ClearPET Project

List Mode Format Implementation: scanner geometry description Version 4.1

CONFIDENTIEL

Magalie KRIEGUER[1]
Luc SIMON[2]
Daniel STRUL[2]
Christian MOREL[2]
December 18, 2002
[1]*IIHE/VUB*
[2]*IPHE/UNIL*

Introduction

This document defines the choices that we have made to describe the scanner geometry of generic cylindrical systems.
Each scanner differs from the others on one or several points, such as:

- radiation detector: scintillator, APD;
- full ring, incomplete ring, hexagon, detector plate;
- single or multi-ring systems;
- with or without inter-ring septa ...

And more precisely we want to describe the implementation of the C functions that allow to calculate the position of events by using the LMF header file (.cch) and the LMF binary file (.ccs).

1 The List Mode Format and the computation of the event positions

All the information required to calculate the event positions is stored in the LMF binary file and in the LMF ASCII header file. This first file contains a description of the scanner topology:

- the number of rings,
- the number of sectors,
- the axial and tangential numbers of modules per rsector,
- the axial and tangential numbers of submodules per module,
- the axial and tangential numbers of crystals per submodule,
- the axial and radial numbers of layers per crystal.

This data is coded in the LMF encoding header and is followed by binary records.
The second file, the LMF ASCII header file (.cch) contains a list of floating point and string information about the scan, the acquisition, and the animal. But this file must also integrate data about the scanner geometry and detector material.

Example of LMF ASCII header file:

```
scan date:  Feb/14/2001
scanner identification:  ClearPET GLS
geometrical design type[1]:1
x shift ring 0:  4 cm
z shift sector 1 mod 2:  1 cm
ring diameter:  100 cm
rsector axial pitch:  35 cm
rsector azimuthal pitch:  45 degree
rsector tangential size:  33 cm
rsector axial size:  44 cm
module axial size:  42 cm
module tangential size:  12 cm
module axial pitch:  10 cm
module tangential pitch:  15 cm
submodule axial size:  10 cm
submodule tangential size:  10 cm
submodule axial pitch:  4 cm
submodule tangential pitch:  4 cm
crystal axial size:  0.2 cm
crystal tangential size:  0.2 cm
crystal radial size:  0.5 cm
crystal axial pitch:  0.25 cm
crystal tangential pitch:  0.25 cm
layer0 radial size:  0.1 cm
layer1 radial size:  0.1 cm
in layer0 interaction length:  0.05 cm
in layer1 interaction length:  0.05 cm
azimuthal step:  2 degree
axial step:  1 cm
clock time step:  0.1 s
energy step:  5 keV
```

The local variable **scannerGeometry** of the type LMF_scanner_geometry will put together all generic information about the scanner geometry and the table called ppShiftValuesList will collect all the data that concern the shift of one or more structures (ring, sector or rsector) of the scanner according to the x-axis or to the y-axis or to the z-axis. In the example above, the scanner consists of 2 rings with each 8 rsectors. Each rsector has 3 rows of modules tangentially and only one column of modules axially. Each module is divided in 4 columns of submodules axially and 1 row tangentially. Finally each submodule is divided in a matrix of 8 rows of 8 columns of crystals. We

[1] geometrical design type = 1, we have a cylindrical geometry

have defined 2 layers radially of each crystal.

```
typedef struct{
  double geometricalDesignType;
  double ringDiameter;
  double rsectorAxialPitch;
  double rsectorAxialSize;
  double rsectorTangentialSize;
  double rsectorAzimuthalPitch;
  double moduleAxialSize;
  double moduleTangentialSize;
  double moduleAxialPitch;
  double moduleTangentialPitch;
  double submoduleAxialSize;
  double submoduleTangentialSize;
  double submoduleAxialPitch;
  double submoduleTangentialPitch;
  double crystalAxialSize;
  double crystalTangentialSize;
  double crystalRadialSize;
  double crystalAxialPitch;
  double crystalTangentialPitch;
  double layer0RadialSize;
  double layer1RadialSize;
  double layer0InteractionLength;
  double layer1InteractionLength;
  double azimuthalStep;
  double axialStep;
  double clockTimeStep;
  double energyStep;
} LMF_scanner_geometry;

LMF_scanner_geometry scannerGeometry;
```

rsector index	x shift [cm]	y shift [cm]	z shift [cm]
rsector 0	4	0	0
rsector 1	4	0	1
rsector 2	4	0	0
rsector 3	4	0	1
rsector 4	4	0	0
rsector 5	4	0	1
rsector 6	4	0	0
rsector 7	4	0	1
rsector 8	0	0	0
.	.	.	.
.	.	.	.
.	.	.	.
rsector 15	0	0	1

Table 1: Example of one shift table with an x shift and z shift.

2 (x,y,z) coordinates

We have defined a 3D coordinates (x,y,z) system as shown in Figure 1. The x-axis is vertical and points to the center of the first ring of detector cells. The z-axis fixes the cylindrical symmetry. $(\overrightarrow{e_r}, \overrightarrow{e_t}, \overrightarrow{e_z})$ is defined as the tangential reference frame which is attached to the rsectors.

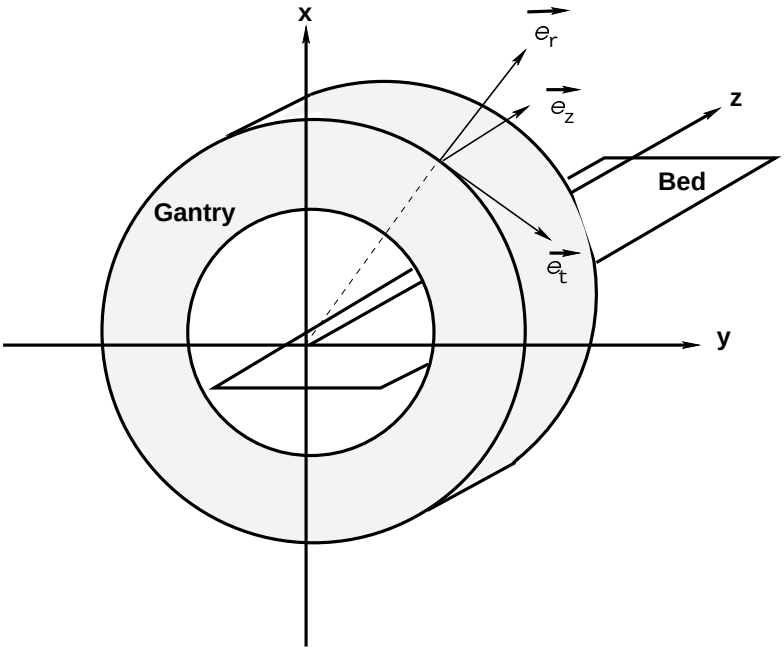

Figure 1: Scanner geometry : (x,y,z) coordinate system and tangential $(\vec{e_r}, \vec{e_t}, \vec{e_z})$ reference frame.

3 Scanner topology

The scanner topology is subdivided in rsectors, modules, submodules, crystals and layers (see the List Mode Format Implementation, version 1.3.2 on April 2002).
Each substructure of the topology is defined with reference to its location within its containing structure, i.e. by its *id*, and its tangential and axial pitches (pitch is the distance between the center of two nearby structures). But the *id* is not directly available to find 3D coordinates of this substructure in the (x,y,z)-plane. That's why we have define two other integers: the tangential *id* (id_t) and the axial *id* (id_z), linked by the relations:

$$\begin{aligned} id &= id_z * tangentialNumberOfSubstructures + id_t \\ id_z &= id/tangentialNumberOfSubstructures \\ id_t &= id \bmod tangentialNumberOfSubstructures \end{aligned}$$

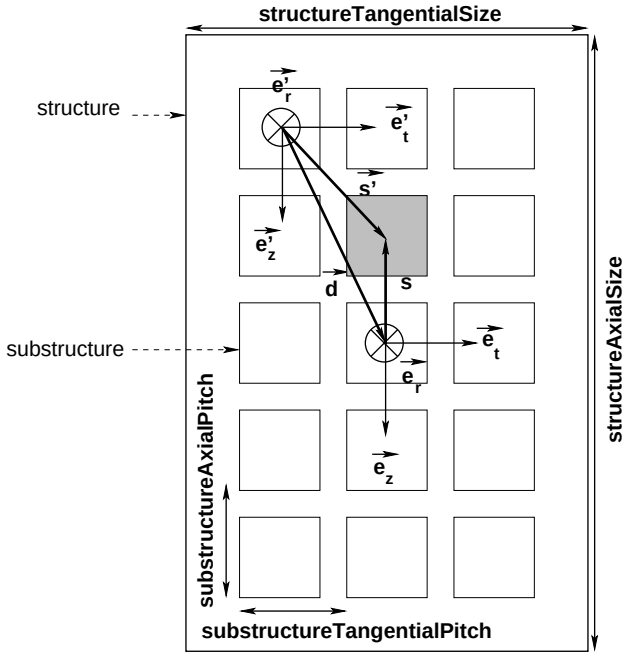

Figure 2: General description of a scanner structure.

4 Calculation of event positions

The aim of this software package is to allow its use in the largest expression of ClearPET scanner topologies with addition of optional shifts of one or more structures (ring, sector or rsector) of the scanner along the x-axis, or the y-axis, or the z-axis.
Moreover we will use only one process to calculate the 3D coordinates of a crystal in a submodule, of a submodule in a module, and of a module in a rsector. Therefore, we have described, in the general case, a scanner structure (cf. Figure 2), that can be a rsector, a module, a submodule or a crystal.

4.1 Static system with no translation, nor rotation movements

The first step of this process is based on the hypothesis that we consider a static system with no translation, nor rotation movements.
In our development we have chosen to start with the smallest structures, which are the crystals, and to end with the largest structures which are the rsectors. In each structure, we need 2 different tangential 3D reference frames: $(\overrightarrow{e_r}, \overrightarrow{e_t}, \overrightarrow{e_z})$ and $(\overrightarrow{e_r}, \overrightarrow{e_t}, \overrightarrow{e_z})$. The first coordinate system is attached

to the center of the substructure number 0 of this structure (*i.e.* corresponding to $id_t = 0$, $id_z = 0$), and the second to the center of the structure itself. The position of the substructure in the $(\overrightarrow{e_r}, \overrightarrow{e_t}, \overrightarrow{e_z})$ reference frame is given by:

$$\overrightarrow{s'} = \begin{pmatrix} 0 \\ id_t * substructureTangentialPitch \\ id_z * substructureAxialPitch \end{pmatrix}$$

By using the Chasles relation: $\overrightarrow{s} = \overrightarrow{s'} - \overrightarrow{d}$ and

$$\overrightarrow{d} = \begin{pmatrix} 0 \\ 1/2 * substructureTangentialPitch * (tangentialNumberOfSubstructures - 1) \\ 1/2 * substructureAxialPitch * (axialNumberOfSubstructures - 1) \end{pmatrix}$$

We can find the position of the substructure in the reference frame attached to the structure $(\overrightarrow{e_r}, \overrightarrow{e_t}, \overrightarrow{e_z})$:

$$\overrightarrow{s}^{structure}(id_{(substructure)}) = \begin{pmatrix} 0 \\ substructureTangentialPitch(id_t - 1/2 * (tangentialNumberOfSubstructures - 1)) \\ substructureAxialPitch(id_z - 1/2 * (axialNumberOfSubstructures - 1)) \end{pmatrix}$$

We define:

- the 3D coordinates of crystal (id_t, id_z) in the tangential reference frame attached to submodule (id_t, id_z) ($\overrightarrow{s}^{submodule}(id_{(crystal)})$);
- the 3D coordinates of submodule (id_t, id_z) in the tangential reference frame attached to module (id_t, id_z) ($\overrightarrow{s}^{module}(id_{(submodule)})$);
- the 3D coordinates of module (id_t, id_z) in the tangential reference frame attached to rsector (id_t, id_z) ($\overrightarrow{s}^{rsector}(id_{(module)})$).

By using the Chasles relation between these vectors, the 3D coordinates of a crystal in the tangential reference frame attached to a rsector is given by:

$$\overrightarrow{s}^{rsector}(id_{(crystal)}) = \overrightarrow{s}^{submodule}(id_{(crystal)}) + \overrightarrow{s}^{module}(id_{(submodule)}) + \overrightarrow{s}^{rsector}(id_{(module)})$$

Finally, the 3D coordinates of crystal (id_t, id_z) in the 3D coordinates (x',y',z') system centered on the first ring of rsectors without scanner rotation is given by:

$$\overrightarrow{s}^{scanner}(id_{(crystal)}) = \mathcal{P}_1 * \overrightarrow{s}^{rsector}(id_{(crystal)})$$

$$\mathrm{P}_1 = \begin{pmatrix} cos(id_{t(rsector)} * rsectorAzimuthalPitch) & -sin(id_{t(rsector)} * rsectorAzimuthalPitch) & 0 \\ sin(id_{t(rsector)} * rsectorAzimuthalPitch) & cos(id_{t(rsector)} * rsectorAzimuthalPitch) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

The calculation of rsector (id_t, id_z) position in the coordinate system (x,y,z) centered of the first ring of crystals is more complex and requires 2 operations:

1. For the coordinate system (x',y',z') so that Ox' points to the center of the rsector 0 (cf. Figure 3), the position of rsector (id_t, id_z) is given by:

$$\overrightarrow{s}^{scanner}(id_{(rsector)}) = \begin{pmatrix} (ringRadius+\Delta r)*\cos(id_{t(rsector)} * rsectorAzimuthalPitch) \\ (ringRadius+\Delta r)*\sin(id_{t(rsector)} * resctorAzimuthalPitch) \\ id_{z(rsector)} * rsectorAxialPitch \end{pmatrix}$$

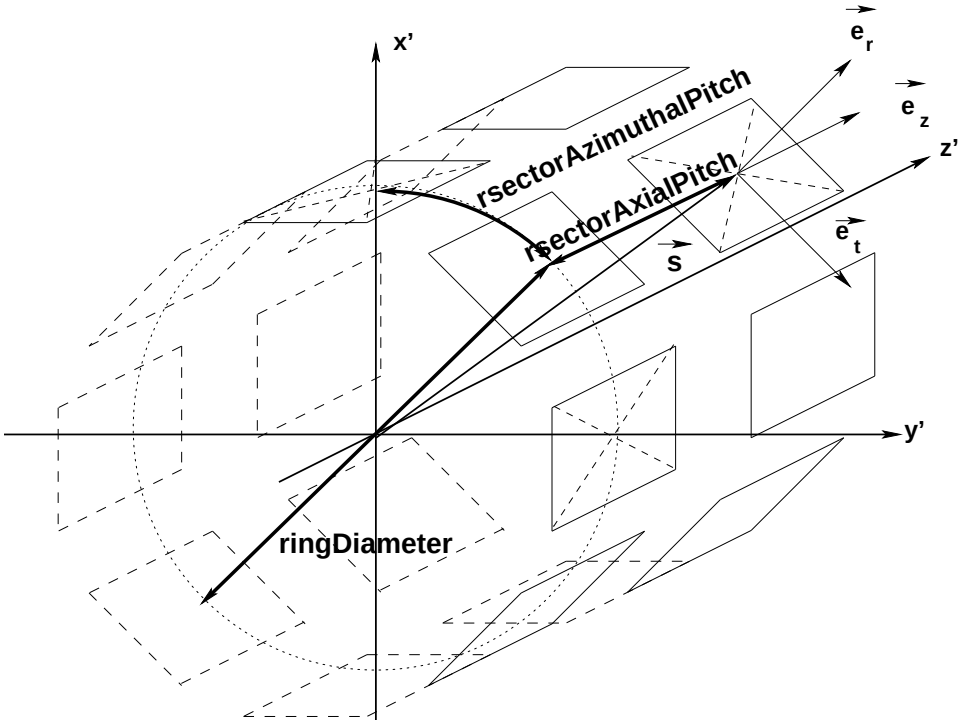

Figure 3: Rsector $(\overrightarrow{e_r}, \overrightarrow{e_t}, \overrightarrow{e_z})$ reference frame and (x',y',z') coordinate system.

2. An axial translation allows to calculate the position of the center of rsector (id_t, id_z) in the coordinate system (x,y,z) with Ox pointing to the center of the first row of crystals (*i.e.* crystal with $id_z = 0$ of submodule with $id_z = 0$ of module with $id_z = 0$) as shown in Figure 4.

$$\vec{s}^{\,laboratory}(id_{(rsector)}) = \begin{pmatrix} (ringRadius + \Delta r) * cos(id_{t(rsector)} * rsectorAzimuthalPitch) \\ (ringRadius + \Delta r) * sin(id_{t(rsector)} * rsectorAzimuthalPitch) \\ id_{z(rsector)} * rsectorAxialPitch + axialPitch \end{pmatrix}$$

$$\Delta r = layerInteractionLength(id_{layer}) + \sum_{i=0}^{(id_{layer}-1)} layerRadialSize(i)$$

$$\begin{aligned} axialPitch = 1/2* \quad [\quad & (axialNumberOfCrystals - 1) * crystalAxialPitch \\ & +(axialNumberOfSubmodules - 1) * submoduleAxialPitch \\ & +(axialNumberOfModules - 1) * moduleAxialPitch] \end{aligned}$$

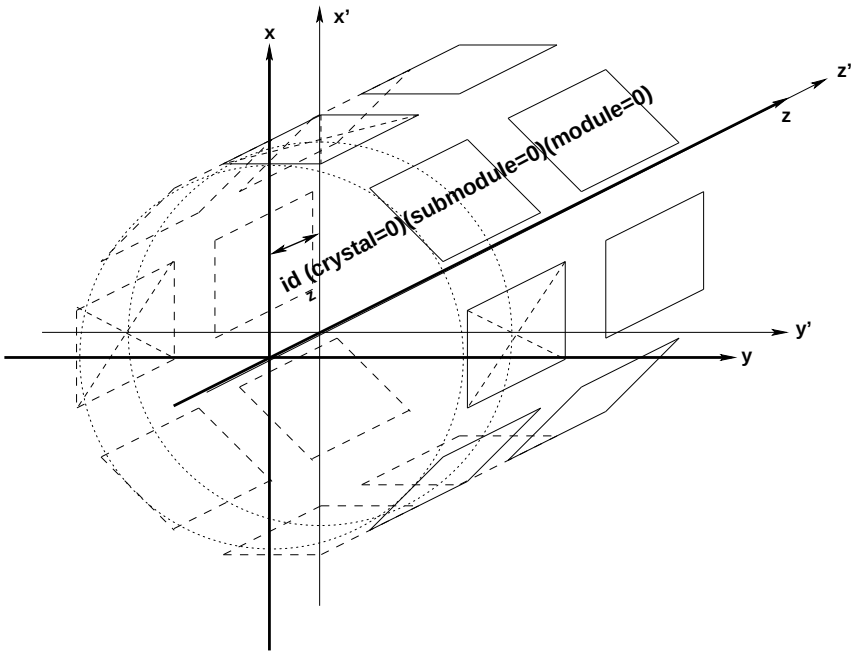

Figure 4: Scanner (x',y',z') coordinate system and laboratory (x,y,z) coordinate system.

Summary:
By using these formulas, we can find the 3D position of crystal (id_t, id_z) in

the (x,y,z) coordinate system without the scanner rotation movement and the rsector shifts:

$\overrightarrow{s}^{laboratory}(id_{(crystal)}) = \overrightarrow{s}^{laboratory}(id_{(rsector)}) + \overrightarrow{s}^{scanner}(id_{(crystal)}) + \overrightarrow{shift}$
$\overrightarrow{shift}$: *the substructure's shift along the x-axis,or the y-axis, or the z-axis* (cf. table 1)

4.2 System in rotation and translation

Finally, by integrating the scanner rotation movement and the rsector shift, we can compute the 3D position of crystal (id_t, id_z) in the (x,y,z) coordinate system:

$\overrightarrow{s}(\mathrm{id}_{(crystal)}) = \mathcal{P}_2 * (\overrightarrow{s}^{laboratory}(id_{(crystal)}) + \overrightarrow{z})$
$\mathrm{P}_2 = \begin{pmatrix} cos(\Delta\varphi) & -sin(\Delta\varphi) & 0 \\ sin(\Delta\varphi) & cos(\Delta\varphi) & 0 \\ 0 & 0 & 1 \end{pmatrix}$
$\overrightarrow{z} = \begin{pmatrix} 0 \\ 0 \\ \Delta z \end{pmatrix}$
$\Delta z = gantryAxialPos * axialStep$ (attached to the scanner translation movement)
$\Delta\varphi = gantryAngularPos * azimuthalStep$ (attached to the scanner rotation movement))

Applied to the description of the scanner topology given by 2 rings of 8 sectors (=2*8=16 rsectors), with 3 modules tangentially per rsector, 4 submodules axially per module, and one 8*8 crystal matrix per submodule, this gives:

- Submodule description
  Parameters, called **crystalTangentialPitch** and **crystalAxialPitch** are respectively the distance between centers of 2 crystals contained in the same row of crystals tangentially and the distance between centers of 2 crystals contained in the same column of crystals axially (cf. Figure 5).

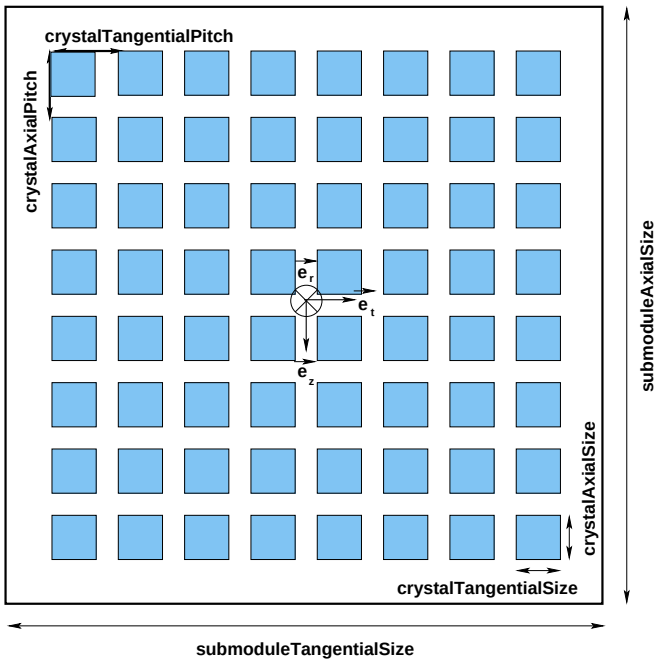

Figure 5: Submodule description.

- Module description
  In the case of above given scanner topology, a module is a column of 4 submodules (cf. Figure 6), and consequently the parameter **submodule-TangentialPitch** is equal to zero.

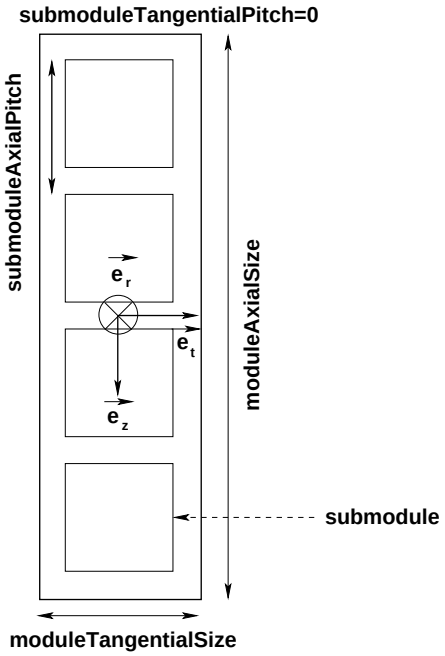

Figure 6: Module description.

- Rsector description
  As you can see in Figure 7, a rsector in this example of scanner is a row of 3 modules tangentially, that's why the parameter **moduleAxialPitch** is equal to zero.

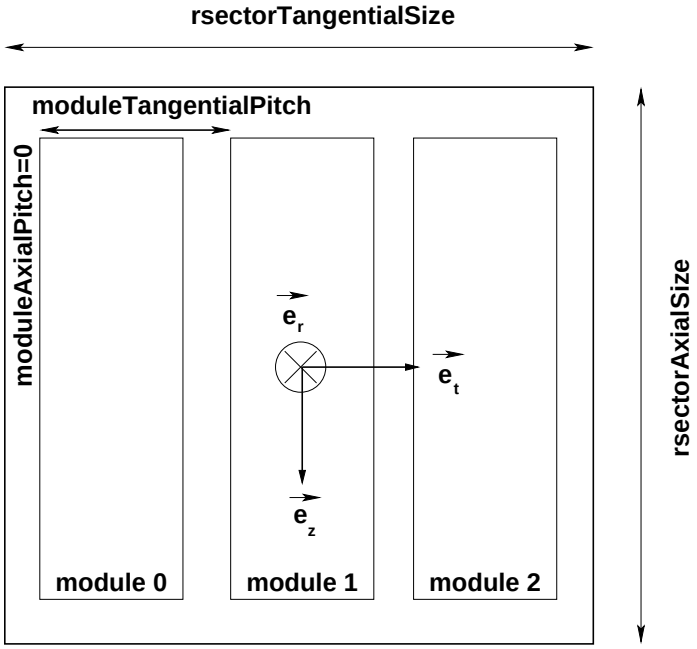

Figure 7: Rsector description.

- Scanner description
  We use the scanner flat development, corresponding to the result of cutting the rings, and unfolding them as shown in Figure 8, to define the **rsectorAxialPitch**. This parameter gives the distance between centers of 2 rings.

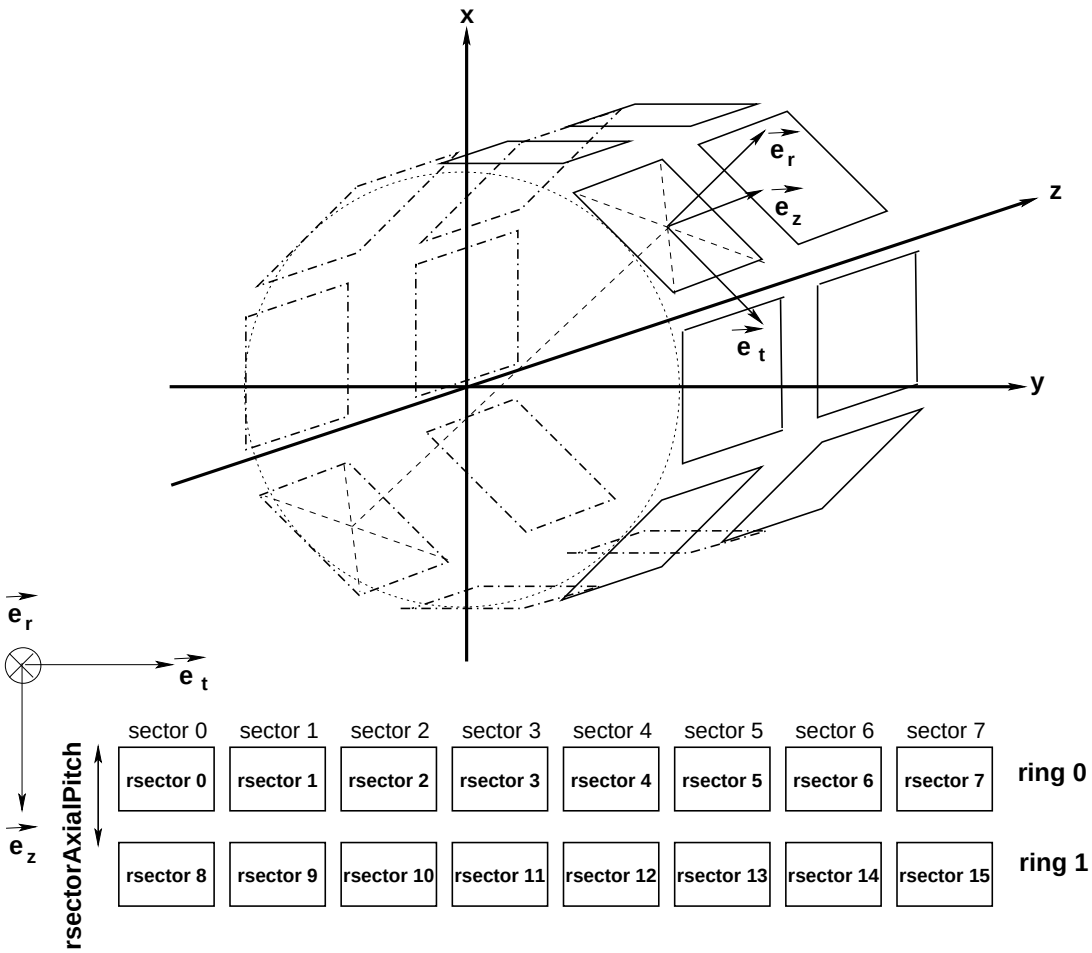

Figure 8: Example of view of an unfolded scanner with 2 rings.

5 LMF Implementation

5.1 C-Functions

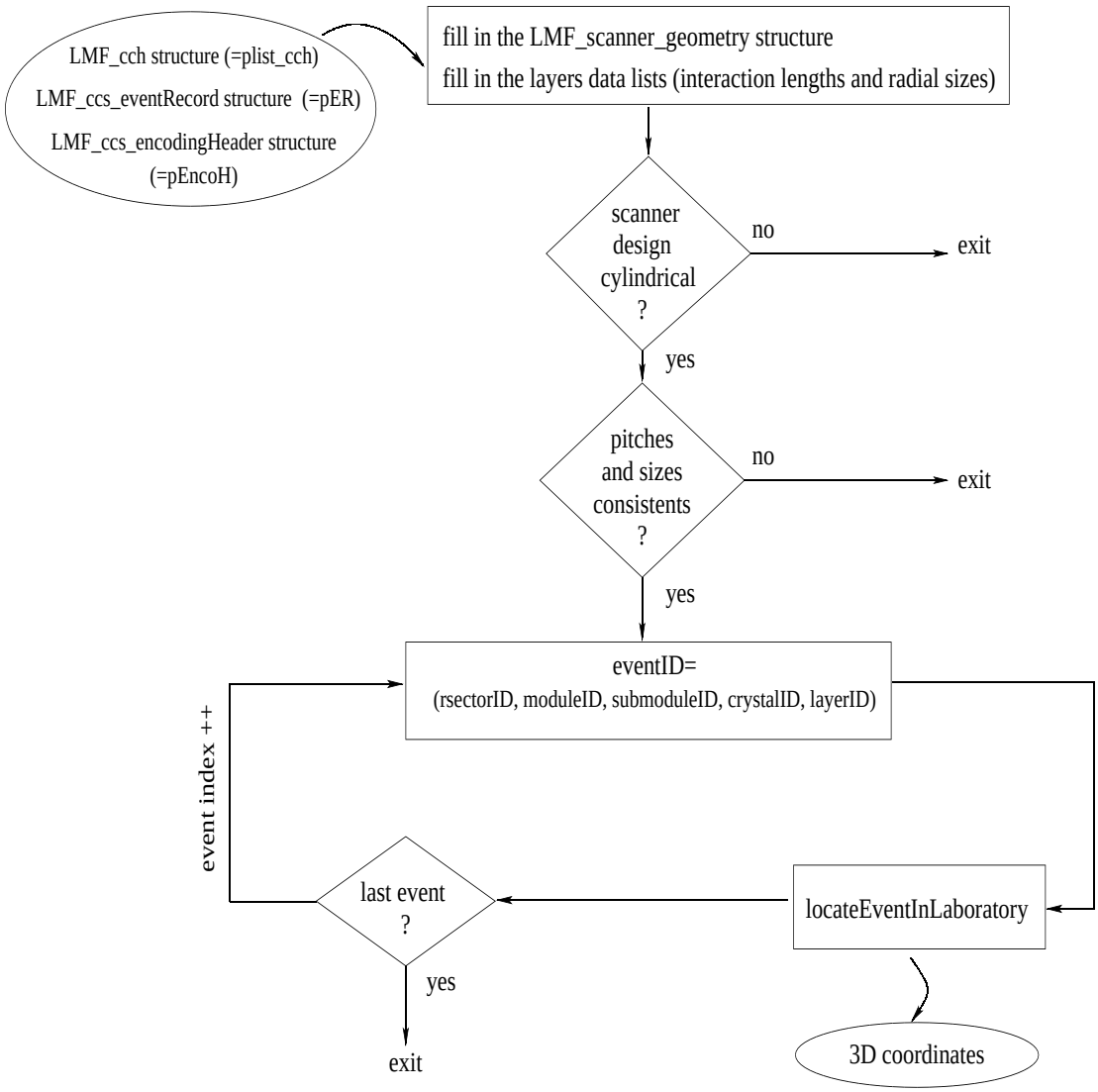

Figure 9: Function used to calculate the position of events in the 3D laboratory coordinates system.

In the LMF library, the 3 functions, locateEventInLaboratory, locateSubstructureInStructure and locateRsectorInLaboratory, allow to calculate the event position in the 3D laboratory coordinates system.

Each of these functions uses the information store by the LMF Record Carrier in the LMF_ccs_eventRecord structure, LMF_ccs_encodingHeader structure and LMF_cch structure (see the Figure 9).
The first step of this computation concerns the scannerGeometry variable initialization with the LMF_cch structure. That's why, before calling the function locateEventInLaboratory, the program must read the asciiHeader file (.cch file) and fill in the structure LMF_cch, with the function LMFcchReader.
The readOneEventRecord function allows to fill the LMF_ccs_eventRecord structure and the readHead function concerns the LMF_ccs_encodingHeader.
After this initialization step, the program controls if the scanner geometry is cylindrical or not, because the calculs of the event position are specific of each type of geometry.
Another test about the structure axial and tangential sizes (crystal sizes, submodule sizes, module sizes and rsector sizes) versus the substructure axial and tangential pitches (crystal pitches, submodule pitches, module pitches and rsector pitches) avoids miscalculations.
In order to calculate the event position in the laboratory coordinate system (x,y,z), these 2 equations must be true for each structure of the geometry (crystal, submodule, module and rsector):

$$\begin{cases} axialStructureSize & \geq & axialSubstructurePitch * (axialNumberOfSubstructures - 1) \\ tangentialStructureSize & \geq & tangentialSubstructurePitch * (tangentialNumberOfSubstructures - 1) \end{cases}$$

If all the tests are successful, the position of one event from the event ID is calculated with the locateEventInLaboratory function (see Annexes A, B and C). The function calls successively the locateSubstructureInStructure function and the locateRsectorInLaboratory function (cf. Figure 10).

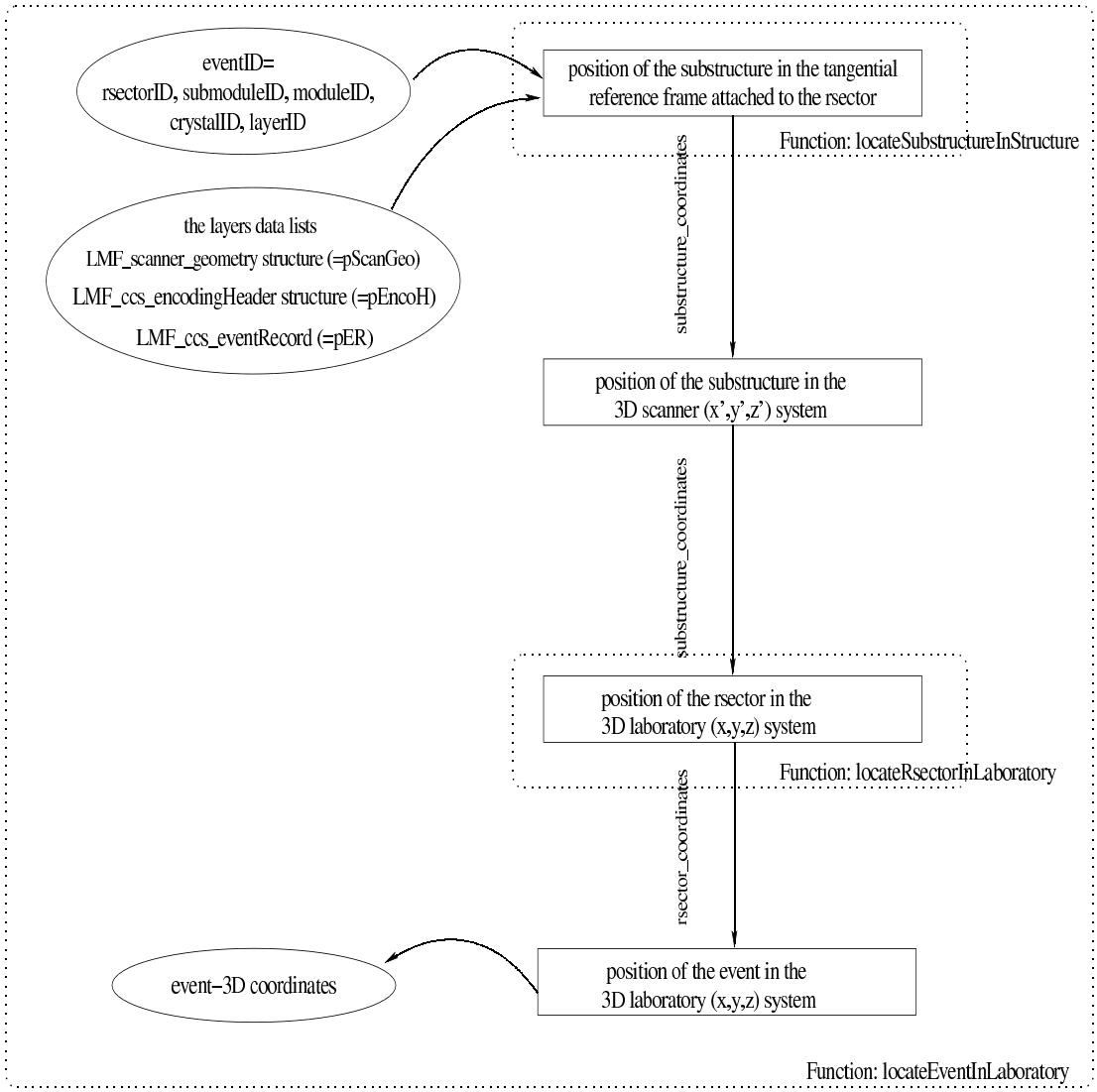

Figure 10: More details about "locateEventInLaboratory" presented in Figure 9.

5.2 How to use this function ?

In the exempleMain_08.c (see Annexe D), the user type the rsector ID, the module ID, the submodule ID, the crystal ID, the layer ID and the gantry's angular and axial position.
The LMF_ccs_eventRecord structure and the LMF_ccs_encodingHeader structure are initialized directly with the information contained in the

file "../includes/constantsLMF_ccs.h".

```
#define BITS_FOR_RSECTORS (4)
#define BITS_FOR_MODULES (2)
#define BITS_FOR_SUBMODULES (2)
#define BITS_FOR_CRYSTALS (6)
#define BITS_FOR_LAYERS (2)
/* The sum of these 5 last constants must be 16 ! */

#define NUMBER_OF_RINGS (1)
#define NUMBER_OF_SECTORS (8)
#define AXIAL_NUMBER_OF_MODULES (1)
#define TANGENTIAL_NUMBER_OF_MODULES (3)
#define AXIAL_NUMBER_OF_SUBMODULES (4)
#define TANGENTIAL_NUMBER_OF_SUBMODULES (1)
#define AXIAL_NUMBER_OF_CRYSTALS (8)
#define TANGENTIAL_NUMBER_OF_CRYSTALS (8)
#define AXIAL_NUMBER_OF_LAYERS (1)
#define RADIAL_NUMBER_OF_LAYERS (2)
```

In this case, we have defined an "one ring scanner" with 8 rsectors. Each rsector has 3 rows of modules tangentially and only one column of modules axially. Each module is divided in 4 columns of submodules and 1 row tangentially. Finally each submodule is divided in a matrix of 8 rows by 8 columns of crystals. In this example, we have defined 2 layers radially per crystal.
If the user wants to test another scanner topology, he must open and change the file "../includes/constantsLMF_ccs.h". In regular use with scan file, this file is not used, since this information is contained in the .ccs scan file.
With all these data, the program finds and prints the event position in the laboratory coordinates system (x,y,z) and frees the allocated memory by a previous call to malloc(), calloc() or realloc().

A Function "locateEventInLaboratory"

```
/*-----------------------------------------------------------------------
                              List Mode Format
                          — locateEventInLaboratory.c —

                              released on july 2002
                              Magalie.Krieguer@iphe.unil.ch
                              Copyright IPHE/UNIL, Lausanne
                              Crystal Clear Collaboration

  Description :
  Find the 3D coordinates of an event in the laboratory coordinates (x,y,z) system
  ->Calculate one event position in the 3D laboratory (x,y,z) system
  -----------------------------------------------------------------------*/
   include <stdio.h>
   include <stdlib.h>
   include <string.h>
   include <math.h>
   include "lmf.h"

   static int initializationDone=FALSE;
   static double *first_pIntLengthLayers=NULL, *first_pRdSizeLayers=NULL;

calculOfEventPosition locateEventInLaboratory(struct LMF_ccs_encodingHeader *pEncoH,
                                              struct LMF_ccs_eventRecord *pER,
                                              int indexID ){

     unsigned short *pcrist;
     static LMF_cch_scannerGeometry myScannerGeometry={0};
     static LMF_cch_scannerGeometry *pScanGeo=myScannerGeometry;
     /*declares pScanGeo to be of pointer to the (struct) LMF_cch_scannerGeometry*/
     calculOfEventPosition resultOfCalculOfEventPosition;
     generalSubstructureID rsectorID={0};
     static double *first_pSubstructuresNumericalValues=NULL;
     static double angleDefaultUnitToRadConversionFactor=0,rotationAngle=0;
     int rdNbOfLayers=0, cch_index=0;

     /* Debuild the ID of one event */
     pcrist = demakeid(pER->crystalIDs[indexID],pEncoH);

     if(initializationDone==FALSE){
        /* Fill in the structure scannerGeometry */
        plist_cch=first_cch_list;
        if(fillInStructScannerGeometry(cch_index, pScanGeo)==1){
           printf(ERROR_GEOMETRY1);
           printf(ERROR5,cchFileName);
           exit(EXIT_FAILURE);
        }
```

```
      /* Test if the scanner geometry is cylindrical or not */
      if(pScanGeo->geometricalDesignType!=1){
         printf(ERROR_GEOMETRY2);
         exit(EXIT_FAILURE);
      }
      /* Initialize the radial size and the interaction length in each layer */
      rdNbOfLayers=(int)pEncoH->scannerTopology.radialNumberOfLayers;
      if((first_pIntLengthLayers=(double*)calloc(rdNbOfLayers,sizeof(double)))==NULL){
         printf(ERROR_GEOMETRY8);
         exit(EXIT_FAILURE);
      }
      if((first_pRdSizeLayers=(double*)calloc(rdNbOfLayers,sizeof(double)))==NULL){
         printf(ERROR_GEOMETRY8);
         exit(EXIT_FAILURE);
      }
     if(setLayersInfo(cch_index,rdNbOfLayers,first_pIntLengthLayers,first_pRdSizeLayers)==1){
         printf(ERROR_GEOMETRY1);
         exit(EXIT_FAILURE);
      }
      /* Define an array, which contains the substructures numerical values: */
      /*layerAxPitch, crystalAxPitch, submoduleAxPitch, moduleAxPitch, rsectorAxPitch*/
      /*layerRdPitch, crystalTgPitch, submoduleTgPitch, moduleTgPitch, rsectorTgPitch*/
      /*axNbOfLayers, axNbOfCrystals, axNbOfSubmodules, axNbOfModules,*nbOfRings*/
      /*rdNbOfLayers, tgNbOfCrystals, tgNbOfSubmodules, tgNbOfModules,**nbOfSectors*/
      /* ( ax = axial, tg = tangential and rd = radial )*/
      /* ( *nbOfRings = axNbOfRsectors and **nbOfSectors = tgNbOfRsectors )*/

first_pSubstructuresNumericalValues=setSubstructuresValues(pScanGeo, pEncoH);

      /* Control the substructurePitches in comparison with the structureSizes */
   if(testPitchVersusSize(first_pSubstructuresNumericalValues,pScanGeo,first_pRdSizeLayers)==1)
     exit(EXIT_FAILURE);

      /* Test if (nbOfRsectors*rsectorAzimuthalPitch)<= 2π rad */
      angleDefaultUnitToRadConversionFactor=testAngleDefaultUnit();
      if(((double)(pEncoH->scannerTopology.numberOfSectors-1)
           *pScanGeo->rsectorAzimuthalPitch*angleDefaultUnitToRadConversionFactor)>6.283185307){
         printf(ERROR_GEOMETRY10, pScanGeo->rsectorAzimuthalPitch,
                                  pEncoH->scannerTopology.numberOfSectors);
         printf(ERROR5,cchFileName);
         exit(EXIT_FAILURE);
      }
      initializationDone=TRUE;
   }
   /* Position of the substructure in the 3D laboratory (x,y,z) system: */
   /* 1->Substructure position in the tangential reference frame (e_r, e_t, e_z): */
   resultOfCalculOfEventPosition.substructureInRsector3DPosition =
               locateSubstructureInStructure(pcrist,1,first_pSubstructuresNumericalValues);
```

```
  /* 2->Substructure position in the scanner coordinates (x',y',z') system: */
  rsectorID = locateID(pcrist,4,first_pSubstructuresNumericalValues);
  rotationAngle = pScanGeo->rsectorAzimuthalPitch*angleDefaultUnitToRadConversionFactor
                  * (double)rsectorID.tangential+((double)(pER->gantryAngularPos
                  * pScanGeo->azimuthalStep*angleDefaultUnitToRadConversionFactor);

  resultOfCalculOfEventPosition.substructureInScanner3DPosition.radial =
              - resultOfCalculOfEventPosition.substructureInRsector3DPosition.tangential
              * sin(rotationAngle);

  resultOfCalculOfEventPosition.substructureInScanner3DPosition.tangential =
              resultOfCalculOfEventPosition.substructureInRsector3DPosition.tangential
              * cos(rotationAngle);

  resultOfCalculOfEventPosition.substructureInScanner3DPosition.axial =
              resultOfCalculOfEventPosition.substructureInRsector3DPosition.axial;

  /* Position of the rsector in the 3D laboratory (x,y,z) system: */
  resultOfCalculOfEventPosition.rsectorInLaboratory3DPosition =
              locateRsectorInLaboratory(pcrist,first_pSubstructuresNumericalValues,
                                        pScanGeo,first_pIntLengthLayer,first_pRdSizeLayers,
                                        pEncoH,pER,angleDefaultUnitToRadConversionFactor);

  /* Position of the event in the laboratory (x,y,z) system: */
  resultOfCalculOfEventPosition.eventInLaboratory3DPosition.radial =
              resultOfCalculOfEventPosition.rsectorInLaboratory3DPosition.radial
              + resultOfCalculOfEventPosition.substructureInScanner3DPosition.radial;

  resultOfCalculOfEventPosition.eventInLaboratory3DPosition.tangential =
              resultOfCalculOfEventPosition.rsectorInLaboratory3DPosition.tangential
              + resultOfCalculOfEventPosition.substructureInScanner3DPosition.tangential;

  resultOfCalculOfEventPosition.eventInLaboratory3DPosition.axial =
              resultOfCalculOfEventPosition.rsectorInLaboratory3DPosition.axial
              + resultOfCalculOfEventPosition.substructureInScanner3DPosition.axial;

  return(resultOfCalculOfEventPosition);
}
```

B Function "locateSubstructureInStructure"

```
/*-------------------------------------------------------------------------
                              List Mode Format
                          — locateSubstructureInStructure.c —

                              released on july 2002
                              Magalie.Krieguer@iphe.unil.ch
                              Copyright IPHE/UNIL, Lausanne
                              Crystal Clear Collaboration

Description :
Find the 3D coordinates of an event in the 3D laboratory (x,y,z) system
->Calculate the substructure position in the structure tangential reference frame (e_r, e_t, e_z):
-------------------------------------------------------------------------*/
   include <stdio.h>
   include <stdlib.h>
   include "lmf.h"

   coordinates locateSubstructureInStructure (unsigned short *pcrist,
                                              int substructureOrder,
                                              double *first_pSubstructuresNumericalValues){
    coordinates structureFrame={0};
    generalSubstructureID substructureID={0};
    double *pTangentialPitch=NULL, *pTangentialNumberOfSubstructures=NULL;
    double *pAxialPitch=NULL, *pAxialNumberOfSubstructures=NULL;
    pAxialPitch=first_pSubstructuresNumericalValues;
    pTangentialPitch=first_pSubstructuresNumericalValues+5;
    pAxialNumberOfSubstructures=first_pSubstructuresNumericalValues+10;
    pTangentialNumberOfSubstructures=first_pSubstructuresNumericalValues+15;

    while(substructureOrder<4) {
       /*Calculation of the axial and the tangential ID (id_t,id_z):*/
       substructureID=locateID(pcrist,substructureOrder,first_pSubstructuresNumericalValues);

       /*Substructure position in the structure reference frame (e_r, e_t, e_z):*/
       structureFrame.tangential += pTangentialPitch[substructureOrder]
                                    *(substructureID.tangential-0.5
                                    *(pTangentialNumberOfSubstructures[substructureOrder]-1));
       structureFrame.axial += pAxialPitch[substructureOrder]*(substructureID.axial
                               -0.5*(pAxialNumberOfSubstructures[substructureOrder]-1));
       substructureOrder++;
       locateSubstructureInStructure(pcrist,
                                     first_pSubstructuresNumericalValues,
                                     substructureOrder);
    }
    return(structureFrame);
   }
```

C Function "locateRsectorInLaboratory

```
/*-------------------------------------------------------------------------
```

List Mode Format
— locateRsectorInLaboratory.c —

released on july 2002
Magalie.Krieguer@iphe.unil.ch

Crystal Clear Collaboration

Description :
Find the coordinates of an event in the 3D laboratory (x,y,z) system
->Calculate the rsector position in the laboratory coordinates (x,y,z) sytem

```
-------------------------------------------------------------------------*/
include <stdio.h>
include <stdlib.h>
include <string.h>
include <math.h>
include "lmf.h"

coordinates locateRsectorInLaboratory( unsigned short *pcrist,
                                       LMF_cch_scannerGeometry *pScanGeo,
                                       struct LMF_ccs_encodingHeader *pEncoH,
                                       struct LMF_ccs_eventRecord *pER,
                                       double *first_pIntLengthLayers,
                                       double *first_pRdSizeLayers,
                                       double angleDefaultUnitToRadConversionFactor,
                                       double *first_pSubstructuresNumericalValues){
 int list_index=0;
 unsigned short *pFirstRowOfCrystalsIDs=NULL;
 coordinates centerOfTheFirstRowOfCrystals={0},rsectorInLaboratoryPosition={0};
 generalSubstructureID rsectorID={0};
 unsigned short firstRowOfCrystalsIDs[5]={0,0,0,0,0};
 /*id_z(layer)=id_z(crystal)=id_z(submodule)=id_z(module)=id_z(rsector)=0*/

 double radialPitch=0,totalLayersRadialSize=0,ringRadius=0,rotationAngle=0;
 double *pRdSizeLayers=NULL,*pIntLengthLayers=NULL,*pSubstructuresNumericalValues=NULL;

 /* Find the axial and tangential rsectorID (id_t,id_z): */
 rsectorID=locateID(pcrist,4,first_pSubstructuresNumericalValues);

 /* Calculation of the radial pitch (Δr): */
 pSubstructuresNumericalValues=first_pSubstructuresNumericalValues+15;
 if((int)pcrist[0]>(int)pSubstructuresNumericalValues[0]){
    printf(ERROR_GEOMETRY3,(int)pcrist[0],(int)pSubstructuresNumericalValues[0]);
    exit(EXIT_FAILURE);
 }
```

```
 for(list_index=0;list_index<(int)pcrist[0];list_index++){
    pRdSizeLayers=first_pRdSizeLayers+list_index;
    totalLayersRadialSize=totalLayersRadialSize+(*pRdSizeLayers);
 }
 pIntLengthLayers = first_pIntLengthLayers+(int)pcrist[0];
 radialPitch = totalLayersRadialSize+(*pIntLengthLayers);

 ringRadius = (double)(0.5*pScanGeo->ringDiameter);
 rotationAngle = pScanGeo->rsectorAzimuthalPitch*angleDefaultUnitToRadConversionFactor
                 * (double)rsectorID.tangential+((double)(pER->gantryAngularPos
                 * pScanGeo->azimuthalStep*angleDefaultUnitToRadConversionFactor);

/*x'-coordinate in the scanner coordinates (x',y',z') system,
  so that 0x' points to the center of the rsector0:*/
/*= x-coordinate in the 3D laboratory (x,y,z) system,
  so that 0x points to the center of the first row of crystals:*/
    rsectorInLaboratoryPosition.radial = (ringRadius+radialPitch)*cos(rotationAngle);

/*y'-coordinate in the scanner coordinates (x',y',z') system,
  so that 0x' points to the center of the rsector0:*/
/*= y-coordinate in the 3D laboratory (x,y,z) system,
  so that 0x points to the center of the first row of crystals:*/
    rsectorInLaboratoryPosition.tangential = (ringRadius+radialPitch)*sin(rotationAngle);

/*z'-coordinate in the scanner coordinates (x',y',z') system,
  so that 0x' points to the center of the rsector0:*/
   rsectorInLaboratoryPosition.axial=(double)rsectorID.axial*pScanGeo->rsectorAxialPitch;

  /* Find the rsector position in the 3D laboratory (x,y,z) system,
    with 0x pointing to the center of the first row of crystals:*/
 pFirstRowOfCrystalsIDs=firstRowOfCrystalsIDs[1];
 centerOfTheFirstRowOfCrystals=
     locateSubstructureInStructure(first_pSubstructuresNumericalValues,
                                   1,
                                   pFirstRowOfCrystalsIDs);

/*z-coordinate in the 3D laboratory (x,y,z) system,
  so that 0x points to the center of the first row of crystals:*/
 rsectorInLaboratoryPosition.axial = (pScanGeo->axialStep*(double)pER->gantryAxialPos)
                                   + rsectorInLaboratoryPosition.axial
                                   - centerOfTheFirstRowOfCrystals.axial;

 return(rsectorInLaboratoryPosition);
}
```

D ExempleMain_08.c

```
/*-------------------------------------------------------------------------
```

List Mode Format
— exempleMain_08.c —

released on july 2002
Magalie.Krieguer@iphe.unil.ch
Copyright IPHE/UNIL, Lausanne
Crystal Clear Collaboration

Description :
Example to use the calculation of events position in the scanner.
The LMF_ccs_eventRecord structure and the LMF_ccs_encodingHeader structure are initialized directly in the main. If the user wants to modify the scanner topology, he must open and change the file "../includes/constantsLMF_ccs.h ".
The user must choose:
-> the rsector, the module, the submodule, the crystal, the layer IDs,
-> the gantry's angular and axial position.
The event position in the 3D laboratory (x,y,z) system is printed.

```
-------------------------------------------------------------------------*/
include <stdio.h>
include <stdlib.h>
include <time.h>
include <string.h>
include "lmf.h"

int main() {
 int index=0,inputData=0,cch_index=0;
 struct LMF_ccs_eventRecord *pER=NULL;
 struct LMF_ccs_encodingHeader *pEncoH=NULL;
 char description[7][charNum]={{"rsector ID"},
                               {"module ID"},
                               {"submodule ID"},
                               {"crystal ID"},
                               {"layer ID"},
                               {"gantry's angular position"},
                               {"gantry's axial position"}};
 unsigned short limits[7]={0};
 int numericalValue[7]={0};
 char input[charNum];
 calculOfEventPosition resultOfCalculOfEventPosition;
```

*/*Initialization of the LMF_ccs_eventRecord structure and the LMF_ccs_encodingHeader structure. We have chose an "one ring scanner" with 8 rsectors. Each rsector has 3 rows of modules tangentially and only one column of modules axially.*
Each module is divided in 4 columns of submodules and 1 row tangentially.
Finally each submodule is divided in a matrix of 8 rows of 8 columns of crystals.

```
        We have defined 2 layers radially of each crystal.*/
        /* fill in the LMF_ccs_encodingHeader structure */
if((pEncoH = (struct LMF_ccs_encodingHeader*) malloc(sizeof(struct LMF_ccs_encodingHeader)))==NULL)
           printf("ERROR: in generateEncoH.c:  impossible to do malloc()\n");
        pEncoH = generateEncoH(1);
        if((pER=(struct LMF_ccs_eventRecord*)malloc(sizeof(struct LMF_ccs_eventRecord)))==NULL)
           printf("ERROR: in generateER.c:  impossible to do malloc()\n");
        if((pER->crystalIDs = malloc(sizeof(unsigned short)))==NULL)
           printf(ERROR: in generateER.c:  impossible to do malloc()\n");

        limits[0]=(unsigned short)pEncoH->scannerTopology.totalNumberOfRsectors;
        limits[1]=(unsigned short)pEncoH->scannerTopology.totalNumberOfModules;
        limits[2]=(unsigned short)pEncoH->scannerTopology.totalNumberOfSubmodules;
        limits[3]=(unsigned short)pEncoH->scannerTopology.totalNumberOfCrystals;
        limits[4]=(unsigned short)pEncoH->scannerTopology.totalNumberOfLayers;
        limits[5]=256;
        limits[6]=256;

      /* read file .cch and fill in structures LMF_cch */
        if(LMFcchReader("")) exit(EXIT_FAILURE);

    /*the user must choose the rsector, the module, the submodule, the crystal, the layer IDs,
     the gantry's angular and axial position.*/
        for(index=0;index<7;index++){
           initialize(input);
           printf("Choose a %s (number between 0 and %d):",description[index],(limits[index]-1));
           if(*gets(input)=='\0') numericalValue[index]=0;
           else {
              inputData=0;
              inputData=atoi(input);
              if((inputData>0  inputData<limits[index])!=1) numericalValue[index]=0;
              else numericalValue[index]=inputData;
           }
        }
        pER->gantryAngularPos=(unsigned short)numericalValue[5];
        pER->gantryAxialPos=(unsigned short)numericalValue[6];

     /*with the rsector, the module, the submodule, the crystal and the layer IDs,
     the makeid function find the event ID*/
        pER->crystalIDs[0]=makeid( numericalValue[0],
                                   numericalValue[1],
                                   numericalValue[2],
                                   numericalValue[3],
                                   numericalValue[4],
                                   pEncoH);

      /* calculation of the event position in the laboratory coordinates system */
        resultOfCalculOfEventPosition = locateEventInLaboratory(pEncoH,pER,0);
        printf("%s:%d\t%s:%d\t%s:%d\t%s:%d\t%s:%d\t%s:%d\n",
```

```
        description[0], numericalValue[0],
        description[1], numericalValue[1],
        description[2], numericalValue[2],
        description[3], numericalValue[3],
        description[4], numericalValue[4],
        description[5], numericalValue[5],
        description[6], numericalValue[6]);
    printf("x:%f:%f:%f",resultOfCalculOfEventPosition.eventInLaboratory3DPosition.radial,
                        resultOfCalculOfEventPosition.eventInLaboratory3DPosition.tangential,
                        resultOfCalculOfEventPosition.eventInLaboratory3DPosition.axial);

/*frees the allocated memory by a previous call to malloc() or realloc()*/
    LMFcchReaderDestructor();
    free(pER->crystalIDs);
    free(pER);
    free(pEncoH);

    return(EXIT_SUCCESS);
    }
```

Contents