Pds csPad

CS-Pad

Cornell-SLAC Pixel Array Detector

CsPad geometry, as described in ElementIterator.hh

```
//  Each "Element" represents one quadrant of a complete detector
//  and they are arranged as follows (viewed from upstream):
//  +---+---+
//  | 0 | 1 |
//  +---+---+
//  | 3 | 2 |
//  +---+---+
//
//  Each "Element" is composed of 8 "Section"s arranged as follows:
//  +---+---+-------+
//  |   |   |   6   |
//  + 5 | 4 +-------+
//  |   |   |   7   |
//  +---+---+---+---+---+   (for quadrant 0)
//  |   2   |   |   |
//  +-------+ 0 | 1 |
//  |   3   |   |   |
//  +-------+---+---+
//  The layout of each successive quadrant is rotated 90 degrees clockwise
//  with respect to the previous quadrant.
//
//  Each "Section" is composed of 2*194 rows by 185 columns with the following
//  orientations (for quadrant 0):
//    Sections 0,1: row index increases from bottom to top, column index increases from left to right
//    Sections 2,3: row index increases from left to right, column index increases from top to bottom
//    Sections 4,5: row index increases from top to bottom, column index increases from right to left
//    Sections 6,7: row index increases from left to right, column index increases from top to bottom
//  Again, the orientations of the Sections for quadrant 1 are rotated 90 degrees clockwise
//  and so on for each successive quadrant.
//
```

Here's a picture showing the approximate position and alignment of the sections in the CSPAD detector image. The value of each pixel is set to increase with increasing row and column number, so the blue corners are (row 0,column 0) in the section array.

Namespace CsPad

Enumerations

```
  enum           { MaxQuadsPerSensor = 4, ASICsPerQuad = 16 }
  enum           { RowsPerBank = 26, FullBanksPerASIC = 7, BanksPerASIC = 8,
                   ColumnsPerASIC = 185, MaxRowsPerASIC = 194 }
  enum           { PotsPerQuad = 80, TwoByTwosPerQuad = 4 }
  enum RunModes  { NoRunning, RunButDrop, RunAndSendToRCE, RunAndSendTriggeredByTTL,
                   ExternalTriggerSendToRCE, ExternalTriggerDrop, NumberOfRunModes }
  enum DataModes { normal = 0, shiftTest = 1, testData = 2, reserved = 3 }
```

class CsPadDigitalPotsCfg

Public Member Functions

```
         // Constructor
         CsPadDigitalPotsCfg ()
uint8_t  value (unsigned i) const
```

Public Attributes

```
uint8_t         pots [PotsPerQuad]
```

class CsPadReadOnlyCfg

Public Member Functions

```
     // Constructor
     CsPadReadOnlyCfg ()
```

Public Attributes

```
  uint32_t         shiftTest
  uint32_t         version
```

class CsPadGainMapCfg

Public Types

```
   typedef uint16_t         GainMap [ColumnsPerASIC][MaxRowsPerASIC]
```

Public Member Functions

```
    // Constructor
    CsPadGainMapCfg ()

    GainMap *  map ()
    const GainMap * map () const
```

Public Attributes

```
 GainMap         _gainMap
```

class ConfigV1QuadReg

Public Member Functions

```
// Constructors
ConfigV1QuadReg ()
ConfigV1QuadReg (uint32_t shiftSelect[],
                 uint32_t edgeSelect[],
                 uint32_t readClkSet,
                 uint32_t readClkHold,
                 uint32_t dataMode,
                 uint32_t prstSel,
                 uint32_t acqDelay,
                 uint32_t intTime,
                 uint32_t digDelay,
                 uint32_t ampIdle,
                 uint32_t injTotal,
                 uint32_t rowColShiftPer)

const uint32_t *         shiftSelect () const
const uint32_t *         edgeSelect () const
uint32_t         readClkSet () const
uint32_t         readClkHold () const
uint32_t         dataMode () const
uint32_t         prstSel () const
uint32_t         acqDelay () const
uint32_t         intTime () const
uint32_t         digDelay () const
uint32_t         ampIdle () const
uint32_t         injTotal () const
uint32_t         rowColShiftPer () const

      Pds::CsPad::CsPadReadOnlyCfg & ro ()
const Pds::CsPad::CsPadReadOnlyCfg & ro () const
      Pds::CsPad::CsPadDigitalPotsCfg & dp ()
const Pds::CsPad::CsPadDigitalPotsCfg & dp () const
      Pds::CsPad::CsPadGainMapCfg *         gm ()
const Pds::CsPad::CsPadGainMapCfg *         gm () const
      Pds::CsPad::CsPadReadOnlyCfg * readOnly ()
const Pds::CsPad::CsPadReadOnlyCfg * readOnly () const
```

class ConfigV1

Public Member Functions

```
// Constructors
ConfigV1 ()
ConfigV1 (uint32_t runDelay,
          uint32_t eventCode,
          uint32_t inactiveRunMode,
          uint32_t activeRunMode,
          uint32_t testDataIndex,
          uint32_t payloadPerQuad,
          uint32_t badAsicMask0,
          uint32_t badAsicMask1,
          uint32_t AsicMask,
          uint32_t quadMask)

 ConfigV1QuadReg *         quads ()
 const ConfigV1QuadReg *         quads () const
 uint32_t         tdi () const
 uint32_t         quadMask () const
 uint32_t         runDelay () const
 uint32_t         eventCode () const
 uint32_t         inactiveRunMode () const
 uint32_t         activeRunMode () const
 uint32_t         payloadSize () const
 uint32_t         badAsicMask0 () const
 uint32_t         badAsicMask1 () const
 uint32_t         asicMask () const
 uint32_t         numAsicsRead () const
 uint32_t         concentratorVersion () const
 uint32_t *         concentratorVersionAddr ()
```

Static Public Member Functions

```
 static const int         version ()
```

Static Public Attributes

```
 static const int         Version = 1
```

class ConfigV2

Public Member Functions

```
// Constructors
  ConfigV2 ()
  ConfigV2 (uint32_t runDelay,
            uint32_t eventCode,
            uint32_t inactiveRunMode,
            uint32_t activeRunMode,
            uint32_t testDataIndex,
            uint32_t payloadPerQuad,
            uint32_t badAsicMask0,
            uint32_t badAsicMask1,
            uint32_t AsicMask,
            uint32_t quadMask,            uint32_t roiMask)

   ConfigV1QuadReg *         quads ()
   const ConfigV1QuadReg *         quads () const
   uint32_t         tdi () const
   uint32_t         quadMask () const
   uint32_t     roiMask      (int iq) const
   uint32_t     runDelay     () const
   uint32_t         eventCode () const
   uint32_t         inactiveRunMode () const
   uint32_t         activeRunMode () const
   uint32_t         payloadSize () const
   uint32_t         badAsicMask0 () const
   uint32_t         badAsicMask1 () const
   uint32_t         asicMask () const
   uint32_t         numAsicsRead () const
   uint32_t         concentratorVersion () const
   uint32_t *         concentratorVersionAddr ()
```

Static Public Member Functions

```
static const int         version ()
```

Static Public Attributes

```
static const int         Version = 2
```

ElementHeader

Public Member Functions

```
    // Constructors
    ElementHeader ()

    // "unsigned" is shorthand for "unsigned int"
    unsigned         virtual_channel () const
    unsigned         lane () const
    unsigned         tid () const
    unsigned         acq_count () const
    unsigned         op_code () const
    unsigned         quad () const
    unsigned         seq_count () const
    unsigned         ticks () const
    unsigned         fiducials () const
    unsigned         sb_temp (unsigned i) const
    unsigned         frame_type () const
```

ElementV1

Inherits from ElementHeader.

Public Types

```
Version = 1
enum  { Version = 1 }
```

Public Member Functions

```
  // Constructors
  ElementV1 ()

  const uint16_t *         data () const
  const uint16_t *         pixel (unsigned asic, unsigned column, unsigned row) const
  const ElementV1 *         next (const ConfigV1 &) const
```

ElementV2

Inherits from ElementHeader.

Public Types

```
Version = 2
enum  { Version = 2 }
```

Public Member Functions

```
  // Constructors
  ElementV2 ()
```

ElementIterator

Iterates through Elements.

Public Types

```
  class Section {
  public:
    uint16_t pixel[ColumnsPerASIC][2*MaxRowsPerASIC];
  };
```

Public Member Functions

```
    ElementIterator();
    ElementIterator(const ConfigV1&, const Xtc&);
    ElementIterator(const ConfigV2&, const Xtc&);

    //  Iterate to the next Element/quadrant (0..3)
    const ElementHeader* next();
    //  Iterate to the next Section (0..7) within the current quadrant
    const Section* next(unsigned& sectionID);
```