```
#ifndef Event_CalDigi_H
#define Event_CalDigi_H 1


// Include files
#include <iostream>
#include <vector>
#include "idents/CalXtalId.h"

#include "GaudiKernel/Kernel.h"
#include "GaudiKernel/StreamBuffer.h"
#include "GaudiKernel/ContainedObject.h"
#include "GaudiKernel/ObjectVector.h"
#include "Event/TopLevel/Definitions.h"



extern const CLID& CLID_CalDigi;

namespace Event {

/** @class CalDigi
 * @brief Transient Data Store class for CAL Digitizations.Actual readout data
 * is contained in nested class CalXtalReadout, holding ACD ADC values and gain ranges
 * from the two crystal ends. Can we describe the ADC and ranges a bit more?  Or at least point
to the CalXtalId enumerations?
 * A vector of CalXtalReadouts is contained in CalDigi,
 * via the typedef CalXtalReadoutCol. This allows up to 4 readouts per end. A collection of
 * CalDigis is typedefed as CalDigiCol
 *
 * Author:  E.Grove
 * $Header$
*/

class CalDigi : virtual public ContainedObject {

public:

/** @class CalXtalReadout
 * @brief Nested class in CalDigi for holding ACD values and gain ranges
 * from the two crystal ends.
 *
 * Author:  E.Grove
 *
```

```
*/

   class CalXtalReadout {  // : virtual public ContainedObject  {

   public:
      CalXtalReadout(char rangeP, unsigned short adcP, char rangeM, unsigned short adcM) :
        m_rangeP(rangeP),
           m_adcP(adcP),
           m_rangeM(rangeM),
           m_adcM(adcM)
        {};

        ~CalXtalReadout() {};


        /// retrieve pulse height from specified face
        inline unsigned short getAdc(idents::CalXtalId::XtalFace face) const {
                  return face == idents::CalXtalId::POS ? m_adcP : m_adcM;};

        /// retrieve energy range from specified face
        inline char getRange(idents::CalXtalId::XtalFace face) const {
                  return face == idents::CalXtalId::POS ? m_rangeP : m_rangeM;};


   private:

      /// ADC value from POSitive face
      unsigned short m_adcP;
      /// ADC value from NEGative face
      unsigned short m_adcM;
      /// gain range from POSitive face
      char m_rangeP;
      /// gain range from NEGative face
      char m_rangeM;

   };

   typedef std::vector<CalXtalReadout> CalXtalReadoutCol;

   /// shifts and masks for packed readout of energy range and Adc value
   enum {POS_OFFSET = 14,                                  // shift for POSitive
face
      RANGE_OFFSET = 12, RANGE_MASK = 0x3000,          // energy range bits
      ADC_VAL_MASK = 0xfff};                                  // Adc value bits

   CalDigi() {};
```

```
   CalDigi(idents::CalXtalId::CalTrigMode mode, idents::CalXtalId xtalId) :
      m_mode(mode), m_xtalId(xtalId) {};

   virtual ~CalDigi() { };

   /// Retrieve readout mode
   inline const idents::CalXtalId::CalTrigMode getMode() const { return m_mode; }

   /// Retrieve Xtal identifier
   inline const idents::CalXtalId& getPackedId() const { return m_xtalId; }

   /// initialize object all at once
   inline void initialize(idents::CalXtalId::CalTrigMode m, idents::CalXtalId id)
         { m_mode = m;  m_xtalId = id;}

   /// add readout to CalXtalReadout collection
   inline void addReadout(CalXtalReadout r) { m_readout.push_back(r); }

   const CalXtalReadoutCol& getReadoutCol() const { return m_readout;}

   /// Retrieve energy range for selected face and readout
   inline char getRange(short readoutIndex, idents::CalXtalId::XtalFace face) const
   {
      return (readoutIndex < m_readout.size()) ? ((m_readout[readoutIndex])).getRange(face) :
(char)-1;
   }

   /// Retrieve pulse height for selected face and readout
   inline short getAdc(short readoutIndex, idents::CalXtalId::XtalFace face) const
   {
      return (readoutIndex < m_readout.size()) ? ((m_readout[readoutIndex])).getAdc(face) :
(short)-1;
   }

   /// Retrieve ranges and pulse heights from both ends of selected readout
   inline const CalXtalReadout* getXtalReadout(short readoutIndex)
   {
      if ( readoutIndex < m_readout.size() )
         return &(m_readout[readoutIndex]);
      else
         return 0;

   }

   /// Retrieve pulse height from selected range
```

```
    inline short getAdcSelectedRange(char range, idents::CalXtalId::XtalFace face) const
    {
        char nRanges = (char)m_readout.size();
        if (nRanges == 1)
            return (range == ((m_readout[0])).getRange(face)) ? ((m_readout[0])).getAdc(face) :
(short)-1;
        else
            return ((m_readout[(nRanges + range - ((m_readout[0])).getRange(face)) %
nRanges])).getAdc(face);
    }

    /// Fill the ASCII output stream
    virtual std::ostream& fillStream( std::ostream& s ) const;


private:

    /// Cal readout mode is based on trigger type
    idents::CalXtalId::CalTrigMode m_mode;
    /// Cal ID
    idents::CalXtalId m_xtalId;
    /// ranges and pulse heights
    CalXtalReadoutCol m_readout;

};

typedef ObjectVector<CalDigi> CalDigiCol;

friend std::ostream& operator << (std::ostream& s, const CalDigi& obj)
        {
            return obj.fillStream(s);
        };

/// Fill the ASCII output stream
inline std::ostream& CalDigi::fillStream( std::ostream& s ) const
{
    char *modeStr;
    if (m_mode == idents::CalXtalId::BESTRANGE) {
        modeStr = "BESTRANGE";
    } else {
        modeStr = "ALLRANGE";
    }

    s << "class CalDigi"
    << " :"
    << "\n    Mode                    = " << modeStr
```

```
    << "\n    Xtal Id  (tower, layer, col)  = (" << m_xtalId.getTower() << ","
    << m_xtalId.getLayer() << "," << m_xtalId.getColumn() << ")"
    << "\n    Number of readouts            =  " << m_readout.size() << "\n";
    unsigned int ixtal;
    for (ixtal = 0; ixtal < m_readout.size(); ixtal++) {
        CalDigi::CalXtalReadout read = m_readout[ixtal];
        int rangeP = int(read.getRange(idents::CalXtalId::POS));
        int rangeM = int(read.getRange(idents::CalXtalId::NEG));
        unsigned int adcP = read.getAdc(idents::CalXtalId::POS);
        unsigned int adcM = read.getAdc(idents::CalXtalId::NEG);
        s << "Readout :"
            << "\n   POS (range, adc) = (" << rangeP
            << "," << adcP << ")"
            << "\n   NEG (range, adc) = (" << rangeM
            << "," << adcM << ")";
    }

    return s;

}



}

#endif
```