```
#ifndef _GlastDigi_CalDigiAlg_H
#define _GlastDigi_CalDigiAlg_H 1


~~// Include files~~
#include "GaudiKernel/Algorithm.h"
#include "GlastSvc/GlastDetSvc/IGlastDetSvc.h"
#include <vector>

/** @class CalDigiAlg
 * @brief Algorithm to convert from McIntegratingHit objects into
 * CalDigi objects and store them in the TDS. Combines contributions from
 * Xtal segments and accounts for light taper along the length of the Xtals.
 * Energies are converted to adc values after pedestal subtraction, and the
 * appropriate gain range is identified. Can we define ADC values as well as gain ranges or at
least point to the enumerations in CalXtalId?

 *
 * Author:  A.Chekhtman
 * $Header$
*/

class CalDigiAlg : public Algorithm {

public:

    CalDigiAlg(const std::string& name, ISvcLocator* pSvcLocator);

    StatusCode initialize();
    StatusCode execute();
    StatusCode finalize();

    //! pair of signals per Xtal. For SignalMap.
/** @class XtalSignal
 * @brief nested class of CalDigiAlg to separately hold the energy deposits in the crystal
 * and the diodes.Vector of diodes holds all 4 per crystal.
 *
 * Author:  A.Chekhtman
 *
*/
    class XtalSignal {
    public:
        XtalSignal();
        /// constructor given signal from both ends of xtal
s1 refers to POS and s2 refers to NEG? - could this be made explicit?
```

```
    XtalSignal(double s1, double s2);
    ~XtalSignal() {};
    ///  return signal from selected diode by specifying the face
    double getSignal(idents::CalXtalId::XtalFace~~nt~~ face) const {return m_signal[face];};
    ///  add to existing diode signals
    void addSignal(double s1, double s2);
    /// fetch diode energy, given the diode number
   How are diode ids assigned?  Could they be assigned to an enumeration?
    double getDiodeEnergy(int diode) const { return m_Diodes_Energy[diode];}
    /// add energy to the selected (already existing) diode
    void addDiodeEnergy(double ene, int diode) { m_Diodes_Energy[diode]+=ene;}
    /// set the (initial) energy for a diode
    void setDiodeEnergy(double ene, int diode) { m_Diodes_Energy[diode]=ene;}

  private:
    /// signal for both xtal faces (POS, NEG)
    double m_signal[2];
    /// direct energy depositions in 4 diodes of one xtal; vector contains all 4 diodes
    std::vector<double> m_d~~D~~iodes_Energy;
  };


  private:

    /// names for volume identifier fields
    enum {fLATObjects, fTowerY, fTowerX, fTowerObjects, fLayer,
      fMeasure, fCALXtal,fCellCmp, fSegment};


    /// local cache for constants defined in xml files
    /// x tower number
    int m_xNum;
    /// y tower number
    int m_yNum;
    /// total number of towers
    int m_nTowers;
    /// detModel identifier for CAL
    int m_eTowerCal;
    /// detModel identifier for LAT Towers
    int m_eLatTowers;
    /// number of layers (ie in z)
    int m_c~~C~~alNLayer;
    // number of Xtals per layer
    int m_nCsIPerLayer;
    int m_eXtal;
    /// number of geometric segments per Xtal
    int m_nCsISeg;
    /// detModel identifier for small minus-side diode
    int m_eDiodeMSmall;
    /// detModel identifier for small plus-side diode
    int m_eDiodePSmall;
    /// detModel identifier for large minus-side diode
    int m_eDiodeMLarge;
    /// detModel identifier for large plus-side diode
    int m_eDiodePLarge;
    /// detModel identifier for xtal measuring 'x'
    int m_eMeasureX;
    /// detModel identifier for xtal measuring 'y'
    int m_eMeasureY;
    /// gain - electrons/MeV 1=Small, 0=Large
    int m_ePerMeV[2];
    /// noise for diodes 1=Small, 0=Large units=electrons
    int m_noise[2];
    /// single pedestal
    int m_pedestal;  How is there only one pedestal?
    /// max value for ADC
    int m_maxAdc;
    /// zero suppression threshold
    double m_thresh;
    /// highest valid energy for each energy range
    double m_maxEnergy[4];
    /// light attenuation factor
    double m_lightAtt;
    /// Xtal length
    double m_c~~C~~sILength;

};


#endif // _GlastDigi_CalDigiAlg_H
```