRecHit() : CaloRecHit(), flagBits_(0) {
012 }
013
014 EcalRecHit::EcalRecHit(const DetId& id, float energy, float time, uint32_t flags, uint32_t fl
015   CaloRecHit(id,energy,time,flags),
016   flagBits_(flagBits)
017 {
018 }
019
020 bool EcalRecHit::isRecovered() const {
021         return (    recoFlag() == kLeadingEdgeRecovered
022                  || recoFlag() == kNeighboursRecovered
023                  || recoFlag() == kTowerRecovered
024                 );
025 }
026
027 float EcalRecHit::chi2() const
028 {
029         uint32_t rawChi2 = 0x7F & (flags()>>4);
030         return (float)rawChi2 / (float)((1<<7)-1) * 64.;
031 }
032
033 float EcalRecHit::chi2Prob() const
034 {
035         std::cerr << "Please retrieve the chi2 value instead of its probability.\n";
036         std::cerr << "Use the method EcalRecHit::chi2() for that purpose.\n";
037         std::cerr << "The chi2 is still in a commissioning phase and further\n";
038         std::cerr << "actions will be taken in the future. Thanks.\n";
039         assert(false);
040         uint32_t rawChi2Prob = 0x7F & (flags()>>4);
041         return (float)rawChi2Prob / (float)((1<<7)-1);
042 }
043
044 float EcalRecHit::outOfTimeChi2Prob() const
045 {
046         std::cerr << "Please retrieve the chi2 value instead of its probability.\n";
047         std::cerr << "Use the method EcalRecHit::outOfTimeChi2() for that purpose.\n";
048         std::cerr << "The chi2 is still in a commissioning phase and further\n";
049         std::cerr << "actions will be taken in the future. Thanks.\n";
050         assert(false);
051         /*
052         uint32_t rawChi2Prob = 0x7F & (flags()>>24);
053         return (float)rawChi2Prob / (float)((1<<7)-1);
054         */
055         return -1; // will never get here
056 }
057
058 float EcalRecHit::outOfTimeChi2() const
059 {
060         uint32_t rawChi2Prob = 0x7F & (flags()>>24);
061         return (float)rawChi2Prob / (float)((1<<7)-1) * 64.;
062 }
063
064 float EcalRecHit::outOfTimeEnergy() const
```

```
065 {
066         uint32_t rawEnergy = (0x1FFF & flags()>>11);
067         uint16_t exponent = rawEnergy>>10;
068         uint16_t significand = ~(0xE<<9) & rawEnergy;
069         return (float) significand*pow(10,exponent-5);
070 }
071
072 void EcalRecHit::setRecoFlag( uint32_t flag )
073 {
074         setFlags( (~0xF & flags()) | (flag & 0xF) );
075 }
076
077 void EcalRecHit::setChi2Prob( float chi2Prob )
078 {
079         /* not used - store the raw chi2 instead */
080         std::cerr << "Please store the chi2 value instead of its probability.\n";
081         std::cerr << "Use the method EcalRecHit::setChi2() for that purpose.\n";
082         std::cerr << "The chi2 is still in a commissioning phase and further\n";
083         std::cerr << "actions will be taken in the future. Thanks.\n";
084         assert(false);
085         /*
086         if ( chi2Prob < 0 || chi2Prob > 1 ) {
087                 edm::LogWarning("EcalRecHit::setChi2Prob") << "chi2Prob outside limits [0, 1]
088         } else {
089                 // use 7 bits
090                 uint32_t rawChi2Prob = lround( chi2Prob * ((1<<7)-1) );
091                 // shift by 4 bits (recoFlag)
092                 setFlags( (~(0x7F<<4) & flags()) | ((rawChi2Prob & 0x7F)<<4) );
093         }
094         */
095 }
096
097 void EcalRecHit::setChi2( float chi2 )
098 {
099         // bound the max value of the chi2
100         if ( chi2 > 64 ) chi2 = 64;
101         // use 7 bits
102         uint32_t rawChi2 = lround( chi2 / 64. * ((1<<7)-1) );
103         // shift by 4 bits (recoFlag)
104         setFlags( (~(0x7F<<4) & flags()) | ((rawChi2 & 0x7F)<<4) );
105 }
106
107 void EcalRecHit::setOutOfTimeEnergy( float energy )
108 {
109         if ( energy > 0.001 ) {
110                 uint16_t exponent = lround(floor(log10(energy)))+3;
111                 uint16_t significand = lround(energy/pow(10,exponent-5));
112                 // use 13 bits (3 exponent, 10 significand)
113                 uint32_t rawEnergy = exponent<<10 | significand;
114                 // shift by 11 bits (recoFlag + chi2)
115                 setFlags( ( ~(0x1FFF<<11) & flags()) | ((rawEnergy & 0x1FFF)<<11) );
116         }
117 }
118
119 void EcalRecHit::setOutOfTimeChi2Prob( float chi2Prob )
120 {
121         /* not used - store the raw chi2 instead */
122         std::cerr << "Please store the chi2 value instead of its probability.\n";
123         std::cerr << "Use the method EcalRecHit::setOutOfTimeChi2() for that purpose.\n";
124         std::cerr << "The chi2 is still in a commissioning phase and further\n";
125         std::cerr << "actions will be taken in the future. Thanks.\n";
126         assert(false);
127         /*
128         if ( chi2Prob < 0 || chi2Prob > 1 ) {
129                 edm::LogWarning("EcalRecHit::setOutOfTimeChi2Prob") << "chi2Prob outside limi
130         } else {
131                 // use 7 bits
```

```
132                 uint32_t rawChi2Prob = lround( chi2Prob * ((1<<7)-1) );
133                 // shift by 24 bits (recoFlag + chi2 + outOfTimeEnergy)
134                 setFlags( (~(0x7F<<24) & flags()) | ((rawChi2Prob & 0x7F)<<24) );
135         }
136         */
137 }
138
139
140 void EcalRecHit::setOutOfTimeChi2( float chi2 )
141 {
142         // bound the max value of chi2
143         if ( chi2 > 64 ) chi2 = 64;
144         // use 7 bits
145         uint32_t rawChi2 = lround( chi2 / 64. * ((1<<7)-1) );
146         // shift by 24 bits (recoFlag + chi2 + outOfTimeEnergy)
147         setFlags( (~(0x7F<<24) & flags()) | ((rawChi2 & 0x7F)<<24) );
148 }
149
150
151 void EcalRecHit::setTimeError( uint8_t timeErrBits )
152 {
153         // take the bits and put them in the right spot
154         setAux( (~0xFF & aux()) | timeErrBits );
155 }
156
157
158 float EcalRecHit::timeError() const
159 {
160         uint32_t timeErrorBits = 0xFF & aux();
161         // all bits off --> time reco bailed out (return negative value)
162         if( (0xFF & timeErrorBits) == 0x00 )
163                 return -1;
164         // all bits on  --> time error over 5 ns (return large value)
165         if( (0xFF & timeErrorBits) == 0xFF )
166                 return 10000;
167
168         float LSB = 1.26008;
169         uint8_t exponent = timeErrorBits>>5;
170         uint8_t significand = timeErrorBits & ~(0x7<<5);
171         return pow(2.,exponent)*significand*LSB/1000.;
172 }
173
174
175 bool EcalRecHit::isTimeValid() const
176 {
177         if(timeError() <= 0)
178           return false;
179         else
180           return true;
181 }
182
183
184 bool EcalRecHit::isTimeErrorValid() const
185 {
186         if(!isTimeValid())
187           return false;
188         if(timeError() >= 10000)
189           return false;
190
191         return true;
192 }
193
194 std::ostream& operator<<(std::ostream& s, const EcalRecHit& hit) {
195   if (hit.detid().det() == DetId::Ecal && hit.detid().subdetId() == EcalBarrel)
196     return s << EBDetId(hit.detid()) << ": " << hit.energy() << " GeV, " << hit.time() << " n
197   else if (hit.detid().det() == DetId::Ecal && hit.detid().subdetId() == EcalEndcap)
198     return s << EEDetId(hit.detid()) << ": " << hit.energy() << " GeV, " << hit.time() << " n
```

```
199   else if (hit.detid().det() == DetId::Ecal && hit.detid().subdetId() == EcalPreshower)
200     return s << ESDetId(hit.detid()) << ": " << hit.energy() << " GeV, " << hit.time() << " n
201   else
202     return s << "EcalRecHit undefined subdetector" ;
203 }
```

-- DavidCockerill - 23-Aug-2010

This topic: Main > EcalRecHitClass
Topic revision: r1 - 2010-08-23 - DavidCockerill