```
#ifndef __CALXTALRECALG_H
#define __CALXTALRECALG_H 1

#include "GaudiKernel/Algorithm.h"
#include "Event/Digi/CalDigi.h"
#include "Event/Recon/CalRecon/CalXtalRecData.h"
#include "GlastSvc/GlastDetSvc/IGlastDetSvc.h"




/** @class CalXtalRecAlg
 *  @brief  Calorimeter crystal reconstruction algorithm
 *
 * This algorithm reconstructs energy and position in each calorimeter crystal
 * It contains computeEnergy() and computePosition() methods for energy
 * and position reconstruction, respectively. See the descriptions.
 * of these methods for details
 *
 *  @author          A.Chekhtman
 *
 * $Header: /nfs/slac/g/glast/ground/cvs/CalRecon/src/CalXtalRecAlg.h,v 1.1 2002/06/13
20:40:36 chehtman Exp $
 */
class CalXtalRecAlg : public Algorithm
{
public:

   /// constructor
   CalXtalRecAlg(const std::string& name, ISvcLocator* pSvcLocator);
   virtual ~CalXtalRecAlg() {}

   StatusCode initialize();
   StatusCode execute();
   StatusCode finalize();
protected:
///  function for setting pointers to the input and output data in Gaudi TDS
   StatusCode retrieve();

private:
   /** @brief method to calculate energy deposited in a crystal
   *
   *    Energy for each crystal face is converted from adc value using simple
   *   linear formula:
   *
```

```
    *    \f[
           E = E_{max} * (ADC - PED)/(ADC_{max} - PED)
         \f]
    *   where \f$ E_{max} \f$ - maximum energy for used energy range,
    *  \f$ PED \f$ - pedestal, \f$ ADC_{max} \f$ - maximum ADC value.
    *
    * @param recData pointer to CalXtalRecData object to store reconstructed energy
    * @param digi pointer to CalDigi object with input data
    */
    void computeEnergy(Event::CalXtalRecData* recData,
                       const Event::CalDigi* digi);

    /** @brief method to calculate longitudinal position in a crystal
    *
    *   Position along the crystal direction is calculated from asymmetry between
    *   energies reconstructed from positive and negative faces of the crystal
    *
    *   \f[ pos = \frac{E_{pos} - E_{neg}}{E_{pos} + E_{neg}} *
                  \frac{1+lightAtt}{1-lightAtt} * \frac{L_{crystal}}{2}
         \f]
    *   where \f$ L_{crystal} \f$ - crystal length and \f$ lightAtt \f$ -
    *   the ratio of signal from "far" crystal face to the signal
    *   from"near" crystal face in the case when the energy deposition is close
    *   to one end of the crystal.
    *
    * @param recData pointer to CalXtalRecData object used as a source of input
    *        information on energy and to store the calculated position
    */
    void computePosition(Event::CalXtalRecData* recData);

private:
    /// pointer to input data collection in TDS
    Event::CalDigiCol* m_CcalDigiCol;

    /// pointer to the output data collection in TDS
    Event::CalXtalRecCol* m_cCalXtalRecCol;

    /// constants defining the position of the fields in VolumeIdentifier
    enum {fLATObjects, fTowerY, fTowerX, fTowerObjects, fLayer,
          fMeasure, fCALXtal,fCellCmp, fSegment};


       // constants defined in xml files
    int m_xNum;  ///< x tower number
    int m_yNum;  ///< y tower number
    int m_nTowers;  ///< total number of towers
```

```
    int m_eTowerCAL;
    ///< the value of fTowerObject field, defining calorimeter module
    int m_eLATTowers; ///< the value of fLATObjects field, defining LAT towers
    int m_cCALnLayer;  ///< number of CAL layers
    int m_nCsIPerLayer;  ///< number of Xtals per layer

    int m_eXtal;      ///< the value of fCellCmp field defining CsI crystal
    int m_nCsISeg;  ///< number of geometric segments per Xtal

    int m_eDiodeMSmall;
    ///< the value of fcellCmp field defining small diode at minus face

    int m_eDiodePSmall;
    ///< the value of fcellCmp field defining small diode at plus face

    int m_eDiodeMLarge;
    ///< the value of fcellCmp field defining large diode at minus face

    int m_eDiodePLarge;
    ///< the value of fcellCmp field defining large diode at plus face

    int m_eMeasureX;
    ///< the value of fmeasure field for crystals along X direction

    int m_eMeasureY;
    ///< the value of fmeasure field for crystals along Y direction

    int m_ePerMeV[2];  ///< gain - electrons/MeV 0=Small, 1=Large
    int m_noise[2];  ///< noise for diodes 0=Small, 1=Large units=electrons
    int m_pedestal;  ///< single pedestal
    int m_maxAdc;  ///< max value for ADC
    int m_thresh;  ///< zero suppression threshold
    double m_maxEnergy[4];  ///< highest energy for each energy range
    double m_lightAtt;  ///< light attenuation factor
    double m_cCsILength;  ///< Xtal length
    IGlastDetSvc* detSvc; ///< pointer to the Glast Detector Service

};


#endif
```