SMT Geometry Package

Package: silicon_geometry

Emanuela Barberis, John Hobbs, Abid Patwa, Thomas Trippe

Contents

1

Introduction

The *silicon_geometry* package provides the functionality to describe the geometry of the SMT detector. The package is divided into several components:

- *silicon_geometry/base* - contains the positioning surfaces for the SMT detector geometry
- *silicon_geometry/channel* - contains the geometrical representation of the "active channels" of the SMT (ie. the silicon detectors strips).
- *silicon_geometry/material* - contains the material representation of the silicon. References the base geometry through the *SiGeometer* class
- *silicon_geometry/utils* - contains utility programs to do simple barrel event drawing and local to global coordinate transformations (and vice versa).
- *silicon_geometry/tests* - contains test programs for the routines contained in the directories above. These are run by issueing 'gmake.test'

The *silicon_geometry* package uses the following packages:

- *geometry_system*
- *thinshells*
- *material*
- *rcp*
- *d0om*

The corresponding library is libsilgeom.a and the d0om reference headers are compiled into the object file *silgeom_ref.o*.

2 SMT Base Geometry (*silicon_geometry/base*)

Names

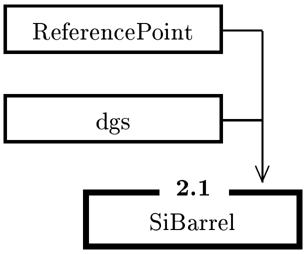
The *base* component of the *silicon_geometry* package contains classes used to describe the positioning surfaces for the SMT detector. All other components of the *silicon_geometry* package use these surfaces as reference positions for their subsequent geometry descriptions. Access to the base geometry should be performed via the Geometer. Correspondingly, access to the full channel information should be done via the ChannelGeometer, described in the following section. Examples of the use of both geometers can be found in the *tests* directory.

2.1

class **SiBarrel** : public dgs ReferencePoint

Barrel section of the SMT

Inheritance



Public Members

SiBarrel () — *Default constructor (empty barrel).*

2.1.1 **SiBarrel** (const std::vector<SiLayer> &layers, bool spun=false)
Construct a barrel containing the given layers 8

2.1.2 const std::list <dgs::GeometryElement*> **get_children** ()
Return the address of the children 8

int **get_layer_count** ()
Return the number of layers.

const SiLayer* **get_layer** (int ilayer)
Return a constant reference to the ilayer-th layer.

SiLayer* **get_mutable_layer** (int ilayer)
Return a modifiable reference to the ilayer-th layer.

friend std::ostream& **operator <<** (std::ostream& os, const SiBarrel& me)
Text dump a barrel.

friend bool **operator==** (const SiBarrel& lhs, const SiBarrel &rhs)
Test that the barrels have the same layers (number and contents)

Barrel section of the SMT.

A SiBarrel is used to position the eight layers of silicon detector units (i.e. ladders) composing each of the SMT barrels.

2.1.1

SiBarrel (const std::vector<SiLayer> &layers, bool spun=false)

Construct a barrel containing the given layers

Construct a barrel containing the given layers. If the ladders are spun to be anti-parallel to the global z direction, flip them around here.

Parameters: `spun` — If spun is false, then the (ideal) ladder local z axis is parallel to global z. If true, local z is anti-parallel to global z.

2.1.2

const std::list <dgs::GeometryElement*> **get_children** ()

Return the address of the children

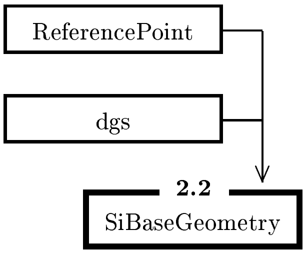
Return the address of the children. Used within dgs::GeometryElement and ReferencePoint to do positioning.

2.2

class **SiBaseGeometry** : public dgs ReferencePoint

Define Baseline Geometry

Inheritance



Public Members

SiBaseGeometry ()
Default constructor

2.2.1 **SiBaseGeometry** (int nc, const float* center_zpos, const SiCenter* proto_center, int ne, const float* end_zpos, const SiEnd* proto_end)
Instantiate detector with nc centers, ne end sections 10

2.2.2 **SiBaseGeometry** (int nc, const float* center_zpos, const std::vector<SiCenter> &centers, int ne, const float* end_zpos, const std::vector<SiEnd> &ends)
Instantiate detector with nc centers, ne end sections 11

2.2.3 static d0_Ref <SiBaseGeometry> **build_default** (edm::RCP* RCPinit=(edm::RCP*)0)
Build default geometry 11

2.2.4 const std::list <dgs::GeometryElement*> **get_children** ()
Return the address of the children 12

int **get_center_count** ()
Return the number of central trackers.

int **get_end_count** () *Return the number of end trackers.*

const SiCenter*

get_center (int icenter)
Return a reference to the icenter-th center in a manner which does not allow the center to be changed

SiCenter* **get_mutable_center** (int icenter)
Return a reference to the icenter-th center allowing changes to the center

const SiEnd*
get_end (int iend) *Return a reference to the iend-th end in a manner which does not allow the end to be changed*

SiEnd* **get_mutable_end** (int iend)
Return a reference to the iend-th end allowing changes to the end

int **get_constants_version** ()
Get and set the alignment constants version number

friend std::ostream&
operator << (std::ostream& os, const SiBaseGeometry& me)
Text dump the silicon detector structure.

2.2.1

SiBaseGeometry (int nc, const float* center_zpos, const SiCenter* proto_center, int ne, const float* end_zpos, const SiEnd* proto_end)

Instantiate detector with nc centers, ne end sections

Instantiate detector with *nc* centers, *ne* end sections. The ideal positions are given in *center_zpos* and *end_zpos*. The center and ends defined as *proto_center* and *proto_end* define the center and ends. If the pointer is null, the defaults are used.

2.2.2

SiBaseGeometry (int nc, const float* center_zpos, const std::vector<SiCenter> &centers, int ne, const float* end_zpos, const std::vector<SiEnd> &ends)

Instantiate detector with nc centers, ne end sections

Instantiate detector with *nc* centers, *ne* end sections. Construct a silicon detector with the centers and ends specified in the vectors *conststd* :: *vector* <*SiCenter*>&*centers* and *conststd* :: *vector*<*SiEnd*>&*ends*.

2.2.3

static d0_Ref <SiBaseGeometry> **build_default** (edm::RCP* RCPinit=(edm::RCP*)0)

Build default geometry

Build default geometry. Create the default geometry from the parameters contained in the RCP file SiBaseGeometry.rcp.

Parameters: `RCPinit` — Pointer to RCP containing default geometry information.

2.2.4

const std::list <dgs::GeometryElement*> **get_children** ()

Return the address of the children

Return the address of the children. Used within dgs::GeometryElement and ReferencePoint to do positioning.

2.3

class **SiLadderType**

Create prototype ladder from RCP

Create prototype ladder from RCP. Instantiate a collection of the RCP parameters describing the id-th ladder type in RCPinit. A prototype of the ladder is instantiated and pointed to by _an_ instance. This prototype is lost when this class is destroyed.

2.4

class **SiLayerType**

Create prototype layer from RCP

Create prototype layer from RCP. Instantiate a collection of the RCP parameters describing the id-th layer type in RCPinit. If you have the ladders already, then an instance of the layer is pointed to by _an_ instance.

2.5 class **SiBarrelType**

Create prototype barrel from RCP

Create prototype barrel from RCP. Instantiate a collection of the RCP parameters describing the id-th barrel type in RCPinit. If you have the layers already, then a prototype of the barrel is pointed to by _ an_ instance.

2.6 class **SiWedgeType**

Create prototype wedge from RCP

Create prototype wedge from RCP. Instantiate a collection of the RCP parameters describing the id-th wedge type in RCPinit. A prototype of the wedge is instantiated and pointed to by _ an _ instance. This prototype is lost when this class is destroyed.

2.7 class **SiWedgeHolderType**

Create prototype wedgeholder from RCP

Create prototype wedgeholder from RCP. Instantiate a collection of the RCP parameters describing the id-th wedgeholder type in RCPinit. If you have the wedgeholders already, then an instance of the wedgeholder is pointed to by _ an _instance.

2.8

class **SiDiskType**

Create prototype disk from RCP

Create prototype disk from RCP. Instantiate a collection of the RCP parameters describing the id-th disk type in RCPinit. If you have the disks already, then an instance of the disk is pointed to by _ an _ instance.

2.9

class **SiCentralType**

Create prototype central from RCP

Create prototype central from RCP. Instantiate a collection of the RCP parameters describing the id-th central tracker type in RCPinit. If you have barrels and disks already, then a prototype of the central is pointed to by _ an _ instance.

2.10

class **SiCenterType**

Create prototype center from RCP

Create prototype center from RCP. Instantiate a collection of the RCP parameters describing the id-th center tracker type in RCPinit. If you have the centrals already, then a prototype of the center is pointed to by _ an _ instance.

2.11 class **SiEndType**

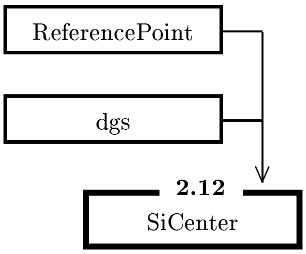
Create prototype end from RCP

Create prototype end from RCP. Instantiate a collection of the RCP parameters describing the id-th end tracker type in RCPinit. If you have the disks already, then a prototype of the end is pointed to by _ an _ instance.

2.12 class **SiCenter** : public dgs ReferencePoint

SMT central detector (default: six barrels and F disks) A SiCenter is used to position the central units

Inheritance



Public Members

	SiCenter ()	*Default constructor (empty center).*	
2.12.1	**SiCenter** (const std::vector<SiCentral> &centrals, const std::vector<float> &z_positions)	*Construct a center containing the given centrals*	16
2.12.2	const std::list <dgs::GeometryElement*>		

SMT central detector (default: six barrels and F disks) A SiCenter is used to position the central units. In the current detector configuration there are eight central units, the outer units contain three F disks, the six inner ones contain one barrel and one F disk.

2.12.1

SiCenter (const std::vector<SiCentral> &centrals, const std::vector<float> &z_positions)

Construct a center containing the given centrals

Construct a center containing the given centrals.

Centrals are positioned according to the vector *z_positions*.

2.12.2

const std::list <dgs::GeometryElement*> **get_children** ()

Return the address of the children

Return the address of the children. Used within dgs::GeometryElement and ReferencePoint to do positioning.

2.12.3

int **get_barrel_count** ()

Simplified interface to barrels

Simplified interface to barrels. This interface knows the distributions of barrels within the set of centrals so users can simply ask for a given barrel.

2.12.4

int **get_disk_count** ()

Simplified interface to disks

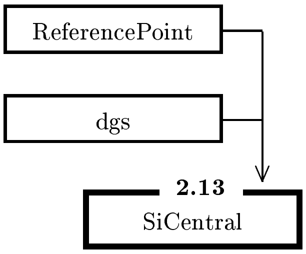
Simplified interface to disks. This interface knows the distributions of disks within the set of centrals so users can simply ask for a given disk.

2.13

class **SiCentral** : public dgs ReferencePoint

SMT central unit

Inheritance



Public Members

SiCentral () — *Default constructor (empty central).*

2.13.1 **SiCentral** (int nb, float* barrel_zpos,
const SiBarrel* proto_barrels, int nd,
float* disk_zpos,
const SiDisk* proto_disks)
Instantiate a central unit with nb barrels and nd disks 19

2.13.2 **SiCentral** (const std::vector<float>& barrel_zpos,
const std::vector<SiBarrel>& barrels,
const std::vector<float>& disk_zpos,
const std::vector<SiDisk>& disks)
Instantiate a central unit with nb barrels, nd disks 20

2.13.3 const std::list <dgs::GeometryElement*>
get_children () *Return the address of the children* 20

int **get_barrel_count** ()
Return the number of barrels.

const SiBarrel*

	get_barrel (int ib)	*Return a constant reference to the ibarrel-th ibarrel.*
SiBarrel*	**get_mutable_barrel** (int ib)	*Return a modifiable reference to the ibarrel-th ibarrel.*
int	**get_disk_count** ()	*Return the number of disks.*
const SiDisk*	**get_disk** (int id)	*Return a constant reference to the idisk-th idisk.*
SiDisk*	**get_mutable_disk** (int id)	*Return a modifiable reference to the idisk-th idisk.*
friend std::ostream&	**operator** << (std::ostream& os, const SiCentral& me)	*Text dump a central unit.*

SMT central unit. Each central contains *nb* barrels and *nd* disks.

2.13.1

SiCentral (int nb, float* barrel_zpos, const SiBarrel* proto_barrels, int nd, float* disk_zpos, const SiDisk* proto_disks)

Instantiate a central unit with nb barrels and nd disks

Instantiate a central unit with *nb* barrels and *nd* disks. The ideal positions are given in *barrel_pos* and *disk_pos*.

2.13.2

SiCentral (const std::vector<float>& barrel_zpos, const std::vector<SiBarrel>& barrels, const std::vector<float>& disk_zpos, const std::vector<SiDisk>& disks)

Instantiate a central unit with nb barrels, nd disks

Instantiate a central unit with *nb* barrels, *nd* disks. Construct a central with the barrels and disks specified in the vectors *conststd* :: *vector*<*SiBarrel*>&*barrels* and *conststd* :: *vector*<*SiDisk*>&*disks*.

2.13.3

const std::list <dgs::GeometryElement*> **get_children** ()

Return the address of the children

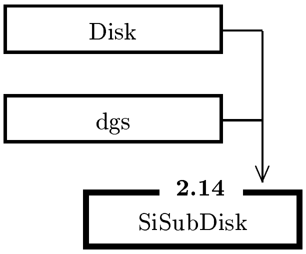
Return the address of the children. Used within dgs::GeometryElement and ReferencePoint to do positioning.

2.14

class **SiSubDisk** : public dgs Disk

Implement a silicon subdisk

Inheritance



Public Members

SiSubDisk () *Default constructor (empty subdisk).*

2.14.1 **SiSubDisk** (float radius, int nwedgeholders, float phi0, const SiWedgeHolder* prototype)
Instantiate a standard subdisk .. 22

2.14.2 **SiSubDisk** (float radius, int nwedgeholders, float phi0, const std::vector<SiWedgeHolder> &wedgeholders)
Instantiate a subdisk containing the given wedgeholders 22

2.14.3 const std::list <dgs::GeometryElement*> **get_children** () *Return the address of the children* 23

int **get_wedgeholder_count** () *Return the number of wedgeholders.*

const SiWedgeHolder* **get_wedgeholder** (int iwedho) *Return constant reference to a wedgeholder.*

SiWedgeHolder* **get_mutable_wedgeholder** (int iwedho) *Return modifiable reference to a wedgeholder.*

2.14.1

SiSubDisk (float radius, int nwedgeholders, float phi0, const SiWedgeHolder* prototype)

Instantiate a standard subdisk

Instantiate a standard subdisk.

Parameters:
`radius` — radius of the subdisk.
`nwedgeholders` — number of wedgeholders attached to the subdisk.
`phi0` — phi value of the first wedgeholder.

2.14.2

SiSubDisk (float radius, int nwedgeholders, float phi0, const std::vector<SiWedgeHolder> &wedgeholders)

Instantiate a subdisk containing the given wedgeholders

Instantiate a subdisk containing the given wedgeholders.

Parameters:
`radius` — radius of the subdisk.
`nwedgeholders` — number of wedgeholders attached to the subdisk.
`phi0` — phi value of the first wedgeholder.

2.14.3

const std::list <dgs::GeometryElement*> **get_children** ()

Return the address of the children

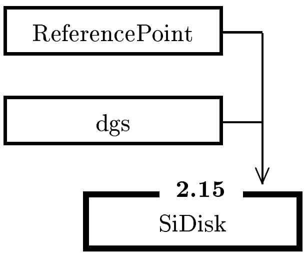
Return the address of the children. Used within dgs::GeometryElement and ReferencePoint to do positioning.

2.15

class **SiDisk** : public dgs ReferencePoint

Implement a silicon disk

Inheritance



Public Members

SiDisk () *Default constructor (empty disk).*

2.15.1 **SiDisk** (float radius, int nwedgeholders, float phi0,
bool flipped,
const SiWedgeHolder* prototype,
float phi1, float separation, bool spun)
Instantiate a standard disk 25

2.15.2 **SiDisk** (float radius, int nwedgeholders, float phi0,
bool flipped, const
std::vector<SiWedgeHolder>& wedgeholders,
float phi1, float separation, bool spun)

Implement a silicon disk. A disk contains wedge holders, which are pure positioners and which hold one wedge in F disks and two wedges in H disks. Within a given disk, wedge holders are numbered from 1 to N.

This implementation defines logical subdisks. These are not part of the standard geometry description, so need not be directly accessed. The subdisk1 (subdisk2) wedge holders are at negative (positive) z in the local disk coordinate system and contain odd (even) numbered wedge holders.

2.15.1

SiDisk (float radius, int nwedgeholders, float phi0, bool flipped, const SiWedgeHolder* prototype, float phi1, float separation, bool spun)

Instantiate a standard disk

Instantiate a standard disk. It consists of subdisk 1 at negative z and subdisk 2 at positive z in local disk coordinates.

Parameters:
`radius` — disk radius.
`phi0` — phi value at the center of the first wedgeholders in subdisks 1.
`phi1` — phi value at the center of the first wedgeholders in subdisks 2. Constraint: 0 < = phi0 < = phi1 < = pi
`flipped` — 0 or 1 for p or n side facing the IP.
`separation` — subdisks 1 (2) are positioned at +(-)separation/2.

2.15.2

SiDisk (float radius, int nwedgeholders, float phi0, bool flipped, const std::vector<SiWedgeHolder>& wedgeholders, float phi1, float separation, bool spun)

Instantiate a disk containing the specified wedgeholders

Instantiate a disk containing the specified wedgeholders. It consists of subdisk 1 at negative z and subdisk 2 at positive z in local disk coordinates.

Parameters:

`radius` — disk radius.
`phi0` — phi value at the center of the first wedgeholders in subdisks 1.
`phi1` — phi value at the center of the first wedgeholders in subdisks 2. Constraint: 0 < = phi0 < = phi1 < = pi
`flipped` — 0 or 1 for p or n side facing the IP.
`separation` — subdisks 1 (2) are positioned at +(-)separation/2. This constructor uses 'friend' access to SiSubDisk.

2.15.3

const std::list <dgs::GeometryElement*> **get_children** ()

Return the address of the children

Return the address of the children. Used within dgs::GeometryElement and ReferencePoint to do positioning.

2.15.4

void **resize** (float dr)

Mimic the Disk positioning and resizing by passing them on to subdisks

Mimic the Disk positioning and resizing by passing them on to subdisks. These will be used during alignment.

2.15.5

void **set_separation** (float dsep)

Vary the separation between subdisks

Vary the separation between subdisks. Used during alignment (not implemented yet).

2.15.6

const SiWedgeHolder* **get_wedgeholder** (int iwedho)

Return a reference to an wedge holder

Return a reference to an wedge holder. The wedge holder may not be modified. This uses the silicon numbering scheme: iwedho = 1 to N for wedge holders, internally mapped onto subdisks. In the mapping, subdisk1 (subdisk2) wedge holders correspond to odd (even) numbered D0 wedge holders.

2.15.7

SiWedgeHolder* **get_mutable_wedgeholder** (int iwedho)

Return a reference to an wedgeholder

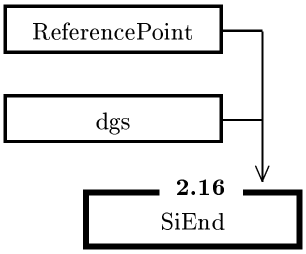
Return a reference to an wedgeholder. The wedgeholder may be modified as a result. This uses the silicon numbering scheme: iwedho = 1 to N for wedge holders, internally mapped onto subdisks. In the mapping, subdisk1 (subdisk2) wedge holders correspond to odd (even) numbered D0 wedge holders.

2.16

class **SiEnd** : public dgs ReferencePoint

SMT end unit

Inheritance



Public Members

		SiEnd ()	*Default constructor (empty end).*	
2.16.1		**SiEnd** (int ndis, std::vector<float> disk_zpos, const SiDisk* proto_disk)	*Instantiate an end unit with nd disks*	29
2.16.2		**SiEnd** (int ndis, std::vector<float> disk_zpos, const std::vector<SiDisk> &disks)	*Instantiate an end unit with nd disks*	29
2.16.3	const std::list <dgs::GeometryElement*>	**get_children** ()	*Return the address of the children*	30
	int	**get_disk_count** ()	*Return the number of disks.*	
	const SiDisk*	**get_disk** (int idisk)	*Return a constant reference to the idisk-th idisk.*	
	SiDisk*	**get_mutable_disk** (int idisk)	*Return a modifiable reference to the idisk-th idisk.*	
	friend std::ostream&			

operator << (std::ostream& os, const SiEnd& me)
Text dump an end unit.

SMT end unit. Each end contains *nd* disks.

2.16.1

SiEnd (int ndis, std::vector<float> disk_zpos, const SiDisk* proto_disk)

Instantiate an end unit with nd disks

Instantiate an end unit with *nd* disks. The ideal positions are given in *disk_pos*.

2.16.2

SiEnd (int ndis, std::vector<float> disk_zpos, const std::vector<SiDisk> &disks)

Instantiate an end unit with nd disks

Instantiate an end unit with *nd* disks. Construct an end with the disks specified in the vector $conststd :: vector{<}SiDisk{>}\&disks$.

2.16.3

const std::list <dgs::GeometryElement*> **get_children** ()

Return the address of the children

Return the address of the children. Used within dgs::GeometryElement and ReferencePoint to do positioning.

2.17

class **SiGeometer**

The singleton providing fast access to the silicon geometry

Public Members

static SiGeometer*
get_instance () *The singleton and update portions of the interface used by GeometryManager via the proxy geometer*

bool **check_silicon** () *Check if a complete silicon geometry already exists.*

void **refresh** (d0_Ref<const SiBaseGeometry> new_silicon) *Update internal information to use the new geometry.*

d0_Ref <const SiBaseGeometry>
get_silicon () *Return a pointer to the complete silicon geometry.*

dgs::Handle <const SiLadder>
get_ladder_handle (int ibarrel, int ilayer, int iladder) *Return an handle to the requested ladder.*

dgs::Handle <const SiWedge>

get_wedge_handle (int idisktype, int idisk, int iwedge)
Return a handle to the requested wedge.

dgs::Handle <const SiWedge>
get_wedge_handle (int idisktype, int idisk, int isubdisk, int iwedge)
Return a handle to the requested wedge.

const SiLadder*
get_ladder (int ibarrel, int ilayer, int iladder)
Return a pointer to a constant ladder.

const SiWedge*
get_wedge (int idisktype, int idisk, int iwedge)
Return a pointer to a constant wedge.

const SiWedge*
get_wedge (int idisktype, int idisk, int isubdisk, int iwedge)
Return a pointer to a constant wedge.

int **make_handle_l_id** (int ibar, int ilay, int ilad)
Generate a ladder id for use in the HandleCache.

int **make_handle_w_id** (int idisty, int idis, int iwed)
Generate a wedge id for use in the HandleCache.

int **make_handle_w_id** (int idisty, int idis, int isubdis, int iwed)
Generate a wedge id for use in the HandleCache.

bool **add_notifier** (notifier)
Register a function to be called when the geometry is refreshed

bool **remove_notifier** (notifier)
Unregister a refresh function

2.18 class **SiGeometerDecoupler**

Define a buffer class to decouple the Silicon Geometer from the geometry manager

Public Members

SiGeometerDecoupler ()
Default constructor.

void **refresh** (d0_Ref<const SiBaseGeometry>replacement_geometry)
Pass the new geometry on to the appropriate Geometer

Define a buffer class to decouple the Silicon Geometer from the geometry manager. When a new geometry is required, the geometry manager calls:

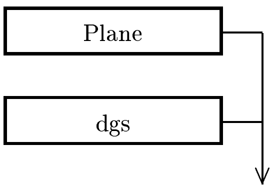
SiGeometerDecoupler::refresh(pointer _to _new _silicon _geometry)

and the implementation of refresh passes this on to the actual silicon geometer.

2.19 class **SiLadder** : public dgs Plane

Define a silicon ladder.

Inheritance



2.19 SiLadder

Public Members

SiLadder () *Default constructor (a ladder of zero dimensions)*

SiLadder (float dxhalf, float dyhalf) *Instantiate a ladder with definite dimensions.*

2.19.1 const std::list <dgs::GeometryElement*>
get_children () *Return the address of the children* 34

virtual SpacePoint
local_to_global (const dgs::CartesianCoordinate& lpt)
Transform a point in the local ladder coordinate system to a global point in the D0 coordinate system

virtual MatrixD
local_to_global (const dgs::CartesianMatrix& theMatrix)
Transform a matrix in the local ladder coordinate system to a matrix in the global D0 coordinate system

virtual dgs::CartesianCoordinate
global_to_local (const SpacePoint& gpt)
Transform a point in the D0 coordinate system to a point in the local ladder coordinate system

friend std::ostream&
operator<< (std::ostream& os, const SiLadder& me)
Text dump a ladder.

friend bool
operator== (const SiLadder& lhs, const SiLadder &rhs)
Compare the size and position of the ladders

friend bool

operator!= (const SiLadder& lhs,
const SiLadder &rhs)
and do it again

2.19.1

const std::list <dgs::GeometryElement*> **get_children** ()

Return the address of the children

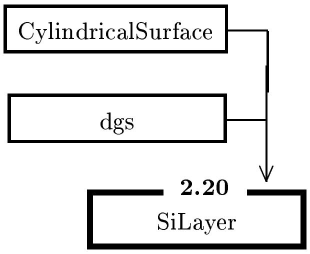
Return the address of the children. Used within dgs::GeometryElement and ReferencePoint to do positioning.

2.20

class **SiLayer** : public dgs CylindricalSurface

Implement the positioning surfaces for a silicon barrel layer

Inheritance



Public Members

SiLayer () *Default constructor (empty layer).*

int **get_ladder_count** ()
Return the number of ladders.

const SiLadder* **get_ladder** (int ilad)
Return a constant reference to the iladder-th iladder.

SiLadder* **get_mutable_ladder** (int ilad)
Return a modifiable reference to the iladder-th iladder.

friend std::ostream& **operator<<** (std::ostream& os, const SiLayer& me)
Is this needed like t00.07.01 (A. Patwa)?

friend bool **operator==** (const SiLayer& lhs, const SiLayer &rhs)
Recursively test that all SiLadders are identical in both layers.

friend bool **operator!=** (const SiLayer& lhs, const SiLayer &rhs)
and the opposite sense...

2.20.1

SiLayer (float radius, float zhalf, int nladders, float phi0, bool flipped=false, const SiLadder* prototype=0)

Instantiate a standard layer

Instantiate a standard layer.

Parameters:
`radius` — radius of the silicon barrel layer.
`zhalf` — hald length, along the z direction of the ladders on the layer.
`nladders` — number of ladders on the layer.
`phi0` — phi value of the first ladder.
`flipped` — 0 or 1 for p or n side facing inward.

2.20.2

SiLayer (float radius, float zhalf, int nladders, float phi0, bool flipped, const std::vector<SiLadder> &ladder-s)

Instantiate a layer containing the specified ladders

Instantiate a layer containing the specified ladders.

Parameters:
`radius` — radius of the silicon barrel layer.
`zhalf` — hald length, along the z direction of the ladders on the layer.
`nladders` — number of ladders on the layer.
`phi0` — phi value of the first ladder.
`flipped` — 0 or 1 for p or n side facing inward.

2.20.3

const std::list <dgs::GeometryElement*> **get_children** ()

Return the address of the children

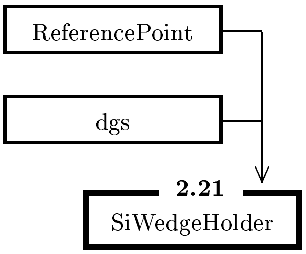
Return the address of the children. Used within dgs::GeometryElement and ReferencePoint to do positioning.

2.21

class **SiWedgeHolder** : public dgs ReferencePoint

Given the different structure of F and H disks, a place holder is created to house the appropriate number of wedges, 1 (double-sided) for F disks and 2 (single-sided, glued back to back on a beryllium support) for H disks

Inheritance



Public Members

	SiWedge ()	*Default constructor (a wedge of zero dimensions)*
	SiWedge (float dswidth, float dheight, float dangle)	*Instantiate a wedge with definite dimensions.*
2.21.1	const std::list <dgs::GeometryElement*> **get_children** ()	*Return the address of the children* 39
	virtual SpacePoint	

local_to_global (const dgs::CartesianCoordinate& lpt)
Transform a point in the local wedge coordinate system (for a p-side: z-axis along the height pointing up, y-axis pointing into the wedge and x-axis to define a right-handed coordinate system) to a global point in the D0 coordinate system

virtual MatrixD
local_to_global (const dgs::CartesianMatrix& theMatrix)
Transform a matrix in the local wedge coordinate system (for a p-side: z-axis along the height pointing up, y-axis pointing into the wedge and x-axis to define a right-handed coordinate system) to a matrix in the global D0 coordinate system

virtual dgs::CartesianCoordinate
global_to_local (const SpacePoint& gpt)
Transform a point in the D0 coordinate system (for a p-side: z-axis along the height pointing up, y-axis pointing into the wedge and x-axis to define a right-handed coordinate system) to a point in the local wedge coordinate system

friend std::ostream&
operator<< (std::ostream& os, const SiWedge& me)
Text dump a wedge.

SiWedgeHolder ()
Default constructor (empty wedgeholder).

2.21.2 **SiWedgeHolder** (int nwedges, const SiWedge* prototype, float separation)

int **get_wedge_count** ()
Return the number of wedgeholders.

const SiWedge* **get_wedge** (int iwedge)
Return constant access to a wedgeholder.

SiWedge* **get_mutable_wedge** (int iwedge)
Return modifiable access to a wedgeholder.

friend std::ostream& **operator** << (std::ostream& os, const SiWedgeHolder& me)
Text dump a wedgeholder.

2.21.1

const std::list <dgs::GeometryElement*> **get_children** ()

Return the address of the children

Return the address of the children. Used within dgs::GeometryElement and ReferencePoint to do positioning.

2.21.2

SiWedgeHolder (int nwedges, const SiWedge* prototype, float separation)

Instantiate a standard wedgeholder

Instantiate a standard wedgeholder.

Parameters: `separation` — distance between the two wedges held by the wedgeholder.

2.21.3

SiWedgeHolder (int nwedges, const std::vector<SiWedge> &wedges, float separation)

Construct a wedgeholder containing the given wedges

Construct a wedgeholder containing the given wedges.

Parameters: `separation` — distance between the two wedges held by the wedgeholder.

2.21.4

const std::list <dgs::GeometryElement*> **get_children** ()

Return the address of the children

Return the address of the children. Used within dgs::GeometryElement and ReferencePoint to do positioning.

3 SMT Channel Geometry (*silicon_geometry/channel*)

Names

The *channel* component of the *silicon_geometry* package provides a wrapper for the silicon geometry needed to convert silicon cluster positions to global positions. It provides information relative to the native (i.e. strips) system such as pitch, stereo angle, offset of the first readout strip and converts a point on a given silicon element into a point in the global detector coordinate system.

3.1 class **SiChannelGeometer**

The Channel geometer singleton

Public Members

static SiChannelGeometer*
get_instance () — *Return a pointer to the (sole) instance of the geometer.*

void **refresh** (edm::RCP* channel_map)
Update the silicon channel geometer using an RCP file.

int **get_view_count** (int barrel, int layer, int ladder)

Return the number of views in a given ladder.

const SiLadderStrips* **get_ladder_strips** (int barrel, int layer, int ladder, int view)
Return the strips corresponding to the given ladder.

const SiWedgeStrips* **get_wedge_strips** (int disktype, int disk, int wedge, int view)
Return the strips corresponding to the given wedge.

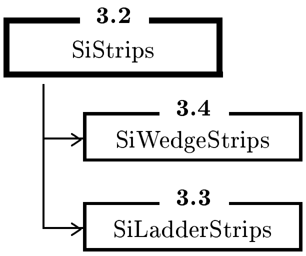
const SiWedgeStrips* **get_wedge_strips** (int disktype, int disk, int subdisk, int wedge, int view)
Return the strips corresponding to the given wedge.

The Channel geometer singleton. It creates instances of SiStrips for converting cluster positions to global DZero coordinate space.

3.2 class **SiStrips**

General strips class

Inheritance



Public Members

General strips class. The strips classes for disks and barrels differ only in the request needed to get the geometry positioning information. Thus, define a base class "SiSrtips" which is the class used by RECO. However, build two different superclasses "SiLadderStrips" and "SiWedgeStrips" whose only purpose is to provide the correct instantiation of an SiStrips class.

3.2.1

SiStrips (float pitch, float stereo, float strip0, float view)

Instantiate a standard strips class

Instantiate a standard strips class.

Parameters:

`pitch` — pitch.
`stereo` — stereo angle.
`strip0` — distance between the first readout strip and the edge of the active area.
`view` — p or n side.

3.2.2

virtual std::vector < std::vector<float> > **transform_error** (const float sxx, const float szz, const float sxz)

Convert errors on the measured positions to global DZero

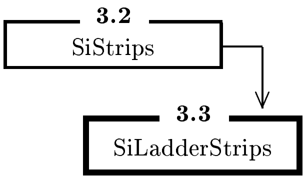
Convert errors on the measured positions to global DZero. The error matrix is represented by a vector of vectors to limit the package contamination.

3.3

class **SiLadderStrips** : public SiStrips

Implement an SiStrips which refers to a ladder

Inheritance



Public Members

SiLadderStrips (float pitch, float stereo, float strip0, float view, dgs::Handle<const SiLadder> location)

Instantiate SiLadderStrips for the given ladder and view.

dgs::CartesianCoordinate
get_global (const dgs::CartesianCoordinate& local_point)
Convert the given local coordinate to its corresponding point in global DZero

std::vector < std::vector<float> >
transform_error (const float sxx, const float szz, const float sxz)
Convert errors to global DZero

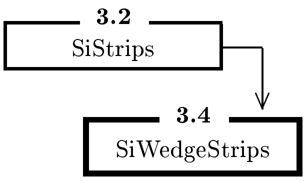
dgs::Handle <const SiLadder>
get_ladder_handle ()
Return the ladder itself.

friend std::ostream&
operator << (std::ostream& os, const SiLadderStrips &me)
Text dump a ladder SiStrips.

3.4 class **SiWedgeStrips** : public SiStrips

Implement an SiStrips which refers to a wedge

Inheritance



Public Members

dgs::CartesianCoordinate

get_global (const dgs::CartesianCoordinate& local_point)
Convert the given local coordinate to its corresponding point in global DZero

std::vector < std::vector<float> >
transform_error (const float sxx, const float szz, const float sxz)
Convert errors to global DZero

dgs::Handle <const SiWedge>
get_wedge_handle ()
Return the wedge itself.

friend std::ostream&
operator << (std::ostream& os, const SiWedgeStrips& me)
Text dump a wedge SiStrips.

4 SMT Material Geometry (*silicon_geometry/material*)

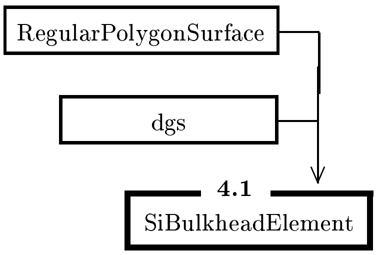
Names

The *material* component of the *silicon_geometry* package provides a wrapper for the silicon geometry needed to support multiple scattering for track reconstruction. Users access this information through the *SiMaterialGeometer* interface.

4.1 class **SiBulkheadElement** : public dgs RegularPolygonSurface

Represent one silicon bulkhead as a set of polygonal structures

Inheritance



Public Members

const std::list <dgs::GeometryElement*>
get_children () *Complete the remaining GeometryElement interface. No kids.*

4.1.1

SiBulkheadElement (const int nsides, const double edge1, const double edge2)

Just construct my dimensions. Don't position me...

Arguments

static SiMaterialGeometer*
make_instance (edm::RCP*rcp)
Build the material from the specified RCP file

static SiMaterialGeometer*
get_instance () *Get the instance*

const PlanarShell <SiWedge> *
get_wedge_material (int idisktype, int idisk, int iwedge)
Get the material (and Handle) for the specified disk

const PlanarShell <SiLadder> *
get_ladder_material (int ibarrel, int ilayer, int iladder)
Get the material (and Handle) for the specified ladder

const PlanarShell <SiBulkhead> *

get_bulkhead_material (const int ibarrel)
Get the material (and Handle) for the specified bulkhead

const TCylindricalShell <SiTrough> *
get_trough_material ()
Get the material for the support trough

void **refresh** () *Get notification that the geometry has changed.*

static SiMaterialGeometer*
_theGeometer *Pointer to the single instance of the material geometry*

D0Material::Material
_composition *The universal ladder/wedge material properties*

All ladders/wedges have same thickness and material composition

float **_thickness** *The universal detector thickness*

D0Material::Material
_bulkhead_composition
The bulkhead material properties

float **_bulkhead_thickness**
The bulkhead thickness

float **_bulkhead_zoffs** *The bulkhead relative positioning information*

D0Material::Material
_trough_composition
The trough material properties

float **_trough_thickness**
The trough thickness

SiMaterialGeometer ()
Hidden constructor and destructor used to manage singleton

mutable std::map <int, PlanarShell<SiLadder>*>
_ladders *Keep caches of all previously requested ladders and disks*

std::vector <SiBulkhead*>

	_bulkheads	*Keep a prototype of each bulkhead and the barrel to which it belongs*
SiTrough*	**_trough**	*Keep a prototype of the sole Trough instance*
	SiTrough (const double length, const double radius, const double dphi)	*Construct a trough*
double	**get_length** ()	*Get the length*
double	**get_radius** ()	*Get the radius information*
double	**get_dphi** ()	*Get the phi extent information*
const std::list <dgs::GeometryElement*>	**get_children** ()	*Complete the GeometryElement interface. No kids.*

4.1.1.1

Provide an interface to the material comprising the RunII silicon detector

Provide an interface to the material comprising the RunII silicon detector. This is a thin surface representation initially for use with reco.

4.2

std::ostream& **operator<<** (ostream& os, const SiBulkhead& me)

Print me...

Arguments

SiBulkhead (const int nl, const std::vector<int> ns, const std::vector<double> inner_edges, const std::vector<double> outer_edges, const std::vector<double> phi_offsets)
Build a bulkhead, explicitly giving necessary information.

const std::list <dgs::GeometryElement*>
get_children () *Complete the remaining GeometryElement interface.*

int **get_layer_count** ()
Get the number of elements

const SiBulkheadElement*
get_layer (const int i)
Get an element

5 SMT Geometry Utilities (*silicon_geometry/utils*)

Utility routines to draw an end view of the barrels(smt_draw_geometry), draw hits on barrels given their local(cluster) coordinates (program smt_draw_data), and perform conversions between cluster coordinates and global coordinates (programs smt_local_to_global and smt_global_to_local)

Class Graph