FTD Mokka Driver and Digitisation

J. Duarte Campderrós, *D. Moya, I. Vila, A. Ruiz*

IFCA - U.Cantabria/CSIC

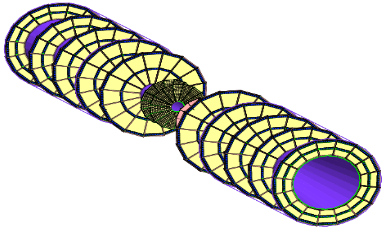
ILD Software Working Group Meeting
July 20 2011

Outline

1. FTD Mokka Driver

SFtd06 driver

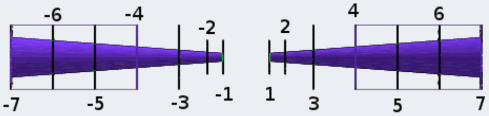
- Self-scaling driver: significant parameters and positioning defined w.r.t surrounding components (VTX, SIT, SET, beam pipe, ...) following the LOI specifications (see backup)
- Sensitive areas placed over petals generating a disk structures
  - Pixel disks: 1,2
  - Micro-strips disks: 3-7 (Included READ-OUT chips and hybrid)
- Supports:
  - Inner cylinder: supports the whole disks structure w.r.t beam tube
  - Outer cylinder: supports the micro-strips disks (4,5,6,7) w.r.t TPC
- Cables located in the inner cylinder as a cone. CABLING TO BE UPDATED

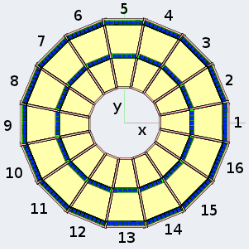


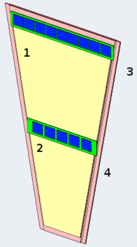
FTD Mokka Driver Characteristics

Elements identification

- Disks (layers): Identified with 4 bits: $\pm 1, \ldots, \pm 7$, in increasing order of (signed) z.
- Petals (ladders): Identified with 5 bits (could be 4), in increasing order of ϕ. Disposed in a turbine-like design.
- Sensors: Identified with 3 bits (could be 2), facing the IP and from *R* great: 1,2,3,4







FTD Mokka Driver Modifications

Work On Going

Modifications

- Remove petal rotation around its plane. Database parameters change
  - petal_inclination_angle_support $4^o \longrightarrow 0^o$
- Within a disk, alternating petals displaced in z to get overlapping petal edges. Driver to be modified and database new parameters inclusion. As the modifications can be introduced in a compatible way with respect the rotated case, the driver is not going to change its name
- Improvement in the petals design including services; slightly change of petal geometry. Driver changes and database new parameters inclusion. Design is ongoing

Two equivalent versions of the subdetector in order to do optimisation and comparation studies (between rotated and non-rotated petals), plus a new future version as soon as the mechanical design including power network components would be ready

Subdetector	driver	database	description
SFtd09	SFtd06.cc	ftd08	Current implementation. Petals rotated around its plane
SFtd10	SFtd06.cc	ftd09	Petals non-rotated and displaced w.r.t. the z-disk position
SFtd11	SFtd07.cc	ftd10	As the previous one but with an improved design

Mokka sub-detector SFtd09

Parameters from the *ftd08* database well documented in the driver. Table parameters are:

common_parameters :

- ftd1_vtx3_distance_z
- ftd7_ecal_distance_z
- ftd1_sit1_radial_diff
- ftd2_sit1_radial_diff
- ftd3_sit2_radial_diff
- ftd4to7_tpc_radial_gap
- beamtube_clearance
- cables_thickness
- cable_shield_thickness
- outer_cylinder_total_thickness
- petal_half_angle_support
- petal_wings_length
- petal_wings_thickness
- petal_y_ratio
- chip_strips_thickness
- chip_strips_width
- chip_strips_length

disks table :

- disk_number
- z_position_ReltoTPCLength
- petal_cp_support_dxMax
- petal_cp_support_thickness
- petal_cp_support_holes_separation
- petal_cp_support_edges_separation
- petal_cp_holes_width_support
- petal_inclination_angle_support
- disk_Si_thickness
- padUp_Si_dxMax
- kapton_petal_thickness
- kapton_petal_interspace

Note that the units used in the database are:

distance	*mm*
angular	*degree*

2. FTD Digitisation

FTD Digitisation

- Code kindly provided by Zbynek Drásal (Charles University Prague)
- Code currently in use for the digitisation of the *SVD* subdetector in *Belle II* $\Rightarrow$ Algorithms tested and validated.
- Incorporates (see `https://docs.google.com/viewer?a=v&pid=explorer&chrome=true&srcid=0Bxr58vjAs-zqM2EwZjk1NTEtYzkzYi00YTI5LTk3MWItNWE0MDBjMDA2OWQ1&hl=ca`):
  - Digitisation of micro-strips:
    - adapted to barrel and forward geometries
    - drift in electric field
    - diffusion due to multiple collisions
    - Lorentz shift in magnetic field
    - mutual micro-strip crosstalks (dependent on AC/DC coupling)
    - electronics noise
  - Clustering: cluster finding algorithm based on COG (centre of gravity) method (cluster size is lower than 3) or on head-tail analog method (cluster-size is equal or higher than 3). Transform electric pulses into real hits
- Integrated in Marlin framework, two main processors:
  - SiStripDigi:
    - input processor parameters to change digitising algorithms
    - LCIO input collection: *SimTrackerHit*
    - LCIO output collection: *TrackerPulse*
  - SiStripClus
    - LCIO input collection: *TrackerPulse* (output of SiStripDigi processor)
    - LCIO output collection: *TrackerHit*

FTD Digitisation

Adapting to FTD

The geometry information required for data processing is accessed via a geometry interface: *SiStripGeom* which uses the information obtained from a given xml file (GEAR constructed). This class encapsulates the detector stuff and manage the methods related with it; the main processors (digitiser and clustering) use the interface to extract all the detector information needed.

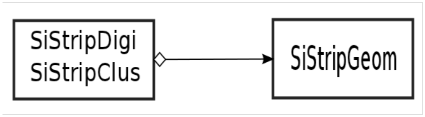
$\Rightarrow$ Potencially can be used for any micro-strip based subdetector: FTD, SIT, SET, ETD

- In order to adapt the processor to FTD (or others subdetectors), we have identified the next issues:
  1. Decouple the geometry interface from the type of subdetector in order to re-use the digitising processor
  2. Modify the geometry interface to the FTD geometry
  3. It will be easier to adapt if previosly the FTD description is included in GEAR

FTD Digitisation

Decoupling the geometry interface

The processors (*SiStripDigi* and *SiStripClus*) use an instance of *SiStripGeom* to deal with the geometry of some concrete detector



SiStripDigi.h

```
class SiStripDigi : public marlin::Processor
{
            ...
     protected:
            SiStripGeom * _geometry;
      ...
}
```

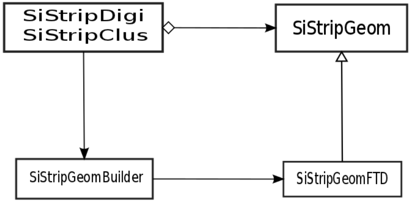
SiStripDigi.cc

```
void SiStripDigi::init()
{
...

_geometry = new SiStripGeom();

...
}

void SiStripDigi::processEvent(LCEvent *evt)
{
        ...
//  Uses the accesors of _geometry
...
}
```

FTD Digitisation

Decoupling the geometry interface (2)

Introduced a builder (*SiStripGeomBuilder*) used by the processors in order to instanciate the concrete *SiStripGeom* representation (FTD,SET,SIT or ETD).



- *SiStripGeomFTD* is a derived class of *SiStripGeom*
- *SiStripGeom* contains pure virtual method which have to be implemented in the concrete classes: methods which are subdetector dependent
- Each subdetector responsable can focussed in the concrete class implementation (SiStripGeomFTD, SiStripGeomSIT, ...)
- The builder decides which concrete class to instance via an input parameter

FTD Digitisation

Decoupling the geometry interface (3)

SiStripDigi.h

```
class SiStripDigi : public marlin::Processor
{
                ...
       protected:
                SiStripGeom * _geometry;
      ...
}
```

SiStripDigi.cc

```
void SiStripDigi::init()
{
...

_geometry = SiStripGeomBuilder::Build(subdetector);

...
}
```

SiStripGeomBuilder.cc

```
SiStripGeom *SiStripGeomBuilder::Build(const
                                 std::string & detector )
{
        SiStripGeom *p = 0;

        // Building the concrete SiStripGeom
        if( detector == "FTD" )
        {
                p = new SiStripGeomFTD("FTD");
        }
        //else if( detector == "VXD" )
        //{
        //      p = new SiStripGeomVXD();
        //}
        // Add another type here and codify a new concrete
        // class derived from SiStripGeom
        else
        {
                streamlog_out(ERROR) <<
                        "Detector␣type␣unknown:␣" <<
                        detector << std::endl;
                exit(-1);
        }

   return p;
}
```

Summary

- FTD Mokka Driver
  - Shown a brief description of the subdetector
  - Decided some modifications:
    - Change the turbine-like design to petals positioned perpendicular to Z-axis
    - Some improvements in the mechanical design are planned, this will include service cabling (DC/DC convertors, opto-links,...)
- FTD Digitisation
  - Code provided by Zbnynek Drásal, used in Belle collaboration for the SVD detector
  - Trying to adapt to our subdetector requirements:
    - Introduced a builder mechanism to allow use this code with all micro-strips subdetector
    - Working on the adaptation of the geometric interface
    - Working on implementation of FTD in GEAR

(auto) Deadline: 2nd week of August

BACKUP

Parameters set passed to Mokka to load the FTD stand-alone

```
/Mokka/init/globalModelParameter  SIT1_Half_Length_Z  371.309
/Mokka/init/globalModelParameter  SIT2_Half_Length_Z  644.906
/Mokka/init/globalModelParameter  SIT1_Radius  165
/Mokka/init/globalModelParameter  SIT2_Radius  309
/Mokka/init/globalModelParameter  TUBE_IPOuterTube_end_z  230
/Mokka/init/globalModelParameter  TUBE_IPOuterTube_end_radius  24
/Mokka/init/globalModelParameter  TUBE_IPOuterBulge_end_z  2364.5
/Mokka/init/globalModelParameter  TUBE_IPOuterBulge_end_radius  184
/Mokka/init/globalModelParameter  Ecal_endcap_zmin  2450
/Mokka/init/globalModelParameter  TPC_Ecal_Hcal_barrel_halfZ   2350
/Mokka/init/globalModelParameter  TPC_inner_radius  329
/Mokka/init/globalModelParameter  TUBE_IPOuterBulge_start_z  150
/Mokka/init/globalModelParameter  TUBE_IPOuterBulge_start_radius  24
/Mokka/init/globalModelParameter  VXD_length_r3  125
```

LOI specifications for FTD

Disk Number	z_{pos} (mm)	R_{inner} (mm)	R_{outer} (mm)
1	220	39	164
2	371.3	49.6	164
3	644.9	70.1	308
4	1046.1	108.3	309
5	1447.3	130.	309
6	1848.5	160.5	309
7	2250	190.5	309