# The DEA geometry package

Konstantin Olchanski, olchansk@b1.phy.bnl.gov v.1.0, 1996-Aug-17

This document describes the DEA geometry package- the data structures that describe the DEA geometry and the function calls to load these data structures. The TeX-formatted Postscript version of this document suitable for printing is *here* <http://lemond.phy.bnl.gov/documentation/geometries/deaGeom.ps>

## Contents



## 1 Drawing

Figure 1 shows how the four DEA segments (1+2, 3, 4 and 5+6) are located inside the MPS magnet.

## 2 DEA geometry definition

The DEA geometry is defined by a set of points and vectors, all of them in the MPS coordinates-

- *origin* - an arbitrary point on the face of the DEA, used as the origin of the "DEA-local" coordinates,
- *normalv* - the normal vector that defines the orientation of the DEA plane,
- *yv* - the up-and-down vector,

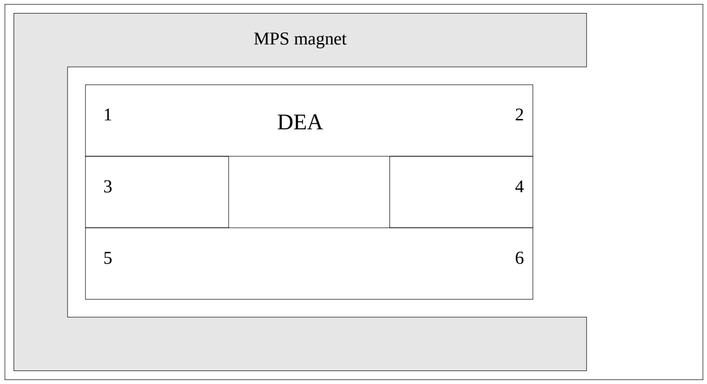


Figure 1: "DEA geometry"

- *xv* - the left-and-right vector. Together, *normalv*, *yv* and *xv* form a axes of the right-handed "DEA-local" coordinate system,

## 2.1 The "DEA-local" coordinates

The DEA face is a plane (a two-dimensional surface), so a point on the face of DEA has two "DEA-local" coordinates- *localX* and *localY*. The depth of a point inside the DEA is given by the third "DEA-local" coordinate- *localZ*.

## 2.2 Conversion to/from MPS coordinates

The conversion between the MPS coordinates and the "DEA-local" coordinates is given by the formulas:

```
vector3_t mps,local;     --- 3D points
vector3_t xv,yv,normalv; --- 3D vectors

mps.x = origin.x + xv.x * local.x + yv.x * local.y + normalv.x * local.z;
mps.y = origin.y + xv.y * local.x + yv.y * local.y + normalv.y * local.z;
mps.z = origin.z + xv.z * local.x + yv.z * local.y + normalv.z * local.z;
```

The inverse conversion is given by:

```
local.x = DotProduct(mps-origin,xv);
local.y = DotProduct(mps-origin,yv);
local.z = DotProduct(mps-origin,normalv);
```

## 2.3 Boundaries and edges

The boundaries of the DEA segments are defined by constraining the "DEA-local" coordinates to the inside of a square box:

```
xMin <= local.x <= xMax
yMin <= local.y <= yMax
   0 <= local.z <= depth
```

### 2.4 Is a given point inside the DEA?

To find out if a point specified in the MPS coordinates is inside or outside of a DEA segment, one has to convert the coordinates of this point to the "DEA-local" coordinates and check if the "DEA-local" coordinates (*local.x* and *local.y*) lay inside the boundaries (*xMin*, *xMax*, *yMin* and *yMax*) of that particular segment.

## 3 Data structures

The values of the vectors that define the DEA geometry are stored in the data structure *deaGeom_t*, which is declared in *deaGeom.h* <http://lemond.phy.bnl.gov/include/deaGeom.h>.

### 3.1 deaGeomBounds_t

Defines the boundaries of one DEA segment in the DEA-local coordinates

```
typedef struct
{
  double xMin;         --- the right-left boundaries of the segment, in cm.
  double xMax;

  double yMin;         --- the up-down boundaries of the segment, in cm.
  double yMax;

} deaGeomBounds_t;
```

### 3.2 deaGeom_t

Defines the geometry of the DEA

```
typedef struct
{
  int runNo;           --- run number for which the data was loaded
  int flags;           --- reserved for future use

  vector3_t origin;    /* origin of the DEA-local coordinates */
  vector3_t normalv;   /* vector defines the normal to the DEA plane */
  vector3_t yv;        /* the "up" vector */
  vector3_t xv;        /* the "left-right" vector. defined to form a right-handed
```

```
                         coordinate system */

  double depth;        --- depth of the DEA along the Z axis

  deaGeomBounds_t dea12;  --- the top segment (channels 1 amd 2)
  deaGeomBounds_t dea3;   --- the closed side segment (channel 3)
  deaGeomBounds_t dea4;   --- the open side segment (channel 4)
  deaGeomBounds_t dea56;  --- the bottom segment (channels 5 and 6)

} deaGeom_t;
```

# 4 Function calls

The functions available to load the DEA geometry for any given run are declared in *deaGeom.h* `<http://lemond.phy.bnl.gov/include/deaGeom.h>`.

## 4.1 deaGeom_get(runNo,geom,flags)

This function loads the DEA geometry for the run given by *runNo* into the structure pointed by *geom*. The *flags* argument should be zero.