PCB API Reference

Summary

This PCB API Reference provides a concise reference of the PCB API as part of the Altium Designer Run Time Library for scripting and server development.

The PCB Application Programming Interface reference covers interfaces for PCB objects such as PCB documents and PCB design objects in the PCB Object Model.

The PCB API consists of the PCB Object model and the API functions. The PCB Object Model part of the PCB API is facilitated by the PCB Editor. The PCB Object Model is defined and implemented in the RT_PCB unit and the PCB functions are defined and implemented in the RT_PCBProcs unit.

The PCB design objects and methods are available to use in your scripts in all script languages that Altium Designer supports and are also available to use in your server projects. You normally do need to add appropriate API units in your scripts, but with server projects, you do need to add these units in the Uses clause in your code unit.

The PCB design objects are wrapped by their corresponding PCB interfaces that make it possible to manipulate them by scripts and server code.

Object Interfaces

Basically an interface is simply a list of methods that a class declares that it implements. That is, each method in the interface is implemented in the corresponding class. Interfaces are declared like classes but cannot be directly instantiated and do not have their own method definitions.

PCBServer function

When you need to work with PCB design objects, the starting point is to invoke the **PCBServer** function which returns you the **IPCB_ServerInterface** interface which represents the PCB Editor. With this interface, you can extract all other PCB interfaces.

For example to get an access to the current PCB document open in Altium Designer, you would invoke the **GetCurrentPCBBoard** method from the **IPCB_ServerInterface** interface object.

Getting the currently open PCB document example

```
Board := PCBServer.GetCurrentPCBBoard;
If Board = Nil then Exit;
TheFilename := Board.FileName;
```

Main PCB Interfaces

The **IPCB_Primitive** interface is a generic ancestor interface for all PCB design object interfaces.

The **IPCB_Board** interface represents an existing PCB document.

The **IPCB_ServerInterface** interface represents the PCB server object.

Script Examples

There are PCB script examples in the **\Examples\Scripts\DelphiScripts\PCB** folder which demonstrate the use of PCB interfaces.

See also

PCB Interfaces

PCB Enumerated Types

PCB Constants

PCB Functions

PCB Object Model

The PCB Object Model compromises of PCB Object Interfaces and standalone utility functions that allow you to deal with PCB objects from a PCB document in Altium Designer via object interfaces.

An interface is just a means of access to an object in memory. To have access to the PCB server and massage certain PCB design objects, you need to invoke the **PCBServer** function which extracts the **IPCB_ServerInterface** interface.

The **IPCB_ServerInterface** interface is the main interface and contains many interfaces within. With this interface, you can proceed further by iterating for PCB objects on a PCB document for example.

A simplified PCB Interfaces hierarchy:

IPCB_Primitive

IPCB_Arc

IPCB_Group

IPCB_Net

The **IPCB_ServerInterface** and **IPCB_Board** interfaces to name the few are the main interfaces that you will be dealing with, when you are dealing data from a PCB document.

See also

IPCB_ServerInterface

IPCB_BoardOutline

IPCB_Board

IPCB_LayerStack

IPCB_LayerObject

IPCB_InternalPlane

IPCB_DrillLayerPair

IPCB_MechanicalLayerPairs

IPCB_SystemOptions

IPCB_InteractiveRoutingOptions

IPCB_Arc

IPCB_Pad

IPCB_Via

IPCB_Track

IPCB_Connection

IPCB_Embedded

IPCB_Violation

IPCB_Text

IPCB_Fill

IPCB_Coordinate

IPCB_Dimension

IPCB_Component

IPCB_Polygon

IPCB_Net

IPCB_LibComponent

PCB System Interfaces

IPCB_ServerInterface

(RT_PCB unit)

Overview

When you need to work with PCB design objects in Altium Designer, the starting point is to invoke the **PCBServer** function which returns the **IPCB_ServerInterface** interface. You can extract the all other derived PCB interfaces that are exposed in the I**PCB_ServerInterface** interface.

Note that these **IServerModule** interfaces represent loaded servers in Altium Designer. The Altium Designer application manages single instances of different server modules. Each server can have multiple server document kinds, for example the PCB server supports two server document kinds – PCB and PCBLIB design documents. A loaded server in Altium Designer typically hosts documents and each document in turn hosts a document view and panel views. Thus a PCB server also has the **IServerModule** interface along with the **IPCB_ServerInterface** interface.

Notes

To get an access to the current PCB document open in Altium Designer, you would invoke the **GetCurrentPCBBoard** method from the **IPCB_ServerInterface** interface object to obtain the **IPCB_Board** interface.

The factory methods produce specialized objects. For example the **PCBObjectFactory** method is invoked to produce a new PCB object. You will need to add this object in a PCB board. The **TObjectCreationKind** type denotes how the attributes of a new PCB object is set (either from software default settings or from global settings as defined in the Preferences dialog within PCB).

The **SendMessageToRobots**, **PreProcess and PostProcess** methods are used when you need to keep the Undo system and other sub systems of the PCB editor in synchronization, when you are adding, deleting or modifying objects to/from the PCB document.

IPCB_ServerInterface methods

```
PCBObjectFactory
PCBClassFactory
PCBClassFactoryByClassMember
PCBRuleFactory
PCBContourFactory

PCBContourMaker
LoadCompFromLibrary
DestroyPCBObject
DestroyPCBLibComp

GetPCBBoardByPath
GetCurrentPCBBoard

GetCurrentComponent
ObjectSupports

PreProcess
PostProcess
SendMessageToRobots

GetState_TTFLettersCache
GetState_TTFontsCache

EnableFastParams;
DisableFastParams;

DocumentLiveHighlight_Start
DocumentLiveHighlight_Stop
RefreshDocumentView
```

IPCB_ServerInterface properties

```
InteractiveRoutingOptions
SystemOptions
InteractiveRoutingOptions
CanFastCrossSelect_Emit
CanFastCrossSelect_Receive
SpecialStringConverter
TTFLettersCache
TTFontsCache
```

See also

Creating/Deleting PCB objects and updating the Undo system

Modifying PCB objects and updating the Undo system

TObjectId enumerated values

TDimensionKind enumerated values

TObjectCreationMode enumerated values

IPCB_ObjectClass interface

IPCB_Rule interface

IPCB_LibComponent interface

IPCB_Primitive interface

IPCB_Board interface

IPCB_SystemOptions interface

IPCB_InteractiveRoutingOptions interface

PCB Scripts `from \Examples\Scripts\Delphiscript\PCB` folder.

GetState and SetState Methods

GetState_SystemOptions method

(IPCB_ServerInterface interface)

Syntax

```
Function  GetState_SystemOptions : IPCB_SystemOptions;
```

Description

The function returns the IPCB_SystemOptions interface.

Example

See also

IPCB_ServerInterface interface

IPCB_SystemOptions interface

GetState_InteractiveRoutingOptions method

(IPCB_ServerInterface interface)

Syntax

```
Function  GetState_InteractiveRoutingOptions : IPCB_InteractiveRoutingOptions;
```

Description

Example

See also

IPCB_ServerInterface interface

GetState_CanFastCrossSelect_Emit method

(IPCB_ServerInterface interface)

Syntax

```
Function  GetState_CanFastCrossSelect_Emit     : Boolean;
```

Description

Example

See also

IPCB_ServerInterface interface

GetState_CanFastCrossSelect_Receive method

(IPCB_ServerInterface interface)

Syntax

```
Function  GetState_CanFastCrossSelect_Receive  : Boolean;
```

Description

Example

See also

IPCB_ServerInterface interface

SetState_CanFastCrossSelect_Emit method

(IPCB_ServerInterface interface)

Syntax

```
Procedure SetState_CanFastCrossSelect_Emit   (B : Boolean);
```

Description

Example

See also

IPCB_ServerInterface interface

SetState_CanFastCrossSelect_Receive method

(IPCB_ServerInterface interface)

Syntax

```
Procedure SetState_CanFastCrossSelect_Receive(B : Boolean);
```

Description

Example

See also

IPCB_ServerInterface interface

GetState_SpecialStringConverter method

(IPCB_ServerInterface interface)

Syntax

```
Function  GetState_SpecialStringConverter  : IPCB_SpecialStringConverter
```

Description

This function returns the Special String converter interface which deals with special string formats of PCB text objects.

Example

See also

IPCB_SpecialStringConverter interface

SpecialStringConverter property

Methods

CreatePCBLibComp method

(IPCB_ServerInterface interface)

Syntax

```
Function  CreatePCBLibComp : IPCB_LibComponent;
```

Description

The CreatePCBLibComp method creates a IPCB_LibComponent based object interface that represents a library component within a library document.

Example

```
Var
    CurrentLib    : IPCB_Library;
    NewPCBLibComp : IPCB_LibComponent;
    NewPad        : IPCB_Pad;
Begin
    If PCBServer        = Nil Then Exit;
    CurrentLib         := PcbServer.GetCurrentPCBLibrary;
    If CurrentLib       = Nil Then Exit;

    NewPCBLibComp      := PCBServer.CreatePCBLibComp;
    NewPcbLibComp.Name := 'ANewComponent';

    CurrentLib.RegisterComponent(NewPCBLibComp);
    PCBServer.PreProcess;

    NewPad := PcbServer.PCBObjectFactory(ePadObject,eNoDimension,eCreate_Default);
    NewPad.X        := MilsToCoord(0);
    NewPad.Y        := MilsToCoord(0);
    NewPad.TopXSize := MilsToCoord(62);
    NewPad.TopYSize := MilsToCoord(62);
    NewPad.HoleSize := MilsToCoord(28);
    NewPad.Layer    := eMultiLayer;
    NewPad.Name     := '1';
    NewPCBLibComp.AddPCBObject(NewPad);

    PCBServer.SendMessageToRobots(CurrentLib.Board.I_ObjectAddress,
        c_Broadcast,
        PCBM_BoardRegisteration,
        NewPCBLibComp.I_ObjectAddress);

    PCBServer.PostProcess;

    CurrentLib.CurrentComponent := NewPcbLibComp;
    CurrentLib.Board.ViewManager_FullUpdate;
End;
```

See also

IPCB_ServerInterface interface

IPCB_LibComponent interface

DestroyPCBLibComp method

IPCB_Board and its ViewManager_FullUpdate method.

DestroyPCBLibObject method

(IPCB_ServerInterface interface)

Syntax

```
Procedure DestroyPCBLibComp  (Var APCBLibComp : IPCB_LibComponent);
```

Description

This procedure destroys a footprint within a library but it is not eliminated from the computer's memory. A library is composed of footprints as pages and each footprint is represented by the **IPCB_LibComponent** interface.

Example

See also

IPCB_ServerInterface interface

PCBDestroyObject method

(IPCB_ServerInterface interface)

Syntax

```
Procedure DestroyPCBObject   (Var APCBObject       : IPCB_Primitive);
```

Description

This procedure destroys a PCB object from the PCB document. It is removed but not eliminated from computer memory. For instance, the Undo system can bring this object back.

Example

```
var
    CurrentPCBBoard : IPCB_Board;
    Iterator        : IPCB_BoardIterator;
    Track           : IPCB_Track;
    OldTrack        : IPCB_Track;
Begin
    CurrentPCBBoard := PCBServer.GetCurrentPCBBoard;
    If CurrentPCBBoard = Nil Then Exit;

    Iterator := CurrentPCBBoard.BoardIterator_Create;
    If Iterator = Nil Then Exit;
    Iterator.AddFilter_ObjectSet(MkSet(eTrackObject));
    Iterator.AddFilter_LayerSet(MkSet(eTopLayer));
    PCBServer.PreProcess;

    Try
        Track := Iterator.FirstPCBObject;
        While Track <> Nil Do
        Begin
            OldTrack := Track;
            Track := Iterator.NextPCBObject;
            CurrentPCBBoard.RemovePCBObject(OldTrack);
            PCBServer.SendMessageToRobots(CurrentPCBBoard.I_ObjectAddress,
                                          c_BroadCast,
                                          PCBM_BoardRegisteration,
                                          OldTrack.I_ObjectAddress);
        End;
    Finally
        CurrentPCBBoard.BoardIterator_Destroy(Iterator);
    End;
    PCBServer.PostProcess;

    // Refresh PCB screen
    Client.SendMessage('PCB:Zoom', 'Action=Redraw' , 255, Client.CurrentView);
```

```
End;
```

See also

IPCB_ServerInterface interface

IPCB_LibComponent interface

GetCurrentPCBBoard method

(IPCB_ServerInterface interface)

Syntax

```
Function  GetCurrentPCBBoard : IPCB_Board;
```

Description

This function returns you the **IPCB_Board** interface which represents the PCB document OR the PCB Library document. The **IPCB_Board** interface has a **IsLibrary** function which determines which type the document is; the PCB or PCBLib document.

Example

```
Var
    Board : IPCB_Board;
Begin
    Board := PCBServer.GetCurrentPCBBoard;
    If Board = Nil Then Exit;

    If Not Board.IsLibrary Then
    Begin
        showMessage('This is not a PCB library document.');
        Exit;
    End;
End;
```

See also

IPCB_ServerInterface interface

GetCurrentPCBLibrary property

(IPCB_ServerInterface interface)

Syntax

```
Function  GetCurrentPCBLibrary : IPCB_Library;
```

Description

This function returns the IPCB_Library interface which represents the PCB library document.

Example

```
Var
    CurrentLib        : IPCB_Library;
    FootprintIterator : IPCB_LibraryIterator;
    Footprint         : IPCB_LibComponent;
Begin
    CurrentLib := PCBServer.GetCurrentPCBLibrary;
    If CurrentLib = Nil Then
    Begin
        ShowMessage('This is not a PCB Library document');
        Exit;
    End;

    // Each page of library is a footprint
    FootprintIterator := CurrentLib.LibraryIterator_Create;
```

```
    FootprintIterator.SetState_FilterAll;
    Footprint := FootprintIterator.FirstPCBObject;
    While Footprint <> Nil Do
    Begin
       // do what you want with the footprint...
       Footprint := FootprintIterator.NextPCBObject;
    End;
Finally
    CurrentLib.LibraryIterator_Destroy(FootprintIterator);
End;
```

See also

IPCB_ServerInterface interface

IPCB_Library interface

GetPCBBoardByPath method

(IPCB_ServerInterface interface)

Syntax

```
Function GetPCBBoardByPath (APath : TPCBString) : IPCB_Board;
```

Description

This function returns you the **IPCB_Board** interface representing the PCB document only if the path (APath parameter) represents a valid PCB document.

Example

See also

IPCB_ServerInterface interface

GetPCBLibraryByPath method

(IPCB_ServerInterface interface)

Syntax

```
Function  GetPCBLibraryByPath (Const APath : TPCBString) : IPCB_Library;
```

Description

This function returns you the **IPCB_Library** interface representing the PCB document only if the path (APath parameter) represents this document.

Example

See also

IPCB_ServerInterface interface

IPCB_Library interface

ObjectSupports method

(IPCB_ServerInterface interface)

Syntax

```
Function ObjectSupports(Const Instance : TObject; Const IID : TGUID; Out Intf) : Boolean;
```

Description

This function checks if the PCB object in question is in fact one of the valid PCB object interfaces.

Example

See also

IPCB_ServerInterface interface

PCBClassObjectFactory method

(IPCB_ServerInterface interface)

Syntax

```
Function  PCBClassFactory(Const AClassKind : TObjectId) : IPCB_ObjectClass;
```

Description

This function produces an object represented by the **IPCB_ObjectClass** interface. An Object class is a Design Rules Class that can store members which represent a group of design objects targetted by the design rules system in the PCB editor.

Example

```
Procedure CreateANewNetClass;
Var
    Board    : IPCB_Board;
    NetClass : IPCB_ObjectClass;
Begin
    Board := PCBServer.GetCurrentPCBBoard;
    If Board = Nil Then Exit;

    Try
        PCBServer.PreProcess;
        NetClass := PCBServer.PCBClassFactoryByClassMember(eClassMemberKind_Net);
        NetClass.SuperClass := False;
        NetClass.Name := 'NetGndClass';
        NetClass.AddMemberByName('GND');
        Board.AddPCBObject(NetClass);

    Finally
        PCBServer.PostProcess;
    End;
End;
```

See also

IPCB_ServerInterface interface

PCBClassObjectFactoryByClassMember method

PCBClassObjectFactoryByClassMember method

(IPCB_ServerInterface interface)

Syntax

```
Function  PCBClassFactoryByClassMember (Const AClassKind : TClassMemberKind) :
IPCB_ObjectClass;
```

Description

This function produces an object represented by the IPCB_ObjectClass interface. An Object class is a Design Rules Class that can store members which represent a group of design objects targetted by the design rules system in the PCB editor.

Example

```
Procedure CreateANewNetClass;
Var
    Board    : IPCB_Board;
    NetClass : IPCB_ObjectClass;
Begin
    Board := PCBServer.GetCurrentPCBBoard;
    If Board = Nil Then Exit;
```

```
    Try
        PCBServer.PreProcess;
        NetClass := PCBServer.PCBClassFactoryByClassMember(eClassMemberKind_Net);
        NetClass.SuperClass := False;
        NetClass.Name := 'NetGndClass';
        NetClass.AddMemberByName('GND');
        Board.AddPCBObject(NetClass);

    Finally
        PCBServer.PostProcess;
    End;
End;
```

See also

IPCB_ServerInterface interface

PCBClassObjectFactory method

LoadCompFromLibrary method

(IPCB_ServerInterface interface)

Syntax

```
Function  LoadCompFromLibrary(Const APattern : TPCBString;
                              Const ALibPath : TPCBString) : IPCB_LibComponent;
```

Description

This function produces an object which is represented by the **IPCB_LibComponent** interface. A footprint in a library is also represented by the **IPCB_LibComponent** interface.

Example

See also

IPCB_ServerInterface interface

IPCB_LibComponent interface

PCBObjectFactory method

(IPCB_ServerInterface interface)

Syntax

```
Function PCBObjectFactory(Const AObjectId      : TObjectId;
                          Const ADimensionKind : TDimensionKind = eNoDimension;
                          Const ACreationMode  : TObjectCreationMode =
                          eCreate_Default) : IPCB_Primitive;
```

Description

This function produces a PCB design object which is represented by the IPCB_Primitive interface. The IPCB_Primitive interface is the ancestor interface for all PCB design objects in Altium Designer.

The **TObjectID** value determines which object you wish to produce.

The **TDimensionKind** value determines which dimension object you wish to produce. By default it is eNoDimension.

The **TObjectCreationMode** type determines which default values are used - from the PCB Preferences dialog or default values used internally from the PCB Editor.

Example

```
Var
    Board : IPCB_Board;
    Via   : IPCB_Via;
Begin
```

```
    Board := PCBServer.GetCurrentPCBBoard;
    If Board = Nil Then Exit;
    // Create a Via object
    Via := PCBServer.PCBObjectFactory(eViaObject, eNoDimension, eCreate_Default);
    Via.X         := MilsToCoord(7500);
    Via.Y         := MilsToCoord(7500);
    Via.Size      := MilsToCoord(50);
    Via.HoleSize  := MilsToCoord(20);
    Via.LowLayer  := eTopLayer;
    Via.HighLayer := eBottomLayer;
    // Put the new Via object in the board object
    Board.AddPCBObject(Via);

    // Refresh the PCB screen
    Client.SendMessage('PCB:Zoom', 'Action=Redraw' , 255, Client.CurrentView);
End;
```

See also

IPCB_ServerInterface interface

PCBRuleFactory method

(IPCB_ServerInterface interface)

Syntax

```
Function  PCBRuleFactory(Const ARuleKind : TRuleKind) : IPCB_Rule;
```

Description

This function produces a design rule object which is represented by the **IPCB_Rule** interface.

Example

See also

IPCB_ServerInterface interface

PostProcess method

(IPCB_ServerInterface interface)

Syntax

```
Procedure PostProcess;
```

Description

This procedure cleans up the robots process in the PCB editor, after a **PreProcess** method and **SendMessageToRobots** messages have been invoked. This also stops the robots from listening to any more PCB messages.

Example

```
PCBServer.PreProcess;

//Notify PCB that the fill object is going to be changed.
PCBServer.SendMessageToRobots(
        Fill.I_ObjectAddress,
        c_Broadcast,
        PCBM_BeginModify ,
        c_NoEventData);

Fill.Layer := eBottomLayer;
```

```
//Notify PCB that the fill object has been changed.
PCBServer.SendMessageToRobots(
        Fill.I_ObjectAddress,
        c_Broadcast,
        PCBM_EndModify ,
        c_NoEventData);

PCBServer.PostProcess;
```

See also

IPCB_ServerInterface interface

PreProcess method

SendMessageToRobots method

Preprocess method

(IPCB_ServerInterface interface)

Syntax

```
Procedure PreProcess;
```

Description

This procedure initializes the PCB robots in the PCB editor so that the robots can listen to any PCB messages being broadcasted. It is highly recommended to use Try Finally End blocks in your scripts or server code so that **PreProcess** and **PostProcess** methods can always be executed. This is imperative to ensure that the PCB editor is in the correct state.

Example

```
Try
    PCBServer.PreProcess;

    //Notify PCB that the fill object is going to be changed.
    PCBServer.SendMessageToRobots(
        Fill.I_ObjectAddress,
        c_Broadcast,
        PCBM_BeginModify ,
        c_NoEventData);

    Fill.Layer := eBottomLayer;

    //Notify PCB that the fill object has been changed.
    PCBServer.SendMessageToRobots(
        Fill.I_ObjectAddress,
        c_Broadcast,
        PCBM_EndModify ,
        c_NoEventData);
Finally
    PCBServer.PostProcess;
End;
```

See also

IPCB_ServerInterface interface

PostProcess method

SendMessageToRobots method

SendMessageToRobots method

(IPCB_ServerInterface interface)

Syntax

```
Procedure SendMessageToRobots(Source, Destination : Pointer; MessageID : Word; MessageData :
Pointer);
```

Description

The **SendMessageToRobots** method sends a specific Message with the Source and Designation parameters into the PCB editor where the PCB robots are listening. It is necessary to invoke the **PreProcess** method first, and to invoke the **PostProcess** method after the **SendMessageToRobots** methods.

Parameters

The **Source** parameter represents the PCB object. You need to pass in the address of this object, thus the **I_ObjectAddress** method of a PCB Object Interface returns the address.

The **Destination** parameter normally has the **c_Broadcast** constant which denotes that the message is being broadcasted into the PCB editor.

The **MessageId** parameter represents one of the PCB message constants. See PCB Messages section for more details.

The **MessageData** parameter can be one of the following values - **c_NoEventData** when a PCB object is being modified, or when this object is being registered into the PCB editor, and you need to pass in the address of this object, thus the **I_ObjectAddress** method of a PCB Object Interface need to be invoked to return the address.

Notes

The PCB Messages are messages that are broadcasted into the PCB Editor server by the **SendMessageToRobots** method. There are different types of messages that describe a specific action within the PCB server.

Example 1 - SendMessageToRobots with BeginModify and EndModify calls

```
//Initialize robots in PCB
PCBServer.PreProcess;

//Notify PCB that the fill object is going to be changed.
PCBServer.SendMessageToRobots(
        Fill.I_ObjectAddress,
        c_Broadcast,
        PCBM_BeginModify ,
        c_NoEventData);

Fill.Layer := eBottomLayer;

//Notify PCB that the fill object has been changed.
PCBServer.SendMessageToRobots(
        Fill.I_ObjectAddress,
        c_Broadcast,
        PCBM_EndModify ,
        c_NoEventData);

// Clean up robots in PCB
PCBServer.PostProcess;
```

Example 2 - SendMessageToRobots with BoardRegistration call

```
//Initialize robots in PCB
PCBServer.PreProcess;

//Create a text object;
TextObj := PCBServer.PCBObjectFactory(eTextObject, eNoDimension, eCreate_Default);
```

```
// notify the event manager that the pcb object is going to be modified
PCBServer.SendMessageToRobots(TextObj.I_ObjectAddress ,c_Broadcast, PCBM_BeginModify ,
c_NoEventData);

TextObj.XLocation := Sheet.SheetX + MilsToCoord(100);
TextObj.YLocation := Sheet.SheetY + MilsToCoord(100);
TextObj.Layer     := eTopOverlay;
TextObj.Text      := 'Text1';
TextObj.Size      := MilsToCoord(90);   // sets the height of the text.
Board.AddPCBObject(TextObj);

// notify the event manager that the pcb object has been modified
PCBServer.SendMessageToRobots(TextObj.I_ObjectAddress, c_Broadcast, PCBM_EndModify         ,
c_NoEventData);

// notify that the pcb object has been registered in PCB.
PCBServer.SendMessageToRobots(Board.I_ObjectAddress, c_Broadcast, PCBM_BoardRegisteration,
TextObj.I_ObjectAddress);

// Clean up robots in PCB
PCBServer.PostProcess;
```

See also

IPCB_ServerInterface interface

PostProcess method

SendMessageToRobots method

PCB Message Constants

PCBContourFactory method

(IPCB_ServerInterface interface)

Syntax

```
Function  PCBContourFactory (AArcResolution       : TCoord;
                             Const ACX            : TCoord = 0;
                             Const ACY            : TCoord = 0;
                             Const ARotation      : TAngle = 0) : IPCB_Contour;
```

Description

The PCBContourFactory function creates a contour object based on the Arc resolution, the centre X and Y coordinates and the orientation of the contour.

Example

See also

IPCB_Contour interface

DestroyPCBContour method

DestroyPCBContour method

(IPCB_ServerInterface interface)

Syntax

```
Procedure DestroyPCBContour  (Var APCBContour      : IPCB_Contour);
```

Description

This DestroyPCBContour method destroys the object represented by the IPCB_Contour interface which was created by the PCBContourFactory method.

Example

See also

IPCB_Contour interface

PCBContourFactory method

Properties

InteractiveRoutingOptions property

(IPCB_ServerInterface interface)

Syntax

```
Property  InteractiveRoutingOptions  : IPCB_InteractiveRoutingOptions Read
GetState_InteractiveRoutingOptions;
```

Description

This property returns you the **IPCB_InteractiveRoutingOptions** interface which represents the interactive routing options in the PCB editor.

Example

See also

IPCB_ServerInterface interface

IPCB_InteractiveRoutingOptions interface

SystemOptions property

(IPCB_ServerInterface interface)

Syntax

```
Property  SystemOptions : IPCB_SystemOptions Read GetState_SystemOptions;
```

Description

The property returns you the **IPCB_SystemOptions** interface. This interface is represented by the System Options in the PCB editor such as PCB document display options,

Example

See also

IPCB_ServerInterface interface

IPCB_SystemOptions interface

CanFastCrossSelect_Emit property

(IPCB_ServerInterface interface)

Syntax

```
Property  CanFastCrossSelect_Emit    : Boolean                        Read
GetState_CanFastCrossSelect_Emit    Write SetState_CanFastCrossSelect_Emit;
```

Description

Example

See also

IPCB_ServerInterface interface

CanFastCrossSelect_Receive property

(IPCB_ServerInterface interface)

Syntax

```
Property  CanFastCrossSelect_Receive : Boolean                          Read
GetState_CanFastCrossSelect_Receive Write SetState_CanFastCrossSelect_Receive;
```

Description

Example

See also

IPCB_ServerInterface interface

SpecialStringConverter property

(IPCB_ServerInterface interface)

Syntax

```
Property  SpecialStringConverter  : IPCB_SpecialStringConverter    Read
GetState_SpecialStringConverter;
```

Description

This property is a read only property, however if you obtain the IPCB_SpecialStringConverter interface, then you can invoke methods or properties to change the data within.

Example

See also

IPCB_ServerInterface interface

IPCB_SpecialStringConverter interface

TTFLettersCache property

(IPCB_ServerInterface interface)

```
Syntax
Property  TTFLettersCache  : IPCB_LettersCache              Read GetState_TTFLettersCache;
```

Description

Example

See also

IPCB_ServerInterface interface

TTFontsCache property

(IPCB_ServerInterface interface)

```
Syntax
Property  TTFontsCache : IPCB_TTFontsCache Read GetState_TTFontsCache;
```

Description

Example

See also

```
IPCB_ServerInterface interface
```

IPCB_Board

(RT_PCB unit)

Overview

The **IPCB_Board** interface encapsulates an opened PCB document in Altium Designer and from this board interface object, you can add, delete PCB design objects, find out which layers are used and so on.

The **IPCB_Board** interface has iterative methods and interactive feedback methods. Basically you can retrieve an object interface for the PCB design object on the PCB that was clicked on. You can also retrieve the coordinates based on the mouse click on the PCB and also you can conduct defined searches on a PCB document with the parameters you have set up for the iterator. Refer to the Iterators section for more details.

This **IPCB_Board** is also used in the **IPCB_Library** interface. A library document is a bit more complex because it has a list of footprints (components with unnamed designators) and each footprint is shown in a PCB Library document. There is a three way relationship: the **IPCB_Board**, the **IPCB_LibComponent** and the **IPCB_Library** interfaces that all work together for the PCB library document.

Notes

Check if the PCB server exists and if there is a PCB document before you invoke any PCB interface methods. For example

```
PCBBoard := PCBServer.GetCurrentPCBBoard;
If PCBBoard = Nil Then Exit;
```

Some properties are only read only, meaning you can only retrieve data from property but not modify the data.

To create a new object and add to the board object, firstly invoke the **PCBObjectFactory** from the **IPCB_ServerInterface** interface and then invoke the **AddPCBObject** method from a **IPCB_Board** interface.

To look for objects on a PCB document, use one of the following iterators; Board Iterator, Group Iterator, Spatial iterator or a library iterator for PCB Library documents.

Interactive feedback from the board can be done with the following methods: **GetObjectAtCursor**, **GetObjectAtXYAskUserIfAmbiguous**, **ChooseRectangleByCorners** and **ChooseLocation** functions.

IPCB_Board methods

```
WindowBoundingRectangle
LayerPositionInSet

BoardIterator_Create
BoardIterator_Destroy
SpatialIterator_Create
SpatialIterator_Destroy

AddPCBObject
RemovePCBObject

AddObjectToHighlightObjectList
GetPrimitiveCount
ConnectivelyValidateNets
ViewManager_Graphically
InvalidatePrimitive
GetPcbComponentByRefDes
Navigate_RedrawChangedObjectsInBoard
SetState_DocumentHasChanged
SetState_Navigate_HighlightObjectList
SetState_SaveCurrentStatusOfObjectsInBoard
SetState_ViewManager_FilterChanging
```

IPCB_Board properties

```
PCBWindow
FileName
XOrigin
YOrigin
XCursor
YCursor
DisplayUnit
CurrentLayer
LayerStack
LayerColor
SnapGridUnit
BigVisibleGridUnit
VisibleGridUnit
BigVisibleGridSize
VisibleGridSize
SnapGridSize
SnapGridSizeX
SnapGridSizeY
TrackGridSize
ViaGridSize
ComponentGridSize
ComponentGridSizeX
ComponentGridSizeY
```

```
ShowPCBObject
HidePCBObject
InvertPCBObject

CreateBoardOutline
UpdateBoardOutline
RebuildBoardOutline

GetObjectAtCursor
GetObjectAtXYAskUserIfAmbiguous
ChooseRectangleByCorners
ChooseLocation

ContextMenuObjectCount
GetLastClickedObject

FindDominantRuleForObject
FindDominantRuleForObjectPair

PrimPrimDistance

AnalyzeNet
CleanNet
GetState_SplitPlaneNets

GetPrimitiveCounter

ClearUndoRedo
NewUndo
EndUndo
DoUndo
DoRedo

GraphicalView_ZoomRedraw;
GraphicalView_ZoomOnRect

GetState_SelectecObjectCount
GetState_SelectecObject

RebuildPadCaches

RuleNameUnique
NetNameIsUnique
DifferentialPairNameIsUnique
ClassNameIsUnique
```

```
DrawDotGrid
OutputOptions
ECOOptions
GerberOptions
PrinterOptions
PlacerOptions
LayerIsDisplayed
LayerIsUsed
InternalPlaneNetName
InternalPlane1NetName
InternalPlane2NetName
InternalPlane3NetName
InternalPlane4NetName
DrillLayerPairsCount
LayerPair
MechanicalPairs
BoardOutline
AutomaticSplitPlanes
PCBSheet
SelectecObjectCount
SelectecObject

PrimitiveCounter
```

See also

TLayer enumerated values

IPCB_Library interface

IPCB_LayerStack interface

IPCB_OutputOptions interface

IPCB_ECOOptions interface

IPCB_GerberOptions interface

IPCB_PrinterOptions interface

IPCB_AdvancedPlacerOptions interface

QueryUsedLayers script in `\Examples\Scripts\PCB` folder

SpatialIterator script in `\Examples\Scripts\PCB` folder

Methods

AddObjectToHighlightObjectList method

(IPCB_Board interface)

Syntax

```
Procedure AddObjectToHighlightObjectList(iPrimitive : IPCB_Primitive);
```

Description

Example

See also

IPCB_Board interface

AddPCBObject method

(IPCB_Board interface)

Syntax

```
Procedure AddPCBObject(PCBObject : IPCB_Primitive);
```

Description

The **AddPCBObject** method adds a new Design Object into the PCB document after this object was created by the **PCBObjectFactory** method from the **IPCB_ServerInterface** interface.

To successfully create and register a PCB design object onto a PCB document, you need to employ the IPCB_PCBServer's PreProcess, PostProcess and SendMessageToRobots messages. A ViewManager_FullUpdate

DelphiScript Example

```
Var
    Board    : IPCB_Board;
    BR       : TCoordRect;
    Sheet    : IPCB_Sheet;
    Via      : IPCB_Via;
    PadCache : TPadCache;
Begin
    // Grab the board interface representing the current PCB document in DXP.
    Board := PCBServer.GetCurrentPCBBoard;

    // If the board interface doesnt exist (no PCB document) then exit.
    If Board = Nil Then Exit;

    // Initialize the systems in the PCB Editor.
    PCBServer.PreProcess;

    Sheet := Board.PCBSheet;
```

```
    // Create a Via object with the PCBObjectFactory method
    // and then with the new attributes.

    // Note we convert values in Mils to internal coordinates
    // using the MilsToCoord function. All PCB objects locations and sizes
    // have internal coordinate units where 1 mil = 10000 internal units

    Via           := PCBServer.PCBObjectFactory(eViaObject, eNoDimension, eCreate_Default);

    // obtain the bottom left coordinates of the board outline
    BR := Board.BoardOutline.BoundingRectangle;
    Via.x := BR.Left   + MilsToCoord(500);
    Via.y := BR.Bottom + MilsToCoord(500);

//    Via.x         := Sheet.SheetX + MilsToCoord(500);
//    Via.y         := Sheet.SheetY + MilsToCoord(500);

    Via.Size      := MilsToCoord(50);
    Via.HoleSize  := MilsToCoord(20);


    // Assign Via to the Top layer and bottom layer.
    Via.LowLayer  := eTopLayer;
    Via.HighLayer := eBottomLayer;

    // Set up Cache info for Via
    // which consists mainly solder mask, paste mask and power plane values from design rules
    Padcache                           := Via.GetState_Cache;
    Padcache.ReliefAirGap              := MilsToCoord(11);
    Padcache.PowerPlaneReliefExpansion := MilsToCoord(11);
    Padcache.PowerPlaneClearance       := MilsToCoord(11);
    Padcache.ReliefConductorWidth      := MilsToCoord(11);
    Padcache.SolderMaskExpansion       := MilsToCoord(11);
    Padcache.SolderMaskExpansionValid  := eCacheManual;
    Padcache.PasteMaskExpansion        := MilsToCoord(11);
    Padcache.PasteMaskExpansionValid   := eCacheManual;

    // Assign the new pad cache to the via
    Via.SetState_Cache                 := Padcache;

    // Put the new Via object on the board
    Board.AddPCBObject(Via);

    // Update the Undo System in DXP that a new VIa object has been added to the board
    PCBServer.SendMessageToRobots(Board  .I_ObjectAddress, c_Broadcast,
PCBM_BoardRegisteration, Via.I_ObjectAddress);

    // Finalize the systems in the PCB Editor.
```

```
    PCBServer.PostProcess;

    //Full PCB system update
    Board.ViewManager_FullUpdate;
    // Refresh PCB screen
    Client.SendMessage('PCB:Zoom', 'Action=Redraw' , 255, Client.CurrentView);
End;
```

See also

IPCB_Board interface

AnalyzeNet method

(IPCB_Board interface)

Syntax

```
Procedure AnalyzeNet(Const ANet : IPCB_Net);
```

Description

This procedure analyzes a supplied net object in the form of **IPCB_Net** interface.

Example

See also

IPCB_Board interface

BoardIterator_Create method

(IPCB_Board interface)

Syntax

```
Function  BoardIterator_Create : IPCB_BoardIterator;
```

Description

The **BoardIterator_Create** method creates a board iterator which is used to search for design objects on the PCB document. After the search has been conducted, invoke the **BoardIterator_Destroy** method to destroy the board iterator object.

Example

```
    // Retrieve the iterator
    Iterator        := Board.BoardIterator_Create;
    Iterator.AddFilter_ObjectSet(MkSet(ePadObject));
    Iterator.AddFilter_LayerSet(AllLayers);
    Iterator.AddFilter_Method(eProcessAll);

    // Search and count pads
    Pad := Iterator.FirstPCBObject;
    While (Pad <> Nil) Do
    Begin
        Inc(PadNumber);
        Pad := Iterator.NextPCBObject;
    End;
    Board.BoardIterator_Destroy(Iterator);
```

See also

IPCB_Board interface

BoardIterator_Destroy method

(IPCB_Board interface)

Syntax

```
Procedure BoardIterator_Destroy(Var AIterator : IPCB_BoardIterator);
```

Description

The **BoardIterator_Destroy** method destroys the board iterator object after it has been used to conduct a search on the PCB document for specified board objects.

Example

```
    // retrieve the iterator
    Iterator        := Board.BoardIterator_Create;
    Iterator.AddFilter_ObjectSet(MkSet(ePadObject));
    Iterator.AddFilter_LayerSet(AllLayers);
    Iterator.AddFilter_Method(eProcessAll);
    // Search and count pads
    Pad := Iterator.FirstPCBObject;
    While (Pad <> Nil) Do
    Begin
        Inc(PadNumber);
        Pad := Iterator.NextPCBObject;
    End;
    Board.BoardIterator_Destroy(Iterator);
```

See also

IPCB_Board interface

BoardIterator_Create method.

ChooseLocation method

(IPCB_Board interface)

Syntax

```
Function  ChooseLocation(Var X1, Y1 : TCoord;
                         Prompt : TPCBString): Boolean;
```

Description

The function returns you the X1 and Y1 coordinates of the PCB Document after you have clicked on a location on the PCB document. When this function is excuted, you are prompted with a cross hair cursor (being in the interactive mode) and the status bar of the Altium Designer appears with the Prompt string.

This function returns a boolean value whether a location has been retrieved or not. if you click Escape key for example, the function does not return the location values and returns a False value.

DelphiScript Example

```
    Try
        Board := PCBServer.GetCurrentPCBBoard;
        If Not Assigned(Board) Then
        Begin
            ShowMessage('The Current Document is not a Protel PCB Document.');
            Exit;
        End;

        Repeat
            Board.ChooseLocation(x,y, 'Choose Component1');
            Comp1 :=
Board.GetObjectAtXYAskUserIfAmbiguous(x,y,MkSet(eComponentObject),AllLayers,
eEditAction_Select);
            If Not Assigned(Comp1) Then Exit;
```

```
            Board.ChooseLocation(x,y, 'Choose Component2');
            Comp2 :=
Board.GetObjectAtXYAskUserIfAmbiguous(x,y,MkSet(eComponentObject),AllLayers,
eEditAction_Select);
            If Not Assigned(Comp2) Then Exit;

            // do what yo want with Comp1 and Comp2
           // click on the board to exit or RMB
        Until (Comp1 = Nil) Or (Comp2 = Nil);

    Finally
        Pcbserver.PostProcess;
        Client.SendMessage('PCB:Zoom', 'Action=Redraw', 255, Client.CurrentView);
    End;
End
```

See also

IPCB_Board interface

ChooseRectangleByCorners method

(IPCB_Board interface)

Syntax

```
Function  ChooseRectangleByCorners(Prompt1    : TPCBString;
                                   Prompt2    : TPCBString;
                                   Var X1, Y1,
                                       X2, Y2 : TCoord) : Boolean;
```

Description

The **ChooseRectangleByCorners** method prompts you twice to choose the two sets of coordinates that define a boundary rectangle on the PCB document. When this method is executed, the PCB is in interactive mode with a cross hair cursor, waiting for the user to click on the PCB document.

The method returns you the X1,Y1, X2, Y2 values that can be used for calculations or for the spatial iterator for example and a True value.

However if the method was exit prematurely for example the user clicks Escape key or the right mouse button, the method returns a false value.

DelphiScript Example

```
Board := PCBServer.GetCurrentPCBBoard;
If Board = Nil Then Exit;

If Not (Board.ChooseRectangleByCorners( 'Choose first corner',
                                        'Choose final corner',
                                         x1,y1,x2,y2)) Then Exit;

// The coordinates from the ChooseRectangleByCorners method
// can be used for a spatial iterator for example
```

See also

IPCB_Board interface

IPCB_SpatialIterator

ChooseLocation method

CleanNet method

(IPCB_Board interface)

Syntax

```
Procedure CleanNet(Const ANet : IPCB_Net);
```

Description

The **CleanNet** procedure cleans up the net represented by the **IPCB_Net** parameter. It cleans up by re-organizing and re-arranging the net topology of this net.

Example

See also

IPCB_Board interface

ClearUndoRedo method

(IPCB_Board interface)

Syntax

```
Procedure ClearUndoRedo;
```

Description

This clears out the UndoRedo facility in the PCB editor.

Example

See also

IPCB_Board interface

ConnectivelyValidateNets method

(IPCB_Board interface)

Syntax

```
Procedure ConnectivelyValidateNets;
```

Description

This procedure validates the connectivity of nets on the PCB document.

Example

See also

IPCB_Board interface

CreateBoardOutline method

(IPCB_Board interface)

Syntax

```
Function  CreateBoardOutline : IPCB_BoardOutline;
```

Description

The function creates a board outline represented by the **IPCB_BoardOutline** interface. To adjust the parameters of the Board outline, please consult the **IPCB_BoardOutline** interface entry.

Example

See also

IPCB_Board interface

IPCB_BoardOutline interface

DoRedo method

(IPCB_Board interface)

Syntax

```
Procedure DoRedo;
```

Description

This procedure invokes the Redo facility in the PCB editor.

Example

See also

IPCB_Board interface

DoUndo method

(IPCB_Board interface)

Syntax

```
Procedure DoUndo;
```

Description

This procedure invokes the Undo facility in the PCB editor.

Example

See also

IPCB_Board interface

EnableAllPrimitives method

(IPCB_Board interface)

Syntax

```
Procedure EnableAllPrimitives(enable : Boolean);
```

Description

This procedure enables all primitives on the PCB document.

Example

See also

IPCB_Board interface

EndUndo method

(IPCB_Board interface)

Syntax

```
Procedure EndUndo;
```

Description

This procedure ends the Undo process in the PCB editor.

Example

See also

IPCB_Board interface

FindDominantRuleForObject method

(IPCB_Board interface)

Syntax

```
Function  FindDominantRuleForObject(APrimitive : IPCB_Primitive;
                                    ARuleKind  : TRuleKInd) : IPCB_Rule;
```

Description

This function returns the dominant specified rule for the primitive which is targetted by this rule.

Example

See also

IPCB_Board interface

FindDominantRuleForObjectPair method

(IPCB_Board interface)

Syntax

```
Function  FindDominantRuleForObjectPair(APrimitive1,
                                        APrimitive2 : IPCB_Primitive;
                                        ARuleKind   : TRuleKInd) : IPCB_Rule;
```

Description

This function returns the dominant specified binary rule for the two primitives which are targetted by this rule.

Example

See also

IPCB_Board interface

GetObjectAtXYAskUserIfAmbiguous method

(IPCB_Board interface)

Syntax

```
Function GetObjectAtXYAskUserIfAmbiguous(HitX,
                                         HitY      : TCoord;
                                         ObjectSet : TObjectSet;
                                         LayerSet  : TLayerSet;
                                         Action    : TEditingAction) : IPCB_Primitive;
```

Description

This function returns you the specified object with the specified X and Y coordinates which could be retrieved by the **ChooseLocation** method for example.

This function is useful when there are overlapping objects on the PCB document and you need to retrieve the specific object type.

The function returns the design object with the following parameters.

Parameters

The HitX parameter specifies the X coordinate value.

The HitY parameter specifies the Y coordinate value.

The ObjectSet parameter specifies which object types can be returned.

The LayerSet parameter specifies the objects on which layers that can be returned.

The Action parameter specifies what is happening when this method is invoked.

DelphiScript Example

```
Var
    Board     : IPCB_Board;
    Comp1     : IPCB_Component;
    Comp2     : IPCB_Component;

    x,y,      : TCoord;
    x1, y1    : TCoord;
    Rotation  : TAngle;
Begin
    Pcbserver.PreProcess;
```

```
    Try
        Board := PCBServer.GetCurrentPCBBoard;
        If Not Assigned(Board) Then
        Begin
            ShowMessage('The Current Document is not a Protel PCB Document.');
            Exit;
        End;

        Repeat
            Board.ChooseLocation(x,y, 'Choose Component1');
            Comp1 :=
Board.GetObjectAtXYAskUserIfAmbiguous(x,y,MkSet(eComponentObject),AllLayers,
eEditAction_Select);
            If Not Assigned(Comp1) Then Exit;


        // click on the board to exit or RMB
        Until (Comp1 = Nil);

    Finally
        Pcbserver.PostProcess;
        Client.SendMessage('PCB:Zoom', 'Action=Redraw', 255, Client.CurrentView);
    End;
End;
```

See also

IPCB_Board interface

ChooseLocation method

TObjectSet type

TLayerSet type

TEditingAction type

GetObjectAtCursor method

(IPCB_Board interface)

Syntax

```
Function  GetObjectAtCursor(ObjectSet     : TObjectSet;
                            LayerSet      : TLayerSet;
                            StatusBarText : TPCBString) : IPCB_Primitive;
```

Description

This function returns the design object that is within the mouse's clicked coordinates on the PCB document.

Parameters

The ObjectSet parameter specifies which object types can be returned.

The LayerSet parameter specifies the objects on which layers that can be returned.

The StatusbarText parameter specifies the text on the status bar of the Altium Designer application when the function is invoked.

Example

See also

IPCB_Board interface

GetPcbComponentByRefDes method

(IPCB_Board interface)

Syntax

```
Function  GetPcbComponentByRefDes(Value : TString) : IPCB_Component;
```

Description

This function returns the component by its valid reference designator.

Example

See also

IPCB_Board interface

GetPrimitiveCount method

(IPCB_Board interface)

Syntax

```
Function  GetPrimitiveCount(AObjSet  : TObjectSet;
                            LayerSet : TLayerSet;
                            AMethod  : TIterationMethod)  :Integer;
```

Description

The function returns the number of primitives which is dependent on the parameters supplied - the object kinds to look for, which layers to look for and how the search is conducted.

Parameters

The ObjectSet parameter specifies which object types can be returned.

The LayerSet parameter specifies the objects on which layers that can be returned.

The AMethod parameter specifies how the search is conducted.

Example

See also

IPCB_Board interface

TObjectSet type

TLayerSet type

TIterationMethod type

GetPrimitiveCounter method

(IPCB_Board interface)

Syntax

```
Function GetPrimitiveCounter : IPCB_PrimitiveCounter;
```

Description

The **IPCB_PrimitiveCounter** interface gives you the means of obtaining the hole count and string count for the focussed PCB document.

Example

See also

IPCB_Board interface

PrimitiveCounter property

IPCB_PrimitiveCounter interface

GetState_SplitPlaneNets method

(IPCB_Board interface)

Syntax

```
Procedure GetState_SplitPlaneNets(NetsList  : TStringList);
```

Description

This procedure retrieves the list of nets for split planes on the PCB document in a TStringList container.

Example

See also

IPCB_Board interface

HidePCBObject method

(IPCB_Board interface)

Syntax

```
Procedure HidePCBObject(Const PCBObject : IPCB_Primitive);
```

Description

This method hides the specified object on the PCB document from view.

Example

See also

IPCB_Board interface

InvertPCBObject method

ShowPCBObject method

InvertPCBObject method

(IPCB_Board interface)

Syntax

```
Procedure InvertPCBObject(Const PCBObject : IPCB_Primitive);
```

Description

This method inverts the colors of the specified object on the PCB document.

Example

See also

IPCB_Board interface

ShowPCBObject method

HidePCBObject method

LayerPositionInSet method

(IPCB_Board interface)

Syntax

```
Function  LayerPositionInSet(ALayerSet : TLayerSet;
                             ALayerObj : IPCB_LayerObject)  : Integer;
```

Description

This function returns a positive value with 1 being the first layer and a higher number being the lower layer in the list. This function is useful for checking low and high layers of a layer pair.

Example

```
Begin
    PCBBoard := PCBServer.GetCurrentPCBBoard;
    If PCBBoard = Nil Then Exit;
```

```
    LayerPairs := TStringList.Create;
    For i := 0 To PCBBoard.DrillLayerPairsCount - 1 Do
    Begin
        PCBLayerPair := PCBBoard.LayerPair[i];
        LowLayerObj  := PCBBoard.LayerStack.LayerObject[PCBLayerPair.LowLayer];
        HighLayerObj := PCBBoard.LayerStack.LayerObject[PCBLayerPair.HighLayer];
        LowPos       := PCBBoard.LayerPositionInSet(SignalLayers + InternalPlanes,
                                                    LowLayerObj);
        HighPos      := PCBBoard.LayerPositionInSet(SignalLayers + InternalPlanes,
                                                    HighLayerObj);
        If LowPos <= HighPos Then
            LayerPairs.Add(LowLayerObj .Name + ' - ' + HighLayerObj.Name)
        Else
            LayerPairs.Add(HighLayerObj.Name + ' - ' + LowLayerObj .Name);
    End;

    // Format the layer pairs data string and display it.
    LS := '';
    For i := 0 to LayerPairs.Count - 1 Do
        LS := LS + LayerPairs[i] + #13#10;
    ShowInfo('Layer Pairs:'#13#10 + LS);
    LayerPairs.Free;
End;
```

See also

IPCB_Board interface

IPCB_LayerObject interface

IPCB_DrillLayerPair interface

Navigate_RedrawChangedObjectsInBoard method

(IPCB_Board interface)

Syntax

```
Procedure Navigate_RedrawChangedObjectsInBoard;
```

Description

Example

See also

IPCB_Board interface

NewUndo method

(IPCB_Board interface)

Syntax

```
Procedure NewUndo;
```

Description

This procedure creates a new undo process in the PCB editor.

Example

See also

IPCB_Board interface

RemovePCBObject method

(IPCB_Board interface)

Syntax

```
Procedure RemovePCBObject(PCBObject : IPCB_Primitive);
```

Description

This method removes the PCB object from the PCB board but it is not completely destroyed, which means it can be undone. When deleting PCB objects, basically you just collect the track object interfaces and put them in a **TInterfaceList** or **TList** Borland Delphi Container objects (which is exposed in the scripting system, but with the Server Development Kit, you need to add the Classes unit in your server project). Then with this **TInterfaceList** or **TList** object, you go thru the items one at a time, and for each item fetched, call the **RemovePCBObject** method from the **IPCB_Board** interface and call the **SendMessageToRobots** to remember this deleted track in the Undo system.

It is generally not a good idea to delete objects while iterating for objects within a While or Repeat loop body because it messes up the data structure that the iterator is traversing.

Bad DelphiScript Example

```
While Track <> Nil Do
Begin
    OldTrack := Track;
    Track := Iterator.NextPCBObject;
    CurrentPCBBoard.RemovePCBObject(OldTrack);
    PCBServer.SendMessageToRobots(CurrentPCBBoard.I_ObjectAddress,
                                  c_BroadCast,
                                  PCBM_BoardRegisteration,
                                  OldTrack.I_ObjectAddress);
End;
```

Correct DelphiScript Example

```
Procedure RemoveTracksOnTopLayer;
var
    CurrentPCBBoard : IPCB_Board;
    Iterator        : IPCB_BoardIterator;
    Track           : IPCB_Track;
    TrackList       : TInterfaceList;
    I               : Integer;
Begin
    CurrentPCBBoard := PCBServer.GetCurrentPCBBoard;
    If CurrentPCBBoard = Nil Then Exit;

    Iterator := CurrentPCBBoard.BoardIterator_Create;
    If Iterator = Nil Then Exit;
    Iterator.AddFilter_ObjectSet(MkSet(eTrackObject));
    Iterator.AddFilter_LayerSet(MkSet(eTopLayer));

    // store tracks in a TInterfacelist that are to be deleted later...
    TrackList := TInterfaceList.Create;

    Try
        Track := Iterator.FirstPCBObject;
```

```
        While Track <> Nil Do
        Begin
            TrackList.Add(Track);
            Track := Iterator.NextPCBObject;
        End;
    Finally
        CurrentPCBBoard.BoardIterator_Destroy(Iterator);
    End;

    Try
        PCBServer.PreProcess;
        For I := 0 to TrackList.Count - 1 Do
        Begin
            Track := TrackList.items[i];
            CurrentPCBBoard.RemovePCBObject(Track);

            PCBServer.SendMessageToRobots(CurrentPCBBoard.I_ObjectAddress,
                                          c_BroadCast,
                                          PCBM_BoardRegisteration,
                                          Track.I_ObjectAddress);
        End;
    Finally
        PCBServer.PostProcess;
        TrackList.Free;
    End;


    // Refresh the PCB document.
    CurrentPCBBoard.ViewManager_FullUpdate;
    Client.SendMessage('PCB:Zoom', 'Action=Redraw' , 255, Client.CurrentView);
End;
```

See also

IPCB_Board interface

ShowPCBObject method

(IPCB_Board interface)

Syntax

```
Procedure ShowPCBObject(Const PCBObject : IPCB_Primitive);
```

Description

This procedure makes this hidden PCB object visible on the PCB document.

Example

See also

IPCB_Board interface

InvertPCBObject method

HidePCBObject method

SetState_DocumentHasChanged method

(IPCB_Board interface)

Syntax

```
Procedure SetState_DocumentHasChanged;
```

Description

This procedure forces the document has changed flag to true denoting that the document has been marked dirty so that when you close this document, you are prompted to save this document.

Example

See also

IPCB_Board interface

SetState_Navigate_HighlightObjectList method

(IPCB_Board interface)

Syntax

```
Procedure SetState_Navigate_HighlightObjectList(
              HighlightMethods : THighlightMethodSet;
              ClearExisting    : Boolean);
```

Description

Example

See also

IPCB_Board interface

SetState_SaveCurrentStatusOfObjectsInBoard method

(IPCB_Board interface)

Syntax

```
Procedure SetState_SaveCurrentStatusOfObjectsInBoard;
```

Description

Example

See also

IPCB_Board interface

SetState_ViewManager_FilterChanging method

(IPCB_Board interface)

Syntax

```
Procedure SetState_ViewManager_FilterChanging;
```

Description

Example

See also

IPCB_Board interface

SpatialIterator_Create method

(IPCB_Board interface)

Syntax

```
Function  SpatialIterator_Create : IPCB_SpatialIterator;
```

Description

This method creates a spatial iterator which conducts a search within defined boundary on a PCB document. A spatial iterator only looks for primitive objects on a PCB document such as tracks and arcs not group objects such as dimensions and components.

Example

```
Iterator := Board.SpatialIterator_Create;

(* Top/Bottom Layers and Arc/Track objects defined
   for the Spatial iterator constraints *)
ASetOfLayers  := MkSet(eTopLayer,eBottomLayer);
ASetOfObjects := MkSet(eArcObject,eTrackObject);

Iterator.AddFilter_ObjectSet(ASetOfObjects);
Iterator.AddFilter_LayerSet(ASetOfLayers);
Iterator.AddFilter_Area(X1,Y1,X2,Y2);

(* Iterate for tracks and arcs on bottom/top layers *)
PCBObject := Iterator.FirstPCBObject;
While PCBObject <> 0 Do
Begin
     PCBObject.Selected := True;
     PCBObject := Iterator.NextPCBObject;
End;
Board.SpatialIterator_Destroy(Iterator);
```

See also

IPCB_Board interface

SpatialIterator_Destroy method

SpatialIterator_Destroy method

(IPCB_Board interface)

Syntax

```
Procedure SpatialIterator_Destroy(Var AIterator : IPCB_SpatialIterator);
```

Description

This method destroys the spatial iterator object after it has finished conducting a search within a defined boundary on the PCB document. A spatial iterator only looks for primitive objects on a PCB document such as tracks and arcs not group objects such as dimensions and components.

Example

```
Iterator := Board.SpatialIterator_Create;

(* Top/Bottom Layers and Arc/Track objects defined
   for the Spatial iterator constraints *)
ASetOfLayers  := MkSet(eTopLayer,eBottomLayer);
ASetOfObjects := MkSet(eArcObject,eTrackObject);

Iterator.AddFilter_ObjectSet(ASetOfObjects);
Iterator.AddFilter_LayerSet(ASetOfLayers);
Iterator.AddFilter_Area(X1,Y1,X2,Y2);
```

```
    (* Iterate for tracks and arcs on bottom/top layers *)
    PCBObject := Iterator.FirstPCBObject;
    While PCBObject <> 0 Do
    Begin
         PCBObject.Selected := True;
         PCBObject := Iterator.NextPCBObject;
    End;
    Board.SpatialIterator_Destroy(Iterator);
```

See also

IPCB_Board interface

SpatialIterator_Create method

UpdateBoardOutline method

(IPCB_Board interface)

Syntax

```
Procedure UpdateBoardOutline;
```

Description

This method refreshes the Board outline on the PCB document for example if you have programmatically altered the outline, it is a good time to invoke the UpdateBoardOutline method to refresh the PCB document.

Example

See also

IPCB_Board interface

ViewManager_GraphicallyInvalidatePrimitive method

(IPCB_Board interface)

Syntax

```
Procedure ViewManager_GraphicallyInvalidatePrimitive(PCBObject : IPCB_Primitive);
```

Description

This procedure forces a repaint of the targeted design object (PCBObject parameter) on the PCB document.

Example

See also

IPCB_Board interface

ViewManager_FullUpdate method

(IPCB_Board interface)

Syntax

```
Procedure ViewManager_FullUpdate;
```

Description

This method invokes a full update of all panels that are associated with the current PCB document. This method is useful if a document has been modified programmatically especially with library documents.

Example

```
Var
    CurrentLib    : IPCB_Library;
    NewPCBLibComp : IPCB_LibComponent;
    NewPad        : IPCB_Pad;
Begin
```

```
    If PCBServer = Nil Then Exit;
    CurrentLib := PcbServer.GetCurrentPCBLibrary;
    If CurrentLib = Nil Then Exit;
    NewPCBLibComp := PCBServer.CreatePCBLibComp;
    NewPcbLibComp.Name := 'ANewComponent';
    CurrentLib.RegisterComponent(NewPCBLibComp);
    CurrentLib.CurrentComponent := NewPcbLibComp;
    PCBServer.PreProcess;
    NewPad := PcbServer.PCBObjectFactory(ePadObject,eNoDimension,eCreate_Default);
    NewPad.X        := MilsToCoord(0);
    NewPad.Y        := MilsToCoord(0);
    NewPad.TopXSize := MilsToCoord(62);
    NewPad.TopYSize := MilsToCoord(62);
    NewPad.HoleSize := MilsToCoord(28);
    NewPad.Layer    := eMultiLayer;
    NewPad.Name     := '1';
    NewPCBLibComp.AddPCBObject(NewPad);

PCBServer.SendMessageToRobots(NewPCBLibComp.I_ObjectAddress,c_Broadcast,PCBM_BoardRegisteratio
n,NewPad.I_ObjectAddress);

PCBServer.SendMessageToRobots(Nil,c_Broadcast,PCBM_BoardRegisteration,NewPCBLibComp.I_ObjectAd
dress);
    PCBServer.PostProcess;
    CurrentLib.Board.ViewManager_FullUpdate;
    RefreshPCB;
End;
```

See also

IPCB_Board interface

WindowBoundingRectangle method

(IPCB_Board interface)

Syntax

```
Function  WindowBoundingRectangle : TCoordRect;
```

Description

This function returns the coordinates of the bounds of a PCB window.

Example

See also

IPCB_Board interface

Properties

AutomaticSplitPlanes property

(IPCB_Board interface)

Syntax

```
Property  AutomaticSplitPlanes : Boolean Read GetState_AutomaticSplitPlanes  Write
SetState_AutomaticSplitPlanes;
```

Description

The AutomaticSplitPlanes property returns you the boolean value whether the split planes are system generated automatically or not. This property is implemented by its GetState_AutomaticSplitPlanes and SetState_AutomaticSplitPlanes methods.

Example

See also

IPCB_Board interface

BigVisibleGridSize property

(IPCB_Board interface)

Syntax

```
BigVisibleGridSize : TReal Read GetState_BigVisibleGridSize    Write
SetState_BigVisibleGridSize;
```

Description

This property retrieves or sets the Big Visible Grid Size in **TReal** type. This Grid Size is used for reference purposes and there are two visible grids.

Example

See also

IPCB_Board interface

VisibleGridSize property

BigVisibleGridUnit property

(IPCB_Board interface)

Syntax

```
Property  BigVisibleGridUnit : TUnit Read GetState_BigVisibleGridUnit    Write
SetState_BigVisibleGridUnit;
```

Description

This property retrieves or sets the big visible grid's measurement units in Imperial or Metric units. There are two visible grids to use for reference purposes.

Example

See also

IPCB_Board interface

VisibleGridUnit property

TUnit type

BoardOutline property

(IPCB_Board interface)

Syntax

```
Property  BoardOutline : IPCB_BoardOutline Read GetState_BoardOutline;
```

Description

The Board Outline represents the board outline which encompasses a board design on a PCB document. The board outline is represented by the **IPCB_BoardOutline** interface and inherited from the **IPCB_Polygon** interface because the Board Outline is composed of vertices (tracks and arcs only).

Example

```
Var
    PCB_Board : IPCB_Board;
    BR        : TCoordRect;
Begin
```

```
    PCB_Board := PCBServer.GetCurrentPCBBoard;
    If PCB_Board = Nil Then Exit;
    If PCB_Board.IsLibrary Then Exit;

    PCB_Board.BoardOutline.Invalidate;
    PCB_Board.BoardOutline.Rebuild;
    PCB_Board.BoardOutline.Validate;
    BR := PCB_Board.BoardOutline.BoundingRectangle;

    // refresh board outline
    PCB_Board.UdateBoardOutline;
End;
```

See also

IPCB_Board interface

IPCB_BoardOutline interface

ComponentGridSize property

(IPCB_Board interface)

Syntax

```
Property  ComponentGridSize : TDouble Read GetState_ComponentGridSize     Write
SetState_ComponentGridSize;
```

Description

This property represents the component grid size for components to be accurately placed on. This component grid size sets the X and Y values simultaneously. If you wish to define different X and Y grid sizes, then use the ComponentGridSizeX and ComponentGridSizeY properties.

Example

See also

IPCB_Board interface

ComponentGridSizeX property

ComponentGridSizeY property

TDouble type

ComponentGridSizeX

(IPCB_Board interface)

Syntax

```
Property  ComponentGridSizeX : TDouble Read GetState_ComponentGridSizeX Write
SetState_ComponentGridSizeX;
```

Description

This property represents the component grid size for components to be accurately placed on. To define different X and Y grid sizes, use the **ComponentGridSizeX** and **ComponentGridSizeY** properties, otherwise to set the same values for the component grid sizes X and Y simultaneously.

Example

See also

IPCB_Board interface

ComponentGridSize

ComponentGridSizeY

ComponentGridSizeY property

(IPCB_Board interface)

Syntax

```
Property  ComponentGridSizeY : TDouble Read GetState_ComponentGridSizeY Write
SetState_ComponentGridSizeY;
```

Description

This property represents the component grid size for components to be accurately placed on. To define different X and Y grid sizes, use the **ComponentGridSizeX** and **ComponentGridSizeY** properties, otherwise to set the same values for the component grid sizes X and Y simultaneously.

Example

See also

IPCB_Board interface

CurrentLayer property

(IPCB_Board interface)

Syntax

```
Property  CurrentLayer : TLayer Read GetState_CurrentLayer;
```

Description

This property denotes the current PCB layer being displayed in the PCB workspace in Altium Designer.

Example

See also

IPCB_Board interface

DisplayUnit property

(IPCB_Board interface)

Syntax

```
Property DisplayUnit : TUnit Read GetState_DisplayUnit Write SetState_DisplayUnit;
```

Description

This property retrieves or sets the measurement units for the PCB document display purposes in Imperial or Metric units.

Example

```
Var
    Board : IPCB_Board;
Begin
    Board := PCBServer.GetCurrentPCBBoard;
    If Board = Nil Then Exit;
    ShowMessage('Board Filename =' + Board.FileName + #13 +
                'Board Units = '   + UnitToString(Board.DisplayUnit));
End;
```

See also

IPCB_Board interface

UnitToString function

DrawDotGrid property

(IPCB_Board interface)

Syntax

```
Property  DrawDotGrid : Boolean Read GetState_DrawDotGrid Write SetState_DrawDotGrid;
```

Description

This property denotes whether the grid has dotted or continuous lines.

Example

See also

IPCB_Board interface

DrillLayersPairsCount property

(IPCB_Board interface)

Syntax

```
Property  DrillLayerPairsCount : Integer Read GetState_DrillLayerPairsCount;
```

Description

This property returns the number of drill layer pairs for the board. A drill layer pair is represented by the **IPCB_DrillLayerPair** interface.

Example

```
Var
    PCBBoard     : IPCB_Board;
    i            : Integer;
    LayerPairs   : TStringList;
    PCBLayerPair : IPCB_DrillLayerPair;
    LowLayerObj  : IPCB_LayerObject;
    HighLayerObj : IPCB_LayerObject;

    LowPos       : Integer;
    HighPos      : Integer;
    LS           : String;
Begin
    PCBBoard := PCBServer.GetCurrentPCBBoard;
    If PCBBoard = Nil Then Exit;

    For i := 0 To PCBBoard.DrillLayerPairsCount - 1 Do
    Begin
        PCBLayerPair := PCBBoard.LayerPair[i];
        LowLayerObj  := PCBBoard.LayerStack.LayerObject[PCBLayerPair.LowLayer];
        HighLayerObj := PCBBoard.LayerStack.LayerObject[PCBLayerPair.HighLayer];

        // do what you want with the LowLayerObj and HighLayerObj objects
    End;
End;
```

See also

IPCB_Board interface

LayerPair property

IPCB_DrillLayerPair interface

FileName property

(IPCB_Board interface)

Syntax

```
Property  FileName : TPCBString Read GetState_FileName;
```

Description

The FileName property denotes the filename of the PCB document that the **IPCB_Board** interface is associated with. The Filename property is read only, which means you can retrieve the filename string only.

Example

```
Var
    Board : IPCB_Board;
Begin
    Board := PCBServer.GetCurrentPCBBoard;
    If Board = Nil Then Exit;
    ShowMessage('Board Filename =' + Board.FileName + #13 +
                'Board Units = '   + UnitToString(Board.DisplayUnit));
End;
```

See also

IPCB_Board interface

InternalPlane1NetName property

(IPCB_Board interface)

Syntax

```
Property  InternalPlane1NetName : TPCBString Read GetState_InternalPlane1NetName Write
SetState_InternalPlane1NetName;
```

Description

This property represents the Internal Plane 1 Netname (for P99SE and earlier products).

Example

See also

IPCB_Board interface

IPCB_LayerStack interface.

InternalPlane2NetName property

(IPCB_Board interface)

Syntax

```
Property  InternalPlane2NetName             : TPCBString                Read
GetState_InternalPlane2NetName Write SetState_InternalPlane2NetName;
```

Description

This property represents the Internal Plane 2 Netname (for P99SE and earlier products).

Example

See also

IPCB_Board interface

IPCB_LayerStack interface.

InternalPlane3NetName property

(IPCB_Board interface)

Syntax

```
Property  InternalPlane3NetName             : TPCBString                Read
GetState_InternalPlane3NetName Write SetState_InternalPlane3NetName;
```

Description

This property represents the Internal Plane 3 Netname (for P99SE and earlier products).

Example

See also

IPCB_Board interface

IPCB_LayerStack interface.

InternalPlane4NetName

(IPCB_Board interface)

Syntax

```
Property  InternalPlane4NetName : TPCBString Read GetState_InternalPlane4NetName Write
SetState_InternalPlane4NetName;
```

Description

This property represents the Internal Plane 1 Netname (for P99SE and earlier products).

Example

See also

IPCB_Board interface

IPCB_LayerStack interface.

InternalPlaneNetName property

(IPCB_Board interface)

Syntax

```
Property  InternalPlaneNetName [L : TLayer] : TPCBString Read GetState_InternalPlaneNetName
Write SetState_InternalPlaneNetName;
```

Description

This property returns or sets the net name for the targetted internal plane.

Example

See also

IPCB_Board interface

TLayer type

LayerColor property

(IPCB_Board interface)

Syntax

```
Property  LayerColor [L : TLayer] : TColorRef Read GetState_LayerColor;
```

Description

This property returns the layer color of TColorRef type. This type is defined in the Windows.pas which is part of the Borland Delphi Run-Time Library.

Example

See also

IPCB_Board interface

TColorRef type

LayerIsDisplayed property

(IPCB_Board interface)

Syntax

```
Property  LayerIsDisplayed     [L : TLayer] : Boolean Read GetState_LayerIsDisplayed
Write SetState_LayerIsDisplayed;
```

Description

The **LayerIsDisplayed** property controls the display of layers for the PCB document. You can fetch or set the

Example

```
PCBBoard := PCBServer.GetCurrentPCBBoard;
If PCBBoard = Nil Then Exit;

// Check for each signal layer for used/display setting
For Layer := eTopLayer to eMultiLayer Do
    If PCBBoard.LayerIsUsed[Layer] Then
        If PCBBoard.LayerIsDisplayed[Layer] Then
            \\ do something
```

See also

IPCB_Board interface

LayerIsUsed property

(IPCB_Board interface)

Syntax

```
Property  LayerIsUsed [L : TLayer] : Boolean Read GetState_LayerIsUsed Write
SetState_LayerIsUsed;
```

Description

This property retrieves or sets the boolean value for whether the layer is used by primitives or not. Normally when a layer has primitives (design objects) on it, the layer is used.

Example

```
PCBBoard := PCBServer.GetCurrentPCBBoard;
If PCBBoard = Nil Then Exit;

// Check for each signal layer for used/display setting
For Layer := eTopLayer to eMultiLayer Do
    If PCBBoard.LayerIsUsed[Layer] Then
        If PCBBoard.LayerIsDisplayed[Layer] Then
            \\ do something
```

See also

IPCB_Board interface

LayerPair property

(IPCB_Board interface)

Syntax

```
Property  LayerPair [I : Integer] : IPCB_DrillLayerPair        Read GetState_LayerPair;
```

Description

This property returns you the layer pair associated with the IPCB_DrillLayerPair interface. A drill layer pair has two drill layers.

Example

```
Var
    PCBBoard     : IPCB_Board;
    i            : Integer;
    LayerPairs   : TStringList;
    PCBLayerPair : IPCB_DrillLayerPair;
    LowLayerObj  : IPCB_LayerObject;
    HighLayerObj : IPCB_LayerObject;
    LowPos       : Integer;
```

```
    HighPos      : Integer;
    LS           : String;
Begin
    PCBBoard := PCBServer.GetCurrentPCBBoard;
    If PCBBoard = Nil Then Exit;

    // Show the Current Layer for the PCB document.
    ShowInfo('Current Layer: ' + Layer2String(PCBBoard.CurrentLayer));

    LayerPairs := TStringList.Create;
    For i := 0 To PCBBoard.DrillLayerPairsCount - 1 Do
    Begin
        PCBLayerPair := PCBBoard.LayerPair[i];
        LowLayerObj  := PCBBoard.LayerStack.LayerObject[PCBLayerPair.LowLayer];
        HighLayerObj := PCBBoard.LayerStack.LayerObject[PCBLayerPair.HighLayer];

        LowPos       := PCBBoard.LayerPositionInSet(SignalLayers + InternalPlanes,
LowLayerObj);
        HighPos      := PCBBoard.LayerPositionInSet(SignalLayers + InternalPlanes,
HighLayerObj);
        If LowPos <= HighPos Then
            LayerPairs.Add(LowLayerObj .Name + ' - ' + HighLayerObj.Name)
        Else
            LayerPairs.Add(HighLayerObj.Name + ' - ' + LowLayerObj .Name);
    End;

   // Display layer pairs.
    LS := '';
    For i := 0 to LayerPairs.Count - 1 Do
        LS := LS + LayerPairs[i] + #13#10;

    ShowInfo('Layer Pairs:'#13#10 + LS);
    LayerPairs.Free;
End;
```

See also

IPCB_Board interface

LayerStack property

(IPCB_Board interface)

Syntax

```
Property  LayerStack : IPCB_LayerStack Read GetState_LayerStack;
```

Description

The layer stack property fetches the **IPCB_LayerStack** interface for the current PCB document. The Layer stack only stores copper layers (signal and internal planes).

Example

```
Var
    PCBBoard      : IPCB_Board;
    TheLayerStack : IPCB_LayerStack;
    i             : Integer;
```

```
    LayerObj      : IPCB_LayerObject;
    LS            : String;
Begin
    PCBBoard := PCBServer.GetCurrentPCBBoard;
    If PCBBoard = Nil Then Exit;

    // Note that the Layer stack only stores existing copper based layers.
    // But you can use the LayerObject property to fetch all layers.
    TheLayerStack := PCBBoard.LayerStack;
    If TheLayerStack = Nil Then Exit;
    LS       := '';
    LayerObj := TheLayerStack.FirstLayer;
    Repeat
        LS       := LS + Layer2String(LayerObj.LayerID) + #13#10;
        LayerObj := TheLayerStack.NextLayer(LayerObj);
    Until LayerObj = Nil;
    ShowInfo('The Layer Stack has :'#13#10 + LS);
End;
```

See also

IPCB_LayerStack interface

IPCB_LayerObject interface

IPCB_Board interface

MechanicalPairs property

(IPCB_Board interface)

Syntax

```
Property  MechanicalPairs : IPCB_MechanicalLayerPairs  Read GetState_MechanicalPairs;
```

Description

There are 16 general purpose mechanical layers for defining the board layout, placing dimensions on, including fabrication details on, or any other mechanical details the design requires.

The purpose of the **IPCB_MechanicalLayerPairs** Interface is to provide which Mechanical layers are paired to one another.

When a component incorporates objects on one or more Mechanical layers which have been paired, the Layer property of those objects changes when the Layer property of the component is toggled (between the Top and Bottom layers), just like objects on the non-Mechanical layers which have always been paired to one another, along with the Top and Bottom (copper) layers, the Top and Bottom Overlay layers, the Top and Bottom Paste Mask layers, and the Top and Bottom Solder Mask layers.

Example

See also

IPCB_Board interface

IPCB_MechanicalPairs interface

PCBSheet property

(IPCB_Board interface)

Syntax

```
Property PCBSheet : IPCB_Sheet Read GetState_PCBSheet;
```

Description

This property returns the IPCB_Sheet interface which is represented by the sheet workspace. A sheet encapsulates the sheet borders, the fabrication and assembly information, and the board outline.

Example

See also

IPCB_Board interface

IPCB_Sheet interface

PCBWindow property

(IPCB_Board interface)

Syntax

```
Property  PCBWindow : HWND Read GetState_Window;
```

Description

This property returns the raw Windows handle for a window handle of a PCB document in Altium Designer.

Example

See also

IPCB_Board interface

SelectecObjectCount property

(IPCB_Board interface)

Syntax

```
Property  SelectecObjectCount : Integer Read GetState_SelectecObjectCount;
```

Description

This property represents the number of selected objects found on the PCB document. This is to be used in conjunction with the SelectecObject property.

Example

See also

IPCB_Board interface

SelectecObject property

SelectecObject property

(IPCB_Board interface)

Syntax

```
Property  SelectecObject [I : Integer] : IPCB_Primitive Read GetState_SelectecObject;
```

Description

This property represents the indexed selected object found on the PCB document. This is to be used in conjunction with the SelectecObjectCount property.

Example

See also

IPCB_Board interface

SelectecObjectCount property

PrimitiveCounter method

(IPCB_Board interface)

Syntax

```
Property  PrimitiveCounter : IPCB_PrimitiveCounter Read GetPrimitiveCounter;
```

Description

The IPCB_PrimitiveCounter interface gives you the means of obtaining the object count, hole count and string count for the focussed PCB document via the IPCB_Board's PrimitiveCounter property.

Example

See also

IPCB_Board interface

GetPrimitiveCounter method

IPCB_PrimitiveCounter interface

SnapGridSizeX

(IPCB_Board interface)

Syntax

```
Property  SnapGridSizeX : TDouble Read GetState_SnapGridSizeX Write SetState_SnapGridSizeX;
```

Description

This property retrieves or sets the Snap Grid size X value. To set both X and Y values simultaneously for the Snap Grid, use the **SnapGridSize** property.

Example

See also

IPCB_Board interface

SnapGridSizeY property

SnapGridSize property

SnapGridSizeY property

(IPCB_Board interface)

Syntax

```
Property  SnapGridSizeY : TDouble Read GetState_SnapGridSizeY Write SetState_SnapGridSizeY;
```

Description

This property retrieves or sets the Snap Grid size Y value. To set both X and Y values simultaneously for the Snap Grid, use the **SnapGridSize** property.

Example

See also

IPCB_Board interface

SnapGridSizeX property

SnapGridSize property

SnapGridSize property

(IPCB_Board interface)

Syntax

```
Property  SnapGridSize                    : TDouble                    Read
GetState_SnapGridSize          Write SetState_SnapGridSize;
```

Description

The SnapGridSize property sets the X and Y values for the Snap Grid simultaneously. If you want to have different X and Y values for this snap grid, use the SnapGridSizeX and SnapGridSizeY properties.

Example

See also

IPCB_Board interface

SnapGridSizeX property

SnapGridSizeY property

SnapGridUnit property

(IPCB_Board interface)

Syntax

```
Property  SnapGridUnit : TUnit Read GetState_SnapGridUnit Write SetState_SnapGridUnit;
```

Description

The SnapGridUnit property retrieves or sets the measurement unit for the Snap Grid Unit. It can be in Imperial or Metric units.

Example

See also

IPCB_Board interface

TUnit type

TrackGridSize property

(IPCB_Board interface)

Syntax

```
Property  TrackGridSize : TDouble  Read GetState_TrackGridSize Write SetState_TrackGridSize;
```

Description

This property retrieves or sets the track grid size in both X and Y directions simultaneously.

Example

See also

IPCB_Board interface

ViaGridSize property

ViaGridSize property

(IPCB_Board interface)

Syntax

```
Property ViaGridSize : TDouble Read GetState_ViaGridSize  Write SetState_ViaGridSize;
```

Description

This property retrieves or sets the via grid size in both X and Y directions simultaneously.

Example

See also

IPCB_Board interface

TrackGridSize property

VisibleGridSize property

(IPCB_Board interface)

Syntax

```
Property  VisibleGridSize : TReal Read GetState_VisibleGridSize Write
SetState_VisibleGridSize;
```

Description

This property retrieves or sets the Visible Grid Size in TReal type. This Grid Size is used for reference purposes and there are two visible grids.

Example

See also

IPCB_Board interface

BigVisibleGridSize property

VisibleGridUnit property

(IPCB_Board interface)

Syntax

```
Property  VisibleGridUnit : TUnit Read GetState_VisibleGridUnit Write
SetState_VisibleGridUnit;
```

Description

This property retrieves or sets the big visible grid's measurement units in Imperial or Metric units. There are two visible grids to use for reference purposes.

Example

See also

IPCB_Board interface

BigVisibleGridUnit interface

TUnit type

XOrigin property

(IPCB_Board interface)

Syntax

```
Property  XOrigin : TCoord  Read GetState_XOrigin Write SetState_XOrigin;
```

Description

This property sets or retrieves the X coordinate of the absolute origin of the board.

Example

See also

IPCB_Board interface

XCursor property

(IPCB_Board interface)

Syntax

```
Property  XCursor : TCoord Read GetState_XCursor  Write SetState_XCursor;
```

Description

This property retrieves or sets the x coordinate of the cursor of the latest mouse click on the PCB document.

Example

See also

IPCB_Board interface

YCursor property

(IPCB_Board interface)

Syntax

```
Property  YCursor : TCoord Read GetState_YCursor Write SetState_YCursor;
```

Description

This property retrieves or sets the Y coordinate of the cursor of the latest mouse click on the PCB document.

Example

See also

IPCB_Board interface

YOrigin property

(IPCB_Board interface)

Syntax

```
Property  YOrigin : TCoord Read GetState_YOrigin Write SetState_YOrigin;
```

Description

This property sets or retrieves the Y coordinate of the absolute origin of the board.

Example

See also

IPCB_Board interface

ECOOptions property

(IPCB_Board interface)

Syntax

```
Property  ECOOptions : IPCB_ECOOptions Read GetState_ECOOptions;
```

Description

This property returns you the IPCB_ECOOptions interface which represents the Options for the Engineering Order Change facility in the PCB editor.

Example

See also

IPCB_Board interface

IPCB_ECOOptions interface

GerberOptions property

(IPCB_Board interface)

Syntax

```
Property  GerberOptions : IPCB_GerberOptions Read GetState_GerberOptions;
```

Description

This property returns you the IPCB_GerberOptions interface which represents the Options for the Gerbers facility in the PCB editor.

Example

See also

IPCB_Board interface

IPCB_GerberOptions interface

PlacerOptions property

(IPCB_Board interface)

Syntax

```
Property  PlacerOptions : IPCB_AdvancedPlacerOptions Read GetState_PlacerOptions;
```

Description

This property returns you the IPCB_PlacerOptions interface which represents the Options for the Placement facility in the PCB editor.

Example

See also

IPCB_Board interface

IPCB_PlacerOptions interface

PrinterOptions property

(IPCB_Board interface)

Syntax

```
Property  PrinterOptions : IPCB_PrinterOptions Read GetState_PrinterOptions;
```

Description

This property returns you the IPCB_PrinterOptions interface which represents the Options for the Printer setup facility in the PCB editor.

Example

See also

IPCB_Board interface

IPCB_PrinterOptions interface

OutputOptions property

(IPCB_Board interface)

Syntax

```
Property  OutputOptions : IPCB_OutputOptions Read GetState_OutputOptions;
```

Description

This property returns you the IPCB_OutputOptions interface which represents the Options for the Output facility in the PCB editor.

Example

See also

IPCB_Board interface

IPCB_OutputOptions interface

IPCB_Library Interface

Overview

The **IPCB_Library** interface represents the library document. A library document has a list of components (footprints). The component in focus in the PCB library is always the current component. This current component is represented by the **IPCB_LibComponent** interface.

To obtain the settings of the library document, you obtain the **IPCB_Board** interface, to obtain the primitives of a component (footprint), you obtain the **IPCB_LibComponent** interface via the Library Iterator interface.

There is a three way relationship: the **IPCB_Board**, the **IPCB_LibComponent** and the **IPCB_Library** interfaces that all work together for a PCB library document.

The **IPCB_Library** interface is a standalone interface.

IPCB_Library methods

GetState_CurrentComponent

SetState_CurrentComponent

GetState_Board

RegisterComponent

DeRegisterComponent

IPCB_Library properties

CurrentComponent

Board

GetUniqueCompName
CreateNewComponent
RemoveComponent
GetComponentByName

SetBoardToComponentByName
Navigate_FirstComponent
SetCurrentComponentReference

LibraryIterator_Create
LibraryIterator_Destroy

Example

```
Var
    CurrentLib    : IPCB_Library;
    NewPCBLibComp : IPCB_LibComponent;
Begin
    If PCBServer = Nil Then Exit;
    CurrentLib := PcbServer.GetCurrentPCBLibrary;
    If CurrentLib = Nil Then Exit;

    // ditto
End;
```

See also

IPCB_ServerInterface interface
IPCB_LibComponent interface
IPCB_LibraryIterator interface

GetState and SetState Methods

GetState_Board method

(IPCB_Library interface)

Syntax

```
Function GetState_Board : IPCB_Board;
```

Description

This function retrieves the **IPCB_Board** interface where the current component (footprint) is in. This **IPCB_Board** interface also contains the system settings such as Snap Grid, Visible and Big Visible Grid Units and Output Options for the PCB library document.

There is a three way relationship: the **IPCB_Board**, the **IPCB_LibComponent** and the **IPCB_Library** interfaces that all work together for the PCB library document.

Example

See also

IPCB_Library interface

GetState_CurrentComponent method

(IPCB_Library interface)

Syntax

```
Function GetState_CurrentComponent : IPCB_LibComponent;
```

Description

This function fetches the current component that is in focus in the PCB library. A footprint (component) in the library is represented by the **IPCB_LibComponent** interface. A PCB Library document is represented differently in regards to a PCB document; a pcb library is composed of footprints and each footprint has its own “window”.

Example

See also

IPCB_Library interface

IPCB_Group interface

SetState_CurrentComponent method

(IPCB_Library interface)

Syntax

```
Procedure SetState_CurrentComponent (Const Component : IPcb_LibComponent);
```

Description

This procedure sets an existing component from the PCB library as the current component (in focus). Basically a component that is currently in focus in the library is the current component.

Note a footprint (component) in the library is represented by the **IPCB_LibComponent** interface.

Example

See also

IPCB_Library interface

Methods

DeRegisterComponent method

(IPCB_Library interface)

Syntax

```
Function DeRegisterComponent(Component : IPcb_LibComponent) : Boolean;
```

Description

This method de-registers this component from the PCB library. That is, the library does not recognize this component after it has been de-registered.

Example

See also

IPCB_Library interface

IPCB_LibComponent interface

GetUniqueCompName method

(IPCB_Library interface)

Syntax

```
Function  GetUniqueCompName  (Const ATestCompName : TPCBString       ) : TPCBString;
```

Description

This function returns you the unique component name and if the supplied component name parameter is taken, this parameter is modified to guarantee its uniqueness.

Example

See also

IPCB_Library interface

IPCB_LibComponent interface

CreateNewComponent method

(IPCB_Library interface)

Syntax

```
Function  CreateNewComponent : IPCB_LibComponent;
```

Description

This function creates a new component and it is represented by the IPCB_LibComponent interface. This is equivalent to the CreatePCBLibComp method from the IPCB_ServerInterface interface.

Example

See also

IPCB_Library interface

IPCB_LibComponent interface

RemoveComponent method

(IPCB_Library interface)

Syntax

```
Procedure RemoveComponent (Var AComponent : IPCB_LibComponent);
```

Description

This procedure removes a component from the current library.

Example

See also

IPCB_Library interface

IPCB_LibComponent interface

SetBoardToComponentByName method

(IPCB_Library interface)

Syntax

```
Function  SetBoardToComponentByName(Const ACompName : TPCBString) : Boolean;
```

Description

This function sets the current library to the specified component by its component name string. If it is successful, a true value is returned.

Example

See also

IPCB_Library interface

IPCB_LibComponent interface

Navigate_FirstComponent method

(IPCB_Library interface)

Syntax

```
Procedure Navigate_FirstComponent;
```

Description

This procedure forces the library to navigate to the first component in the library and set the focus to it.

Example

See also

IPCB_Library interface

IPCB_LibComponent interface

SetCurrentComponentReference method

(IPCB_Library interface)

Syntax

```
Procedure SetCurrentComponentReference (AX : TCoord;
                                        AY : TCoord);
```

Description

This procedure sets the component reference of the currently focused component as the center.

Example

See also

IPCB_Library interface

IPCB_LibComponent interface

GetComponentByName method

(IPCB_Library interface)

Syntax

```
Function GetComponentByName (Const CompName : TPCBString ) : IPCB_LibComponent;
```

Description

This function returns you the **IPCB_LibComponent** of a PCB component (footprint) if the CompName string.

Example

See also

IPCB_Library interface

IPCB_LibComponent interface

LibraryIterator_Create method

(IPCB_Library interface)

Syntax

```
Function LibraryIterator_Create : IPCB_LibraryIterator;
```

Description

This function creates a library iterator that fetches footprints in a PCB library. Each footprint fetched by the iterator is a **IPCB_LibComponent** interface which is inherited by the **IPCB_Group** interface.

DelphiScript Example

```
Var
    CurrentLib        : IPCB_Library;
    FootprintIterator : IPCB_LibraryIterator;
    Footprint         : IPCB_LibComponent;
Begin
    CurrentLib := PCBServer.GetCurrentPCBLibrary;
    If CurrentLib = Nil Then
    Begin
        ShowMessage('This is not a PCB Library document');
        Exit;
    End;

    // For each page of library is a footprint
```

```
    FootprintIterator := CurrentLib.LibraryIterator_Create;
    FootprintIterator.SetState_FilterAll;

    Try
        // Within each footprint page, fetch primitives of the footprint
        // A footprint is a IPCB_LibComponent inherited from
        // the IPCB_Group. A container object that stores primitives.
        Footprint := FootprintIterator.FirstPCBObject;
        While Footprint <> Nil Do
        Begin
           // do what you want with the footprint
           Footprint := FootprintIterator.NextPCBObject;
        End;
    Finally
        CurrentLib.LibraryIterator_Destroy(FootprintIterator);
    End;
End;
```

See also

IPCB_LibraryIterator interface

IPCB_Library interface

IPCB_LibComponent interface

LibraryIterator_Destroy method

(IPCB_Library interface)

Syntax

```
Procedure LibraryIterator_Destroy(Var AIterator : IPCB_LibraryIterator);
```

Description

This **LibraryIterator_Destroy** method destroys the library iterator after it was used in iterating for footprints in a PCB library document.

Example

```
Var
    CurrentLib        : IPCB_Library;
    FootprintIterator : IPCB_LibraryIterator;
    Footprint         : IPCB_LibComponent;
Begin
    CurrentLib := PCBServer.GetCurrentPCBLibrary;
    If CurrentLib = Nil Then
    Begin
        ShowMessage('This is not a PCB Library document');
        Exit;
    End;

    // For each page of library is a footprint
    FootprintIterator := CurrentLib.LibraryIterator_Create;
    FootprintIterator.SetState_FilterAll;

    Try
        // Within each page, fetch primitives of the footprint
        // A footprint is a IPCB_LibComponent inherited from
```

```
        // IPCB_Group which is a container object that stores primitives.
        Footprint := FootprintIterator.FirstPCBObject;
        While Footprint <> Nil Do
        Begin
           // do what you want with the footprint
           Footprint := FootprintIterator.NextPCBObject;
        End;
    Finally
        CurrentLib.LibraryIterator_Destroy(FootprintIterator);
    End;
End;
```

See also

IPCB_Library interface

IPCB_LibComponent interface

IPCB_LibraryIterator interface

RegisterComponent method

(IPCB_Library interface)

Syntax

```
Function RegisterComponent (Component : IPcb_LibComponent) : Boolean;
```

Description

The **RegisterComponent** method registers the new footprint in the PCB library document, so that the PCB system is aware of this new footprint.

For example when creating a new footprint programmatically, this footprint needs to be registered in the PCB library first before setting it to be the current component.

Example

```
Var
    CurrentLib    : IPCB_Library;
    NewPCBLibComp : IPCB_LibComponent;
    NewPad        : IPCB_Pad;
Begin
    If PCBServer = Nil Then Exit;
    CurrentLib := PcbServer.GetCurrentPCBLibrary;
    If CurrentLib = Nil Then Exit;

    NewPCBLibComp := PCBServer.CreatePCBLibComp;
    NewPcbLibComp.Name := 'ANewComponent';

    CurrentLib.RegisterComponent(NewPCBLibComp);
    CurrentLib.CurrentComponent := NewPcbLibComp;
    // ditto
End;
```

See also

IPCB_Library interface

IPCB_LibComponent interface

Properties

Board property

(IPCB_Library interface)

Syntax

```
Property Board : IPCB_Board Read GetState_Board;
```

Description

The property represents the board that the current component is residing on in the PCB library document. This **IPCB_Board** interface also contains the system settings such as Snap Grid, Visible and Big Visible Grid Units and Output Options for the PCB library document.

The read only **Board** property is supported by the **GetState_Board** method.

There is a three way relationship: the **IPCB_Board**, the **IPCB_LibComponent** and the **IPCB_Library** interfaces that all work together for a PCB library document.

Example

See also

IPCB_Library interface

CurrentComponent property

(IPCB_Library interface)

Syntax

```
Property CurrentComponent : IPCB_LibComponent Read GetState_CurrentComponent Write
SetState_CurrentComponent;
```

Description

This property determines the current component (footprint) that is in focus or displayed in the PCB library document in Altium Designer.

When creating a new footprint programmatically, this footprint needs to be registered in the PCB library first before setting it to be the current component.

This CurrentComponent property is supported by GetState_CurrentComponent and SetState_CurrentComponent methods.

Example

```
Var
    CurrentLib    : IPCB_Library;
    NewPCBLibComp : IPCB_LibComponent;
    NewPad        : IPCB_Pad;
Begin
    If PCBServer = Nil Then Exit;
    CurrentLib := PcbServer.GetCurrentPCBLibrary;
    If CurrentLib = Nil Then Exit;

    NewPCBLibComp := PCBServer.CreatePCBLibComp;
    NewPcbLibComp.Name := 'ANewComponent';

    CurrentLib.RegisterComponent(NewPCBLibComp);
    CurrentLib.CurrentComponent := NewPcbLibComp;
    // ditto
End;
```

See also

IPCB_Library interface

IPCB_LibComponent interface

IPCB_Sheet

Overview

The **IPCB_Sheet** interface represents the background workspace for the PCB document and can include fabrication and assembly documentation as well as the board outline. The **IPCB_Board** interface has the **IPCB_Sheet** interface as an aggregation interface because a sheet is part of the PCB document.

Notes

The sheet behind the PCB can be shown or not.

The coordinates of the PCB sheet can be defined programmatically.

IPCB_Sheet methods	IPCB_Sheet properties
I_ObjectAddress	SheetX
	SheetY
	SheetWidth
	SheetHeight
	ShowSheet
	LockSheet

See also

IPCB_Board

Methods

I_ObjectAddress method

(IPCB_AbstractIterator, IPCB_BoardIterator, IPCB_SpatialIterator, IPCB_GroupIterator, IPCB_Sheet)

Syntax

```
Function I_ObjectAddress : TPCBObjectHandle;
```

Description

The **I_ObjectAddress** property retrieves the pointer to the iterator object. This property is useful for situations where you need to have references to objects (not to object interfaces) and store them in a TList container for example.

See also

IPCB_Sheet interface

Properties

SheetHeight property

(IPCB_Board interface)

Syntax

```
Property  SheetHeight : TCoord Read GetState_SheetHeight Write SetState_SheetHeight;
```

Description

The SheetHeight property represents the sheet's height.

Example

See also

IPCB_Sheet interface

SheetWidth property

(IPCB_Sheet interface)

Syntax

```
Property  SheetWidth : TCoord Read GetState_SheetWidth Write SetState_SheetWidth;
```

Description

The SheetWidth property represents the width of the sheet.

Example

See also

IPCB_Sheet interface

SheetX property

(IPCB_Sheet interface)

Syntax

```
Property  SheetX : TCoord Read GetState_SheetX Write SetState_SheetX;
```

Description

The SheetX property represents the X coordinate of the sheet.

Example

See also

IPCB_Sheet interface

SheetY property

(IPCB_Sheet interface)

Syntax

```
Property  SheetY : TCoord Read GetState_SheetY Write SetState_SheetY;
```

Description

The SheetY property represents the Y coordinate of the sheet.

Example

See also

IPCB_Sheet interface

ShowSheet method

(IPCB_Sheet interface)

Syntax

```
Property  ShowSheet : Boolean Read GetState_ShowSheet Write SetState_ShowSheet;
```

Description

This property retrieves or sets the boolean value. The Sheet property represents the bounds where a board outline and assembly / fabrication details are included within.

Example

```
Function UnitToString(U : TUnit) : TPCBString;
Begin
    Result := '';
    Case U of
       eImperial : Result := 'Imperial (mil)';
       eMetric   : Result := 'Metric (mm)';
    End;
End;
{.....................................................................}
{.....................................................................}
Function BoolToString(B : Boolean) : TPCBString;
Begin
    Result := 'False';
    If B Then Result := True;
End;
{.....................................................................}
```

```
{.....................................................................}
Procedure Query_Board;
Var
    Board         : IPCB_Board;
    LibraryExists : TPCBString;
    AShowSheet    : TPCBString;
    ALockSheet    : TPCBString;
Begin
    Board := PCBServer.GetCurrentPCBBoard;
    If Board = Nil Then Exit;
    LibraryExists := BoolToString(Board.IsLibrary);
    AShowSheet    := BoolToString(Board.PCBSheet.ShowSheet);
    ALockSheet    := BoolToString(Board.PCBSheet.LockSheet);
    ShowMessage(
        'Board Handle  = '   + IntToStr    (Board.I_ObjectAddress)      + #13 +
        'Window Handle = '   + IntToStr    (Board.PCBWindow)            + #13 +
        'Board Filename ='   +               Board.FileName             + #13 +
        'Is a Library = '    +               LibraryExists              + #13 +
        'Origin X = '        + IntToStr    (Board.XOrigin)              + #13 +
        'Origin Y = '        + IntToStr    (Board.YOrigin)              + #13 +
        'Board Units = '     + UnitToString(Board.DisplayUnit)          + #13 +
        'Current layer = '   + Layer2String(Board.CurrentLayer)         + #13 +
        'Sheet.X = '         + IntToStr    (Board.PCBSheet.SheetX)      + #13 +
        'Sheet.Y = '         + IntToStr    (Board.PCBSheet.SheetY)      + #13 +
        'Sheet.Height = '    + IntToStr    (Board.PCBSheet.SheetHeight) + #13 +
        'Sheet.Width = '     + IntToStr    (Board.PCBSheet.SheetWidth)
        'Sheet is shown = '  +               AShowSheet                 + #13 +
        'Sheet is locked = ' +               ALockSheet
    );
End;
```

See also

IPCB_Sheet interface

LockSheet method

(IPCB_Sheet interface)

Syntax

```
Property  LockSheet : Boolean Read GetState_LockSheet Write SetState_LockSheet;
```

Description

The LockSheet property represents whether the objects on a mechanical layer linked to the sheet is locked or not.

Example

See also

IPCB_Sheet interface

IPCB_LayerStack

Overview

The **IPCB_LayerStack** interface represents the layer stack for the current PCB document. This Layer Stack interface is a property within in the **IPCB_Board** interface.

Strictly speaking, the **IPCB_LayerStack** interface represents the layer stack and therefore only has copper based layers such as top, mid1-30, bottom layers and internal planes. However you can use the **LayerObject** property with the **IPCB_Board** parameter passed in to obtain any PCB layer for the PCB document.

Iterating copper layers within the Layer Stack

To query for existing copper layers (signal layers and internal players) within the layer stack, you can use the **FirstLayer** and **NextLayer** properties of the **IPCB_LayerStack** interface to iterate for such layers.

Notes

Each layer can be represented as a IPCB_LayerObject, IPCB_InternalPlane, IPCB_DrillLayerPair or IPCB_MechanicalLayerPairs interfaces.

A layer can have dielectric properties which is represented by a **IPCB_DielectricObject** interface.

To have access to other layers of the PCB document, use the **LayerObject** property of the **IPCB_LayerStack** interface.

IPCB_LayerStack methods

`FirstLayer`
`NextLayer`
`PreviousLayer`
`LastLayer`
`InsertLayer`
`LastInternalPlane`
`FirstAvailableSignalLayer`
`FirstAvailableInternalPlane`
`SignalLayerCount`

IPCB_LayerStack properties

`Board`
`LayerObject`
`DielectricTop`
`DielectricBottom`
`ShowDielectricTop`
`ShowDielectricBottom`

See also

Using PCB Layers
Using the PCB Layer Stack
IPCB_LayerObject interface
IPCB_InternalPlane interface
IPCB_Board interface
IPCB_DielectricObject interface
QueryLayerStack and QueryMechLayers script in the **\Examples\Scripts\Delphiscript\PCB** folder

Methods

FirstLayer method

(IPCB_LayerStack interface)

Syntax

```
Function  FirstLayer : IPCB_LayerObject;
```

Description

The Firstlayer property fetches the first layer stored in the layer stack for the PCB document. To fetch the next layer in the layer stack, invoke the NextLayer property. Notice that the layer stack only stores signal and internal (copper based) layers.

Example

```
Var
    PCBBoard      : IPCB_Board;
    TheLayerStack : IPCB_LayerStack;
    i             : Integer;
    LayerObj      : IPCB_LayerObject;
    LS            : String;
Begin
    PCBBoard := PCBServer.GetCurrentPCBBoard;
    If PCBBoard = Nil Then Exit;
```

```
    TheLayerStack := PCBBoard.LayerStack;
    If TheLayerStack = Nil Then Exit;
    LS       := '';
    LayerObj := TheLayerStack.FirstLayer;
    Repeat
        LS       := LS + Layer2String(LayerObj.LayerID) + #13#10;
        LayerObj := TheLayerStack.NextLayer(LayerObj);
    Until LayerObj = Nil;
    ShowInfo('The Layer Stack has :'#13#10 + LS);
End;
```

See also

IPCB_LayerStack interface

FirstAvailableInternalPlane method

(IPCB_LayerStack interface)

Syntax

```
Function FirstAvailableInternalPlane : IPCB_InternalPlane;
```

Description

This function retrieves the first available internal plane object interface for the PCB document.

Example

See also

IPCB_LayerStack interface

FirstAvailableSignalLayer method

(IPCB_LayerStack interface)

Syntax

```
Function  FirstAvailableSignalLayer : IPCB_LayerObject;
```

Description

This function retrieves the first available signal layer from the layer stack. A layer stack only stores copper based layers such as signal and internal plane layers.

Example

See also

IPCB_LayerStack interface

IPCB_LayerObject interface

InsertLayer method

(IPCB_LayerStack interface)

Syntax

```
Procedure InsertLayer(L : TLayer);
```

Description

Example

See also

IPCB_LayerStack interface

LastInternalPlane method

(IPCB_LayerStack interface)

Syntax

```
Function LastInternalPlane : IPCB_InternalPlane;
```

Description

This function retrieves the last internal plane from the layer stack if it exists. If there is no internal planes in the layer stack, the function will return a Nil value.

Example

See also

IPCB_LayerStack interface

IPCB_InternalPlane interface

LastLayer property

(IPCB_LayerStack interface)

Syntax

```
Function LastLayer : IPCB_LayerObject;
```

Description

Example

See also

IPCB_LayerStack interface

NextLayer property

(IPCB_LayerStack interface)

Syntax

```
Function  NextLayer(L : IPCB_LayerObject) : IPCB_LayerObject;
```

Description

The **Nextlayer** property fetches the next layer stored in the layer stack for the PCB document after the **FirstLayer** property has been invoked. Notice that the layer stack only stores signal and internal (copper based) layers.

Example

```
Var
    PCBBoard      : IPCB_Board;
    TheLayerStack : IPCB_LayerStack;
    i             : Integer;
    LayerObj      : IPCB_LayerObject;
    LS            : String;
Begin
    PCBBoard := PCBServer.GetCurrentPCBBoard;
    If PCBBoard = Nil Then Exit;

    // Note that the Layer stack only stores existing copper based layers.
    TheLayerStack := PCBBoard.LayerStack;
    If TheLayerStack = Nil Then Exit;
    LS       := '';
    LayerObj := TheLayerStack.FirstLayer;
    Repeat
        LS       := LS + Layer2String(LayerObj.LayerID) + #13#10;
```

```
        LayerObj := TheLayerStack.NextLayer(LayerObj);
    Until LayerObj = Nil;

    ShowInfo('The Layer Stack has :'#13#10 + LS);
End;
```

See also

IPCB_LayerStack interface

PreviousLayer method

(IPCB_LayerStack interface)

Syntax

```
Function  PreviousLayer(L : IPCB_LayerObject) : IPCB_LayerObject;
```

Description

The **Previouslayer** property fetches the previous layer stored in the layer stack for the PCB document after the **FirstLayer** property has been invoked. Notice that the layer stack only stores signal and internal (copper based) layers.

Example

See also

IPCB_LayerStack interface

SignalLayerCount method

(IPCB_LayerStack interface)

Syntax

```
Function SignalLayerCount : Integer;
```

Description

This function returns the number of signal layers in the layer stack for the PCB document.

Example

See also

IPCB_LayerStack interface

RemoveFromStack method

(IPCB_LayerStack interface)

Syntax

```
Procedure RemoveFromStack(L : IPCB_LayerObject);
```

Description

This procedure removes the targeted layer (represented by the IPCB_LayerObject interface) from the layer stack.

Example

See also

IPCB_LayerStack interface

IPCB_LayerObject interface

InsertInStackBelow method

(IPCB_LayerStack interface)

Syntax

```
Procedure InsertInStackBelow(RefL   : IPCB_LayerObject;
                             L      : IPCB_LayerObject);
```

Description

Example

See also

IPCB_LayerStack interface

IPCB_LayerObject interface

InsertInStackAbove method

(IPCB_LayerStack interface)

Syntax

```
Procedure InsertInStackAbove(RefL   : IPCB_LayerObject;
                             L      : IPCB_LayerObject);
```

Description

Example

See also

IPCB_LayerStack interface

IPCB_LayerObject interface

Properties

Board property

(IPCB_LayerStack interface)

Syntax

```
Property Board : IPCB_Board Read GetState_Board;
```

Description

This property returns the PCB document that is represented by the **IPCB_Board** interface, that the layer stack is associated with.

Example

See also

IPCB_LayerStack interface

IPCB_Board interface

DielectricBottom property

(IPCB_Board interface)

Syntax

```
Property  DielectricBottom : IPCB_DielectricObject Read GetState_DielectricBottom;
```

Description

This property returns the **IPCB_DielectricObject** interface associated with the dielectric information for the bottom layer of the layer stack.

Example

See also

IPCB_DielectricObject interface

DielectricTop property

(IPCB_Board interface)

Syntax

```
Property  DielectricTop : IPCB_DielectricObject Read GetState_DielectricTop;
```

Description

This property returns the **IPCB_DielectricObject** interface associated with the dielectric information for the top layer of the layer stack.

Example

See also

IPCB_DielectricObject interface

LayerObject property

(IPCB_LayerStack interface)

Syntax

```
Property  LayerObject [L : TLayer] : IPCB_LayerObject  Read GetState_LayerObject;
```

Description

The LayerObject property retrieves the layer object interface for the specified layer, L of TLayer type. It is a read only property.

Example

```
Var
    PCBBoard      : IPCB_Board;
    TheLayerStack : IPCB_LayerStack;
    i             : Integer;
    LayerObj      : IPCB_LayerObject;
    LS            : String;
Begin
    PCBBoard := PCBServer.GetCurrentPCBBoard;
    If PCBBoard = Nil Then Exit;

    TheLayerStack := PCBBoard.LayerStack;
    If TheLayerStack = Nil Then Exit;
    LS       := '';
    LayerObj := TheLayerStack.FirstLayer;
    Repeat
        LS       := LS + Layer2String(LayerObj.LayerID) + #13#10;
        LayerObj := TheLayerStack.NextLayer(LayerObj);
    Until LayerObj = Nil;
    ShowInfo('The Layer Stack has :'#13#10 + LS);
End;
```

See also

IPCB_LayerStack interface

IPCB_LayerObject interface

TLayer type

ShowDielectricBottom property

(IPCB_LayerStack interface)

Syntax

```
Property  ShowDielectricBottom           : Boolean          Read GetState_ShowBotDielectric
Write SetState_ShowBotDielectric;
    End;
```

Description

This property enables or disables the dielectric layer for the bottom layer.

Example

See also

IPCB_LayerStack interface

ShowDielectricTop property

(IPCB_LayerStack interface)

Syntax

```
Property ShowDielectricTop : Boolean Read GetState_ShowTopDielectric Write
SetState_ShowTopDielectric;
```

Description

This property enables or disables the dielectric layer for the top layer.

Example

See also

IPCB_LayerStack interface

IPCB_SpecialStringConverter Interface

Overview

The **IPCB_SpecialStringConverter** interface provides a way to fetch special strings in a PCB Project. You would need to pass the document as a parameter in the Convert function and obtain the special strings.

IPCB_SpecialStringConverter methods	IPCB_SpecialStringConverter properties
`FirstSpecialStringName`	
`NextSpecialStringName`	
`Convert`	

See also

IPCB_ServerInterface interface

IPCB_Text interface

Methods

Convert method

(IPCB_SpecialStringConverter interface)

Syntax

```
Function Convert(Const Primitive : IPCB_Primitive;Const aString : TString;Out ConvertedString
: TPCBString) : Boolean;
```

Description

The convert function converts a special string as a formatted string and returns a boolean result whether the conversion is a success or not.

Example

See also

IPCB_SpecialStringConverter interface

FirstSpecialStringName method

(IPCB_SpecialStringConverter interface)

Syntax

```
Function FirstSpecialStringName : TPCBString;
```

Description

This function obtains the first special string name used in a design project (for example a PCB Project).

Example

See also

IPCB_SpecialStringConverter interface

NextSpecialStringName method

NextSpecialStringName method

(IPCB_SpecialStringConverter interface)

Syntax

```
Function NextSpecialStringName : TPCBString;
```

Description

This function obtains the next special string name used in a design project (for example a PCB Project).

Example

See also

IPCB_SpecialStringConverter interface

IPCB_PrimitiveCounter Interface

Overview

The IPCB_PrimitiveCounter interface gives you the means of obtaining the hole count and string count for the focussed PCB document via the IPCB_Board's PrimitiveCounter property.

IPCB_PrimitiveCounter methods	IPCB_PrimitiveCounter properties
GetObjectCount	HoleCount
GetCount	StringCount
GetHoleCount	
GetStringCount	

See also

IPCB_Board interface

Methods

GetCount method

(IPCB_PrimitiveCounter interface)

Syntax

```
Function GetCount (ObjectSet : TObjectSet) : Cardinal;
```

Description

The GetCount function counts the objects of a set of object types specified by the ObjectSet parameter.

Example

See also

IPCB_PrimitiveCounter interface

TObjectSet type

GetHoleCount method

(IPCB_PrimitiveCounter interface)

Syntax

```
Function GetHoleCount : Cardinal;
```

Description

This function counts the holes (pads and vias) on the current PCB document.

Example

See also

IPCB_PrimitiveCounter interface

GetObjectCount method

(IPCB_PrimitiveCounter interface)

Syntax

```
Function GetObjectCount (ObjectId: TObjectId) : Cardinal;
```

Description

This function counts objects of a specific object type.

Example

See also

IPCB_PrimitiveCounter interface

GetStringCount method

(IPCB_PrimitiveCounter interface)

Syntax

```
Function GetStringCount : Cardinal
```

Description

This function counts text strings on the PCB document.

Example

See also

IPCB_PrimitiveCounter interface

Properties

HoleCount property

(IPCB_PrimitiveCounter interface)

Syntax

```
Property HoleCount : Cardinal Read GetHoleCount;
```

Description

This property obtains the hole count from the PCB document (Pads and Vias).

Example

See also

IPCB_PrimitiveCounter interface

StringCount property

(IPCB_PrimitiveCounter interface)

Syntax

```
Property StringCount : Cardinal Read GetStringCount;
```

Description

This property obtains string (text object) count from the PCB document.

Example

See also

IPCB_PrimitiveCounter interface

PCB Layer Interfaces

IPCB_LayerObject

Overview

The **IPCB_LayerObject** interface represents a layer used in a PCB document. Each layer has properties such as layer id, name, used by primitives and whether it is displayed for example. This interface is a property in the **IPCB_LayerStack** interface.

The layer stack for a PCB document only deals with copper based layers such as signal and internal plane layers. Each layer in the layer stack can have dielectric information and layer pairs can be specified. However there is a **LayerObject** property in the **IPCB_LayerStack** interface which allows you to access any PCB layer for the PCB board.

Iterating for any PCB layer of a PCB document

Although the **IPCB_LayerStack** interface basically deals with copper based layers that are used in the layer stack, this Layer Stack interface can be used to look for other PCB layers that are not in the layer stack. The **LayerObject** property from this layer stack interface obtains any PCB layer whether it is a keep out layer, top signal layer or a mechanical 16 layer.

Methods

```
Function  I_ObjectAddress : TPCBObjectHandle;
Function  IsInLayerStack  : Boolean;
```

Properties

```
Property LayerStack                     : IPCB_LayerStack
Property LayerID                        : TLayer
Property Name                           : TPCBString
Property CopperThickness                : TCoord
Property Dielectric                     : IPCB_DielectricObject
Property UsedByPrims                    : Boolean
Property IsDisplayed[Board : IPCB_Board] : Boolean
Property PreviousLayer                  : TLayer
Property NextLayer                      : TLayer
```

Example

```
Var
    PCBBoard      : IPCB_Board;
    TheLayerStack : IPCB_LayerStack;
    i             : Integer;
    LayerObj      : IPCB_LayerObject;
    LS            : String;
Begin
    PCBBoard := PCBServer.GetCurrentPCBBoard;
    If PCBBoard = Nil Then Exit;

    // Note that the Layer stack only stores
    // existing copper based layers.
    // But you can use the LayerObject property to fetch all layers.
    TheLayerStack := PCBBoard.LayerStack;
    If TheLayerStack = Nil Then Exit;
    LS       := '';

    LayerObj := TheLayerStack.FirstLayer;
    Repeat
        LS       := LS + Layer2String(LayerObj.LayerID) + #13#10;
        LayerObj := TheLayerStack.NextLayer(LayerObj);
```

```
    Until LayerObj = Nil;
    ShowInfo('The Layer Stack has :'#13#10 + LS);
End;
```

See also

TLayer enumerated values

TCoord value

IPCB_DielectricObject interface

IPCB_LayerStack interface

IPCB_MechanicalLayer

Overview

There are 16 general purpose mechanical layers for defining the board layout, placing dimensions on, including fabrication details on, or any other mechanical details the design requires.

To obtain mechanical layers, you iterate for layers on a PCB document, and once you determine it is a mechanical layer, you can wrap the layer as a **IPCB_MechanicalLayer** interface.

Note that the Layer stack only stores existing copper based layers, but you can use the LayerObject property from the **IPCB_LayerStack** interface to fetch all layers by using a Layer loop.

Code snippet

```
    TheLayerStack := PCBBoard.LayerStack;
    If TheLayerStack = Nil Then Exit;
    For Layer := eMechanical1 to eMechanical16 Do
    Begin
        MechLayerObj := TheLayerStack.LayerObject[Layer];
        // where MechLayerObj is a IPCB_MechanicalLayer type
    End;
```

The **IPCB_MechanicalLayer** interface hierarchy is as follows;

IPCB_LayerObject

IPCB_MechanicalLayer

IPCB_MechanicalLayer methods

```
GetState_MechLayerEnabled
GetState_DisplayInSingleLayerMode
GetState_LinkToSheet
SetState_MechLayerEnabled
SetState_DisplayInSingleLayerMode
SetState_LinkToSheet
```

IPCB_MechanicalLayer properties

```
MechanicalLayerEnabled
DisplayInSingleLayerMode
LinkToSheet
```

Example

```
Var
    PCBBoard      : IPCB_Board;
    TheLayerStack : IPCB_LayerStack;
    i             : Integer;
    LayerObj      : IPCB_MechanicalLayer;
    Layer         : TLayer;
    LS            : String;
Begin
    PCBBoard := PCBServer.GetCurrentPCBBoard;
    If PCBBoard = Nil Then Exit;
```

```
    // Note that the Layer stack only stores existing copper based layers.
    // But you can use the LayerObject property to fetch all layers.
    TheLayerStack := PCBBoard.LayerStack;
    If TheLayerStack = Nil Then Exit;
    LS := '';
    For Layer := eMechanical1 to eMechanical16 Do
    Begin
        LayerObj := TheLayerStack.LayerObject[Layer];
        ShowMessage(Layer2String(Layer) + #13 +
                    ' linked '                          + BooleanToString(LayerObj.LinkToSheet)
+ #13 +
                    ' enabled '                         +
BooleanToString(LayerObj.MechanicalLayerEnabled) + #13 +
                    ' displayed in single layer mode ' +
BooleanToString(LayerObj.DisplayInSingleLayerMode) + #13);
        End;
    End;
End;
```

See also

IPCB_LayerObject interface

IPCB_LayerStack interface

TLayer enumerated values

Methods

SetState_MechLayerEnabled method

(IPCB_MechanicalLayer interface)

Syntax

```
Procedure SetState_MechLayerEnabled (Value : Boolean);
```

Description

This method determines whether this mechanical layer is enabled or not for the current PCB document. You cannot disable the mechanical layers that already have design objects on them.

This method is used by the **MechLayerEnabled** property.

Example

See also

IPCB_MechanicalLayer interface

SetState_LinkToSheet method

(IPCB_MechanicalLayer interface)

Syntax

```
Procedure SetState_LinkToSheet (Value : Boolean);
```

Description

This method determines whether this mechanical layer is linked to the sheet on the PCB document or not. Once a sheet is linked to the mechanical layer, the sheet is re-sized automatically to fit the objects on the linked layer when a zoom command is executed.

This method is used for the **LinkToSheet** property.

Example

See also

IPCB_MechanicalLayer interface

SetState_DisplayInSingleLayerMode method

(IPCB_MechanicalLayer interface)

Syntax

```
Procedure SetState_DisplayInSingleLayerMode(Value : Boolean);
```

Description

This method determines whether the document is displayed in the single layer mode. Set it true to override the system's single layer mode setting and the design objects on these enabled single layer mode mechanical layers still show up in the single layer mode.

This method is used by the DisplayInSingleLayerMode property.

Example

See also

IPCB_MechanicalLayer interface

GetState_MechLayerEnabled method

(IPCB_MechanicalLayer interface)

Syntax

```
Function GetState_MechLayerEnabled : Boolean;
```

Description

This method determines whether this mechanical layer is enabled or not for the current PCB document. You cannot disable the mechanical layers that already have design objects on them.

This method is used by the **MechLayerEnabled** property.

Example

See also

IPCB_MechanicalLayer interface

GetState_LinkToSheet method

(IPCB_MechanicalLayer interface)

Syntax

```
Function GetState_LinkToSheet : Boolean;
```

Description

This method determines whether this mechanical layer is linked to the sheet on the PCB document or not. Once a sheet is linked to the mechanical layer, the sheet is re-sized automatically to fit the objects on the linked layer when a zoom command is executed.

This method is used for the **LinkToSheet** property.

Example

See also

IPCB_MechanicalLayer interface

GetState_DisplayInSingleLayerMode method

(IPCB_MechanicalLayer interface)

Syntax

```
Function GetState_DisplayInSingleLayerMode : Boolean;
```

Description

This method determines whether the document is displayed in the single layer mode. Set it true to override the system's single layer mode setting and the design objects on these enabled single layer mode mechanical layers still show up in the single layer mode.

This method is used by the DisplayInSingleLayerMode property.

Example

See also

IPCB_MechanicalLayer interface

Properties

MechanicalLayerEnabled property

(IPCB_MechanicalLayer interface)

Syntax

```
Property MechanicalLayerEnabled : Boolean Read GetState_MechLayerEnabled Write
SetState_MechLayerEnabled;
```

Description

This property determines whether this mechanical layer is enabled or not for the current PCB document. You cannot disable the mechanical layers that already have design objects on them.

This property is supported by the GetState_MechLayerEnabled and SetState_MechLayerEnabled methods.

Example

See also

IPCB_MechanicalLayer interface

LinkToSheet property

(IPCB_MechanicalLayer interface)

Syntax

```
Property LinkToSheet : Boolean Read GetState_LinkToSheet Write SetState_LinkToSheet;
```

Description

This property determines whether this mechanical layer is linked to the sheet on the PCB document or not. Once a sheet is linked to the mechanical layer, the sheet is re-sized automatically to fit the objects on the linked layer when a zoom command is executed.

This property is supported by the SetState_LinkToSheet and GetState_LinkToSheet methods.

Example

See also

IPCB_MechanicalLayer interface

DisplayInSingleLayerMode property

(IPCB_MechanicalLayer interface)

Syntax

```
Property DisplayInSingleLayerMode : Boolean Read GetState_DisplayInSingleLayerMode Write
SetState_DisplayInSingleLayerMode;
```

Description

This property determines whether the document is displayed in the single layer mode. Set it true to override the system's single layer mode setting and the design objects on these enabled single layer mode mechanical layers still show up in the single layer mode.

This property is supported by the GetState_DisplayInSingleLayerMode and SetState_DisplayInSingleLayerMode methods.

Example

See also

IPCB_MechanicalLayer interface

IPCB_DielectricObject

Overview

The **IPCB_DielectricObject** interface represents the dielectric properties for the specified PCB layer.

Notes

The **IPCB_DielectricObject** interface is used by the **IPCB_LayerStack** interface.

Properties

```
Property  DielectricMaterial : TPCBString
Property  DielectricType     : TDielectricType
Property  DielectricConstant : TReal
Property  DielectricHeight   : TCoord
Function  I_ObjectAddress    : TPCBObjectHandle;
```

Example

```
Function ConvertDielectricTypeTOString (DT : TDielectricType): String;
Begin
    Result := 'Unknown Type';
    Case DT Of
        eNoDielectric    : Result := 'No Dielectric';
        eCore            : Result := 'Core';
        ePrePreg         : Result := 'PrePreg';
        eSurfaceMaterial : Result := 'Surface Material';
    End;
End;
{.....................................................................}
{.....................................................................}
Function GetLayerInfo(Board : IPCB_Board; Var LayerID : TLayer) : String;
Var
    LayerObj : IPCB_LayerObject;
Begin
    LayerObj := Board.LayerStack.LayerObject[LayerId];
    Result := Layer2String(LayerID) + ', ' + LayerObj.Name + ', ' +
              'Copper' + ', ' + FloatToStr(LayerObj.CopperThickness / 10000) + ', ';
    If LayerObj.Dielectric.DielectricType <> eNoDielectric Then
    Begin
       Result := Result + ConvertDielectricTypeTOString(LayerObj.Dielectric.DielectricType) +
', ' +
                  LayerObj.Dielectric.DielectricMaterial +  ', ' +
FloatToStr(LayerObj.Dielectric.DielectricHeight / 10000) + ', ' +
                  FloatToStr(LayerObj.Dielectric.DielectricConstant);
    End;
    LayerObj := Board.LayerStack.NextLayer(LayerObj);

    If LayerObj <> Nil Then
        LayerID := LayerObj.LayerID
    Else
        LayerID := eNoLayer;
```

```
End;
{.....................................................................}
{.....................................................................}
Procedure FetchLayersInformation;
Var
    Board : IPCB_Board;
    Str   : String;
    Layer : TLayer;
Begin
    Board := PCBServer.GetCurrentPCBBoard;
    If Board = Nil Then Exit;

    Str := 'Layer, Name, Material, Cu Thickness,  Dielectric Material, type, constant, height
' + #13#10;
    Layer := MinLayer;
    Repeat
       Str := Str + GetLayerInfo(Board, Layer) + #13#10;
    Until Layer = eNoLayer;

    // Do what you want with the Str string.
End;
```

See also

IPCB_LayerStack interface

LayerReport script in the **Examples**\Scripts\DelphiScript\PCB\ folder.

IPCB_DrillLayerPair

Overview

The **IPCB_DrillLayerPair** interface represents the paired drill layer for the layer stack up for the PCB document.

Notes

The **IPCB_DrillLayerPair** interface is a standalone interface

The **IPCB_DrillLayerPair** interface is a **DrillLayerPair** property from the **IPCB_Board** interface

Methods

```
Function  I_ObjectAddress       : TPCBObjectHandle;
Function  GetState_Description  : TPCBString;
Function  IsSimilarTo(ADLP : IPCB_DrillLayerPair) : Boolean;
Procedure OrderLayers;
```

Properties

```
Property LowLayer         : TLayer
Property HighLayer        : TLayer
Property StartLayer       : IPCB_LayerObject
Property StopLayer        : IPCB_LayerObject
Property Board            : IPCB_Board
Property PlotDrillDrawing : Boolean
Property PlotDrillGuide   : Boolean
```

Example

```
Var
    PCBBoard     : IPCB_Board;
    i            : Integer;
    LayerPairs   : TStringList;
```

```
    PCBLayerPair : IPCB_DrillLayerPair;
    LowLayerObj  : IPCB_LayerObject;
    HighLayerObj : IPCB_LayerObject;
    LowPos       : Integer;
    HighPos      : Integer;
    LS           : String;
Begin
    PCBBoard := PCBServer.GetCurrentPCBBoard;
    If PCBBoard = Nil Then Exit;

    // Show the current layer
    ShowInfo('Current Layer: ' + Layer2String(PCBBoard.CurrentLayer));

    LayerPairs := TStringList.Create;
    For i := 0 To PCBBoard.DrillLayerPairsCount - 1 Do
    Begin
        PCBLayerPair := PCBBoard.LayerPair[i];
        LowLayerObj  := PCBBoard.LayerStack.LayerObject[PCBLayerPair.LowLayer];
        HighLayerObj := PCBBoard.LayerStack.LayerObject[PCBLayerPair.HighLayer];
        LowPos       := PCBBoard.LayerPositionInSet(SignalLayers + InternalPlanes,
LowLayerObj);
        HighPos      := PCBBoard.LayerPositionInSet(SignalLayers + InternalPlanes,
HighLayerObj);
        If LowPos <= HighPos Then
            LayerPairs.Add(LowLayerObj .Name + ' - ' + HighLayerObj.Name)
        Else
            LayerPairs.Add(HighLayerObj.Name + ' - ' + LowLayerObj .Name);
    End;

    //Display layer pairs.
    LS := '';
    For i := 0 to LayerPairs.Count - 1 Do
        LS := LS + LayerPairs[i] + #13#10;
    ShowInfo('Layer Pairs:'#13#10 + LS);
    LayerPairs.Free;
End;
```

See also

TLayer enumerated values

TCoord value

IPCB_LayerObject interface

IPCB_Board interface

IPCB_InternalPlane

Overview

This **IPCB_InternalPlane** interface represents an existing internal plane used on a PCB document. 16 internal planes are supported, and a net can be assigned to each of these layers or share a power plane between a number of nets by splitting the it into two or more isolated areas.

Pad and via connections to power planes are controlled by the Plane design rules.

The **IPCB_InternalPlane** interface is used by the **IPCB_LayerStack** interface.

Properties

```
Property PullBackDistance         : TCoord
Property NetName                  : TPCBString
Property FirstPreviousSignalLayer : TLayer  //Read only
Property FirstNextSignalLayer     : TLayer  //Read only
```

See also

TLayer enumerated values

TCoord value

IPCB_LayerStack interface

IPCB_MechanicalLayerPairs

Overview

There are 16 general purpose mechanical layers for defining the board layout, placing dimensions on, including fabrication details on, or any other mechanical details the design requires.

The purpose of the **IPCB_MechanicalLayerPairs** Interface is to provide which Mechanical layers are paired to one another.

When a component incorporates objects on one or more Mechanical layers which have been paired, the Layer property of those objects changes when the Layer property of the component is toggled (between the Top and Bottom layers), just like objects on the non-Mechanical layers which have always been paired to one another, to wit the Top and Bottom (copper) layers, the Top and Bottom Overlay layers, the Top and Bottom Paste Mask layers, and the Top and Bottom Solder Mask layers.

Notes

The **IPCB_MechanicalLayerPairs** interface is a MechanicalPairs property of the **IPCB_Board** interface.

Invoke the **Count** method to obtain the number of mechanical layer pairs for the existing PCB document. Indexed mechanical layer pairs which is a **LayerPair[]** property can be returned. This property returns a **TMechanicalLayerPair** record of two PCB layers.

Methods

```
Procedure Clear;
Function  Count                          : Integer;
Function  AddPair    (Layer1,
                      Layer2 : TLayer)   : Integer;
Function  RemovePair (Layer1,
                      Layer2 : TLayer)   : Boolean;
Function  PairDefined(Layer1,
                      Layer2 : TLayer)    : Boolean;
Function  LayerUsed   (Layer  : TLayer)   : Boolean;
Function  FlipLayer(Var L  : TLayer)      : Boolean;

Procedure Import_FromParameters(Params : PChar);
Procedure Export_ToParameters  (Params : PChar);
```

Properties

```
LayerPair [I : Integer] : TMechanicalLayerPair
```

Example

```
Var
    Board   : IPCB_Board;
    Layer   : TLayer;
    LS      : IPCB_LayerStack;
    LObject : IPCB_LayerObject;
    S       : TPCBString;
Begin
    Board := PCBServer.GetCurrentPCBBoard;
```

```
    If Board = Nil Then Exit;
    LS := Board.LayerStack;
    If LS = Nil Then Exit;
    S := '';
    For Layer := eMechanical1 to eMechanical16 Do
    Begin
        LObject := LS.LayerObject[Layer];
        // If a mechanical layer is not enabled (as per the Board Layers and
        //   Colors dialog) then this layer cannot be displayed nor have any objects on it.
        If Not (LObject.MechanicalLayerEnabled) Then
            S := S + LObject.Name + ' is NOT enabled (thus it cannot be displayed nor have any
objects on it).' + #13
        Else
        Begin
           If (LObject.IsDisplayed[Board] = True) and (LObject.UsedByPrims) Then
               S := S + LObject.Name + ' is displayed and there are objects on it.' + #13;
           If (LObject.IsDisplayed[Board] = True) and Not (LObject.UsedByPrims) Then
               S := S+ LObject.Name + ' is displayed and there are NO objects on it.' + #13;
           If (LObject.IsDisplayed[Board] = False) and (LObject.UsedByPrims) Then
                S := S + LObject.Name + ' is NOT displayed and there are objects on it.' +
#13;
           If (LObject.IsDisplayed[Board] = False) and Not (LObject.UsedByPrims) Then
              S := S + LObject.Name + ' is NOT displayed and there are NO objects on it.' +
#13;
        End;
    End;
    ShowMessage(S);
End;
```

See also

TLayer enumerated values

TMechanicalLayerPair values

IPCB_LayerStack interface

PCB Options Interfaces

IPCB_AbstractOptions

Overview

The IPCB_AbstractOptions interface is the base interface for other options related interfaces such as SystemOptions and InteractiveRoutingOptions through IPCB_ServerInterface. These option objects are global objects created by the PCB Server.

The other OutputOptions, ECOOptions, GerberOptions, PrinterOptions and PlacerOptions interfaces are referenced through IPCB_Board interface.

Notes

Ancestor interface for ECO Options, Output Options, Gerber Options, Printer Options, Advanced Placer Options, SystemOptions, Design Rule Checker Options, SpecctraRouter Options and Interactive Routing options interfaces.

Methods

```
Procedure Import_FromParameters          (DisplayUnit : TUnit;
                                          Parameters  : PChar);
Procedure Export_ToParameters            (Parameters  : PChar);
Procedure Import_FromParameters_Version4 (DisplayUnit : TUnit;
```

```
                                          Parameters  : PChar);
Procedure Export_ToParameters_Version4   (Parameters  : PChar);
Procedure Import_FromParameters_Version3 (DisplayUnit : TUnit;
                                          Parameters  : PChar);
Procedure Export_ToParameters_Version3   (Parameters  : PChar);
Function  I_ObjectAddress : TPCBObjectHandle;
```

Properties

```
OptionsObjectID : TOptionsObjectId
```

See also

IPCB_ECOOptions interface

IPCB_OutputOptions interface

IPCB_GerberOptions interface

IPCB_PrinterOptions interface

IPCB_AdvancedPlacerOptions interface

IPCB_SystemOptions interface

IPCB_DesignRuleCheckerOptions interface

IPCB_SpecctraRouterOptions interface

IPCB_InteractiveRoutingOptions interface

IPCB_AdvancedPlacerOptions

Overview

The IPCB_AdvancedPlacerOptions interface represents the options for the placement application.

Notes

Derived from IPCB_AbstractOptions interface

IPCB_ Properties

```
Property PlaceLargeClear       : TCoord
Property PlaceSmallClear       : TCoord
Property PlaceUseRotation      : Boolean
Property PlaceUseLayerSwap     : Boolean
Property PlaceByPassNet1       : TPCBString
Property PlaceByPassNet2       : TPCBString
Property PlaceUseAdvancedPlace : Boolean
Property PlaceUseGrouping      : Boolean
```

See also

IPCB_AbstractOptions interface

IPCB_DesignRuleCheckerOptions

Overview

The IPCB_DesignRuleCheckerOptions interface deals with the DRC options.

Notes

Derived from IPCB_AbstractOptions interface

IPCB_DesignRuleCheckerOptions Methods

```
Procedure Export_ToParameters_GeneralOptions        (Parameters : PChar);
Procedure Export_ToParameters_RulesToCheck          (Parameters : PChar);
Procedure Export_ToParameters_RulesToCheck_Version3 (Parameters : PChar);
Procedure Import_FromParameters_GeneralOptions      (Parameters : PChar);
Procedure Import_FromParameters_RulesToCheck        (Parameters : PChar);
```

IPCB_DesignRuleCheckerOptions Properties

```
Property OnLineRuleSetToCheck           : TRuleSet
Property DoMakeDRCFile                  : Boolean
Property DoMakeDRCErrorList             : Boolean
Property DoSubNetDetails                : Boolean
Property RuleSetToCheck                 : TRuleSet
Property ReportFilename                 : TPCBString
Property ExternalNetListFileName        : TPCBString
Property CheckExternalNetList           : Boolean
Property MaxViolationCount              : Integer
Property InternalPlaneWarnings          : Boolean
Property VerifyShortingCopper           : Boolean
```

See also

IPCB_AbstractOptions interface

IPCB_ECOOptions

Overview

The IPCB_ECOOptions represents an existing Engineering Change Order options object in a PCB document.

Notes

Derived from IPCB_AbstractOptions interface

Properties

```
Property ECOIsActive : Boolean
Property ECOFileName : TString
```

See also

IPCB_AbstractOptions interface

IPCB_GerberOptions

Overview

The tolerance range used when matching apertures for each item in the plots. If no exact match for an item is available in the current aperture list, the software checks to see if a larger aperture exists within this tolerance range and uses it instead.

If no suitable aperture exists within the tolerance range, the software will attempt to "paint" with a larger aperture to create the required shape. This requires that a suitable larger aperture is available, and that this aperture can be used for "painting".

Note: Match tolerances are normally only used when you are targeting a vector photoplotter, which require a fixed, or supplied aperture file. They will not be required if the apertures have been created from the PCB. If match tolerances are not required they should be left at the default of 0.005 mil.

Notes

Derived from IPCB_AbstractOptions interface

Properties

```
Property SortOutput          : Boolean
Property UseSoftwareArcs     : Boolean
Property CenterPhotoPlots    : Boolean
Property EmbedApertures      : Boolean
Property Panelize            : Boolean
Property G54                 : Boolean
Property PlusTol             : TCoord
Property MinusTol            : TCoord
Property FilmSizeX           : TCoord
Property FilmSizeY           : TCoord
Property BorderSize          : TCoord
Property AptTable            : TPCBString
```

```
Property MaxAperSize         : TCoord
Property ReliefShapesAllowed : Boolean
Property PadsFlashOnly       : Boolean
Property GerberUnits         : Integer
Property GerberDecs          : Integer
```

See also

IPCB_AbstractOptions interface

IPCB_InteractiveRoutingOptions

Overview

The IPCB_InteractiveRoutingOptions interface represents the options for the interactive routing module in the PCB editor.

Notes

Derived from IPCB_AbstractOptions interface

Methods

```
Procedure Export_ToParameters_GeneralOptions(Parameters : PChar);
Procedure Export_ToParameters_LayerOptions  (Parameters : PChar);
Procedure Export_ToParameters_LayerOptions_Version3(Parameters : PChar);
```

Properties

```
PlaceTrackMode    : TPlaceTrackMode
OldTrackDrawLayer : TLayer
TrackArcX         : TCoord
TrackArcY         : TCoord
TrackArcRadius    : TCoord
TrackArcAngle1    : TCoord
TrackArcAngle2    : TCoord
OldTrackArcX      : TCoord
OldTrackArcY      : TCoord
OldTrackArcRadius : TCoord
OldTrackArcAngle1 : TCoord
OldTrackArcAngle2 : TCoord
OldTrackDrawSize  : TCoord
OldMidx           : TCoord
OldMidy           : TCoord
OldCx             : TCoord
OldCy             : TCoord
EndLineX          : TCoord
EndLineY          : TCoord
Midx              : TCoord
MidY              : TCoord
StartX            : TCoord
StartY            : TCoord
Beginx            : TCoord
Beginy            : TCoord
```

See also

IPCB_AbstractOptions interface

IPCB_MechanicalLayerPairs

Overview

There are 16 general purpose mechanical layers for defining the board layout, placing dimensions on, including fabrication details on, or any other mechanical details the design requires.

The purpose of the **IPCB_MechanicalLayerPairs** Interface is to provide which Mechanical layers are paired to one another.

When a component incorporates objects on one or more Mechanical layers which have been paired, the Layer property of those objects changes when the Layer property of the component is toggled (between the Top and Bottom layers), just like objects on the non-Mechanical layers which have always been paired to one another, to wit the Top and Bottom (copper) layers, the Top and Bottom Overlay layers, the Top and Bottom Paste Mask layers, and the Top and Bottom Solder Mask layers.

Notes

The **IPCB_MechanicalLayerPairs** interface is a MechanicalPairs property of the **IPCB_Board** interface.

Invoke the **Count** method to obtain the number of mechanical layer pairs for the existing PCB document. Indexed mechanical layer pairs which is a **LayerPair[]** property can be returned. This property returns a **TMechanicalLayerPair** record of two PCB layers.

Methods

```
Procedure Clear;
Function  Count                          : Integer;
Function  AddPair    (Layer1,
                      Layer2 : TLayer)   : Integer;
Function  RemovePair (Layer1,
                      Layer2 : TLayer)   : Boolean;
Function  PairDefined(Layer1,
                      Layer2 : TLayer)    : Boolean;
Function  LayerUsed   (Layer  : TLayer)   : Boolean;
Function  FlipLayer(Var L  : TLayer)      : Boolean;

Procedure Import_FromParameters(Params : PChar);
Procedure Export_ToParameters  (Params : PChar);
```

Properties

```
LayerPair [I : Integer] : TMechanicalLayerPair
```

Example

```
Var
    Board   : IPCB_Board;
    Layer   : TLayer;
    LS      : IPCB_LayerStack;
    LObject : IPCB_LayerObject;
    S       : TPCBString;
Begin
    Board := PCBServer.GetCurrentPCBBoard;
    If Board = Nil Then Exit;
    LS := Board.LayerStack;
    If LS = Nil Then Exit;
    S := '';
    For Layer := eMechanical1 to eMechanical16 Do
    Begin
        LObject := LS.LayerObject[Layer];
        // If a mechanical layer is not enabled (as per the Board Layers and
        //   Colors dialog) then this layer cannot be displayed nor have any objects on it.
        If Not (LObject.MechanicalLayerEnabled) Then
            S := S + LObject.Name + ' is NOT enabled (thus it cannot be displayed nor have any
objects on it).' + #13
```

```
        Else
        Begin
           If (LObject.IsDisplayed[Board] = True) and (LObject.UsedByPrims) Then
               S := S + LObject.Name + ' is displayed and there are objects on it.' + #13;
           If (LObject.IsDisplayed[Board] = True) and Not (LObject.UsedByPrims) Then
               S := S+ LObject.Name + ' is displayed and there are NO objects on it.' + #13;
           If (LObject.IsDisplayed[Board] = False) and (LObject.UsedByPrims) Then
                S := S + LObject.Name + ' is NOT displayed and there are objects on it.' +
#13;
           If (LObject.IsDisplayed[Board] = False) and Not (LObject.UsedByPrims) Then
              S := S + LObject.Name + ' is NOT displayed and there are NO objects on it.' +
#13;
        End;
    End;
    ShowMessage(S);
End;
```

See also

TLayer enumerated values

TMechanicalLayerPair values

IPCB_LayerStack interface

IPCB_OutputOptions

Overview

The IPCB_OutputOptions interface represents the options for the generation of PCB output such as including mechanical layers in plots etc.

Notes

Derived from IPCB_AbstractOptions interface

Methods

```
Procedure Import_FromParameters_GeneralOptions (DisplayUnit : TUnit;
                                                Parameters  : PChar);
Procedure Import_FromParameters_LayerOptions   (Parameters  : PChar);
Procedure Import_FromParameters_LayerOptions_Version3 (Parameters  : PChar);
Procedure Export_ToParameters_GeneralOptions   (Parameters  : PChar);
Procedure Export_ToParameters_LayerOptions     (Parameters  : PChar);
Procedure Export_ToParameters_LayerOptions_Version3   (Parameters  : PChar);
```

Properties

```
Property DrillGuideHoleSize              : TCoord
Property DrillDrawSymbolSize             : TCoord
Property DrillSymbolKind                 : TDrillS
Property MultiLayerOnPadMaster           : Boolean
Property TopLayerOnPadMaster             : Boolean
Property BottomLayerOnPadMaster          : Boolean
Property IncludeViasInSolderMask         : Boolean

Property IncludeUnconnectedPads          : Boolean
Property PlotLayer [PL : TPlotLayer]     : Boolean
Property FlipLayer [PL : TPlotLayer]     : Boolean
```

See also

IPCB_AbstractOptions interface

IPCB_PrinterOptions

Overview

The **IPCB_PrinterOptions** interface represents the Printer options setup in the PCB Editor server.

Notes

Derived from IPCB_AbstractOptions interface

Methods

```
Procedure Import_FromParameters_GeneralOptions        (DisplayUnit : TUnit;
                                                       Parameters  : PChar);
Procedure Import_FromParameters_LayerOptions          (Parameters  : PChar);
Procedure Import_FromParameters_LayerOptions_Version3 (Parameters  : PChar);
Procedure Export_ToParameters_GeneralOptions          (Parameters  : PChar);
Procedure Export_ToParameters_LayerOptions            (Parameters  : PChar);
Procedure Export_ToParameters_LayerOptions_Version3   (Parameters  : PChar);
```

Properties

```
Property Device           : TPCBString
Property Driver           : TPCBString
Property OutPut           : TPCBString
Property OutputDriverType : TOutputDriverType
Property ShowHoles        : Boolean
Property ScaleToFitPage   : Boolean
Property UsePrinterFonts  : Boolean
Property UseSoftwareArcs  : Boolean
Property BatchType        : TPrinterBatch
Property CompositeType    : TPrinterComposite
Property cBorderSize      : TCoord
Property Scale            : TGeometry
Property XCorrect         : TGeometry
Property YCorrect         : TGeometry
Property PlotMode [OId : TObjectId]            : TDrawMode
Property PlotPadNets      : Boolean
Property PlotPadNumbers   : Boolean
Property PlotterScale     : TGeometry
Property PlotterXCorrect  : TGeometry
Property PlotterYCorrect  : TGeometry
Property PlotterXOffset   : TCoord
Property PlotterYOffset   : TCoord
Property PlotterShowHoles : Boolean
Property PlotterUseSoftwareArcs : Boolean
Property PlotterWaitBetweenSheets : Boolean
Property PlotterOutputPort                     : TOutputPort
Property PlotterLanguage                       : TPlotterLanguage
Property PlotterPens [PId : Integer]           : TPlotterPen
Property CompositePlotMonoLayers  [L : TLayer] : TColor
Property CompositePlotColorLayers [L : TLayer] : TColor
Property CompositePlotLayers      [L : TLayer] : Boolean
Property CompositePlotPens        [L : TLayer] : Integer
```

See also

IPCB_AbstractOptions interface

IPCB_SpecctraRouterOptions

Overview

The IPCB_SpecctraRouterOptions interface represents the options for the Specctra Router application.

Notes

Derived from IPCB_AbstractOptions interface

Properties

```
Property Setback      [I : Integer]      : TCoord
Property DoSetback    [I : Integer]      : Boolean
Property DoBus                           : Boolean
Property BusDiagonal                     : Boolean
Property DoQuit                          : Boolean
Property WireGrid                        : TReal
Property ViaGrid                         : TReal
Property DoSeedVias                      : Boolean
Property NoConflicts                     : Boolean
Property AdvancedDo                      : Boolean
Property ReorderNets                     : Boolean
Property ProtectPreRoutes                : Boolean
Property SeedViaLimit                    : TCoord
Property RoutePasses                     : Integer
Property CleanPasses                     : Integer
Property FilterPasses                    : Integer
Property LayerCost    [L : TLayer]       : TCCTCost
Property LayerWWCost  [L : TLayer]       : TCCTCost
Property WwCost                          : TCCTCost
Property CrossCost                       : TCCTCost
Property ViaCost                         : TCCTCost
Property OffGridCost                     : TCCTCost
Property OffCenterCost                   : TCCTCost
Property SideExitCost                    : TCCTCost
Property SqueezeCost                     : TCCTCost
Property LayerTax     [L : TLayer]       : TCCTTax
Property LayerWWTax   [L : TLayer]       : TCCTTax
Property WwTax                           : TCCTTax
Property CrossTax                        : TCCTTax
Property ViaTax                          : TCCTTax
Property OffGridTax                      : TCCTTax
Property OffCenterTax                    : TCCTTax
Property SideExitTax                     : TCCTTax
Property SqueezeTax                      : TCCTTax
Property DoCritic                        : Boolean
Property DoMiter                         : Boolean
Property DoRecorner                      : Boolean
Property DoFanout                        : Boolean
Property FoPower                         : Boolean
Property FoSignal                        : Boolean
Property FoIn                            : Boolean
Property FoOut                           : Boolean
Property FoVias                          : Boolean
```

```
Property FoPads                          : Boolean
Property FoPasses                        : Integer
Property ForceVias                       : Boolean
Property DoSpread                        : Boolean
Property SortKind                        : TCCTSort
Property SortDir                         : TCCTSortDir
Property Adv10                           : Boolean
Property Dfm10                           : Boolean
Property Hyb10                           : Boolean
Property SpVersion                       : Integer
Property MinimizePads                    : Boolean
```

See also

IPCB_AbstractOptions interface

IPCB_SystemOptions

Overview

The **IPCB_SystemOptions** interface represents the global system options in the PCB Editor server.

Notes

Derived from IPCB_AbstractOptions interface

Methods

```
Procedure Import_FromIniFile;
Procedure Export_ToIniFile;
Procedure AddComponentMapping (Value : TComponentTypeMapping);
```

Properties

```
{DisplayOptions}
Property UndoRedoStackSize                 : Integer
Property SingleLayerMode                   : Boolean
Property LockPreRoutes                     : Boolean
Property DrawMode  [OId : TObjectID]       : TDrawMode
Property FromTosDisplayMode                : TFromToDisplayMode
Property PadTypesDisplayMode               : TFromToDisplayMode
Property DraftTrackThreshold               : TCoord
Property CleanRedraw                       : Boolean
Property ShowInvisibleObjects              : Boolean
Property DisplaySpecialStrings             : Boolean
Property RedrawLayerOnToggle               : Boolean
Property UseCurrentForMultiLayer           : Boolean
Property UseNetColorForHighlight           : Boolean
Property HighlightFull                     : Boolean
Property ShowAllPrimitivesInHighlightedNets : Boolean
Property UseTransparent                    : Boolean
Property UseDithered                       : Boolean
Property ShowPadNets                       : Boolean
Property ShowPadNumbers                    : Boolean
Property ShowTestPoints                    : Boolean
Property ShowViaNets                       : Boolean
Property ShowStatusInfo                    : Boolean
Property ShowStatusInterval                : Integer
Property BoardCursorType                   : TGraphicsCursor
```

```
Property TextToRectSize                     : Integer
Property AutoPan                            : Boolean
Property LayerDrawingOrder [I : Integer]    : TLayer

{Paste Options}
Property  Paste_InSameClass       : Boolean
Property  Paste_OnSameLayer       : Boolean
Property  Paste_InSameNet         : Boolean
Property  Paste_HasSameDesignator : Boolean

{PlaceArray Options}
Property RepeatRotateItem                   : Boolean
Property RepeatCircular                     : Boolean
Property RepeatDegrees                      : TGeometry
Property RepeatX                            : TGeometry
Property RepeatY                            : TGeometry
Property RepeatXUnit                        : TUnit
Property RepeatYUnit                        : TUnit
Property RepeatCountDefault                 : Integer
Property RepeatInc                          : TPCBString

{Com Port Options}
Property Com1Parameters    : TSerialParameters
Property Com2Parameters    : TSerialParameters
Property Com3Parameters    : TSerialParameters
Property Com4Parameters    : TSerialParameters

{Netlist load options}
Property CheckPatterns          : Boolean
Property CheckComments          : Boolean
Property NetlistReportFile      : Boolean
Property NetlistReportDialog    : Boolean
Property DeleteUnconnectedComps : Boolean
Property DeleteUnconnectedPrims : Boolean

{Misc System Options}
Property GlobalEditIncludeArcsWithTracks : Boolean
Property ValidateOnLoad                  : Boolean
Property SaveDefs                        : Boolean
Property DoOnlineDRC                     : Boolean
Property LoopRemoval                     : Boolean
Property UseSmartTrackEnds               : Boolean
Property DeleteDeadEnds                  : Boolean
Property QuestionDelete                  : Boolean
Property QuestionGlobalChange            : Boolean
Property QuestionDrag                    : Boolean
Property NearestComponent                : Boolean
Property RemoveDuplicatesOnOutput        : Boolean
Property DuplicateDesignatorsAllowed     : Boolean
```

```
Property AutoVia                          : Boolean
Property SnapToCentre                     : Boolean
Property ReportsCSV                       : Boolean
Property ClickClearsSelection             : Boolean
Property HoldShiftToSelectObjectId [OId : TObjectID] : Boolean
Property MustHoldShiftToSelect            : Boolean
Property DoubleClickRunsInspector         : Boolean
Property DefaultPrimsPermanent            : Boolean
Property DragMode                         : TPcbDragMode
Property RotationStep                     : TAngle
Property OnlySelectVisible                : Boolean
Property PlaceShoveDepth                  : Integer
Property LayerColors[L : TLayer]          : TColor
Property AutoPanMode                      : TAutoPanMode
Property AutoPanSmallStep                 : Integer
Property AutoPanLargeStep                 : Integer
Property AutoPanUnit                      : TAutoPanUnit
Property AutoPanSpeed                     : Integer
Property InteractiveRouteMode             : TInteractiveRouteMode
Property PolygonThreshold                 : Integer
Property PolygonRepour                    : TPolygonRepourMode
Property PlowThroughPolygons              : Boolean
Property ProtectLockedPrimitives          : Boolean
Property ConfirmSelectionMemoryClear      : Boolean
Property ComponentMoveKind                : TComponentMoveKind
Property SameNamePadstackReplacementMode : TSameNamePadstackReplacementMode
Property PadstackUpdateFromGlobalsOnLoad : TSameNamePadstackReplacementMode
Property PlaneDrawMode           : TPlaneDrawMode
Property BoardAreaColor          : TColor
Property BoardLineColor          : TColor
Property SheetAreaColor          : TColor
Property SheetLineColor          : TColor
Property WorkspaceColor1         : TColor
Property WorkspaceColor2         : TColor

DefaultTTFont
PadViaFontName
PadViaFontColor
PadViaFontStyle
PadViaMinFontSize
PadViaFontBkColor
MinPadViaObjectSizeInPixels
```

Example

```
Var
    PCBSystemOptions : IPCB_SystemOptions;
Begin
    PCBSystemOptions := PCBServer.SystemOptions;
    If PCBSystemOptions = Nil Then Exit;
    If PcbSystemOptions.BoardCursorType = eCurShapeCross90 Then
```

```
        PcbSystemOptions.BoardCursorType := eCurShapeBigCross
    Else If PcbSystemOptions.BoardCursorType = eCurShapeBigCross Then
        PcbSystemOptions.BoardCursorType := eCurShapeCross45
    Else
        PcbSystemOptions.BoardCursorType := eCurShapeCross90;
End.
```

See also

IPCB_AbstractOptions interface

TPCBDragMode enumerated values

TGraphicsCursor enumerated values

TComponentTypeMapping enumerated values

TComponentMoveKind enumerated values

TPolygonRepourMode enumerated values

TSameNamePadstackReplacementMode enumerated values

TPlaneDrawMode enumerated values

TAutoPanUnit enumerated values

TAutoPanMode enumerated values

TInteractiveRouteMode enumerated values

PCB Design Objects Interfaces

A PCB design object on a PCB document is represented by its interface. An interface represents an existing object in memory and its properties and methods can be invoked.

A PCB design object is basically a primitive or a group object. A primitive can be a track or an arc object. A group object is an object that is composed of child objects. For example a board outline or a component is a group object.

Since many design objects are descended from ancestor interfaces and thus the ancestor methods and properties are also available to use.

For example the IPCB_Text interface is inherited from an immediate IPCB_RectangularPrimitive interface and in turn inherited from the IPCB_Primitive interface. If you check the IPCB_Text entry in this online help you will see the following information;

The IPCB_Text Interface hierarchy is as follows;

IPCB_Primitive

IPCB_RectangularPrimitive

IPCB_Text

and so on.

This PCB Design Objects section is broken up into several categories- Primitives, Dimensions, Group Objects and Rectangular Objects.

Primitives include arcs, embedded objects, fills, fromtos, pads, nets, tracks, vias, violations, object classes and connections.

Dimensions include Linear, Angular, Radial, Leader, Datum, Baseline, Center, Linear Diameter and Radial Diameter objects

Group objects include board outlines, coordinates, components, polygons, library components (footprints) and nets.

Rectangular objects include text objects.

See also

IPCB_Primitive interface

IPCB_Group interface

IPCB_Arc

IPCB_ObjectClass

IPCB_Pad

IPCB_Via

IPCB_Track

IPCB_Embedded
IPCB_Violation
IPCB_Text
IPCB_Fill
IPCB_Coordinate
IPCB_Dimension
IPCB_Component
IPCB_Polygon
IPCB_Net
IPCB_LibComponent

IPCB_Primitive Interface

Overview

The **IPCB_Primitive** interface is the ancestor interface object for all other PCB interface objects and therefore the methods and properties declared in the **IPCB_Primitive** interface are also declared in the descendant interfaces.

Notes

Every PCB object has an unique object address stored in a PCB design database for that document this object resides on. Each PCB object address has the **TPCBObjectHandle** type.

Every existing PCB design object on a PCB document has the Board owner which represents the specific board document.

Each existing PCB design object on a PCB document has Query Rule Properties which can be queried.

A primitive has a bounding rectangle which encapsulates the region of the primitive. There are two other bounding rectangles which are for selection and for painting (refreshing and updating).

IPCB_Primitive methods

```
GetState_Board
GetState_ObjectId
GetState_Layer
GetState_Selected
SetState_Selected
GetState_IsPreRoute
SetState_IsPreRoute
GetState_InSelectionMemory
SetState_InSelectionMemory
GetState_PadCacheRobotFlag
SetState_PadCacheRobotFlag
GetState_Enabled
SetState_Enabled
GetState_Enabled_Direct
SetState_Enabled_Direct
GetState_Enabled_vNet
SetState_Enabled_vNet
GetState_Enabled_vPolygon
SetState_Enabled_vPolygon
GetState_Enabled_vComponent
SetState_Enabled_vComponent
GetState_Enabled_vCoordinate
SetState_Enabled_vCoordinate
GetState_Enabled_vDimension
```

IPCB_Primitive properties

```
Board
ObjectId
Layer
Index
Selected
IsPreRoute
InSelectionMemory
PadCacheRobotFlag
Enabled
Enabled_Direct
Enabled_vNet
Enabled_vPolygon
Enabled_vComponent
Enabled_vCoordinate
Enabled_vDimension
Used
DRCError
MiscFlag1
MiscFlag2
MiscFlag3
EnableDraw
Moveable
UserRouted
TearDrop
```

SetState_Enabled_vDimension
GetState_Used
SetState_Used
GetState_DRCError
SetState_DRCError
GetState_MiscFlag1
SetState_MiscFlag1
GetState_MiscFlag2
SetState_MiscFlag2
GetState_MiscFlag3
SetState_MiscFlag3
GetState_EnableDraw
SetState_EnableDraw
GetState_Moveable
SetState_Moveable
GetState_UserRouted
SetState_UserRouted
GetState_TearDrop
SetState_TearDrop
GetState_IsTenting
SetState_IsTenting
GetState_IsTenting_Top
SetState_IsTenting_Top
GetState_IsTenting_Bottom
SetState_IsTenting_Bottom
GetState_IsTestPoint_Top
SetState_IsTestPoint_Top
GetState_IsTestPoint_Bottom
SetState_IsTestPoint_Bottom
GetState_IsKeepout
SetState_IsKeepout
GetState_AllowGlobalEdit
SetState_AllowGlobalEdit
GetState_PolygonOutline
SetState_PolygonOutline
GetState_InBoard
SetState_InBoard
GetState_InPolygon
SetState_InPolygon
GetState_InComponent
SetState_InComponent
GetState_InNet
SetState_InNet
GetState_InCoordinate
SetState_InCoordinate
GetState_InDimension
SetState_InDimension
GetState_IsElectricalPrim
SetState_Board

IsTenting
IsTenting_Top
IsTenting_Bottom
IsTestpoint_Top
IsTestpoint_Bottom
IsKeepout
AllowGlobalEdit
PolygonOutline
InBoard
InPolygon
InComponent
InNet
InCoordinate
InDimension
IsElectricalPrim
ObjectIDString
Identifier
Descriptor
Detail
PowerPlaneConnectStyle
ReliefConductorWidth
ReliefEntries
ReliefAirGap
PasteMaskExpansion
SolderMaskExpansion
PowerPlaneClearance
PowerPlaneReliefExpansion
Net
Component
Polygon
Coordinate
Dimension
ViewableObjectID
UnionIndex

```
SetState_Layer
GetState_ObjectIDString
GetState_Identifier
GetState_DescriptorString
GetState_DetailString
GetState_Index
SetState_Index

GetState_UnionIndex
SetState_UnionIndex

GetState_PowerPlaneConnectStyle
GetState_ReliefConductorWidth
GetState_ReliefEntries
GetState_ReliefAirGap
GetState_PasteMaskExpansion
GetState_SolderMaskExpansion
GetState_PowerPlaneClearance
GetState_PowerPlaneReliefExpansion
GetState_Net
GetState_Component
GetState_Polygon
GetState_Coordinate
GetState_Dimension
GetState_ViewableObjectID
SetState_Net
SetState_Component
SetState_Polygon
SetState_Coordinate
SetState_Dimension

I_ObjectAddress
BoundingRectangle
BoundingRectangleForSelection
BoundingRectangleForPainting
IsHidden
IsFreePrimitive
IsSaveable
AddPCBObject
RemovePCBObject

MoveByXY
MoveToXY
RotateBy
FlipXY
Mirror
SwapLayerPairs
GraphicallyInvalidate
```

```
BeginModify
EndModify
CancelModify

Export_ToParameters
RequiredParamterSpace
```

See also

PCB Design Objects

Methods

BeginModify method

(IPCB_Primitive interface)

Syntax

```
Procedure BeginModify;
```

Description

Example

See also

IPCB_Primitive interface

BoundingRectangle method

(IPCB_Primitive interface)

Syntax

```
Function BoundingRectangle : TCoordRect;
```

Description

This function returns the coordinates of the bounding rectangle that encapsulates the design object on a PCB document. There are other two bounding rectangle methods.

Example

```
Var
    R : TCoordRect;
Begin
    // check for comment / name objects
    If P.ObjectId <> eTextObject Then
    Begin
        R := P.BoundingRectangle;
        If R.left   < MinX Then MinX := R.left;
        If R.bottom < MinY Then MinY := R.bottom;
        If R.right  > MaxX Then MaxX := R.right;
        If R.top    > MaxY Then MaxY := R.top;
    End;
End;
```

See also

IPCB_Primitive interface

TCoordRect type

BoundingRectangle script from `\Examples\Scripts\Delphiscript Scripts\Pcb\` folder.

BoundingRectangleForSelection method

(IPCB_Primitive interface)

Syntax

```
Function BoundingRectangleForSelection : TCoordRect;
```

Description

The bounding rectangle of a design object used for selection is a bit bigger than the bounding rectangle of a design object itself.

Example

See also

IPCB_Primitive interface

BoundingRectangleForPainting method

(IPCB_Primitive interface)

Syntax

```
Function BoundingRectangleForPainting : TCoordRect;
```

Description

The bounding rectangle of a design object for painting is potentially the largest of all bounding rectangles because for example a component can have comment and designator objects as well.

Example

See also

IPCB_Primitive interface

CancelModify method

(IPCB_Primitive interface)

Syntax

```
Procedure CancelModify;
```

Description

Example

See also

IPCB_Primitive interface

EndModify method

(IPCB_Primitive interface)

Syntax

```
Procedure EndModify;
```

Description

Example

See also

IPCB_Primitive interface

FlipXY method

(IPCB_Primitive interface)

Syntax

```
Procedure FlipXY (Axis : TCoord;MirrOp : TMirrorOperation);
```

Description

This procedure flips the object about the axis depending on Axis and MirrOp parameters.

Example

See also

IPCB_Primitive interface

TMirrorOperation type

GraphicallyInvalidate method

(IPCB_Primitive interface)

Syntax

```
Procedure GraphicallyInvalidate;
```

Description

This procedure renders the object graphically invalidate which forces a system graphical update /refresh on the PCB document.

Example

See also

IPCB_Primitive interface

I_ObjectAddress method

(IPCB_Primitive interface)

Syntax

```
Function I_ObjectAddress : TPCBObjectHandle;
```

Description

This function returns the true pointer value of the object interface of a design object.

Note

The IPCB_ServerInterface.**SendMessageToRobots** method needs the **I_ObjectAddress** parameter which is the handle of a design object.

Example

```
//Notify PCB that the fill object is going to be changed.
PCBServer.SendMessageToRobots(
        Fill.I_ObjectAddress,
        c_Broadcast,
        PCBM_BeginModify ,
        c_NoEventData);
```

See also

IPCB_Primitive interface

IsFreePrimitive method

(IPCB_Primitive interface)

Syntax

```
Function IsFreePrimitive : Boolean;
```

Description

This function determines whether the object is a free primitive (not connected to a net) or just a standalone object.

Example

See also

IPCB_Primitive interface

IsHidden method

(IPCB_Primitive interface)

Syntax

```
Function IsHidden : Boolean;
```

Description

This function determines whether this object is hidden from view or not.

Example

See also

IPCB_Primitive interface

IsSaveable method

(IPCB_Primitive interface)

Syntax

```
Function IsSaveable (AVer : TAdvPCBFileFormatVersion) : Boolean;
```

Description

This function determines whether this particular object can be saved in a specified file format version according to the **TAdvPCBFileFormatVersion** type.

Example

See also

IPCB_Primitive interface

TAdvPCBFileFormatVersion type

Mirror method

(IPCB_Primitive interface)

Syntax

```
Procedure Mirror (Axis : TCoord;MirrOp : TMirrorOperation);
```

Description

This procedure mirrors the design object across the axis depending on the mirror operation.

Example

See also

IPCB_Primitive interface

TMirrorOperation type

MoveByXY method

(IPCB_Primitive interface)

Syntax

```
Procedure MoveByXY (AX, AY : TCoord);
```

Description

This procedure moves the design object by an offset in horizontal and vertical directions specified by the AX and AY parameters.

Example

```
//Move the object by a specified offset
XStep := DistanceStep * Cos(AngleStep);
```

```
YStep := DistanceStep * Sin(AngleStep);
PcbObject.MoveByXY(XStep,YStep);
```

See also

IPCB_Primitive interface

MoveToXY method

(IPCB_Primitive interface)

Syntax

```
Procedure MoveToXY (AX, AY : TCoord);
```

Description

This procedure moves the design object to a new location specified by the AX and AY parameters.

Example

See also

IPCB_Primitive interface

RotateBy method

(IPCB_Primitive interface)

Syntax

```
Procedure RotateBy (Angle : TAngle);
```

Description

Example

See also

IPCB_Primitive interface

SwapLayerPairs method

(IPCB_Primitive interface)

Syntax

```
Procedure SwapLayerPairs;
```

Description

This procedure swaps the current layer pair that the PCB design object (vias and pads only) has.

Example

See also

IPCB_Primitive interface

GetState and SetState Methods

GetState_AllowGlobalEdit method

(IPCB_Primitive interface)

Syntax

```
Function GetState_AllowGlobalEdit : Boolean;
```

Description

Example

See also

IPCB_Primitive interface

GetState_Board method

(IPCB_Primitive interface)

Syntax

```
Function GetState_Board : IPCB_Board;
```

Description

The Board property determines the PCB document that the object itself is associated with. This method is used by the Board property.

Example

See also

IPCB_Primitive interface

GetState_Component method

(IPCB_Primitive interface)

Syntax

```
Function GetState_Component : IPCB_Component;
```

Description

This property determines whether the object itself is associated with the component or not. This method retrieves the Component and is used in the Component property.

Example

See also

IPCB_Primitive interface

GetState_Coordinate method

(IPCB_Primitive interface)

Syntax

```
Function GetState_Coordinate : IPCB_Coordinate;
```

Description

This property determines whether the object itself is associated with the coordinate object or not. This method retrieves the coordinate object and is used in the Coordinate property.

Example

See also

IPCB_Primitive interface

GetState_DescriptorString method

(IPCB_Primitive interface)

Syntax

```
Function GetState_DescriptorString : TPCBString;
```

Description

Example

See also

IPCB_Primitive interface

GetState_DetailString method

(IPCB_Primitive interface)

Syntax

```
Function GetState_DetailString : TPCBString;
```

Description

Example

See also

IPCB_Primitive interface

GetState_Dimension method

(IPCB_Primitive interface)

Syntax

```
Function GetState_Dimension : IPCB_Dimension;
```

Description

This property determines whether the object itself is associated with the dimension object or not. This method retrieves the Dimension and is used in the Dimension property.

Example

See also

IPCB_Primitive interface

GetState_EnableDraw method

(IPCB_Primitive interface)

Syntax

```
Function GetState_EnableDraw : Boolean;
```

Description

Example

See also

IPCB_Primitive interface

GetState_Identifier method

(IPCB_Primitive interface)

Syntax

```
Function GetState_Identifier : TPCBString;
```

Description

Example

See also

IPCB_Primitive interface

GetState_InBoard method

(IPCB_Primitive interface)

Syntax

```
Function GetState_InBoard : Boolean;
```

Description

Example

See also

IPCB_Primitive interface

GetState_InComponent method

(IPCB_Primitive interface)

Syntax

```
Function GetState_InComponent : Boolean;
```

Description

Example

See also

IPCB_Primitive interface

GetState_InCoordinate method

(IPCB_Primitive interface)

Syntax

```
Function GetState_InCoordinate : Boolean;
```

Description

Example

See also

IPCB_Primitive interface

GetState_Index method

(IPCB_Primitive interface)

Syntax

```
Function GetState_Index : Word;
```

Description

Example

See also

IPCB_Primitive interface

GetState_InDimension method

(IPCB_Primitive interface)

Syntax

```
Function GetState_InDimension : Boolean;
```

Description

Example

See also

IPCB_Primitive interface

GetState_Enabled_vDimension method

(IPCB_Primitive interface)

Syntax

```
Function GetState_Enabled_vDimension : Boolean;
```

Description

Example

See also

IPCB_Primitive interface

GetState_InNet method

(IPCB_Primitive interface)

Syntax

```
Function GetState_InNet : Boolean;
```

Description

Example

See also

IPCB_Primitive interface

GetState_Enabled_vPolygon method

(IPCB_Primitive interface)

Syntax

```
Function GetState_Enabled_vPolygon : Boolean;
```

Description

Example

See also

IPCB_Primitive interface

GetState_Enabled_vNet method

(IPCB_Primitive interface)

Syntax

```
Function GetState_Enabled_vNet : Boolean;
```

Description

Example

See also

IPCB_Primitive interface

GetState_DRCError method

(IPCB_Primitive interface)

Syntax

```
Function GetState_DRCError : Boolean;
```

Description

The DRCError property determines whether the object is affected by the Design Rule Checker and thus if the object breaks one of the design rules, the DRCError is true.

Example

See also

IPCB_Primitive interface

GetState_Enabled method

(IPCB_Primitive interface)

Syntax

```
Function GetState_Enabled : Boolean;
```

Description

Example

See also

IPCB_Primitive interface

GetState_Enabled_Direct method

(IPCB_Primitive interface)

Syntax

```
Function GetState_Enabled_Direct : Boolean;
```

Description

Example

See also

IPCB_Primitive interface

GetState_Enabled_vComponent method

(IPCB_Primitive interface)

Syntax

```
Function GetState_Enabled_vComponent : Boolean;
```

Description

Example

See also

IPCB_Primitive interface

GetState_Enabled_vCoordinate method

(IPCB_Primitive interface)

Syntax

```
Function GetState_Enabled_vCoordinate : Boolean;
```

Description

Example

See also

IPCB_Primitive interface

GetState_InPolygon method

(IPCB_Primitive interface)

Syntax

```
Function GetState_InPolygon : Boolean;
```

Description

Example

See also

IPCB_Primitive interface

GetState_InSelectionMemory method

(IPCB_Primitive interface)

Syntax

```
Function GetState_InSelectionMemory (Index : Integer) : Boolean;
```

Description

Example

See also

IPCB_Primitive interface

GetState_IsElectricalPrim method

(IPCB_Primitive interface)

Syntax

```
Function GetState_IsElectricalPrim : Boolean;
```

Description

Example

See also

IPCB_Primitive interface

GetState_IsTenting method

(IPCB_Primitive interface)

Syntax

```
Function GetState_IsTenting : Boolean;
```

Description

Example

See also

IPCB_Primitive interface

GetState_IsKeepout method

(IPCB_Primitive interface)

Syntax

```
Function GetState_IsKeepout : Boolean;
```

Description

The keepout layer generally defines areas on the PCB document that you don't want automatically or manually routed, and this can include clearance areas around mounting hole pads or high voltage components for example.

This function determines whether the object itself is used for the keep out boundary.

Example

See also

IPCB_Primitive interface

GetState_Moveable method

(IPCB_Primitive interface)

Syntax

```
Function GetState_Moveable : Boolean;
```

Description

This method determines whether this design object can be moved or not (by the autorouter for example).

This method is used by the Moveable property.

Example

See also

IPCB_Primitive interface

GetState_IsTenting_Bottom method

(IPCB_Primitive interface)

Syntax

```
Function GetState_IsTenting_Bottom : Boolean;
```

Description

Example

See also

IPCB_Primitive interface

GetState_IsTenting_Top method

(IPCB_Primitive interface)

Syntax

```
Function GetState_IsTenting_Top : Boolean;
```

Description

Example

See also

IPCB_Primitive interface

GetState_IsTestPoint_Bottom method

(IPCB_Primitive interface)

Syntax

```
Function GetState_IsTestPoint_Bottom : Boolean;
```

Description

Example

See also

IPCB_Primitive interface

GetState_IsTestPoint_Top method

(IPCB_Primitive interface)

Syntax

```
Function GetState_IsTestPoint_Top : Boolean;
```

Description

Example

See also

IPCB_Primitive interface

GetState_Layer method

(IPCB_Primitive interface)

Syntax

```
Function GetState_Layer : TLayer;
```

Description

Example

See also

IPCB_Primitive interface

GetState_MiscFlag1 method

(IPCB_Primitive interface)

Syntax

```
Function GetState_MiscFlag1 : Boolean;
```

Description

Example

See also

IPCB_Primitive interface

GetState_MiscFlag3 method

(IPCB_Primitive interface)

Syntax

```
Function GetState_MiscFlag3 : Boolean;
```

Description

Example

See also

IPCB_Primitive interface

GetState_Net method

(IPCB_Primitive interface)

Syntax

```
Function GetState_Net : IPCB_Net;
```

Description

The net property of an object denotes it has an electrical property, meaning it is connected from one node to another. The method fetches the net of an object (if it has one).

This method is used for the Net property.

Example

See also

IPCB_Primitive interface

GetState_MiscFlag2 method

(IPCB_Primitive interface)

Syntax

```
Function GetState_MiscFlag2 : Boolean;
```

Description

Example

See also

IPCB_Primitive interface

GetState_IsPreRoute method

(IPCB_Primitive interface)

Syntax

```
Function GetState_IsPreRoute : Boolean;
```

Description

Example

See also

IPCB_Primitive interface

GetState_ReliefEntries method

(IPCB_Primitive interface)

Syntax

```
Function GetState_ReliefEntries : Integer;
```

Description

This method retrieves the number of relief entries for a pad/via object.

Example

See also

IPCB_Primitive interface

GetState_PasteMaskExpansion method

(IPCB_Primitive interface)

Syntax

```
Function GetState_PasteMaskExpansion : TCoord;
```

Description

Example

See also

IPCB_Primitive interface

GetState_Polygon method

(IPCB_Primitive interface)

Syntax

```
Function GetState_Polygon : IPCB_Polygon;
```

Description

This function retrieves the IPCB_Polygon interface that the design object primitive is associated with. For example, a polygon may contain arcs and tracks, and when you only have the arc, you can retreive the polygon the arc is associated with.

Example

See also

IPCB_Primitive interface

GetState_PolygonOutline method

(IPCB_Primitive interface)

Syntax

```
Function GetState_PolygonOutline : Boolean;
```

Description

This function determines whether the design object primitive is part of the polygon outline or not.

Example

See also

IPCB_Primitive interface

GetState_PowerPlaneClearance method

(IPCB_Primitive interface)

Syntax

```
Function GetState_PowerPlaneClearance : TCoord;
```

Description

Example

See also

IPCB_Primitive interface

GetState_PowerPlaneConnectStyle method

(IPCB_Primitive interface)

Syntax

```
Function GetState_PowerPlaneConnectStyle : TPlaneConnectStyle;
```

Description

Example

See also

IPCB_Primitive interface

GetState_PowerPlaneReliefExpansion method

(IPCB_Primitive interface)

Syntax

```
Function GetState_PowerPlaneReliefExpansion : TCoord;
```

Description

Example

See also

IPCB_Primitive interface

GetState_ObjectIDString method

(IPCB_Primitive interface)

Syntax

```
Function GetState_ObjectIDString : TPCBString;
```

Description

This **ObjectIDString** property returns the Object Id string. For example eTrackObject type will yield a Track string.

The method returns a object id string for the associated object and is used in the ObjectIDString property.

Example

See also

IPCB_Primitive interface

GetState_ReliefConductorWidth method

(IPCB_Primitive interface)

Syntax

```
Function GetState_ReliefConductorWidth : TCoord;
```

Description

This method retrieves the relief conductor width of a pad or via object as a TCoord value.

Example

See also

IPCB_Primitive interface

GetState_ObjectId method

(IPCB_Primitive interface)

Syntax

```
Function GetState_ObjectId : TObjectId;
```

Description

Example

See also

IPCB_Primitive interface

GetState_Selected method

(IPCB_Primitive interface)

Syntax

```
Function GetState_Selected : Boolean;
```

Description

This method determines whether this object is selected or not on the PCB document. This method is used by the Selected property.

Example

See also

IPCB_Primitive interface

GetState_SolderMaskExpansion method

(IPCB_Primitive interface)

Syntax

```
Function GetState_SolderMaskExpansion : TCoord;
```

Description

The solder mask expansion property determines the shape that is created on the solder mask layer at each pad and via site. This shape is expanded or contracted radially by the amount specified by this rule. This property over-rides the solder mask expansion design rule.

This method is used for the SolderMaskExpansion property.

Example

See also

IPCB_Primitive interface

GetState_TearDrop method

(IPCB_Primitive interface)

Syntax

```
Function GetState_TearDrop : Boolean;
```

Description

This method determines whether the PCB object (an arc or track object) is used for as a tear drop.

This TearDrop property is supported by the GetState_TearDrop and SetState_TearDrop methods.

Example

See also

IPCB_Primitive interface

GetState_Used method

(IPCB_Primitive interface)

Syntax

```
Function GetState_Used : Boolean;
```

Description

Example

See also

IPCB_Primitive interface

GetState_ReliefAirGap method

(IPCB_Primitive interface)

Syntax

```
Function GetState_ReliefAirGap : TCoord;
```

Description

Example

See also

IPCB_Primitive interface

GetState_PadCacheRobotFlag method

(IPCB_Primitive interface)

Syntax

```
Function GetState_PadCacheRobotFlag : Boolean;
```

Description

Example

See also

IPCB_Primitive interface

GetState_UserRouted method

(IPCB_Primitive interface)

Syntax

```
Function GetState_UserRouted : Boolean;
```

Description

Example

See also

IPCB_Primitive interface

GetState_ViewableObjectID method

(IPCB_Primitive interface)

Syntax

```
Function GetState_ViewableObjectID : TViewableObjectID;
```

Description

The property determines the ViewableObjectID of the design object. The TViewableObjectID type is a more descriptive ID of a design object than the TObjectID type.

For example any type of dimension object is a eDimension type according to the TObjectID but could be one of the eViewableObject_LinearDimension...eViewableObject_RadialDiameterDimension value.

This function returns the TViewableObjectID and is used in the ViewableObjectID property.

Example

See also

IPCB_Primitive interface

SetState_InComponent method

(IPCB_Primitive interface)

Syntax

```
Procedure SetState_InComponent (Value : Boolean);
```

Description

Example

See also

IPCB_Primitive interface

SetState_Enabled_Direct method

(IPCB_Primitive interface)

Syntax

```
Procedure SetState_Enabled_Direct (Value : Boolean);
```

Description

Example

See also

IPCB_Primitive interface

SetState_Enabled_vComponent method

(IPCB_Primitive interface)

Syntax

```
Procedure SetState_Enabled_vComponent (Value : Boolean);
```

Description

Example

See also

IPCB_Primitive interface

SetState_Enabled_vCoordinate method

(IPCB_Primitive interface)

Syntax

```
Procedure SetState_Enabled_vCoordinate(Value : Boolean);
```

Description

Example

See also

IPCB_Primitive interface

SetState_Enabled_vDimension method

(IPCB_Primitive interface)

Syntax

```
Procedure SetState_Enabled_vDimension (Value : Boolean);
```

Description

Example

See also

IPCB_Primitive interface

SetState_Enabled_vNet method

(IPCB_Primitive interface)

Syntax

```
Procedure SetState_Enabled_vNet (Value : Boolean);
```

Description

Example

See also

IPCB_Primitive interface

SetState_Enabled_vPolygon method

(IPCB_Primitive interface)

Syntax

```
Procedure SetState_Enabled_vPolygon (Value : Boolean);
```

Description

Example

See also

IPCB_Primitive interface

SetState_DRCError method

(IPCB_Primitive interface)

Syntax

```
Procedure SetState_DRCError (Value : Boolean);
```

Description

The DRCError property determines whether the object is affected by the Design Rule Checker and thus if the object breaks one of the design rules, the DRCError property is true. This method is used in the DRCError property.

Example

See also

IPCB_Primitive interface

SetState_InBoard method

(IPCB_Primitive interface)

Syntax

```
Procedure SetState_InBoard (Value : Boolean);
```

Description

Example

See also

IPCB_Primitive interface

SetState_Dimension method

(IPCB_Primitive interface)

Syntax

```
Procedure SetState_Dimension (Value : IPCB_Dimension);
```

Description

This property determines whether the object itself is associated with the dimension object or not. This method sets the dimension object and is used in the Dimension property.

Example

See also

IPCB_Primitive interface

SetState_EnableDraw method

(IPCB_Primitive interface)

Syntax

```
Procedure SetState_EnableDraw (Value : Boolean);
```

Description

Example

See also

IPCB_Primitive interface

SetState_Enabled method

(IPCB_Primitive interface)

Syntax

```
Procedure SetState_Enabled (Value : Boolean);
```

Description

Example

See also

IPCB_Primitive interface

SetState_AllowGlobalEdit method

(IPCB_Primitive interface)

Syntax

```
Procedure SetState_AllowGlobalEdit (Value : Boolean);
```

Description

Example

See also

IPCB_Primitive interface

SetState_Board method

(IPCB_Primitive interface)

Syntax

```
Procedure SetState_Board (ABoard : IPCB_Board);
```

Description

The Board property determines the PCB document that the object itself is associated with. This method sets the PCB document that the object is associated with and is used in the Board property.

Example

See also

IPCB_Primitive interface

SetState_Component method

(IPCB_Primitive interface)

Syntax

```
Procedure SetState_Component (Value : IPCB_Component);
```

Description

This property determines whether the object itself is associated with the component or not. This method sets the Component and is used in the Component property.

Example

See also

IPCB_Primitive interface

SetState_Coordinate method

(IPCB_Primitive interface)

Syntax

```
Procedure SetState_Coordinate (Value : IPCB_Coordinate);
```

Description

This property determines whether the object itself is associated with the coordinate object or not. This method retrieves the Coordinate object and is used in the Coordinate property.

Example

See also

IPCB_Primitive interface

SetState_InDimension method

(IPCB_Primitive interface)

Syntax

```
Procedure SetState_InDimension (Value : Boolean);
```

Description

Example

See also

IPCB_Primitive interface

SetState_InCoordinate method

(IPCB_Primitive interface)

Syntax

```
Procedure SetState_InCoordinate (Value : Boolean);
```

Description

Example

See also

IPCB_Primitive interface

SetState_InNet method

(IPCB_Primitive interface)

Syntax

```
Procedure SetState_InNet (Value : Boolean);
```

Description

Example

See also

IPCB_Primitive interface

SetState_Index method

(IPCB_Primitive interface)

Syntax

```
Procedure SetState_Index (AIndex : Word);
```

Description

Example

See also

IPCB_Primitive interface

SetState_Layer method

(IPCB_Primitive interface)

Syntax

```
Procedure SetState_Layer (ALayer : TLayer);
```

Description

Example

See also

IPCB_Primitive interface

SetState_InPolygon method

(IPCB_Primitive interface)

Syntax

```
Procedure SetState_InPolygon (Value : Boolean);
```

Description

Example

See also

IPCB_Primitive interface

SetState_InSelectionMemory method

(IPCB_Primitive interface)

Syntax

```
Procedure SetState_InSelectionMemory (Index : Integer;Value : Boolean);
```

Description

Example

See also

IPCB_Primitive interface

SetState_IsKeepout method

(IPCB_Primitive interface)

Syntax

```
Procedure SetState_IsKeepout (Value : Boolean);
```

Description

The keepout layer generally defines areas on the PCB document that you don't want automatically or manually routed, and this can include clearance areas around mounting hole pads or high voltage components for example.

Example

See also

IPCB_Primitive interface

SetState_IsPreRoute method

(IPCB_Primitive interface)

Syntax

```
Procedure SetState_IsPreRoute (B : Boolean);
```

Description

Example

See also

IPCB_Primitive interface

SetState_IsTenting method

(IPCB_Primitive interface)

Syntax

```
Procedure SetState_IsTenting (Value : Boolean);
```

Description

Example

See also

IPCB_Primitive interface

SetState_IsTenting_Bottom method

(IPCB_Primitive interface)

Syntax

```
Procedure SetState_IsTenting_Bottom (Value : Boolean);
```

Description

Example

See also

IPCB_Primitive interface

SetState_IsTenting_Top method

(IPCB_Primitive interface)

Syntax

```
Procedure SetState_IsTenting_Top (Value : Boolean);
```

Description

Example

See also

IPCB_Primitive interface

SetState_IsTestPoint_Top method

(IPCB_Primitive interface)

Syntax

```
Procedure SetState_IsTestPoint_Top (Value : Boolean);
```

Description

Example

See also

IPCB_Primitive interface

SetState_MiscFlag1 method

(IPCB_Primitive interface)

Syntax

```
Procedure SetState_MiscFlag1 (Value : Boolean);
```

Description

This method sets a boolean value to the MiscFlag1 field and can be used for custom purposes.

Example

See also

IPCB_Primitive interface

SetState_MiscFlag2 method

(IPCB_Primitive interface)

Syntax

```
Procedure SetState_MiscFlag2 (Value : Boolean);
```

Description

This method sets a boolean value to the MiscFlag2 field and can be used for custom purposes.

Example

See also

IPCB_Primitive interface

SetState_IsTestPoint_Bottom method

(IPCB_Primitive interface)

Syntax

```
Procedure SetState_IsTestPoint_Bottom (Value : Boolean);
```

Description

Example

See also

IPCB_Primitive interface

SetState_UserRouted method

(IPCB_Primitive interface)

Syntax

```
Procedure SetState_UserRouted (Value : Boolean);
```

Description

Example

See also

IPCB_Primitive interface

SetState_MiscFlag3 method

(IPCB_Primitive interface)

Syntax

```
Procedure SetState_MiscFlag3 (Value : Boolean);
```

Description

This method sets a boolean value to the MiscFlag3 field and can be used for custom purposes.

Example

See also

IPCB_Primitive interface

SetState_Moveable method

(IPCB_Primitive interface)

Syntax

```
Procedure SetState_Moveable (Value : Boolean);
```

Description

This method sets whether this design object can be moved or not (by the autorouter for example).

This method is used by the Moveable property.

Example

See also

IPCB_Primitive interface

SetState_Net method

(IPCB_Primitive interface)

Syntax

```
Procedure SetState_Net (Value : IPCB_Net);
```

Description

The net property of an object denotes it has an electrical property, meaning it is connected from one node to another. The method sets the valid net to an object.

This method is used for the Net property.

Example

See also

IPCB_Primitive interface

SetState_PadCacheRobotFlag method

(IPCB_Primitive interface)

Syntax

```
Procedure SetState_PadCacheRobotFlag (Value : Boolean);
```

Description

Example

See also

IPCB_Primitive interface

SetState_Polygon method

(IPCB_Primitive interface)

Syntax

```
Procedure SetState_Polygon (Value : IPCB_Polygon);
```

Description

Example

See also

IPCB_Primitive interface

SetState_PolygonOutline method

(IPCB_Primitive interface)

Syntax

```
Procedure SetState_PolygonOutline (Value : Boolean);
```

Description

Example

See also

IPCB_Primitive interface

SetState_Selected method

(IPCB_Primitive interface)

Syntax

```
Procedure SetState_Selected (B : Boolean);
```

Description

This method determines whether this object is selected or not on the PCB document by passing in a boolean parameter.

This method is used by the Selected property.

Example

See also

IPCB_Primitive interface

SetState_Used method

(IPCB_Primitive interface)

Syntax

```
Procedure SetState_Used (Value : Boolean);
```

Description

Example

See also

IPCB_Primitive interface

SetState_TearDrop method

(IPCB_Primitive interface)

Syntax

```
Procedure SetState_TearDrop (Value : Boolean);
```

Description

This method determines whether the PCB object (an arc or track object) is used for as a tear drop.

This TearDrop property is supported by the GetState_TearDrop and SetState_TearDrop methods.

Example

See also

IPCB_Primitive interface

Properties

AllowGlobalEdit property

(IPCB_Primitive interface)

Syntax

```
Property AllowGlobalEdit : Boolean Read GetState_AllowGlobalEdit Write
SetState_AllowGlobalEdit;
```

Description

This property is supported by the GetState_AllowGlobalEdit and SetState_AllowGlobalEdit methods.

Example

See also

IPCB_Primitive interface

Board property

(IPCB_Primitive interface)

Syntax

```
Property Board : IPCB_Board Read GetState_Board Write SetState_Board;
```

Description

The Board property determines the PCB document that the object itself is associated with. This property is supported by the GetState_Board and SetState_Board methods.

Example

See also

IPCB_Primitive interface

Component property

(IPCB_Primitive interface)

Syntax

```
Property Component : IPCB_Component Read GetState_Component Write SetState_Component;
```

Description

This property determines whether the object itself is associated with the component or not. This property is supported by the GetState_Component and SetState_Component methods.

Example

See also

IPCB_Primitive interface

Coordinate property

(IPCB_Primitive interface)

Syntax

```
Property Coordinate : IPCB_Coordinate Read GetState_Coordinate Write SetState_Coordinate;
```

Description

The Coordinate property returns the IPCB_Coordinate only if this object itself is part of the IPCB_Coordinate type. A coordinate object is a group object and is composed of design object primitives such as tracks, arcs and text objects.

This property is supported by the GetState_Coordinate and SetState_Coordinate methods.

Example

See also

IPCB_Primitive interface

Detail property

(IPCB_Primitive interface)

Syntax

```
Property Detail : TPCBString Read GetState_DetailString;
```

Description

This property retrieves the Detail text string for this design object. This property is supported by the GetState_Detail method.

Example

See also

IPCB_Primitive interface

Dimension property

(IPCB_Primitive interface)

Syntax

```
Property Dimension : IPCB_Dimension Read GetState_Dimension Write SetState_Dimension;
```

Description

The Coordinate property returns the IPCB_Dimension only if this object itself is part of the IPCB_Dimension type. A dimension object is a group object and is composed of design object primitives such as tracks, arcs and text objects.

This property is supported by the GetState_Dimension and SetState_Dimension methods.

Example

See also

IPCB_Primitive interface

DRCError property

(IPCB_Primitive interface)

Syntax

```
Property DRCError : Boolean Read GetState_DRCError Write SetState_DRCError;
```

Description

The DRCError property determines whether the object is affected by the Design Rule Checker and thus if the object breaks one of the design rules, the DRCError is true.

This property is supported by the GetState_DRCError and SetState_DRCError methods.

Example

See also

IPCB_Primitive interface

Descriptor property

(IPCB_Primitive interface)

Syntax

```
Property Descriptor : TPCBString Read GetState_DescriptorString;
```

Description

The Descriptor read only property fetches the Descriptor string.

This property is supported by the GetState_Descriptor methods.

Example

See also

IPCB_Primitive interface

Enabled_Direct property

(IPCB_Primitive interface)

Syntax

```
Property Enabled_Direct : Boolean Read GetState_Enabled_Direct Write SetState_Enabled_Direct;
```

Description

This property is supported by the GetState_Direct and SetState_Direct methods.

Example

See also

IPCB_Primitive interface

Enabled property

(IPCB_Primitive interface)

Syntax

```
Property Enabled : Boolean Read GetState_Enabled Write SetState_Enabled;
```

Description

This property is supported by the GetState_Enabled and SetState_Enabled methods.

Example

See also

IPCB_Primitive interface

Enabled_vComponent property

(IPCB_Primitive interface)

Syntax

```
Property Enabled_vComponent : Boolean Read GetState_Enabled_vComponent Write
SetState_Enabled_vComponent;
```

Description

This property is supported by the GetState_vComponent and SetState_vComponent methods.

Example

See also

IPCB_Primitive interface

Enabled_vCoordinate property

(IPCB_Primitive interface)

Syntax

```
Property Enabled_vCoordinate : Boolean Read GetState_Enabled_vCoordinate Write
SetState_Enabled_vCoordinate;
```

Description

This property is supported by the GetState_vCoordinate and SetState_vCoordinate methods.

Example

See also

IPCB_Primitive interface

Enabled_vDimension property

(IPCB_Primitive interface)

Syntax

```
Property Enabled_vDimension : Boolean Read GetState_Enabled_vDimension Write
SetState_Enabled_vDimension;
```

Description

This property is supported by the GetState_vDimension and SetState_vDimension methods.

Example

See also

IPCB_Primitive interface

Enabled_vNet property

(IPCB_Primitive interface)

Syntax

```
Property Enabled_vNet : Boolean Read GetState_Enabled_vNet Write SetState_Enabled_vNet;
```

Description

This property is supported by the GetState_vNet and SetState_vNet methods.

Example

See also

IPCB_Primitive interface

Enabled_vPolygon property

(IPCB_Primitive interface)

Syntax

```
Property Enabled_vPolygon : Boolean Read GetState_Enabled_vPolygon Write
SetState_Enabled_vPolygon;
```

Description

This property is supported by the GetState_vPolygon and SetState_vPolygon methods.

Example

See also

IPCB_Primitive interface

EnableDraw property

(IPCB_Primitive interface)

Syntax

```
Property EnableDraw : Boolean Read GetState_EnableDraw Write SetState_EnableDraw;
```

Description

This property is supported by the GetState_EnableDraw and SetState_EnableDraw methods.

Example

See also

IPCB_Primitive interface

InDimension property

(IPCB_Primitive interface)

Syntax

```
Property InDimension : Boolean Read GetState_InDimension Write SetState_InDimension;
```

Description

This InDimension property determines whether the obejct itself is part of the dimension object or not.

This property is supported by the GetState_InDimension and SetState_InDimension methods.

Example

See also

IPCB_Primitive interface

Identifier property

(IPCB_Primitive interface)

Syntax

```
Property Identifier : TPCBString Read GetState_Identifier;
```

Description

This property is supported by the GetState_Identifier method.

Example

See also

IPCB_Primitive interface

InBoard property

(IPCB_Primitive interface)

Syntax

```
Property InBoard : Boolean Read GetState_InBoard Write SetState_InBoard;
```

Description

This InBoard property determines whether the object itself is part of the board object or not.

This property is supported by the GetState_InBoard and SetState_InBoard methods.

Example

See also

IPCB_Primitive interface

InComponent property

(IPCB_Primitive interface)

Syntax

```
Property InComponent : Boolean Read GetState_InComponent Write SetState_InComponent;
```

Description

This InComponent property determines whether the object itself is part of the component object or not.

This property is supported by the GetState_InComponent and SetState_InComponent methods.

Example

See also

IPCB_Primitive interface

Index property

(IPCB_Primitive interface)

Syntax

```
Property Index : Word Read GetState_Index Write SetState_Index;
```

Description

This property is supported by the GetState_Index and SetState_Index methods.

Example

See also

IPCB_Primitive interface

InNet property

(IPCB_Primitive interface)

Syntax

```
Property InNet : Boolean Read GetState_InNet Write SetState_InNet;
```

Description

This property is supported by the GetState_InNet and SetState_InNet methods.

Example

See also

IPCB_Primitive interface

InPolygon property

(IPCB_Primitive interface)

Syntax

```
Property InPolygon : Boolean Read GetState_InPolygon Write SetState_InPolygon;
```

Description

This InPolygon property determines whether the object itself is part of the polygon object or not.

This property is supported by the GetState_InPolygon and SetState_InPolygon methods.

Example

See also

IPCB_Primitive interface

InSelectionMemory property

(IPCB_Primitive interface)

Syntax

```
Property InSelectionMemory [I : Integer] : Boolean Read GetState_InSelectionMemory Write
SetState_InSelectionMemory;
```

Description

This property is supported by the GetState_InSelectionMemory and SetState_InSelectionMemory methods.

Example

See also

IPCB_Primitive interface

IsElectricalPrim property

(IPCB_Primitive interface)

Syntax

```
Property IsElectricalPrim : Boolean Read GetState_IsElectricalPrim;
```

Description

This property determines whether this PCB object possesses an electrical property- tracks, fills, polygons, arcs, vias all have electrical properties - basically those objects that have a Net property will possess an electrical property.

Embedded boards and Embedded objects etc don't have an electrical property for example.

This property is supported by the GetState_IsElectricalPrim and SetState_IsElectricalPrim methods.

Example

See also

IPCB_Primitive interface

InCoordinate property

(IPCB_Primitive interface)

Syntax

```
Property InCoordinate : Boolean Read GetState_InCoordinate Write SetState_InCoordinate;
```

Description

This InCoordinate property determines whether the object itself is part of the coordinate object or not.

This property is supported by the GetState_InCoordinate and SetState_InCoordinate methods.

Example

See also

IPCB_Primitive interface

IsKeepout property

(IPCB_Primitive interface)

Syntax

```
Property IsKeepout : Boolean Read GetState_IsKeepout Write SetState_IsKeepout;
```

Description

This property determines whether a PCB object is used as a keep-out object. Currently arc, track and fill objects are used as keep out objects. The keepout layer generally defines areas on the PCB document that you don't want automatically or manually routed, and this can include clearance areas around mounting hole pads or high voltage components for example.

This property is supported by the GetState_IsKeepOut and SetState_IsKeepOut methods.

Example

See also

IPCB_Primitive interface

IsPreRoute property

(IPCB_Primitive interface)

Syntax

```
Property IsPreRoute : Boolean Read GetState_IsPreRoute Write SetState_IsPreRoute;
```

Description

This property is supported by the GetState_IsPreRoute and SetState_IsPreRoute methods.

Example

See also

IPCB_Primitive interface

IsTestpoint_Bottom property

(IPCB_Primitive interface)

Syntax

```
Property IsTestpoint_Bottom : Boolean Read GetState_IsTestpoint_Bottom Write
SetState_IsTestpoint_Bottom;
```

Description

This property determines whether a pad or via is used as a test point on the bottom layer.

This property is supported by the GetState_IsTestpoint_Bottom and SetState_IsTestPoint_Bottom methods.

Example

See also

IPCB_Primitive interface

IsTenting property

(IPCB_Primitive interface)

Syntax

```
Property IsTenting : Boolean Read GetState_IsTenting Write SetState_IsTenting;
```

Description

This property determines whether the solder mask of pad and via objects are tented on top and bottom layers. A tenting closes an opening in the mask of pad or via objects.

This property is supported by the GetState_IsTenting and SetState_IsTenting methods.

Example

See also

IPCB_Primitive interface

IsTenting_Top property

(IPCB_Primitive interface)

Syntax

```
Property IsTenting_Top : Boolean Read GetState_IsTenting_Top Write SetState_IsTenting_Top;
```

Description

This property determines whether the solder mask of pad and via objects are tented or not on the top layer. A tenting closes an opening in the mask of pad or via objects.

This property is supported by the GetState_IsTenting_Top and SetState_IsTenting_Top methods.

Example

See also

IPCB_Primitive interface

IsTestpoint_Top property

(IPCB_Primitive interface)

Syntax

```
Property IsTestpoint_Top : Boolean Read GetState_IsTestpoint_Top Write
SetState_IsTestpoint_Top;
```

Description

This property determines whether a pad or via is used as a test point on the top layer.

This property is supported by the GetState_IsTestpoint_Top and SetState_IsTestpoint_Top methods.

Example

See also

IPCB_Primitive interface

Layer property

(IPCB_Primitive interface)

Syntax

```
Property Layer : TLayer Read GetState_Layer Write SetState_Layer;
```

Description

This layer denotes which layer the object is on.

This property is supported by the GetState_Layer and SetState_layer methods.

Example

See also

IPCB_Primitive interface

TLayer type

MiscFlag1 property

(IPCB_Primitive interface)

Syntax

```
Property MiscFlag1 : Boolean Read GetState_MiscFlag1 Write SetState_MiscFlag1;
```

Description

This property determines the boolean value from the MiscFlag1 property and can be used for custom purposes.

This property is supported by the GetState_MiscFlag1 and SetState_MiscFlag1 methods.

Example

See also

IPCB_Primitive interface

MiscFlag2 property

(IPCB_Primitive interface)

Syntax

```
Property MiscFlag2 : Boolean Read GetState_MiscFlag2 Write SetState_MiscFlag2;
```

Description

This property determines the boolean value from the MiscFlag2 property and can be used for custom purposes.

This property is supported by the GetState_MiscFlag2 and SetState_MiscFlag2 methods.

Example

See also

IPCB_Primitive interface

MiscFlag3 property

(IPCB_Primitive interface)

Syntax

```
Property MiscFlag3 : Boolean Read GetState_MiscFlag3 Write SetState_MiscFlag3;
```

Description

This property determines the boolean value from the MiscFlag3 property and can be used for custom purposes.

This property is supported by the GetState_MiscFlag3 and SetState_MiscFlag3 methods.

Example

See also

IPCB_Primitive interface

IsTenting_Bottom property

(IPCB_Primitive interface)

Syntax

```
Property IsTenting_Bottom : Boolean Read GetState_IsTenting_Bottom Write
SetState_IsTenting_Bottom;
```

Description

This property determines whether the solder mask of pad and via objects are tented or not on the bottom layer. A tenting closes an opening in the mask of pad or via objects.

This property is supported by the GetState_IsTenting_Bottom and SetState_IsTenting_Bottom methods.

Example

See also

IPCB_Primitive interface

Moveable property

(IPCB_Primitive interface)

Syntax

```
Property Moveable : Boolean Read GetState_Moveable Write SetState_Moveable;
```

Description

This property determines whether this design object can be moved or not (by the autorouter for example).

This property is supported by the GetState_Moveable and SetState_Moveable methods.

Example

See also

IPCB_Primitive interface

Net property

(IPCB_Primitive interface)

Syntax

```
Property Net : IPCB_Net Read GetState_Net Write SetState_Net;
```

Description

The Net property of an object denotes it has an electrical property, meaning it is connected from one node to another.

This property is supported by the GetState_Net and SetState_Net methods.

Example

See also

IPCB_Primitive interface

NetObjectAssign script from the **Example**s\Scripts\Delphiscript Scripts\Pcb\

ObjectId property

(IPCB_Primitive interface)

Syntax

```
Property ObjectId : TObjectId Read GetState_ObjectId;
```

Description

This ObjectId property determines what Object Id this object is. Please note that this ObjectId type is a limited set and to have a wider range of Object IDs, check the TViewableObjectId type.

This read only property is supported by the GetState_ObjectId method.

Example

See also

IPCB_Primitive interface

ViewableObjectId property

ObjectIDString property

(IPCB_Primitive interface)

Syntax

```
Property ObjectIDString : TPCBString Read GetState_ObjectIDString;
```

Description

This ObjectIDString property returns the Object Id string. For example eTrackObject type will yield a Track string.

This read only property is supported by the GetState_ObjectIDString method.

Example

See also

IPCB_Primitive interface

PadCacheRobotFlag property

(IPCB_Primitive interface)

Syntax

```
Property PadCacheRobotFlag : Boolean Read GetState_PadCacheRobotFlag Write
SetState_PadCacheRobotFlag;
```

Description

This property is supported by the GetState_PadCacheRobotFlag and SetState_PadCacheRobotFlag methods.

Example

See also

IPCB_Primitive interface

PasteMaskExpansion property

(IPCB_Primitive interface)

Syntax

```
Property PasteMaskExpansion : TCoord Read GetState_PasteMaskExpansion;
```

Description

This property is supported by the GetState_PasteMaskExpansion and SetState_PasteMaskExpansion methods.

Example

See also

IPCB_Primitive interface

Polygon property

(IPCB_Primitive interface)

Syntax

```
Property Polygon : IPCB_Polygon Read GetState_Polygon Write SetState_Polygon;
```

Description

This property is supported by the GetState_Polygon and SetState_Polygon methods.

Example

See also

IPCB_Primitive interface

PolygonOutline property

(IPCB_Primitive interface)

Syntax

```
Property PolygonOutline : Boolean Read GetState_PolygonOutline Write SetState_PolygonOutline;
```

Description

This property is supported by the GetState_PolygonOutline and SetState_PolygonOutline methods.

Example

See also

IPCB_Primitive interface

PowerPlaneReliefExpansion property

(IPCB_Primitive interface)

Syntax

```
Property PowerPlaneReliefExpansion : TCoord Read GetState_PowerPlaneReliefExpansion;
```

Description

This property is supported by the GetState_PowerPlaneReliefExpansion method.

Example

See also

IPCB_Primitive interface

PowerPlaneClearance property

(IPCB_Primitive interface)

Syntax

```
Property PowerPlaneClearance : TCoord Read GetState_PowerPlaneClearance;
```

Description

This property is supported by the GetState_PowerPlaneClearance method.

Example

See also

IPCB_Primitive interface

PowerPlaneConnectStyle property

(IPCB_Primitive interface)

Syntax

```
Property PowerPlaneConnectStyle : TPlaneConnectStyle Read GetState_PowerPlaneConnectStyle;
```

Description

This property is supported by the GetState_PowerPlaneConnectStyle method.

Example

See also

IPCB_Primitive interface

TPlaneConnectStyle type

ReliefAirGap property

(IPCB_Primitive interface)

Syntax

```
Property ReliefAirGap : TCoord Read GetState_ReliefAirGap;
```

Description

The ReliefAirGap property retrieves the relief air gap value for this pad/via object.

This read only property is supported by the GetState_ReliefAirGap method.

Example

See also

IPCB_Primitive interface

ReliefConductorWidth property

(IPCB_Primitive interface)

Syntax

```
Property ReliefConductorWidth : TCoord Read GetState_ReliefConductorWidth;
```

Description

The ReliefConductorWidth property retrieves the relief conductor width value for a this pad/via object.

This read only property is supported by the GetState_ReliefConductorWidth method

Example

See also

IPCB_Primitive interface

ReliefEntries property

(IPCB_Primitive interface)

Syntax

```
Property ReliefEntries : Integer Read GetState_ReliefEntries;
```

Description

This property retrieves the number of relief entries for a pad/via object.

This read only property is supported by the GetState_ReliefEntries method.

Example

See also

IPCB_Primitive interface

Selected property

(IPCB_Primitive interface)

Syntax

```
Property Selected : Boolean Read GetState_Selected Write SetState_Selected;
```

Description

This property determines whether this object is selected or not on the PCB document.

This property is supported by the GetState_Selected and SetState_Selected methods.

Example

See also

IPCB_Primitive interface

SolderMaskExpansion property

(IPCB_Primitive interface)

Syntax

```
Property SolderMaskExpansion : TCoord Read GetState_SolderMaskExpansion;
```

Description

The solder mask expansion property determines the shape that is created on the solder mask layer at each pad and via site. This shape is expanded or contracted radially by the amount specified by this rule. This property over-rides the solder mask expansion design rule.

This read-only property is supported by the GetState_SolderMaskExpansion method.

Notes

A Solder Mask expansion property for a pad object is currently relevant just for pads on top and bottom copper layers.

Paste mask layers are used to design stencils which will selectively place solder paste on a blank PCB. Vias do not have a paste mask layer.

Solder paste is only placed on pads where component leads are to be soldered to them. Vias normally don't have anything soldered onto them.

Example

See also

IPCB_Primitive interface

TearDrop property

(IPCB_Primitive interface)

Syntax

```
Property TearDrop : Boolean Read GetState_TearDrop Write SetState_TearDrop;
```

Description

This property determines whether the PCB object (an arc or track object) is used for as a tear drop.

This property is supported by the GetState_TearDrop and SetState_TearDrop methods.

Example

See also

IPCB_Primitive interface

Used property

(IPCB_Primitive interface)

Syntax

```
Property Used : Boolean Read GetState_Used Write SetState_Used;
```

Description

This property is supported by the GetState_Used and SetState_Used methods.

Example

See also

IPCB_Primitive interface

UserRouted property

(IPCB_Primitive interface)

Syntax

```
Property UserRouted : Boolean Read GetState_UserRouted Write SetState_UserRouted;
```

Description

This property is supported by the GetState_UserRouted and SetState_UserRouted methods.

Example

See also

IPCB_Primitive interface

ViewableObjectID property

(IPCB_Primitive interface)

Syntax

```
Property ViewableObjectID : TViewableObjectID Read GetState_ViewableObjectID;
```

Description

The read only property determines the ViewableObjectID of the design object. The TViewableObjectID type is a more descriptive ID of a design object than the TObjectID type.

For example any type of dimension object is a eDimension type according to the TObjectID but could be one of the eViewableObject_LinearDimension...eViewableObject_RadialDiameterDimension value.

This property is supported by the GetState_ViewableObjectID and SetState_ViewableObjectId methods.

Notes

This **TViewableObjectID** type is mainly used by the Inspector and List views in Altium Designer and is an extension of **TObjectID** type.

Example

See also

IPCB_Primitive interface

TViewableObjectID type

TObjectID type

IPCB_Arc Interface

Overview

Arcs are circular track segments with a definable width and can be placed on any layer. Arcs can have resizeable angles. You can set the angles to 0 and 360 respectively to obtain a circle object. Arcs have a variety of uses in the PCB design layout.

For example, arcs can be used to outline component shapes. Arcs can also be placed on a signal layer and be electrically connected to tracks.

Note

You can use **IPCB_Primitive** methods and properties that are relevant to the **IPCB_Arc** interface.

The **IPCB_Arc** interface hierarchy is as follows;

IPCB_Primitive

IPCB_Arc

IPCB_Arc methods	IPCB_Arc properties
GetState_CenterX	XCenter
GetState_CenterY	YCenter
GetState_Radius	Radius
GetState_LineWidth	LineWidth
GetState_StartAngle	StartAngle
GetState_EndAngle	EndAngle
GetState_StartX	StartX
GetState_StartY	StartY
GetState_EndX	EndX
GetState_EndY	EndY
SetState_CenterX	
SetState_CenterY	
SetState_Radius	
SetState_LineWidth	
SetState_StartAngle	
SetState_EndAngle	
RotateAroundXY	
GetState_StrictHitTest	

Example

```
Var
    Board     : IPCB_Board;
    WorkSpace : IWorkSpace;
    Arc       : IPCB_Arc;
Begin
    // Create a new PCB documen
    WorkSpace := GetWorkSpace;
    If WorkSpace = Nil Then Exit;
    Workspace.DM_CreateNewDocument('PCB');

    Board := PCBServer.GetCurrentPCBBoard;
    If Board = Nil then exit;

    Arc := PCBServer.PCBObjectFactory(eArcObject, eNoDimension, eCreate_Default);
    // need the board origin marker to appear on the PCB document
    // in order to obtain the Board.Xorigin and YOrigin values.
    Arc.XCenter    := MilsToCoord(Board.XOrigin + 1800);
    Arc.YCenter    := MilsToCoord(Board.YOrigin + 1800);
    Arc.Radius     := MilsToCoord(200);
    Arc.LineWidth  := MilsToCoord(50);
    Arc.StartAngle := 0;
    Arc.EndAngle   := 270;
    Arc.Layer      := eBottomLayer;

    // Add the new arc object to the PCB database.
```

```
    Board.AddPCBObject(Arc);

    // Repaint the PCB Worksheet
    ResetParameters;
    AddStringParameter('Action', 'All');
    RunProcess('PCB:Zoom');
End;
```

See also

IPCB_Primitive interface

PCB Design Objects

Methods

RotateAroundXY method

(IPCB_Arc interface)

Syntax

```
Procedure RotateAroundXY (AX,AY : TCoord; Angle : TAngle);
```

Description

This method rotates an arc on the PCB document about the AX, AY coordinates with an angle in degrees. To ensure the arc rotates without moving about, pass in its XCenter and YCenter attributes for the AX,AY parameters.

Example

```
//rotate the arc about its original center
Arc.RotateAroundXY(Arc.XCenter,Arc.YCenter,45);
```

See also

IPCB_Arc interface

GetState and SetState Methods

GetState_CenterX method

(IPCB_Arc interface)

Syntax

```
Function GetState_CenterX : TCoord;
```

Description

This method is used for the CenterX property.

Example

See also

IPCB_Arc interface

GetState_CenterY method

(IPCB_Arc interface)

Syntax

```
Function GetState_CenterY : TCoord;
```

Description

This method is used for the CenterY property.

Example

See also

IPCB_Arc interface

GetState_EndAngle method

(IPCB_Arc interface)

Syntax

```
Function GetState_EndAngle : TAngle;
```

Description

This method is used for the EndAngle property.

Example

See also

IPCB_Arc interface

GetState_EndX method

(IPCB_Arc interface)

Syntax

```
Function GetState_EndX : TCoord;
```

Description

This method is used for the EndX property.

Example

See also

IPCB_Arc interface

GetState_EndY method

(IPCB_Arc interface)

Syntax

```
Function GetState_EndY : TCoord;
```

Description

This method is used for the EndY property.

Example

See also

IPCB_Arc interface

GetState_LineWidth method

(IPCB_Arc interface)

Syntax

```
Function GetState_LineWidth : TCoord;
```

Description

This method is used for the LineWidth property.

Example

See also

IPCB_Arc interface

GetState_Radius method

(IPCB_Arc interface)

Syntax

```
Function GetState_Radius : TCoord;
```

Description

This method is used for the Radius property.

Example

See also

IPCB_Arc interface

GetState_StartAngle method

(IPCB_Arc interface)

Syntax

```
Function GetState_StartAngle : TAngle;
```

Description

This method is used for the StartAngle property.

Example

See also

IPCB_Arc interface

GetState_StartX method

(IPCB_Arc interface)

Syntax

```
Function GetState_StartX : TCoord;
```

Description

This method is used for the StartX property.

Example

See also

IPCB_Arc interface

GetState_StartY method

(IPCB_Arc interface)

Syntax

```
Function GetState_StartY : TCoord;
```

Description

This method is used for the StartY property.

Example

See also

IPCB_Arc interface

GetState_StrictHitTest method

(IPCB_Arc interface)

Syntax

```
Function GetState_StrictHitTest (HitX,HitY : TCoord) : Boolean;
```

Description

Example

See also

IPCB_Arc interface

SetState_CenterX method

(IPCB_Arc interface)

Syntax

```
Procedure SetState_CenterX (AX : TCoord);
```

Description

This method is used for the CenterX property.

Example

See also

IPCB_Arc interface

SetState_CenterY method

(IPCB_Arc interface)

Syntax

```
Procedure SetState_CenterY (AY : TCoord);
```

Description

This method is used for the CenterY property.

Example

See also

IPCB_Arc interface

SetState_EndAngle method

(IPCB_Arc interface)

Syntax

```
Procedure SetState_EndAngle (Angle : TAngle);
```

Description

This method is used for the EndAngle property.

Example

See also

IPCB_Arc interface

SetState_LineWidth method

(IPCB_Arc interface)

Syntax

```
Procedure SetState_LineWidth (Width : TCoord);
```

Description

This method is used for the Linewidth property.

Example

See also

IPCB_Arc interface

SetState_Radius method

(IPCB_Arc interface)

Syntax

```
Procedure SetState_Radius (Radius : TCoord);
```

Description

This method is used for the Radius property.

Example

See also

IPCB_Arc interface

SetState_StartAngle method

(IPCB_Arc interface)

Syntax

```
Procedure SetState_StartAngle (Angle : TAngle);
```

Description

This method is used for the StartAngle property.

Example

See also

IPCB_Arc interface

Properties

EndAngle property

(IPCB_Arc interface)

Syntax

```
Property EndAngle : TAngle Read GetState_EndAngle Write SetState_EndAngle;
```

Description

The EndAngle property denotes the end angle of the arc. It is supported by the GetState_EndAngle / SetState_EndAngle and complemented by the GetState_StartAngle/SetState_StartAngle methods.

Example

See also

IPCB_Arc interface

EndX property

(IPCB_Arc interface)

Syntax

```
Property EndX : TCoord Read GetState_EndX;
```

Description

The EndX property denotes the end X coordinate of the arc. It is supported by the GetState_EndX method.

Example

See also

IPCB_Arc interface

EndY property

(IPCB_Arc interface)

Syntax

```
Property EndY : TCoord Read GetState_EndY;
```

Description

The EndY property denotes the end Y coordinate of the arc. It is supported by the GetState_EndY method.

Example

See also

IPCB_Arc interface

LineWidth property

(IPCB_Arc interface)

Syntax

```
Property LineWidth : TCoord Read GetState_LineWidth Write SetState_LineWidth;
```

Description

The LineWidth property denotes the line thickness or width of the arc. It is supported by the GetState_LineWidth and SetState_LineWidth methods.

Example

See also

IPCB_Arc interface

Radius property

(IPCB_Arc interface)

Syntax

```
Property Radius : TCoord Read GetState_Radius Write SetState_Radius;
```

Description

The Radius property denotes the radius of the arc. It is supported by the GetState_Radius and SetState_Radius methods.

Example

See also

IPCB_Arc interface

StartY property

(IPCB_Arc interface)

Syntax

```
Property StartY : TCoord Read GetState_StartY;
```

Description

The StartY property denotes the end Y coordinate of the arc. It is supported by the GetState_StartY method.

Example

See also

IPCB_Arc interface

StartX property

(IPCB_Arc interface)

Syntax

```
Property StartX : TCoord Read GetState_StartX;
```

Description

The StartX property denotes the starting X coordinate of the arc. It is supported by the GetState_StartX method.

Example

See also

IPCB_Arc interface

StartAngle property

(IPCB_Arc interface)

Syntax

```
Property StartAngle : TAngle Read GetState_StartAngle Write SetState_StartAngle;
```

Description

The StartAngle property denotes the initial angle of the arc. It is supported by the GetState_StartAngle / SetState_StartAngle and complemented by the GetState_EndAngle/SetState_EndAngle methods.

Example

```
Arc := PCBServer.PCBObjectFactory(eArcObject,eNoDimension,eCreate_Default);
Arc.XCenter    := MilsToCoord(Board.XOrigin + 1800);
Arc.YCenter    := MilsToCoord(Board.YOrigin + 1800);
Arc.Radius     := MilsToCoord(200);
Arc.LineWidth  := MilsToCoord(50);
Arc.StartAngle := 0;
Arc.EndAngle   := 270;
Arc.Layer      := eBottomLayer;
```

See also

IPCB_Arc interface

XCenter property

(IPCB_Arc interface)

Syntax

```
Property XCenter : TCoord Read GetState_CenterX Write SetState_CenterX;
```

Description

The XCenter property denotes the X coordinate of the center of the arc. It is supported by the GetState_CenterX and SetState_CenterX methods.

Example

See also

IPCB_Arc interface

YCenter property

(IPCB_Arc interface)

Syntax

```
Property YCenter : TCoord Read GetState_CenterY Write SetState_CenterY;
```

Description

The YCenter property denotes the X coordinate of the center of the arc. It is supported by the GetState_CenterY and SetState_CenterY methods.

Example

See also

IPCB_Arc interface

IPCB_BoardOutline

Overview

The board outline object represents the board shape which defines the extents or boundary of the board in the PCB Editor. A board outline object is essentially a closed polygon and is inherited from the **IPCB_Polygon** interface.

The PCB Editor uses the board outline shape to determine the extents of the power planes for plane edge pull back, used when splitting power planes and for calculating the board edge when design data is exported to other tools such as the 3D viewer tool.

A board outline is a group object therefore it is composed of pull back primitives namely tracks and arcs as the vertices for the closed polygon of the board outline. Although the board outline object interface is inherited from the **IPCB_Polygon** interface, you cannot use layer, net assignment and repour polygon behaviours for a board outline.

The **IPCB_BoardOutline** interface hierarchy is as follows;

IPCB_Primitive

IPCB_Group

IPCB_BoardOutline

Notes

The **IPCB_BoardOutline** interface is inherited from **IPCB_Polygon** interface and in turn from **IPCB_Group** interface.

To iterate the board outline for the pullback primitives, you create and use a group iterator because the board outline is a group object which in turn is composed of child objects.

The **IPCB_BoardOutline** interface is used by the **BoardOutline** property from the **IPCB_Board** interface.

Each new PCB document in Altium Designer is created with a board outline, so if you wish to update a board outline of a PCB document, you modify the existing board outline by massaging the board outline's vertices and then update the board outline..

IPCB_Group methods

```
FreePrimitives
GetPrimitiveAt
GetPrimitiveCount
SetState_XSizeYSize
FastSetState_XSizeYSize
SetState_LayersUsedArray
GroupIterator_Create
GroupIterator_Destroy
AddPCBObject
RemovePCBObject
```

IPCB_Group properties

```
X
Y
PrimitiveLock
LayerUsed
```

IPCB_BoardOutline methods

```
GetState_HitPrimitive
Rebuild
Validate
Invalidate
InvalidatePlane
```

IPCB_BoardOutline properties

Example

```
Procedure Query_Board_Outline;
Var
    PCB_Board : IPCB_Board;
    BR        : TCoordRect;
    NewUnit   : TUnit;
Begin
```

```
    PCB_Board := PCBServer.GetCurrentPCBBoard;
    If PCB_Board = Nil Then Exit;
    If PCB_Board.IsLibrary Then Exit;

    PCB_Board.BoardOutline.Invalidate;
    PCB_Board.BoardOutline.Rebuild;
    PCB_Board.BoardOutline.Validate;

    // The BoundingRectangle method is defined in IPCB_Primitive interface
    BR := PCB_Board.BoardOutline.BoundingRectangle;
    If PCB_Board.DisplayUnit = eImperial Then NewUnit := eMetric
                                         Else NewUnit := eImperial;
    ShowMessage(
        'Board Outline Width  : ' +
        CoordUnitToString(BR.right - BR.left,
                          PCB_Board.DisplayUnit) + #13 +
        'Board Outline Height : ' +
        CoordUnitToString(BR.top - BR.bottom,
                          PCB_Board.DisplayUnit));
End;
```

See also

PCB Design Objects

PCB_Primitive interface

IPCB_Group interface

IPCB_Polygon interface

IPCB_GroupIterator interface

PCB_Outline script in **Example**s\Scripts\Delphiscript\PCB folder.

BoardOutlineDetails script in **Example**s\Scripts\Delphiscript\PCB folder.

Methods

GetState_HitPrimitive method

(IPCB_BoardOutline interface)

Syntax

```
Function  GetState_HitPrimitive (APrimitive : IPCB_Primitive)  : Boolean;
```

Description

This function checks if a primitive that is not part of the board outline is touching or overlapping on the edge of the outline (whether being touched or enclosed by the outline).

This primitive could be placed by the user or created and placed programmatically. If the result is false, it means the primitive is definitely outside the outline.

Example

See also

IPCB_BoardOutline interface

Invalidate method

(IPCB_BoardOutline interface)

Syntax

```
Procedure Invalidate;
```

Description

This procedure renders the board outline in an invalidated state. This state needs to be rebuilt and validated by the system.

Example

See also

IPCB_BoardOutline interface

Validate method

InvalidatePlane method

(IPCB_BoardOutline interface)

Syntax

```
Procedure InvalidatePlane(Layer : TLayer);
```

Description

This procedure invalidates the specified layer the board outline is connected to, because the outline has been modified and this particular layer needs to be rebuilt.

Example

See also

IPCB_BoardOutline interface

Rebuild method

(IPCB_BoardOutline interface)

Syntax

```
Procedure Rebuild;
```

Description

This Rebuild procedure is called by the Validate method. This method rebuilds the board outline after it has been graphically altered which potentially could affect the internal/split planes that are connected to this outline.

Example

See also

IPCB_BoardOutline interface

Validate method

(IPCB_BoardOutline interface)

Syntax

```
Procedure Validate;
```

Description

The Validate method refreshes and updates the board outline object and its connections to the internal/split planes after it has been altered programmatically (layers or the coordinates of the outline).

The Rebuild method is called implicitly by the Validate method, so executing the Invalidate then the Valid methods are sufficient when the coordinates of a board outline has been modified programmatically.

Example

See also

IPCB_BoardOutline interface

IPCB_Component Interface

Overview

Components are defined by footprints, which are stored in a PCB library (or part of an integrated library). Note, a footprint can be linked to a schematic component.

When a footprint is placed in the workspace, it is assigned a designator (and optional comment). It is then referred to as a component. A component is composed of primitives (normally tracks, arcs, and pads).

Components are defined by footprints, which are stored in a PCB library. When a footprint is placed in the workspace, it is assigned a designator (and optional comment). It is then referred to as a component with the defined reference. The origin in the library editor defines the reference point of a footprint.

The **IPCB_Component** interface hierarchy is as follows;

IPCB_Primitive

IPCB_Group

IPCB_Component

Notes

The reference point of a component is set by the X,Y fields inherited from **IPCB_Group** interface. You can obtain the bounding rectangle of the component and calculate the mid point X and Y values to enable rotation about the center of the component if desired.

The rotation property of a component is set according to the reference point of a component, therefore the Rotation property and the RotateAroundXY method are equivalent only if you use the X,Y parameters for the RotateAroundXY method that are the same as the reference point of the component.

A component is a group object and therefore composes of child objects such as arcs and tracks. You use a group iterator to fetch the child objects for that component.

The **IPCB_Component** interface hierarchy is as follows;

IPCB_Group methods	**IPCB_Group properties**
FreePrimitives	X
GetPrimitiveAt	Y
GetPrimitiveCount	PrimitiveLock
SetState_XSizeYSize	LayerUsed
FastSetState_XSizeYSize	
SetState_LayersUsedArray	
GroupIterator_Create	
GroupIterator_Destroy	
AddPCBObject	
RemovePCBObject	

IPCB_Component methods	**IPCB_Component properties**
GetState_ChannelOffset	ChannelOffset
GetState_ComponentKind	ComponentKind
GetState_Name	Name
GetState_Comment	Comment
GetState_Pattern	Pattern
GetState_NameOn	NameOn
GetState_CommentOn	CommentOn
GetState_LockStrings	LockStrings
GetState_GroupNum	GroupNum
GetState_UnionIndex	UnionIndex
GetState_Rotation	Rotation
GetState_Height	Height

GetState_NameAutoPos
GetState_CommentAutoPos
GetState_SourceDesignator
GetState_SourceUniqueId
GetState_SourceHierarchicalPath
GetState_SourceFootprintLibrary
GetState_SourceComponentLibrary
GetState_SourceLibReference
GetState_SourceDescription
GetState_FootprintDescription
GetState_DefaultPCB3DModel
GetState_IsBGA
BoundingRectangleNoNameComment
BoundingRectangleNoNameCommentForSignals

SetState_ChannelOffset
SetState_ComponentKind
SetState_Pattern
SetState_NameOn
SetState_CommentOn
SetState_LockStrings
SetState_GroupNum
SetState_UnionIndex
SetState_Rotation
SetState_Height
SetState_NameAutoPos
SetState_CommentAutoPos
SetState_SourceDesignator
SetState_SourceUniqueId
SetState_SourceHierarchicalPath
SetState_SourceFootprintLibrary
SetState_SourceComponentLibrary
SetState_SourceLibReference
SetState_SourceDescription
SetState_FootprintDescription
SetState_DefaultPCB3DModel

ChangeNameAutoposition
ChangeCommentAutoposition
SetState_xSizeySize
RotateAroundXY
FlipComponent
Rebuild
Getstate_PadByName
LoadCompFromLibrary

NameAutoPosition
CommentAutoPosition
SourceDesignator
SourceUniqueId
SourceHierarchicalPath
SourceFootprintLibrary
SourceComponentLibrary
SourceLibReference
SourceDescription
FootprintDescription
DefaultPCB3DModel

IsBGA
EnablePinSwapping
EnablePartSwapping

LoadFromLibrary
AutoPosition_NameComment

SetState_EnablePinSwapping
SetState_EnablePartSwapping
GetState_EnablePinSwapping
GetState_EnablePartSwapping

See also
PCB Design Objects
IPCB_Primitive interface
IPCB_Group interface
IPCB_GroupIterator interface
IPCB_Text interface
TComponentKind enumerated values
TTextAutoposition enumerated values

Methods

AutoPosition_NameComment method

(IPCB_Component interface)

Syntax

```
Procedure AutoPosition_NameComment;
```

Description

This procedure invokes the auto positioning of the name and comment objects associated with the component after the Name and Comment objects' positions have been updated.

Example

See also
IPCB_Component interface

ChangeCommentAutoposition method

(IPCB_Component interface)

Syntax

```
Function ChangeCommentAutoposition (Value : TTextAutoposition) : Boolean;
```

Description

Example

See also
IPCB_Component interface

ChangeNameAutoposition method

(IPCB_Component interface)

Syntax

```
Function ChangeNameAutoposition (Value : TTextAutoposition) : Boolean;
```

Description

Example

See also

IPCB_Component interface

FlipComponent method

(IPCB_Component interface)

Syntax

```
Procedure FlipComponent;
```

Description

This method flips the component from one layer to the other, for example top layer to the bottom layer.

Example

See also

IPCB_Component interface

Getstate_PadByName method

(IPCB_Component interface)

Syntax

```
Function Getstate_PadByName (S : TPCBString) : IPCB_Primitive;
```

Description

This method retrieves the pad object interface only if the pad's name is found which is associated with this component.

Example

See also

IPCB_Component interface

LoadFromLibrary method

(IPCB_Component interface)

Syntax

```
Function LoadFromLibrary : Boolean;
```

Description

This function refreshes the specified component from the library. If it is successful a true value is returned otherwise false.

Example

See also

IPCB_Component interface

LoadCompFromLibrary method

(IPCB_Component interface)

Syntax

```
Function LoadCompFromLibrary : Boolean;
```

Description

This function refreshes the component from the library. If it is successful a true value is returned otherwise false.

Example

See also

IPCB_Component interface

Rebuild method

(IPCB_Component interface)

Syntax

```
Procedure Rebuild;
```

Description

This procedure forces a rebuild of the whole component graphically.

Example

See also

IPCB_Component interface

RotateAroundXY method

(IPCB_Component interface)

Syntax

```
Procedure RotateAroundXY (AX,AY : TCoord;Angle : TAngle);
```

Description

This method rotates a component object on the PCB document about the AX, AY coordinates with an angle in degrees. To ensure the component rotates without moving about, pass in its midpoint (between X1,X2 and Y1, Y2) attributes for the AX,AY parameters or use the **Rotation** property.

Example

See also

IPCB_Component interface

Rotation property

SetState_xSizeySize method

(IPCB_Component interface)

Syntax

```
Function SetState_xSizeySize : Boolean;
```

Description

After a component has been rebuilt programmatically for example the name and comment positions have changed, do a SetState_xSizeySize method to update the bounding rectangle of the whole component.

Example

See also

IPCB_Component interface

GetState and SetState Methods

GetState_ChannelOffset method

(IPCB_Component interface)

Syntax

```
Function GetState_ChannelOffset : TChannelOffset;
```

Description

The ChannelOffset represents the Channel Offset parameter for the component. A channel offset denotes where the component is in a room especially when a room is being copied and a copy is created on the same document. The copies of rooms containing components are created based on their offsets.

This method is used for the ChannelOffset property.

Example

See also

IPCB_Component interface

GetState_Comment method

(IPCB_Component interface)

Syntax

```
Function GetState_Comment : IPCB_Text;
```

Description

This property denotes the comment object associated with the IPCB_Component component object on the PCB document.

This method is used for the Comment property.

Example

See also

IPCB_Component interface

GetState_CommentAutoPos method

(IPCB_Component interface)

Syntax

```
Function GetState_CommentAutoPos : TTextAutoposition;
```

Description

This property denotes that the Comment text object is to be positioned relative to the component object depending on what the **TTextAutoposition** parameter is.

This method is used by the **CommentAutoPos** property.

Example

See also

IPCB_Component interface

GetState_CommentOn method

(IPCB_Component interface)

Syntax

```
Function GetState_CommentOn : Boolean;
```

Description

The CommentOn property denotes the visibility of the Name object associated with the component.

This method is used for the CommentOn property.

Example

See also

IPCB_Component interface

GetState_ComponentKind method

(IPCB_Component interface)

Syntax

```
Function GetState_ComponentKind : TComponentKind;
```

Description

A component kind can be one of the following:

eComponentKind_Standard: These components possess standard electrical properties, are always synchronized and are the type most commonly used on a board.

eComponentKind_Mechanical: These components do not have electrical properties and will appear in the BOM. They are synchronized if the same components exist on both the Schematic and PCB documents. An example is a heatsink.

eComponentKind_Graphical: These components are not used during synchronization or checked for electrical errors. These components are used, for example, when adding company logos to documents.

eComponentKind_NetTie_BOM: These components short two or more different nets for routing and these components will appear in the BOM and are maintained during synchronization.

eComponentKind_NetTie_NoBOM: These components short two or more different nets for routing and these components will NOT appear in the BOM and are maintained during synchronization. Note

Note, the TComponentKind type is defined from RT_Workspace unit.

This method is used by the ComponentKind property.

Example

See also

IPCB_Component interface

GetState_DefaultPCB3DModel method

(IPCB_Component interface)

Syntax

```
Function GetState_DefaultPCB3DModel : TPCBString;
```

Description

The DefaultPCB3DModel method denotes the default PCB 3D Model name as the default to be linked to this PCB component.

This method is used for the DeafultPCB3DModel property.

Example

See also

IPCB_Component interface

GetState_FootprintDescription method

(IPCB_Component interface)

Syntax

```
Function GetState_FootprintDescription : TPCBString;
```

Description

This property denotes the descriptive account of the footprint. This method is used for the Footprint**Description** property.

Example

See also

IPCB_Component interface

GetState_GroupNum method

(IPCB_Component interface)

Syntax

```
Function GetState_GroupNum : Integer;
```

Description

This GroupNum is not used internally. Can use for specific purposes such as a tag or an index.

This GroupNum method is used for the GroupNum property.

Example

See also

IPCB_Component interface

GetState_Height method

(IPCB_Component interface)

Syntax

```
Function GetState_Height : TCoord;
```

Description

The height of the component denotes the height of the component. It is used for the 3D viewer which works out the heights of components before displaying components in a 3D view.

This method is used for the Height property.

Example

See also

IPCB_Component interface

GetState_LockStrings method

(IPCB_Component interface)

Syntax

```
Function GetState_LockStrings : Boolean;
```

Description

The LockStrings property of the component denotes whether the strings of a component can be locked or not. This method is used for the LockStrings property.

Example

See also

IPCB_Component interface

GetState_Name method

(IPCB_Component interface)

Syntax

```
Function GetState_Name : IPCB_Text;
```

Description

This property denotes the name object associated with the IPCB_Component component object on the PCB document.

This method is used for the Name property.

Example

See also

IPCB_Component interface

GetState_NameAutoPos method

(IPCB_Component interface)

Syntax

```
Function GetState_NameAutoPos : TTextAutoposition;
```

Description

The CommentAutoPos denotes that the Comment text object is to be positioned relative to the component object depending on what the **TTextAutoposition** parameter is.

This method is used for the CommentAutoPos property.

Example

See also

IPCB_Component interface

GetState_NameOn method

(IPCB_Component interface)

Syntax

```
Function GetState_NameOn : Boolean;
```

Description

The NameOn property denotes the visibility of the Name object associated with the component.

This method is used for the NameOn property.

Example

See also

IPCB_Component interface

GetState_Pattern method

(IPCB_Component interface)

Syntax

```
Function GetState_Pattern : TPCBString;
```

Description

The Pattern denotes the footprint name of this component which is a widestring. This method is used for the Pattern property.

Example

See also

IPCB_Component interface

GetState_Rotation method

(IPCB_Component interface)

Syntax

```
Function GetState_Rotation : TAngle;
```

Description

The Rotation of the component denotes the angle of the component with respect to the horizontal axis. The rotation parameter of **TAngle** type is between 0 and 360 degrees inclusive.

This method is used for the **Rotation** property.

Example

See also

IPCB_Component interface

GetState_SourceComponentLibrary method

(IPCB_Component interface)

Syntax

```
Function GetState_SourceComponentLibrary : TPCBString;
```

Description

This source library field denotes the integrated library where the PCB component comes from. Note: When a schematic is transferred to a blank PCB using the Update command, the source reference links for each PCB footprint is populated with source library path names.

This method is used for the SourceComponentLibrary property.

Example

See also

IPCB_Component interface

GetState_SourceDescription method

(IPCB_Component interface)

Syntax

```
Function GetState_SourceDescription : TPCBString;
```

Description

This method can include a descriptive account of the reference link to a source component or a device name.

This method is used for the SourceDescription property.

Example

See also

IPCB_Component interface

GetState_SourceDesignator method

(IPCB_Component interface)

Syntax

```
Function GetState_SourceDesignator : TPCBString;
```

Description

This method represents the current designator of the source component from the corresponding schematic.

This method is used for the SourceDesignator property.

Example

See also

IPCB_Component interface

GetState_SourceFootprintLibrary method

(IPCB_Component interface)

Syntax

```
Function GetState_SourceFootprintLibrary : TPCBString;
```

Description

This method denotes the descriptive account of the footprint. This method is used for the SourceFootprintLibrary property.

Example

See also

IPCB_Component interface

GetState_SourceHierarchicalPath method

(IPCB_Component interface)

Syntax

```
Function GetState_SourceHierarchicalPath : TPCBString;
```

Description

This uniquely identifies the source reference path to the PCB component. The path can be multi-level depending on whether it is a multi channel (sheet symbols) or a normal design (schematic sheets).

Note: When a schematic is transferred to a blank PCB using the Update command, the source reference links for each PCB footprint is populated with source library path names.

This method is used for the SourceHierarchicalPath property.

Example

See also

IPCB_Component interface

GetState_SourceLibReference method

(IPCB_Component interface)

Syntax

```
Function GetState_SourceLibReference : TPCBString;
```

Description

The source library reference property is the name of the component from the library. This method is used for the SourceLibReference property.

Example

See also

IPCB_Component interface

GetState_SourceUniqueId method

(IPCB_Component interface)

Syntax

```
Function GetState_SourceUniqueId : TPCBString;
```

Description

Unique IDs (UIDs) are used to match each schematic component to the corresponding PCB component. When a schematic is transferred to a blank PCB using the Update command, the source reference links for each PCB footprint is populated with source library pathnames.

The Unique Identifier (UID) is a system generated value that uniquely identifies the source component.

This method is used for the SourceUniqueID property.

Example

See also

IPCB_Component interface

GetState_UnionIndex method

(IPCB_Component interface)

Syntax

```
Function GetState_UnionIndex : Integer;
```

Description

The UnionIndex property denotes the union index. Unions are sets of components that will be manipulated as a block for the PCB placement. Components in a union maintain their relative positions within the union as they are moved for example.

This method is used for the UnionIndex property.

Example

See also

IPCB_Component interface

SetState_ChannelOffset method

(IPCB_Component interface)

Syntax

```
Procedure SetState_ChannelOffset (Value : TChannelOffset);
```

Description

The ChannelOffset represents the Channel Offset parameter for the component. A channel offset denotes where the component is in a room especially when a room is being copied and a copy is created on the same document. The copies of rooms containing components are created based on their offsets.

This method is used for the ChannelOffset property.

Example

See also

IPCB_Component interface

SetState_CommentAutoPos method

(IPCB_Component interface)

Syntax

```
Procedure SetState_CommentAutoPos (Value : TTextAutoposition);
```

Description

This property denotes that the Comment text object is to be positioned relative to the component object depending on what the **TTextAutoposition** parameter is.

This method is used by the **CommentAutoPos** property.

Example

See also

IPCB_Component interface

SetState_CommentOn method

(IPCB_Component interface)

Syntax

```
Procedure SetState_CommentOn (Value : Boolean);
```

Description

The CommentOn property denotes the visibility of the Comment object associated with the component. This method is used for the CommentOn property.

Example

See also

IPCB_Component interface

SetState_ComponentKind method

(IPCB_Component interface)

Syntax

```
Procedure SetState_ComponentKind (Value : TComponentKind);
```

Description

A component kind can be one of the following:

eComponentKind_Standard: These components possess standard electrical properties, are always synchronized and are the type most commonly used on a board.

eComponentKind_Mechanical: These components do not have electrical properties and will appear in the BOM. They are synchronized if the same components exist on both the Schematic and PCB documents. An example is a heatsink.

eComponentKind_Graphical: These components are not used during synchronization or checked for electrical errors. These components are used, for example, when adding company logos to documents.

eComponentKind_NetTie_BOM: These components short two or more different nets for routing and these components will appear in the BOM and are maintained during synchronization.

eComponentKind_NetTie_NoBOM: These components short two or more different nets for routing and these components will NOT appear in the BOM and are maintained during synchronization. Note

Note, the TComponentKind type is defined from RT_Workspace unit.

This method is used by the ComponentKind property.

Example

See also

IPCB_Component interface

SetState_DefaultPCB3DModel method

(IPCB_Component interface)

Syntax

```
Procedure SetState_DefaultPCB3DModel (Value : TPCBString);
```

Description

The DefaultPCB3DModel method denotes the default PCB 3D Model name as the default to be linked to this PCB component.

This method is used for the DeafultPCB3DModel property.

Example

See also

IPCB_Component interface

SetState_FootprintDescription method

(IPCB_Component interface)

Syntax

```
Procedure SetState_FootprintDescription (Value : TPCBString);
```

Description

This property denotes the descriptive account of the footprint. This method is used for the Footprint**Description** property.

Example

See also

IPCB_Component interface

SetState_GroupNum method

(IPCB_Component interface)

Syntax

```
Procedure SetState_GroupNum (Value : Integer);
```

Description

This GroupNum is not used internally. Can use for specific purposes such as a tag or an index.

This GroupNum method is used for the GroupNum property.

Example

See also

IPCB_Component interface

SetState_Height method

(IPCB_Component interface)

Syntax

```
Procedure SetState_Height (Value : TCoord);
```

Description

The height of the component denotes the height of the component. It is used for the 3D viewer which works out the heights of components before displaying components in a 3D view.

This method is used for the Height property.

Example

See also

IPCB_Component interface

SetState_LockStrings method

(IPCB_Component interface)

Syntax

```
Procedure SetState_LockStrings (Value : Boolean);
```

Description

The LockStrings property of the component denotes whether the strings of a component can be locked or not. This method is used for the LockStrings property.

Example

See also

IPCB_Component interface

SetState_NameAutoPos method

(IPCB_Component interface)

Syntax

```
Procedure SetState_NameAutoPos (Value : TTextAutoposition);
```

Description

The NameAutoPos denotes that the Name text object is to be positioned relative to the component object depending on what the **TTextAutoposition** parameter is.

This method is used for the NameAutoPos property.

Example

See also

IPCB_Component interface

SetState_NameOn method

(IPCB_Component interface)

Syntax

```
Procedure SetState_NameOn (Value : Boolean);
```

Description

The NameOn property denotes the visibility of the Name object associated with the component.

This method is used for the NameOn property.

Example

See also

IPCB_Component interface

SetState_Pattern method

(IPCB_Component interface)

Syntax

```
Procedure SetState_Pattern (Value : TPCBString);
```

Description

The Pattern denotes the footprint name of this component which is a widestring. This method is used for the Pattern property.

Example

See also

IPCB_Component interface

SetState_Rotation method

(IPCB_Component interface)

Syntax

```
Procedure SetState_Rotation (Value : TAngle);
```

Description

The Rotation of the component denotes the angle of the component with respect to the horizontal axis. The rotation parameter of **TAngle** type is between 0 and 360 degrees inclusive.

This method is used for the Rotation property.

Example

See also

IPCB_Component interface

SetState_SourceComponentLibrary method

(IPCB_Component interface)

Syntax

```
Procedure SetState_SourceComponentLibrary(Value : TPCBString);
```

Description

This source library field denotes the integrated library where the PCB component comes from. Note: When a schematic is transferred to a blank PCB using the Update command, the source reference links for each PCB footprint is populated with source library path names.

This method is used for the SourceComponentLibrary property.

Example

See also

IPCB_Component interface

SetState_SourceDescription method

(IPCB_Component interface)

Syntax

```
Procedure SetState_SourceDescription (Value : TPCBString);
```

Description

This method can include a descriptive account of the reference link to a source component or a device name.

This method is used for the Source**Description** property.

Example

See also

IPCB_Component interface

SetState_SourceDesignator method

(IPCB_Component interface)

Syntax

```
Procedure SetState_SourceDesignator (Value : TPCBString);
```

Description

This method represents the current designator of the source component from the corresponding schematic.

This method is used for the SourceDesignator property.

Example

See also

IPCB_Component interface

SetState_SourceFootprintLibrary method

(IPCB_Component interface)

Syntax

```
Procedure SetState_SourceFootprintLibrary(Value : TPCBString);
```

Description

This method denotes the descriptive account of the footprint. This method is used for the SourceFootprintLibrary property.

Example

See also

IPCB_Component interface

SetState_SourceHierarchicalPath method

(IPCB_Component interface)

Syntax

```
Procedure SetState_SourceHierarchicalPath(Value : TPCBString);
```

Description

This uniquely identifies the source reference path to the PCB component. The path can be multi-level depending on whether it is a multi channel (sheet symbols) or a normal design (schematic sheets).

Note: When a schematic is transferred to a blank PCB using the Update command, the source reference links for each PCB footprint is populated with source library path names.

This method is used for the SourceHierarchicalPath property.

Example

See also

IPCB_Component interface

SetState_SourceLibReference method

(IPCB_Component interface)

Syntax

```
Procedure SetState_SourceLibReference (Value : TPCBString);
```

Description

The source library reference property is the name of the component from the library. This method is used for the SourceLibReference property.

Example

See also

IPCB_Component interface

SetState_SourceUniqueId method

(IPCB_Component interface)

Syntax

```
Procedure SetState_SourceUniqueId (Value : TPCBString);
```

Description

Unique IDs (UIDs) are used to match each schematic component to the corresponding PCB component. When a schematic is transferred to a blank PCB using the Update command, the source reference links for each PCB footprint is populated with source library pathnames.

The Unique Identifier (UID) is a system generated value that uniquely identifies the source component.

This method is used for the SourceUniqueId property.

Example

See also

IPCB_Component interface

SetState_UnionIndex method

(IPCB_Component interface)

Syntax

```
Procedure SetState_UnionIndex (Value : Integer);
```

Description

The UnionIndex property denotes the union index. Unions are sets of components that will be manipulated as a block for the PCB placement. Components in a union maintain their relative positions within the union as they are moved for example.

This method is used for the UnionIndex property.

Example

See also

IPCB_Component interface

Properties

ChannelOffset property

(IPCB_Component interface)

Syntax

```
Property ChannelOffset : TChannelOffset Read GetState_ChannelOffset Write
SetState_ChannelOffset;
```

Description

The ChannelOffset represents the Channel Offset parameter for the component. A channel offset denotes where the component is in a room especially when a room is being copied and a copy is created on the same document. The copies of rooms containing components are created based on their offsets.

This property is supported by the GetState_ChannelOffset and SetState_ChannelOffset methods.

Example

See also

IPCB_Component interface

Comment property

(IPCB_Component interface)

Syntax

```
Property Comment : IPCB_Text Read GetState_Comment;
```

Description

This property denotes the comment object associated with the IPCB_Component component object on the PCB document.

This read only property is supported by the GetState_Comment method.

Example

See also

IPCB_Component interface

IPCB_Text interface

CommentAutoPosition property

(IPCB_Component interface)

Syntax

```
Property CommentAutoPosition : TTextAutoposition Read GetState_CommentAutoPos Write
SetState_CommentAutoPos;
```

Description

This property denotes that the Comment text object is to be positioned relative to the component object depending on what the **TTextAutoposition** parameter is.

This property is supported by the GetState_CommentAutoPosition and SetState_CommentAutoPosition methods.

Example

See also

IPCB_Component interface

TTextAutoposition type

CommentOn property

(IPCB_Component interface)

Syntax

```
Property CommentOn : Boolean Read GetState_CommentOn Write SetState_CommentOn;
```

Description

The CommentOn property denotes the visibility of the Comment object associated with the component.

This property is supported by the GetState_CommentOn and SetState_CommentOn methods.

Example

See also

IPCB_Component interface

ComponentKind property

(IPCB_Component interface)

Syntax

```
Property ComponentKind : TComponentKind Read GetState_ComponentKind Write
SetState_ComponentKind;
```

Description

A component kind can be one of the following:

eComponentKind_Standard: These components possess standard electrical properties, are always synchronized and are the type most commonly used on a board.

eComponentKind_Mechanical: These components do not have electrical properties and will appear in the BOM. They are synchronized if the same components exist on both the Schematic and PCB documents. An example is a heatsink.

eComponentKind_Graphical: These components are not used during synchronization or checked for electrical errors. These components are used, for example, when adding company logos to documents.

eComponentKind_NetTie_BOM: These components short two or more different nets for routing and these components will appear in the BOM and are maintained during synchronization.

eComponentKind_NetTie_NoBOM: These components short two or more different nets for routing and these components will NOT appear in the BOM and are maintained during synchronization. Note

Note, the TComponentKind type is defined from RT_Workspace unit.

This property is supported by the GetState_ComponentKind and SetState_ComponentKind methods.

Example

See also

IPCB_Component interface

TComponentKind type in the RT_Workspace unit.

DefaultPCB3DModel property

(IPCB_Component interface)

Syntax

```
Property DefaultPCB3DModel : TPCBString Read GetState_DefaultPCB3DModel Write
SetState_DefaultPCB3DModel;
```

Description

The property denotes the default PCB 3D Model name as the default to be linked to this PCB component.

This property is supported by the GetState_DefaultPCB3DModel and SetState_DefaultPCB3DModel methods.

Example

See also

IPCB_Component interface

FootprintDescription property

(IPCB_Component interface)

Syntax

```
Property FootprintDescription : TPCBString Read GetState_FootprintDescription Write
SetState_FootprintDescription;
```

Description

This property denotes the descriptive account of the footprint.

This property is supported by the GetState_Footprint**Description** and SetState_Footprint**Description** methods.

Example

See also

IPCB_Component interface

GroupNum property

(IPCB_Component interface)

Syntax

```
Property GroupNum : Integer Read GetState_GroupNum Write SetState_GroupNum;
```

Description

This property is not used internally. Can use for specific purposes such as a tag or an index.

This property is supported by the GetState_GroupNum and SetState_GroupNum methods.

Example

See also

IPCB_Component interface

Height property

(IPCB_Component interface)

Syntax

```
Property Height : TCoord Read GetState_Height Write SetState_Height;
```

Description

The height property denotes the height of the component. It is used for the 3D viewer which works out the heights of components before displaying components in a 3D view.

This property is supported by the GetState_Height and SetState_Height methods.

Example

See also

IPCB_Component interface

LockStrings property

(IPCB_Component interface)

Syntax

```
Property LockStrings : Boolean Read GetState_LockStrings Write SetState_LockStrings;
```

Description

The LockStrings property denotes whether the strings of a component can be locked or not.

This property is supported by the GetState_LockStrings and SetState_LockStrings methods.

Example

See also

IPCB_Component interface

Name property

(IPCB_Component interface)

Syntax

```
Property Name : IPCB_Text Read GetState_Name;
```

Description

This property denotes the name object associated with the IPCB_Component object on the PCB document and represents the pattern string.

This read only property is supported by the GetState_Name method.

Example

See also

IPCB_Component interface

IPCB_Text interface

NameAutoPosition property

(IPCB_Component interface)

Syntax

```
Property NameAutoPosition : TTextAutoposition Read GetState_NameAutoPos Write
SetState_NameAutoPos;
```

Description

This property denotes that the Name text object is to be positioned relative to the component object depending on what the **TTextAutoposition** parameter is.

This property is supported by the GetState_NameAutoPos and SetState_NameAutoPos methods.

Example

See also

IPCB_Component interface

TTextAutoposition type

NameOn property

(IPCB_Component interface)

Syntax

```
Property NameOn : Boolean Read GetState_NameOn Write SetState_NameOn;
```

Description

The NameOn property denotes the visibility of the Name object associated with the component.

This property is supported by the GetState_NameOn and SetState_NameOn methods.

Example

See also

IPCB_Component interface

Pattern property

(IPCB_Component interface)

Syntax

```
Property Pattern : TPCBString Read GetState_Pattern Write SetState_Pattern;
```

Description

The property denotes the footprint name of this component which is a widestring.

This property is supported by the GetState_Pattern and SetState_Pattern methods.

Example

See also

IPCB_Component interface

Rotation property

(IPCB_Component interface)

Syntax

```
Property Rotation : TAngle Read GetState_Rotation Write SetState_Rotation;
```

Description

This property denotes the angle of the component with respect to the horizontal axis. The rotation parameter of **TAngle** type is between 0 and 360 degrees inclusive.

This property is supported by the GetState_Rotation and SetState_Rotation methods.

Example

See also

IPCB_Component interface

TAngle type

SourceComponentLibrary property

(IPCB_Component interface)

Syntax

```
Property SourceComponentLibrary : TPCBString Read GetState_SourceComponentLibrary Write
SetState_SourceComponentLibrary;
```

Description

This source library field denotes the integrated library where the PCB component comes from. Note: When a schematic is transferred to a blank PCB using the Update command, the source reference links for each PCB footprint is populated with source library path names.

This property is supported by the GetState_SourceComponentLibrary and SetState_SourceComponentLibrary methods.

Example

See also

IPCB_Component interface

SourceDescription **property**

(IPCB_Component interface)

Syntax

```
Property SourceDescription : TPCBString Read GetState_SourceDescription Write
SetState_SourceDescription;
```

Description

This property can include a descriptive account of the reference link to a source component or a device name.

This property is supported by the GetState_Source**Description** and SetState_Source**Description** methods.

Example

See also

IPCB_Component interface

SourceDesignator property

(IPCB_Component interface)

Syntax

```
Property SourceDesignator : TPCBString Read GetState_SourceDesignator Write
SetState_SourceDesignator;
```

Description

This property represents the current designator of the source component from the corresponding schematic.

This property is supported by the GetState_SourceDesignator and SetState_SourceDesignator methods.

Example

See also

IPCB_Component interface

SourceFootprintLibrary property

(IPCB_Component interface)

Syntax

```
Property SourceFootprintLibrary : TPCBString Read GetState_SourceFootprintLibrary Write
SetState_SourceFootprintLibrary;
```

Description

This field shows the name of the footprint. The footprint is the graphical representation of a PCB component and is used to display it on the PCB, and usually contains component outline and connection pads along with an unique designator.

Footprints are stored in PCB library files or Integrated libraries, which can be edited using the PCB Library Editor to create new footprints or edit existing ones.

This property is supported by the GetState_SourceFootprintLibrary and SetState_SourceFootprintLibrary methods.

Example

See also

IPCB_Component interface

SourceHierarchicalPath property

(IPCB_Component interface)

Syntax

```
Property SourceHierarchicalPath : TPCBString Read GetState_SourceHierarchicalPath Write
SetState_SourceHierarchicalPath;
```

Description

This property uniquely identifies the source reference path to the PCB component. The path can be multi-level depending on whether it is a multi channel (sheet symbols) or a normal design (schematic sheets).

Note: When a schematic is transferred to a blank PCB using the Update command, the source reference links for each PCB footprint is populated with source library path names.

This property is supported by the GetState_SourceHierarchicalPath and SetState_SourceHierarchicalPath methods.

Example

See also

IPCB_Component interface

SourceLibReference property

(IPCB_Component interface)

Syntax

```
Property SourceLibReference : TPCBString Read GetState_SourceLibReference Write
SetState_SourceLibReference;
```

Description

The source library reference property is the name of the component from the library.

This property is supported by the GetState_SourceLibReference and SetState_SourceLibReference methods.

Example

See also

IPCB_Component interface

SourceUniqueId property

(IPCB_Component interface)

Syntax

```
Property SourceUniqueId : TPCBString Read GetState_SourceUniqueId Write
SetState_SourceUniqueId;
```

Description

Unique IDs (UIDs) are used to match each schematic component to the corresponding PCB component. When a schematic is transferred to a blank PCB using the Update command, the source reference links for each PCB footprint is populated with source library pathnames.

The Unique Identifier (UID) is a system generated value that uniquely identifies the source component.

This property is supported by the GetState_SourceUniqueId and SetState_SourceUniqueId methods.

Example

See also

IPCB_Component interface

UnionIndex property

(IPCB_Component interface)

Syntax

```
Property UnionIndex : Integer Read GetState_UnionIndex Write SetState_UnionIndex;
```

Description

The property denotes the union index. Unions are sets of components that will be manipulated as a block for the PCB placement. Components in a union maintain their relative positions within the union as they are moved for example.

The UnionIndex property is supported by the GetState_UnionIndex and SetState_UnionIndex methods.

Example

See also

IPCB_Component interface

EnablePinSwapping property

(IPCB_Component interface)

Syntax

```
Property  EnablePinSwapping                 : Boolean             Read
GetState_EnablePinSwapping      Write SetState_EnablePinSwapping ;
```

Description

The property denotes the pin swapping for the pins of this component. In this case, these pins can be swapped if the EnablePinSwapping is set to true.

The EnablePinSwapping property is supported by the GetState_EnablePinSwapping and SetState_EnablePinSwapping methods.

Example

See also

IPCB_Component interface

EnablePartSwapping property

(IPCB_Component interface)

Syntax

```
Property  EnablePartSwapping : Boolean Read GetState_EnablePartSwapping Write
SetState_EnablePartSwapping;
```

Description

The property denotes the part swapping. Components can have multi-parts and in this case, these multi parts can be swapped if the EnablePartSwapping is set to True.

The UnionIndex property is supported by the GetState_EnablePartSwapping and SetState_EnablePartSwapping methods.

Example

See also

IPCB_Component interface

IPCB_ComponentBody Interface

Overview

A component body is a body that encapsulates a component in 3 dimensions on a PCB document. Component bodies are handled in the same way as other primitives, and they are contained in the component itself, whether in a library or on a board.

A component body object is a group object that contain child objects, thus in order to retrieve component bodies from within a component, use an iterator on this component.

The **IPCB_ComponentBody** interface hierarchy is as follows;

IPCB_ComponentBody methods

IPCB_ComponentBody properties

GetStandoffHeight
GetOverallHeight
GetBodyProjection
SetStandoffHeight
SetOverallHeight
SetBodyProjection

StandoffHeight
OverallHeight
BodyProjection

See also

IPCB_Component interface

Methods

SetStandoffHeight method

(IPCB_ComponentBody interface)

Syntax

```
Procedure SetStandoffHeight(Value : TCoord );
```

Description

Example

See also

IPCB_ComponentBody interface

SetOverallHeight method

(IPCB_ComponentBody interface)

Syntax

```
Procedure SetOverallHeight (Value : TCoord );
```

Description

Example

See also

IPCB_ComponentBody interface

SetBodyProjection method

(IPCB_ComponentBody interface)

Syntax

```
Procedure SetBodyProjection (Value : TBoardSide);
```

Description

Example

See also

IPCB_ComponentBody interface

GetStandoffHeight method

(IPCB_ComponentBody interface)

Syntax

```
Function GetStandoffHeight : TCoord;
```

Description

Example

See also

IPCB_ComponentBody interface

GetOverallHeight method

(IPCB_ComponentBody interface)

Syntax

```
Function GetOverallHeight : TCoord;
```

Description

Example

See also

IPCB_ComponentBody interface

GetBodyProjection method

(IPCB_ComponentBody interface)

Syntax

```
Function GetBodyProjection : TBoardSide;
```

Description

Example

See also

IPCB_ComponentBody interface

Properties

OverallHeight property

(IPCB_ComponentBody interface)

Syntax

```
Property OverallHeight : TCoord Read GetOverallHeight Write SetOverallHeight;
```

Description

Example

See also

IPCB_ComponentBody interface

BodyProjection property

(IPCB_ComponentBody interface)

Syntax

```
Property BodyProjection : TBoardSide Read GetBodyProjection Write SetBodyProjection;
```

Description

Example

See also

IPCB_ComponentBody interface

StandoffHeight property

(IPCB_ComponentBody interface)

Syntax

```
Property StandoffHeight : TCoord Read GetStandoffHeight Write SetStandoffHeight;
```

Description

Example

See also

IPCB_ComponentBody interface

IPCB_Coordinate

Overview

Coordinate markers are used to indicate the coordinates of specific points in a PCB workspace. A coordinate marker consists of a point marker and the X and Y coordinates of the position

The **IPCB_Coordinate** interface hierarchy is as follows;

IPCB_Primitive

IPCB_Group

IPCB_Coordinate

IPCB_Group methods	IPCB_Group properties
FreePrimitives	X
GetPrimitiveAt	Y
GetPrimitiveCount	PrimitiveLock
SetState_XSizeYSize	LayerUsed
FastSetState_XSizeYSize	
SetState_LayersUsedArray	
GroupIterator_Create	
GroupIterator_Destroy	
AddPCBObject	
RemovePCBObject	

IPCB_Coordinate methods	IPCB_Coordinate properties
GetState_Size	Size
GetState_LineWidth	LineWidth
GetState_TextHeight	TextHeight
GetState_TextWidth	TextWidth
GetState_TextFont	TextFont
GetState_Style	Style
GetState_Rotation	Rotation
GetState_UseTTFonts	
GetState_Bold	UseTTFonts
GetState_Italic	Bold
GetState_FontName	Italic

FontName
SetState_Size
SetState_LineWidth
SetState_TextHeight
SetState_TextWidth
SetState_TextFont
SetState_Style
SetState_Rotation
SetState_UseTTFonts
SetState_Bold
SetState_Italic
SetState_FontName

SetState_xSizeySize
RotateAroundXY
Text
Track1
Track2

GetState_StrictHitTest

Methods

SetState_xSizeySize
RotateAroundXY
Text
Track1
Track2

GetState_StrictHitTest

Properties

Size property

(IPCB_Coordinate interface)

Syntax

```
Property  Size : TCoord Read GetState_Size Write SetState_Size;
```

Description

The Size property determines the size of the coordinate object.

Example

See also

IPCB_Coordinate interface

LineWidth property

(IPCB_Coordinate interface)

Syntax

```
Property  LineWidth                          : TCoord              Read GetState_LineWidth
Write SetState_LineWidth;
```

Description

The LineWidth property determines the line width or the outline of the coordinate object.

Example

See also

IPCB_Coordinate interface

TextHeight property

(IPCB_Coordinate interface)

Syntax

```
Property  TextHeight : TCoord Read GetState_TextHeight Write SetState_TextHeight;
```

Description

The TextHeight property determines the text height of the coordinate object.

Example

See also

IPCB_Coordinate interface

TextWidth property

(IPCB_Coordinate interface)

Syntax

```
Property  TextWidth : TCoord Read GetState_TextWidth Write SetState_TextWidth;
```

Description

The TextHeight property determines the text width of the coordinate object.

Example

See also

```
IPCB_Coordinate interface
```

TextFont property

(IPCB_Coordinate interface)

Syntax

```
Property  TextFont : TFontID Read GetState_TextFont Write SetState_TextFont;
```

Description

The TextFont property determines the font id of TFontID type used for the coordinate object.

Example

See also

IPCB_Coordinate interface

```
TFontID
```

Style property

(IPCB_Coordinate interface)

Syntax

```
Property  Style : TUnitStyle Read GetState_Style Write SetState_Style;
```

Description

The Style property determines the style used for the measurement units of the coordinate object. Display no units, show units as Mils or MM or show Units with parenthesises.

Example

See also

IPCB_Coordinate interface

`TUnitStyle type`

Rotation property

(IPCB_Coordinate interface)

Syntax

```
Property  Rotation : TAngle Read GetState_Rotation Write SetState_Rotation;
```

Description

The Rotation property determines the coordinate object's orientation of TAngle type.

Example

See also

IPCB_Coordinate interface

`TAngle type`

UseTTFonts property

(IPCB_Coordinate interface)

Syntax

```
Property  UseTTFonts : Boolean Read GetState_UseTTFonts Write SetState_UseTTFonts;
```

Description

The UseTTFonts property determines whether the text of the coordinate object is of True Type Font type.

Example

See also

IPCB_Coordinate interface

`TAngle type`

Bold property

(IPCB_Coordinate interface)

Syntax

```
Property  Bold : Boolean Read GetState_Bold Write SetState_Bold;
```

Description

This property sets or gets the bold property of the PCB string True Type text on a PCB document. This property is supported by the GetState_Bold and SetState_Bold methods.

Once the UseTTFonts property is enabled, you can use the Bold, Italic, FontName, Inverted, InvertedTTTextBorder and TTTextOutline properties.

Example

See also

IPCB_Coordinate interface

Italic property

(IPCB_Coordinate interface)

Syntax

```
Property Italic : Boolean Read GetState_Italic Write SetState_Italic;
```

Description

The Italic property sets or gets the italic property of the PCB string True Type text on a PCB document. This property is supported by the GetState_Italic and SetState_Italic methods.

Once the UseTTFonts property is enabled, you can use the Bold, Italic, FontName, Inverted, InvertedTTTextBorder and TTTextOutline properties.

Example

See also

IPCB_Coordinate interface

FontName property

(IPCB_Coordinate interface)

Syntax

```
Property FontName : TPCBString Read GetState_FontName Write SetState_FontName;
```

Description

This property sets or gets the FontName property of the PCB string True Type text on a PCB document. For example one of the True Type font strings could be 'Arial', 'Arial Narrow', 'Courier New' and 'Verdana'. This property is supported by the GetState_Bold and SetState_Bold methods.

Once the UseTTFonts property is enabled, you can use the Bold, Italic, FontName, Inverted, InvertedTTTextBorder and TTTextOutline properties.

Delphiscript Script **Example**

```
TextObj := PCBServer.PCBObjectFactory(eTextObject, eNoDimension, eCreate_Default);

// notify that the pcb object is going to be modified
PCBServer.SendMessageToRobots(TextObj.I_ObjectAddress, c_Broadcast, PCBM_BeginModify,
c_NoEventData);
TextObj.XLocation := Sheet.SheetX + MilsToCoord(1000);
TextObj.YLocation := Sheet.SheetY + MilsToCoord(1000);
TextObj.Layer     := eBottomOverlay;
TextObj.UseTTFonts := True;
TextObj.Italic := True;
TextObj.Bold := False;
TextObj.FontName := 'ARIAL';
// inverts the text object and a text boundary is created around the text
// The Inverted and InvertedTTTextBorder properties are useful for situations
// if text is to be placed on a copper region and create a cutout in the region.
// the color of the inverted border is the layer color and the text color itself
// is black.
TextObj.Inverted := True;
// The InvertedTTTextBorder property determines the distance between the boundary of the
// the text object itself to the inverted text border boundary.
TextObj.InvertedTTTextBorder := MilsToCoord(100);
TextObj.Text      := 'Text with True Type Property enabled.';
TextObj.Size       := MilsToCoord(200);    // sets the height of the text.
```

See also

IPCB_Coordinate interface

See also

PCB Design Objects

IPCB_Primitive interface
IPCB_Group interface
IPCB_GroupIterator interface

IPCB_Connection Interface

Overview

The **IPCB_Connection** interface represents a connection between two nodes on a PCB document. The two nodes can be on two different layers and the connection style can be a connected line or a broken specially marked connection.

The IPCB_Connection hierarchy;

IPCB_Primitive
IPCB_Connection

IPCB_Connection methods
GetState_X1
GetState_Y1
GetState_X2
GetState_Y2
GetState_Layer1
GetState_Layer2
GetState_Mode

SetState_X1
SetState_Y1
SetState_X2
SetState_Y2
SetState_Layer1
SetState_Layer2
SetState_Mode

IsRedundant
RotateAroundXY

IPCB_Connection properties
X1
Y1
X2
Y2
Layer1
Layer2
Mode

See also

IPCB_Primitive interface
TLayer enumerated values
TConnectionMode enumerated values
PCB Design Objects

GetState and SetState Methods

GetState_Layer2 method

(IPCB_Connection interface)

Syntax

```
Function GetState_Layer2 : TLayer;
```

Description

This method retrieves the Layer 2 attribute which represents a connection from the first layer to the second layer on a PCB document. This function is used for the Layer2 property.

Example

See also

IPCB_Connection interface

GetState_Mode method

(IPCB_Connection interface)

Syntax

```
Function GetState_Mode : TConnectionMode;
```

Description

This method retrieves the connection mode for the connection object. This method is used for the Mode property.

Example

See also

IPCB_Connection interface

TConnectionMode type

GetState_X1 method

(IPCB_Connection interface)

Syntax

```
Function GetState_X1 : TCoord;
```

Description

This function represents the X1 (initial X) coordinate of the connection object. This method is used by the X1 property.

Example

See also

IPCB_Connection interface

GetState_X2 method

(IPCB_Connection interface)

Syntax

```
Function GetState_X2 : TCoord;
```

Description

This function represents the X2 (final X) coordinate of the connection object. This method is used by the X2 property.

Example

See also

IPCB_Connection interface

GetState_Y1 method

(IPCB_Connection interface)

Syntax

```
Function GetState_Y1 : TCoord;
```

Description

This function represents the Y1 (initial Y) coordinate of the connection object. This method is used by the Y1 property.

Example

See also

IPCB_Connection interface

GetState_Y2 method

(IPCB_Connection interface)

Syntax

```
Function GetState_Y2 : TCoord;
```

Description

This function represents the Y2 (final Y) coordinate of the connection object. This method is used by the Y2 property.

Example

See also

IPCB_Connection interface

SetState_Mode method

(IPCB_Connection interface)

Syntax

```
Procedure SetState_Mode (Value : TConnectionMode);
```

Description

This function represents the Connection Mode for the connection object. This method is used by the Mode property.

Example

See also

IPCB_Connection interface

TConnectionMode type

SetState_X1 method

(IPCB_Connection interface)

Syntax

```
Procedure SetState_X1 (Value : TCoord);
```

Description

This method represents the X1 (initial X) coordinate of the connection object. This method is used by the X1 property.

Example

See also

IPCB_Connection interface

SetState_X2 method

(IPCB_Connection interface)

Syntax

```
Procedure SetState_X2 (Value : TCoord);
```

Description

This method represents the X2 (finall X) coordinate of the connection object. This method is used by the X2 property.

Example

See also

IPCB_Connection interface

SetState_Y1 method

(IPCB_Connection interface)

Syntax

```
Procedure SetState_Y1 (Value : TCoord);
```

Description

This method represents the Y1 (initial Y) coordinate of the connection object. This method is used by the Y1 property.

Example

See also

IPCB_Connection interface

SetState_Y2 method

(IPCB_Connection interface)

Syntax

```
Procedure SetState_Y2 (Value : TCoord);
```

Description

This method represents the Y2 (final Y) coordinate of the connection object. This method is used by the Y2 property.

Example

See also

IPCB_Connection interface

Methods

RotateAroundXY method

(IPCB_Connection interface)

Syntax

```
Procedure RotateAroundXY (AX, AY : TCoord;Angle : TAngle);
```

Description

This method rotates a connection object on the PCB document about the AX, AY coordinates with an angle in degrees. To ensure the connection rotates without moving about, pass in its midpoint (between X1,X2 and Y1, Y2) attributes for the AX,AY parameters.

Example

See also

IPCB_Connection interface

IsRedundant method

(IPCB_Connection interface)

Syntax

```
Function IsRedundant : Boolean;
```

Description

This method determines whether the object is redundant (unused object) on the PCB document or not.

Example

See also

IPCB_Connection interface

Properties

X1 property

(IPCB_Connection interface)

Syntax

```
Property X1 : TCoord Read GetState_X1 Write SetState_X1;
```

Description

This property represents the X1 (initial X) coordinate of the connection object.

Example

See also

IPCB_Connection interface

Y1 property

(IPCB_Connection interface)

Syntax

```
Property Y1 : TCoord Read GetState_Y1 Write SetState_Y1;
```

Description

This property represents the Y1 (initial Y) coordinate of the connection object.

Example

See also

IPCB_Connection interface

X2 property

(IPCB_Connection interface)

Syntax

```
Property X2 : TCoord Read GetState_X2 Write SetState_X2;
```

Description

This property represents the X2 (finall X) coordinate of the connection object.

Example

See also

IPCB_Connection interface

Y2 property

(IPCB_Connection interface)

Syntax

```
Property Y2 : TCoord Read GetState_Y2 Write SetState_Y2;
```

Description

This property represents the Y2 (final Y) coordinate of the connection object.

Example

See also

IPCB_Connection interface

Mode property

(IPCB_Connection interface)

Syntax

```
Property Mode : TConnectionMode Read GetState_Mode Write SetState_Mode;
```

Description

The Mode property represents the connection mode type of the connection; whether it is part of the rats nest, or as a broken net marker.

Example

See also

IPCB_Connection interface

TConnectionMode type

Layer2 property

(IPCB_Connection interface)

Syntax

```
Property Layer2 : TLayer Read GetState_Layer2;
```

Description

This property retrieves the Layer 2 attribute which represents a connection from the first layer to the second layer on a PCB document.

Example

See also

IPCB_Connection interface

Layer1 property

(IPCB_Connection interface)

Syntax

```
Property Layer1 : TLayer Read GetState_Layer1;
```

Description

This property retrieves the Layer 1 attribute which represents a connection from the first layer to the second layer on a PCB document.

Example

See also

IPCB_Connection interface

IPCB_ DifferentialPair Interface

Overview

A differential signaling system is one where a signal is transmitted down a pair of tightly coupled carriers, one of these carrying the signal, the other carrying an equal but opposite image of the signal. Differential signaling was developed to cater for situations where the logic reference ground of the signal source could not be well connected to the logic reference ground of the load. Differential signaling is inherently immune to common mode electrical noise, the most common interference artifact present in an electronic product. Another major advantage of differential signaling is that it minimizes electromagnetic interference (EMI) generated from the signal pair.

Differential pair routing is a design technique employed to create a balanced transmission system able to carry differential (equal and opposite) signals across a printed circuit board. Typically this differential routing will interface to an external differential transmission system, such as a connector and cable.

It is important to note that while the coupling ratio achieved in a twisted pair differential cable may be better than 99%, the coupling achieved in differential pair routing will typically be less than 50%. Current expert opinion is that the PCB routing task is not to try to ensure a specific *differential impedance* is achieved, rather the objective is to maintain the properties required to ensure the differential signal arrives in good condition at the target component as it travels from the external cabling.

Notes

The IPCB_DifferentialPair Interface hierarchy is as follows;

IPCB_Primitive

IPCB_DifferentialPair

IPCB_DifferentialPair methods

```
GetState_Name
GetState_PositiveNet
GetState_NegativeNet
GetState_GatherControl

SetState_Name
SetState_PositiveNet
SetState_NegativeNet
SetState_GatherControl
```

IPCB_DifferentialPair properties

```
Name
PositiveNet
NegativeNet
GatherControl
```

Example

See also

PCB Design Objects

Methods

Properties

IPCB_Embedded Interface

Overview

An IPCB_Embedded interface represents an embedded object in a PCB document. An embedded object is not a visible object and cannot be manipulated by normal means in Altium Designer. An embedded object can be used to store information which gets saved in the PCB document file when this file is saved. Each embedded object is identified by its Name property and the **Description** property can be used to store information.

The IPCB_Embedded hierarchy;

IPCB_Primitive

IPCB_Embedded

IPCB_Embedded methods

GetState_Name

GetState_Description

SetState_Name

SetState_Description

IPCB_Embedded properties

Name

Description

Example

```
Var
    Board      : IPCB_Board;
    EmbdObject : IPCB_Embedded;
Begin
    // Check if PCB board exists
    Board := PCBServer.GetCurrentPCBBoard;
```

```
If Board = Nil Then
Begin
    ShowWarning('This document is not a PCB document!');
    Exit;
End;

// Embedded object created.
EmbdObject := PCBServer.PCBObjectFactory(eEmbeddedObject, eNoDimension, eCreate_Default);
EmbdObject.Name        := 'Embedded Object Name';
EmbdObject.Description := 'Embedded object  can store many chars.';
Board.AddPCBObject(EmbdObject);
```

See also

IPCB_Primitive interface

PCB Design Objects

The EmbeddedObjects script in the `Examples\Scripts\Delphiscript Scripts\Pcb\` folder

Methods

SetState_Name method

(IPCB_Embedded interface)

Syntax

```
Procedure SetState_Name (Value : TPCBString);
```

Description

This method sets the name for the embedded object. This method represents the Name property.

Example

See also

IPCB_Embedded interface

SetState_Description method

(IPCB_Embedded interface)

Syntax

```
Procedure SetState_Description (Value : TPCBString);
```

Description

This method sets the description for the embedded object. This method represents the **Description** property. The **Description** field can be used to store data.

Example

See also

IPCB_Embedded interface

GetState_Name method

(IPCB_Embedded interface)

Syntax

```
Function GetState_Name : TPCBString;
```

Description

This method gets the name for the embedded object. This method represents the Name property.

Example

See also

IPCB_Embedded interface

GetState_Description method

(IPCB_Embedded interface)

Syntax

```
Function GetState_Description : TPCBString;
```

Description

This method gets the description for the embedded object. This method represents the **Description** property. The **Description** field can be used to store data.

Example

See also

IPCB_Embedded interface

Properties

Name property

(IPCB_Embedded interface)

Syntax

```
Property Name : TPCBString Read GetState_Name Write SetState_Name;
```

Description

The Name property represents the name identifier of the embedded object. This property is supported by its GetState_Name and SetState_Name methods.

Example

```
Var
    Board    : IPCB_Board;
    Iterator : IPCB_BoardIterator;
    Embd     : IPCB_Embedded;
Begin
    Iterator := PCBServer.GetCurrentPCBBoard.BoardIterator_Create;
    Iterator.AddFilter_ObjectSet(MkSet(eEmbeddedObject));
    Iterator.AddFilter_LayerSet (AllLayers);
    Iterator.AddFilter_Method   (eProcessAll);

    Embd   := Iterator.FirstPCBObject;
    While Embd <> Nil Do
    Begin
        ShowInfo('Name : '        + Embd.Name + #13#10 +
                 'Description : ' + Embd.Description);
        Embd := Iterator.NextPCBObject;
    End;
    PCBServer.GetCurrentPCBBoard.BoardIterator_Destroy(Iterator);
End;
```

See also

IPCB_Embedded interface

TPCBString type

Description property

(IPCB_Embedded interface)

Syntax

```
Property Description : TPCBString Read GetState_Description Write SetState_Description;
```

Description

The **Description** property represents the **Description** field of the embedded object. This property is supported by its GetState_**Description** and SetState_**Description** methods.

The **Description** field can be used to store data that represents this embedded object.

Example

```
Var
    Board    : IPCB_Board;
    Iterator : IPCB_BoardIterator;
    Embd     : IPCB_Embedded;
Begin
    Iterator := PCBServer.GetCurrentPCBBoard.BoardIterator_Create;
    Iterator.AddFilter_ObjectSet(MkSet(eEmbeddedObject));
    Iterator.AddFilter_LayerSet (AllLayers);
    Iterator.AddFilter_Method   (eProcessAll);

    Embd   := Iterator.FirstPCBObject;
    While Embd <> Nil Do
    Begin
        ShowInfo('Name : '        + Embd.Name + #13#10 +
                 'Description : ' + Embd.Description);
        Embd := Iterator.NextPCBObject;
    End;
    PCBServer.GetCurrentPCBBoard.BoardIterator_Destroy(Iterator);
End;
```

See also

IPCB_Embedded interface

TPCBString type

IPCB_EmbeddedBoard Interface

Overview

The IPCB_EmbeddedBoard interface represents an embedded board object consisting of multiple child PCBs in a matrix of rows and columns which is an embedded board array feature. Each board array can reference a different pcb file.

Notes

• The IPCB_EmbeddedBoard interface is inherited from the IPCB_RectangularPrimitive interface.

• The RowSpacing and ColSpacing values determine the gap between items in the matrix of rows and columns.

• The DocumentPath string refers to the referenced PCB file. The corresponding ChildBoard interface represents the child referenced PCB.

• The OriginMode property denotes how the array is referenced from the origin of the embedded board or let the PCB editor build the array based on the bottom left of the objects in the referenced board's workspace.

• The MirrorFlag denotes whether the embedded board is to be flipped over or not.

The **IPCB_EmbeddedBoard** interface hierarchy is as follows;

The IPCB_EmbeddedBoard hierarchy;

IPCB_RectangularPrimitive

IPCB_EmbeddedBoard

IPCB_RectangularPrimitive methods
RotateAroundXY
IsRedundant
SetState_XSizeYSize

IPCB_RectangularPrimitive properties
XLocation
YLocation
X1Location
Y1Location
X2Location
Y2Location
Rotation

IPCB_EmbeddedBoard methods
GetState_RowCount
GetState_ColCount
GetState_RowSpacing
GetState_ColSpacing
GetState_DocumentPath
GetState_ChildBoard
GetState_Mirror
GetState_OriginMode
SetState_RowCount
SetState_ColCount
SetState_RowSpacing
SetState_ColSpacing
SetState_DocumentPath
SetState_Mirror
SetState_OriginMode

IPCB_EmbeddedBoard properties
RowCount
ColCount
RowSpacing
ColSpacing
DocumentPath
ChildBoard
MirrorFlag
OriginMode

See also

IPCB_RectangularPrimitive interface
PCB Design Objects

Methods

GetState_ChildBoard method

(IPCB_EmbeddedBoard interface)

Syntax

```
Function GetState_ChildBoard : IPCB_Board;
```

Description

This method retrieves the reference PCB document to be used for the embedded board panellization. This method is used for the ChildBoard property.

Example

See also

IPCB_EmbeddedBoard interface

GetState_ColCount method

(IPCB_EmbeddedBoard interface)

Syntax

```
Function GetState_ColCount : Integer;
```

Description

This method retrieves the number of columns that the board array will have. You can also obtain the RowCount as well to determine the size of the matrix for the board array.

This method is used for the ColCount property.

Example

See also

IPCB_EmbeddedBoard interface

GetState_ColSpacing method

(IPCB_EmbeddedBoard interface)

Syntax

```
Function GetState_ColSpacing : TCoord;
```

Description

This method sets the height of the first board and the gap between two boards. This row spacing and the column spacing values are used to generate an embedded board array.

This method is used by the ColSpacing property.

Example

See also

IPCB_EmbeddedBoard interface

GetState_DocumentPath method

(IPCB_EmbeddedBoard interface)

Syntax

```
Function GetState_DocumentPath : TPCBString;
```

Description

This method obtains the path to the referenced PCB for the board panellization. This method is used by the **DocumentPath** property.

Example

See also

IPCB_EmbeddedBoard interface

GetState_Mirror method

(IPCB_EmbeddedBoard interface)

Syntax

```
Function GetState_Mirror : Boolean;
```

Description

The MirrorFlag property obtains the mirrored state of the embedded board panel of PCBs. Set true to mirror it, or False to leave the embedded board panel as is.

This method is used by the MirrorFlag property.

Example

See also

IPCB_EmbeddedBoard interface

GetState_OriginMode method

(IPCB_EmbeddedBoard interface)

Syntax

```
Function GetState_OriginMode : TEmbeddedBoardOriginMode;
```

Description

This method obtains the board array from the origin of the embedded board or from the bottom left of the referenced board's workspace.

From the bottom left is the default value which has the software build the array based on the bottom left of the objects in the referenced board's workspace (which is the child PCB document).

This method is used by the **OriginMode** property.

Note that the reference point (as a red cross) of the board array is defined by the child PCB document that is used as the base for the board array to place on a PCB document. To change the reference point (origin) of the child board object, click Edit » Origin » Reset / Set menu items to set the origin marker from the PCB menu.

Example

See also

IPCB_EmbeddedBoard interface

GetState_RowCount method

(IPCB_EmbeddedBoard interface)

Syntax

```
Function GetState_RowCount : Integer;
```

Description

This method retrieves the number of rows that the board array will have. You can also obtain the RowCount as well to determine the size of the matrix for the board array.

This method is used for the RowCount property.

Example

See also

IPCB_EmbeddedBoard interface

GetState_RowSpacing method

(IPCB_EmbeddedBoard interface)

Syntax

```
Function GetState_RowSpacing : TCoord;
```

Description

This method obtains the width of the first board and the gap between two boards. This row spacing and the column spacing values are used to generate an embedded board array.

This method is used by the RowSpacing property.

Example

See also

IPCB_EmbeddedBoard interface

SetState_ColCount method

(IPCB_EmbeddedBoard interface)

Syntax

```
Procedure SetState_ColCount (Value : Integer);
```

Description

This method sets the number of columns that the board array will have. You can also set the RowCount as well to determine the size of the matrix for the board array.

This method is used for the ColCount property.

Example

See also

IPCB_EmbeddedBoard interface

SetState_ColSpacing method

(IPCB_EmbeddedBoard interface)

Syntax

```
Procedure SetState_ColSpacing (Value : TCoord );
```

Description

This method sets the width of the first board and the gap between two boards. This row spacing and the column spacing values are used to generate an embedded board array.

This method is used by the ColSpacing property.

Example

See also

IPCB_EmbeddedBoard interface

SetState_DocumentPath method

(IPCB_EmbeddedBoard interface)

Syntax

```
Procedure SetState_DocumentPath (Value : TPCBString);
```

Description

This method sets the path to the referenced PCB for the board panellization. This method is used by the DocumentPath property.

Example

See also

IPCB_EmbeddedBoard interface

SetState_Mirror method

(IPCB_EmbeddedBoard interface)

Syntax

```
Procedure SetState_Mirror (Value : Boolean);
```

Description

The MirrorFlag property sets the mirrored state of the embedded board panel of PCBs. Set true to mirror it, or False to leave the embedded board panel as is.

This method is used by the MirrorFlag property.

Example

See also

IPCB_EmbeddedBoard interface

SetState_OriginMode method

(IPCB_EmbeddedBoard interface)

Syntax

```
Procedure SetState_OriginMode (Value : TEmbeddedBoardOriginMode);
```

Description

This method sets the board array from the origin of the embedded board or from the bottom left of the referenced board's workspace.

From the bottom left is the default value which has the software build the array based on the bottom left of the objects in the referenced board's workspace (which is the child PCB document).

This method is used by the **OriginMode** property..

Note that the reference point (as a red cross) of the board array is defined by the child PCB document that is used as the base for the board array to place on a PCB document. To change the reference point (origin) of the child board object, click Edit » Origin » Reset / Set menu items to set the origin marker from the PCB menu.

Example

See also

IPCB_EmbeddedBoard interface

SetState_RowCount method

(IPCB_EmbeddedBoard interface)

Syntax

```
Procedure SetState_RowCount (Value : Integer);
```

Description

This method sets the number of rows that the board array will have. You can also set the ColCount as well to determine the size of the matrix for the board array.

This method is used for the RowCount property.

Example

See also

IPCB_EmbeddedBoard interface

SetState_RowSpacing method

(IPCB_EmbeddedBoard interface)

Syntax

```
Procedure SetState_RowSpacing (Value : TCoord );
```

Description

This method sets the width of the first board and the gap between two boards. This row spacing and the column spacing values are used to generate an embedded board array.

This method is used by the RowSpacing property.

Example

See also

IPCB_EmbeddedBoard interface

Properties

ChildBoard property

(IPCB_EmbeddedBoard interface)

Syntax

```
Property ChildBoard : IPCB_Board Read GetState_ChildBoard;
```

Description

This **ChildBoard** property represents the reference PCB document to be used for the embedded board panellization.

This read only property is supported by the GetState_ChildBoard method.

Example

See also

IPCB_EmbeddedBoard interface

ColCount property

(IPCB_EmbeddedBoard interface)

Syntax

```
Property ColCount : Integer Read GetState_ColCount Write SetState_ColCount;
```

Description

This **ColCount** property represents the number of columns that the board array will have. You can also define the RowCount property as well to define the size of the matrix for the board array.

This property is represented by the GetState_ColCount and SetState_ColCount methods.

Example

See also

IPCB_EmbeddedBoard interface

ColSpacing property

(IPCB_EmbeddedBoard interface)

Syntax

```
Property ColSpacing : TCoord Read GetState_ColSpacing Write SetState_ColSpacing;
```

Description

The **ColSpacing** property determines the height of the first board and the gap between two boards. This column spacing and the row spacing values are used to generate an embedded board array.

This property is supported by the GetState_ColSpacing and SetState_ColSpacing methods.

Example

See also

IPCB_EmbeddedBoard interface

DocumentPath property

(IPCB_EmbeddedBoard interface)

Syntax

```
Property DocumentPath : TPCBString Read GetState_DocumentPath Write SetState_DocumentPath;
```

Description

This **DocumentPath** property represents the path to the referenced PCB for the board panellization. This property is supported by the **GetState_DocumentPath** and **SetState_DocumentPath** methods.

Example

See also

IPCB_EmbeddedBoard interface

MirrorFlag property

(IPCB_EmbeddedBoard interface)

Syntax

```
Property MirrorFlag : Boolean Read GetState_Mirror Write SetState_Mirror;
```

Description

The **MirrorFlag** property represents the mirrored state of the embedded board panel of PCBs. Set true to mirror it, or False to leave the embedded board panel as is.

This property is supported by the GetState_MirrorFlag and SetState_MirrorFlag methods.

Example

See also

IPCB_EmbeddedBoard interface

OriginMode property

(IPCB_EmbeddedBoard interface)

Syntax

```
Property OriginMode : TEmbeddedBoardOriginMode Read GetState_OriginMode Write
SetState_OriginMode;
```

Description

This **OriginMode** property references the board array from the origin of the embedded board or from the bottom left of the referenced board's workspace.

From the bottom left is the default value which has the software build the array based on the bottom left of the objects in the referenced board's workspace (which is the child PCB document).

This **OriginMode** property is supported by the **GetState_OriginMode** and **SetState_OriginMode** methods.

Note that the reference point (as a red cross) of the board array is defined by the child PCB document that is used as the base for the board array to place on a PCB document. To change the reference point (origin) of the child board object, click Edit » Origin » Reset / Set menu items to set the origin marker from the PCB menu.

Example

See also

IPCB_EmbeddedBoard interface

TEmbeddedBoardOriginMode type

RowCount property

(IPCB_EmbeddedBoard interface)

Syntax

```
Property RowCount : Integer Read GetState_RowCount Write SetState_RowCount;
```

Description

This **RowCount** property represents the number of rows that the board array will have. You can also define the ColCount property as well to define the size of the matrix for the board array.

This property is represented by the GetState_RowCount and SetState_RowCount methods.

Example

See also

IPCB_EmbeddedBoard interface

RowSpacing property

(IPCB_EmbeddedBoard interface)

Syntax

```
Property RowSpacing : TCoord Read GetState_RowSpacing Write SetState_RowSpacing;
```

Description

The **RowSpacing** property determines the width of the first board and the gap between two boards. This row spacing and the column spacing values are used to generate an embedded board array.

This property is supported by the GetState_RowSpacing and SetState_RowSpacing methods.

Example

See also

IPCB_EmbeddedBoard interface

IPCB_Fill

Overview

The **IPCB_Fill** interface represents a PCB fill object on a PCB document. A fill object is a rectangular object and thus is inherited from the IPCB_RectangularPrimitive interface.

Notes

The IPCB_Fill interface hierarchy is as follows;

IPCB_Primitive

IPCB_RectangularPrimitive

IPCB_Fill

IPCB_RectangularPrimitive methods	IPCB_RectangularPrimitive properties
	XLocation
RotateAroundXY	YLocation
IsRedundant	X1Location
SetState_XSizeYSize	Y1Location
	X2Location
	Y2Location
	Rotation

IPCB_Fill methods	IPCB_Fill properties

Example

```
Var
    WorkSpace : IWorkSpace;
    Board     : IPCB_Board;
    Fill      : IPCB_Fill;
Begin
    //Create a new PCB document
    WorkSpace := GetWorkSpace;
    If WorkSpace = Nil Then Exit;
    Workspace.DM_CreateNewDocument('PCB');

    Board := PCBServer.GetCurrentPCBBoard;
    If Board = Nil then exit;

    // Create a Fill object
    Fill             := PCBServer.PCBObjectFactory(eFillObject, eNoDimension,eCreate_Default);

    Fill.X1Location  := MilsToCoord(2000);
    Fill.Y1Location  := MilsToCoord(2000);
    Fill.X2Location  := MilsToCoord(2500);
```

```
    Fill.Y2Location  := MilsToCoord(2500);
    Fill.Layer       := eBottomLayer;
    Fill.Rotation    := 45;

    // Add a new Fill into the PCB design database.
    Board.AddPCBObject(Fill);

    // Refresh the PCB document
    ResetParameters;
    AddStringParameter('Action', 'All');
    RunProcess('PCB:Zoom');
End;
```

See also

PCB Design Objects

IPCB_Primitive interface

IPCB_RectangularPrimitive interface

Undo script in **Example**s\Scripts\PCB folder.

IPCB_FromTo Interface

Overview

The **IPCB_FromTo** interface represents a FromTo object on a PCB document, as a node to a node (a pad of a component to a pad of another component for example) and has a NetName property.

The IPCB_FromTo hierarchy;

IPCB_Primitive

IPCB_FromTo

IPCB_FromTo methods

GetState_FromPad

GetState_ToPad

GetState_NetName

SetState_FromPad

SetState_ToPad

SetState_NetName

GetNet

GetFromPad

GetToPad

GetState_RoutedLength

IPCB_FromTo properties

FromPad

ToPad

NetName

See also

IPCB_Primitive interface

IPCB_Pad interface

IPCB_Net interface

PCB Design Objects

GetState and SetState Methods

GetState_FromPad method

(IPCB_FromTo interface)

Syntax

```
Function GetState_FromPad : TPCBString;
```

Description

A FromTo object has a node to a node (a pin to a pin for example) represented FromPad and ToPad properties.

This method is used for the FromPad property.

Example

See also

IPCB_FromTo interface

TPCBString

GetState_NetName method

(IPCB_FromTo interface)

Syntax

```
Function GetState_NetName : TPCBString;
```

Description

The FromTo object has two nodes, FromPad and ToPad. These **Notes** have their Net Name properties.

This method gets the net name for the FromTo object and is for the NetName property.

Example

See also

IPCB_FromTo interface

GetState_ToPad method

(IPCB_FromTo interface)

Syntax

```
Function GetState_ToPad : TPCBString;
```

Description

A FromTo object has a node to a node (a pin to a pin for example) represented FromPad and ToPad properties.

This method is used for the ToPad property.

Example

See also

IPCB_FromTo interface

SetState_FromPad method

(IPCB_FromTo interface)

Syntax

```
Procedure SetState_FromPad (Value : TPCBString);
```

Description

A FromTo object has a node to a node (a pin to a pin for example) represented FromPad and ToPad properties.

This method sets the FromPad and is for the FromPad property.

Example

See also

IPCB_FromTo interface

SetState_NetName method

(IPCB_FromTo interface)

Syntax

```
Procedure SetState_NetName (Value : TPCBString);
```

Description

The FromTo object has two nodes, FromPad and ToPad. These **Notes** have their Net Name properties.

This method sets the net name for the FromTo object and is for the NetName property.

Example

See also

IPCB_FromTo interface

SetState_ToPad method

(IPCB_FromTo interface)

Syntax

```
Procedure SetState_ToPad (Value : TPCBString);
```

Description

A FromTo object has a node to a node (a pin to a pin for example) represented FromPad and ToPad properties.

This method sets the ToPad and is for the ToPad property.

Example

See also

IPCB_FromTo interface

Methods

GetFromPad method

(IPCB_FromTo interface)

Syntax

```
Function GetFromPad : IPCB_Pad;
```

Description

This function returns the pad interface associated with the FromPad of the FromTo object.

Example

See also

IPCB_FromTo interface

GetNet method

(IPCB_FromTo interface)

Syntax

```
Function GetNet : IPCB_Net;
```

Description

This function returns the net interface associated with the net of the FromTo object.

Example

See also

IPCB_FromTo interface

GetToPad method

(IPCB_FromTo interface)

Syntax

```
Function GetToPad : IPCB_Pad;
```

Description

This function returns the pad interface associated with the ToPad of the FromTo object.

Example

See also

IPCB_FromTo interface

GetState_RoutedLength method

(IPCB_FromTo interface)

Syntax

```
Function GetState_RoutedLength : TCoord;
```

Description

This function returns the routed length of the FromTo object in TCoord units.

Example

See also

IPCB_FromTo interface

Properties

FromPad property

(IPCB_FromTo interface)

Syntax

```
Property FromPad : TPCBString Read GetState_FromPad Write SetState_FromPad;
```

Description

The FromTo object has two nodes, FromPad and ToPad. These **Notes** have their Net Name properties.

This property represents the FromPad node and returns the name of the FromPad property.

Example

See also

IPCB_FromTo interface

NetName property

(IPCB_FromTo interface)

Syntax

```
Property NetName : TPCBString Read GetState_NetName Write SetState_NetName;
```

Description

The FromTo object has two nodes, FromPad and ToPad. These **Notes** have their Net Name properties.

This property represents the net name of the FromTo object.

Example

See also

IPCB_FromTo interface

ToPad property

(IPCB_FromTo interface)

Syntax

```
Property ToPad : TPCBString Read GetState_ToPad Write SetState_ToPad;
```

Description

The FromTo object has two nodes, FromPad and ToPad. These **Notes** have their Net Name properties.

This property represents the ToPad node and returns the name of the ToPad property..

Example

See also

IPCB_FromTo interface

IPCB_Group

Overview

The **IPCB_Group** interface is an immediate ancestor for **IPCB_Net**, **IPCB_LibComponent**, **IPCB_Polygon**, **IPCB_Coordinate**, **IPCB_Dimension** and its descendant interfaces.

The **IPCB_Group** interface is a composite object interface which means it can store objects. Thus a group object is an object composed of primitives such as arcs, tracks and fills. For example a polygon consists of child tracks and arcs. A footprint in a PCB library consists of child objects such as arcs, pads and tracks.

The **IPCB_Group** interface hierarchy is as follows;

IPCB_Primitive

IPCB_Group

Notes

To fetch objects of a group object, you employ the Group Iterator with the **GroupIterator_Create** and **GroupIterator_Destroy** methods.

To add or remove child objects from a group object, you employ the **AddPCBObject** or the **RemovePCBObject** methods.

To fetch the reference coordinates of a group object, the X,Y properties define the reference point.

IPCB_Group methods

```
FreePrimitives
GetPrimitiveAt
GetPrimitiveCount
SetState_XSizeYSize
FastSetState_XSizeYSize
SetState_LayersUsedArray
GroupIterator_Create
GroupIterator_Destroy
AddPCBObject
RemovePCBObject
```

IPCB_Group properties

```
X
Y
PrimitiveLock
LayerUsed
```

See also

IPCB_Primitive interface

IPCB_Net interface

IPCB_LibComponent interface

IPCB_Polygon interface

IPCB_Coordinate interface

IPCB_Dimension interface

IPCB_GroupIterator interface

PCB Design Objects

Methods

AddPCBObject method

(IPCB_Group interface)

Syntax

```
Procedure AddPCBObject(PCBObject : IPCB_Primitive);
```

Description

Example

See also

IPCB_Group interface

FastSetState_XSizeYSize method

(IPCB_Group interface)

Syntax

```
Function FastSetState_XSizeYSize : Boolean;
```

Description

Example

See also

IPCB_Group interface

FreePrimitives method

(IPCB_Group interface)

Syntax

```
Procedure FreePrimitives;
```

Description

Example

See also

IPCB_Group

GetPrimitiveAt method

(IPCB_Group interface)

Syntax

```
Function  GetPrimitiveAt(I        : Integer;
                         ObjectId : TObjectId) : IPCB_Primitive;
```

Description

Example

See also

IPCB_Group interface

GetPrimitiveCount method

(IPCB_Group interface)

Syntax

```
Function GetPrimitiveAt(I        : Integer;
                        ObjectId : TObjectId): IPCB_Primitive;
```

Description

Example

See also

IPCB_Group

GroupIterator_Create

(IPCB_Group interface)

Syntax

```
Function GroupIterator_Create : IPCB_GroupIterator;
```

Description

The GroupIterator_Create method creates a group iterator for the group object, so that the child objects can be searched from within the group object. This group iterator searches for child objects of a group object, such as a component, footprint, polygon, dimension, board layout and so on.

Example

```
Var
    Track                   : IPCB_Primitive;
    TrackIteratorHandle     : IPCB_GroupIterator;
    Component               : IPCB_Component;
    ComponentIteratorHandle : IPCB_BoardIterator;
    TrackCount              : Integer;
    ComponentCount          : Integer;
Begin
    TrackCount     := 0;
    ComponentCount := 0;
    If PCBServer.GetCurrentPCBBoard = Nil Then Exit;

    ComponentIteratorHandle := PCBServer.GetCurrentPCBBoard.BoardIterator_Create;
    ComponentIteratorHandle.AddFilter_ObjectSet(MkSet(eComponentObject));
    ComponentIteratorHandle.AddFilter_LayerSet(AllLayers);
    ComponentIteratorHandle.AddFilter_Method(eProcessAll);
    Component := ComponentIteratorHandle.FirstPCBObject;

    While (Component <> Nil) Do
    Begin
        TrackIteratorHandle := Component.GroupIterator_Create;
        TrackIteratorHandle.AddFilter_ObjectSet(MkSet(eTrackObject));
        TrackIteratorHandle.AddFilter_LayerSet(MkSet(eTopOverlay));
        Track := TrackIteratorHandle.FirstPCBObject;
        While (Track <> Nil) Do
        Begin
            Inc(TrackCount);
            Track := TrackIteratorHandle.NextPCBObject;
```

```
        End;
        ShowInfo('This component ' + Component.SourceDesignator  + ' has ' +
IntToStr(TrackCount)  + ' tracks.');
        TrackCount := 0;
        Component.GroupIterator_Destroy(TrackIteratorHandle);
        Component := ComponentIteratorHandle.NextPCBObject;
        Inc(ComponentCount);
        If (ComponentCount > 5) Then Break;
    End;
    PCBServer.GetCurrentPCBBoard.BoardIterator_Destroy(ComponentIteratorHandle);
End;
```

See also

IPCB_Group interface

IPCB_GroupIterator interface

GroupIterator_Destroy

(IPCB_Group interface)

Syntax

```
Procedure GroupIterator_Destroy(Var AIterator : IPCB_GroupIterator);
```

Description

Example

See also

IPCB_Group interface

IPCB_GroupIterator interface

RemovePCBObject method

(IPCB_Group interface)

Syntax

```
Procedure RemovePCBObject(PCBObject : IPCB_Primitive);
```

Description

Example

See also

IPCB_Group interface

SetState_LayersUsedArray method

(IPCB_Group interface)

Syntax

```
Procedure SetState_LayersUsedArray;
```

Description

Example

See also

IPCB_Group interface

SetState_XSizeYSize method

(IPCB_Group interface)

Syntax

```
Function  SetState_XSizeYSize : Boolean;
```

Description

Example

See also

IPCB_Group interface

Properties

LayerIsUsed property

(IPCB_Group interface)

Syntax

```
Property  LayerUsed [L : TLayer] : Boolean Read GetState_LayerUsed Write SetState_LayerUsed;
```

Description

Example

See also

IPCB_Group

PrimitiveLock property

(IPCB_Group interface)

Syntax

```
Property PrimitiveLock : Boolean Read GetState_PrimitiveLock Write SetState_PrimitiveLock;
```

Description

The PrimitiveLock property denotes whether the primitives of the group object can be edited individually or not. Normally all the child objects or primitives of a group can only be accessed as a group object.

Example

See also

IPCB_Group

X property

(IPCB_Group interface)

Syntax

```
Property X : TCoord Read GetState_XLocation Write SetState_XLocation;
```

Description

The X property defines the reference point of the group object.

Example

See also

IPCB_Group interface

Y property

(IPCB_Group interface)

Syntax

```
Property Y : TCoord Read GetState_YLocation Write SetState_YLocation;
```

Description

The Y property defines the reference point of the group object.

Example

See also

IPCB_Group interface

IPCB_LettersCache Interface

Overview

IPCB_LettersCache methods

I_ObjectAddress

PlotText

IPCB_LettersCache properties

Example

```
Var
```

See also

PCB Design Objects

Methods

I_ObjectAddress method

(IPCB_LettersCache interface)

Syntax

```
Function  I_ObjectAddress : TPCBObjectHandle;
```

Description

Example

See also

IPCB_LettersCache interface

PlotText method

(IPCB_LettersCache interface)

Syntax

```
Procedure PlotText(ATextHandle  : TPCBObjectHandle;
                   PlotProc     : TPlotPolygonProc;
             Const ADisplayText : TPCBString);
```

Description

Example

See also

IPCB_LettersCache interface

IPCB_LibComponent Interface

Overview

The **IPCB_LibComponent** object represents the current footprint in a PCB library document. The footprints of a PCB library is equivalent to "pages" of a library.

The library document is represented by two interfaces - the current footprint and the IPCB_Library document.

The **IPCB_LibraryIterator** object interface iterates through a loaded PCB library in Altium Designer to fetch PCB footprints which are represented by the **IPCB_LibComponent** interfaces. The IPCB_LibraryIterator interface is used in the IPCB_Library interface - LibraryIterator_Create and LibraryIterator_Destory methods.

Notes

A library is represented by the IPCB_Library interface.

A PCB footprint (as a page of the library) is represented by its IPCB_LibComponent interface which is inherited from the IPCB_Group object interface.

A PCB footprint is composed of child objects such as pads and tracks. Therefore the footprint has its own IPCB_GroupIterator to fetch its own child objects.

DelphiScript doesnt support sets, therefore to pass in a set of layers or a set of objects, you need to use the **MkSet** function to create a pseudo set of objects or layers for the **AddFilter_ObjectSet** or **AddFilterLayerSet** methods. For example LibraryIterator.AddFilter_ObjectSet(**MkSet**(eTrackObject,eFillObject));

The **IPCB_LibComponent** interface hierarchy is as follows;

IPCB_Primitive

IPCB_Group

IPCB_LibComponent

IPCB_Group methods

```
FreePrimitives
GetPrimitiveAt
GetPrimitiveCount
SetState_XSizeYSize
FastSetState_XSizeYSize
SetState_LayersUsedArray

GroupIterator_Create
GroupIterator_Destroy

AddPCBObject
RemovePCBObject
```

IPCB_Group properties

```
X
Y
PrimitiveLock
LayerUsed
```

IPCB_LibComponent methods

GetState_Pattern

GetState_Height

GetState_Description

SetState_Pattern

SetState_Height

SetState_Description

IPCB_LibComponent properties

Name

Height

Description

DelphiScript Example

```
Procedure ReportFootprintInfo;
Var
    CurrentLib        : IPCB_Library;
    FootprintIterator : IPCB_LibraryIterator;
    Footprint         : IPCB_LibComponent;
    FootprintList     : TStringList;
    ReportDocument    : IServerDocument;
    Filename          : TString;
    S                 : TString;
    I                 : Integer;
Begin
    CurrentLib := PCBServer.GetCurrentPCBLibrary;
    If CurrentLib = Nil Then Exit;

    Filename := ExtractFilePath(CurrentLib.Board.FileName) + 'PCBLib_Report.csv';
    S := '';
    FootprintList := TStringList.Create;

    FootprintIterator := CurrentLib.LibraryIterator_Create;
    FootprintIterator.SetState_FilterAll;
    Try
        Footprint := FootprintIterator.FirstPCBObject;
        While Footprint <> Nil Do
        Begin
            // Determine which units are in use. at the mo it is the other way around!!!
            If CurrentLib.Board.DisplayUnit = eMetric Then
                S := footprint.name + ',' + FloatToStr(CoordToMils(Footprint.Height)) + ',' +
Footprint.Description
            Else
                S := footprint.name + ',' + FloatToStr(CoordToMMs(Footprint.Height)) + ',' +
Footprint.Description;

            FootprintList.Add(S);
            Footprint := FootprintIterator.NextPCBObject;
        End;
     Finally
        CurrentLib.LibraryIterator_Destroy(FootprintIterator);
        FootprintList.SaveToFile(FileName);
        FootprintList.Free;
    End;

    //Display and save report.
    ReportDocument := Client.OpenDocument('Text', FileName);
    If ReportDocument <> Nil Then
        Client.ShowDocument(ReportDocument);
End;
```

See also

PCB Design Objects

IPCB_Primitive interface

IPCB_Group interface

IPCB_GroupIterator interface

LibraryIterator example from **\Examples\Scripts\DelphiScript\PCB** folder.

GetState and SetState Methods

GetState_Description method

(IPCB_LibComponent interface)

Syntax

```
Function GetState_Description : TPCBString;
```

Description

The **Description** property denotes the footprint's description. This method is used for the **Description** property.

Example

See also

IPCB_LibComponent interface

GetState_Height method

(IPCB_LibComponent interface)

Syntax

```
Function GetState_Height : TCoord;
```

Description

The Height property denotes the footprint's height. This method is used by the Height property.

Example

See also

IPCB_LibComponent interface

GetState_Pattern method

(IPCB_LibComponent interface)

Syntax

```
Function GetState_Pattern : TPCBString;
```

Description

The Name property denotes the pattern name of the footprint. This pattern method is used by the Name property.

Example

See also

IPCB_LibComponent interface

SetState_Description method

(IPCB_LibComponent interface)

Syntax

```
Procedure SetState_Description (Value : TPCBString);
```

Description

The **Description** property denotes the footprint's description. This method is used for the **Description** property.

Example

See also

IPCB_LibComponent interface

SetState_Height method

(IPCB_LibComponent interface)

Syntax

```
Procedure SetState_Height (Value : TCoord);
```

Description

The Height property denotes the footprint's height. This method is used by the Height property.

Example

See also

IPCB_LibComponent interface

SetState_Pattern method

(IPCB_LibComponent interface)

Syntax

```
Procedure SetState_Pattern (Value : TPCBString);
```

Description

The Name property denotes the pattern name of the footprint. This pattern method is used by the Name property.

Example

See also

IPCB_LibComponent interface

Properties

Description **property**

(IPCB_LibComponent interface)

Syntax

```
Property Description : TPCBString Read getState_Description Write SetState_Description;
```

Description

The **Description** property denotes the footprint's description. This **Description** property is supported by the GetState_**Description** and SetState_**Description** methods.

Note, the IPCB_LibComponent interface represents the current footprint in the PCB Library editor workspace.

Example

See also

IPCB_LibComponent interface

Height property

(IPCB_LibComponent interface)

Syntax

```
Property Height : TCoord Read GetState_Height Write SetState_Height;
```

Description

The Height property denotes the footprint's height. This Height property is supported by the GetState_Height and SetState_Height methods.

Note, the IPCB_LibComponent interface represents the current footprint in the PCB Library editor workspace.

Example

See also

IPCB_LibComponent interface

Name property

(IPCB_LibComponent interface)

Syntax

```
Property Name : TPCBString Read GetState_Pattern Write SetState_Pattern;
```

Description

The Name property denotes the pattern name of the footprint. This Name property is supported by the GetState_Pattern and SetState_Pattern methods.

Note, the **IPCB_LibComponent** interface represents the current footprint in the PCB Library editor workspace.

Example

See also

IPCB_LibComponent interface

IPCB_Net Interface

Overview

A net object can store net information from a PCB document. The net object contains information about the components used in the design, and the connectivity created in the design, stored in the form of nets. A net object is a list of pin to pin connections that are electrically connected in the design. The arrangement of the pin to pin connections is called the net topology.

The net objects are system generated objects, which means, you can retrieve the net names of PCB objects that have a net property on a PCB document.

By default the PCB editor arranges the pin to pin connections of each net to give the shortest overall connection length. To have control of the arrangement of the pin to pin connections in a net, the PCB editor allows the user to define a set of From-Tos.

The IPCB_Net interface hierarchy is as follows;

IPCB_Primitive

IPCB_Group

IPCB_Net

Notes

The ConnectsVisible property denotes the visibility of a net. If True, connections are visible.

IPCB_Group table

IPCB_Group methods	IPCB_Group properties
FreePrimitives	X
GetPrimitiveAt	Y
GetPrimitiveCount	PrimitiveLock
SetState_XSizeYSize	LayerUsed
FastSetState_XSizeYSize	
SetState_LayersUsedArray	
GroupIterator_Create	
GroupIterator_Destroy	
AddPCBObject	
RemovePCBObject	

IPCB_Net table

IPCB_Net methods	IPCB_Net properties
GetState_Color	Color

GetState_Name	Name
GetState_ConnectsVisible	ConnectsVisible
GetState_ConnectivelyInvalid	ConnectivelyInvalid
GetState_RoutedLength	RoutedLength
GetState_ViaCount	ViaCount
GetState_PinCount	PinCount
Getstate_PadByName	PadByName
Getstate_PadByPinDescription	PadByPinDescription
GetState_IsHighlighted	IsHighlighted
GetState_LoopRemoval	LoopRemoval
GetState_DifferentialPair	DifferentialPair
GetState_InDifferentialPair	InDifferentialPair
GetState_LiveHighlightMode	LiveHighlightMode

SetState_Color
SetState_Name
SetState_ConnectsVisible
SetState_IsHighlighted
SetState LoopRemoval
SetState_DifferentialPair
SetState_LiveHighlightMode

Rebuild
HideNetConnects
ShowNetConnects
ConnectivelyInValidate;Procedure
CancelGroupWarehouseRegistration
CancelGroupWarehouseRegistration
RegisterWithGroupWarehouse

GetLogicalNet

SubnetIndices_Set
SubnetIndices_Reset

Example

```
Procedure IterateNetObjects;
Var
    Board       : IPCB_Board;
    Net         : IPCB_Net;
    Iterator    : IPCB_BoardIterator;
    LS          : TPCBString;
Begin
```

```
    // Retrieve the current board
    Board := PCBServer.GetCurrentPCBBoard;
    If Board = Nil Then Exit;
    // Create the iterator that will look for Net objects only
    Iterator        := Board.BoardIterator_Create;
    Iterator.AddFilter_ObjectSet(MkSet(eNetObject));
    Iterator.AddFilter_LayerSet(AllLayers);
    Iterator.AddFilter_Method(eProcessAll);
    // Search for Net objects and get their Net Name values
    LS := '';
    Net := Iterator.FirstPCBObject;
    While (Net <> Nil) Do
    Begin
        LS := LS + Net.Name + ', ';
        Net := Iterator.NextPCBObject;
    End;
    Board.BoardIterator_Destroy(Iterator);
    // Display the Net Names on a dialog.
    ShowInfo('Nets = ' + LS);
End;
```

See also

PCB Design Objects

IPCB_Primitive interface

IPCB_Group interface

IPCB_GroupIterator interface

IterateNets example from the **\Examples\Scripts\DelphiScript\PCB** folder.

NetObjectAssign example from the **\Examples\Scripts\DelphiScript\PCB** folder.

GetState and SetState methods

GetState_Color method

(IPCB_Net interface)

Syntax

```
Function GetState_Color : TColor;
```

Description

Example

See also

IPCB_Net interface

GetState_ConnectivelyInvalid method

(IPCB_Net interface)

Syntax

```
Function GetState_ConnectivelyInvalid : Boolean;
```

Description

Example

See also

IPCB_Net interface

GetState_ConnectsVisible method

(IPCB_Net interface)

Syntax

```
Function GetState_ConnectsVisible : Boolean;
```

Description

Example

See also

IPCB_Net interface

GetState_IsHighlighted method

(IPCB_Net interface)

Syntax

```
Function GetState_IsHighlighted : Boolean;
```

Description

Example

See also

IPCB_Net interface

GetState_Name method

(IPCB_Net interface)

Syntax

```
Function GetState_Name : TPCBString;
```

Description

Example

See also

IPCB_Net interface

Getstate_PadByName method

(IPCB_Net interface)

Syntax

```
Function Getstate_PadByName (PadName : TPCBString) : IPCB_Primitive;
```

Description

Example

See also

IPCB_Net interface

Getstate_PadByPinDescription method

(IPCB_Net interface)

Syntax

```
Function Getstate_PadByPinDescription (PinDes : TPCBString) : IPCB_Primitive;
```

Description

Example

See also

IPCB_Net interface

GetState_PinCount method

(IPCB_Net interface)

Syntax

```
Function GetState_PinCount : Integer;
```

Description

Example

See also

IPCB_Net interface

GetState_RoutedLength method

(IPCB_Net interface)

Syntax

```
Function GetState_RoutedLength : TCoord;
```

Description

Example

See also

IPCB_Net interface

GetState_ViaCount method

(IPCB_Net interface)

Syntax

```
Function GetState_ViaCount : Integer;
```

Description

Example

See also

IPCB_Net interface

SetState_Color method

(IPCB_Net interface)

Syntax

```
Procedure SetState_Color (Color : TColor);
```

Description

Example

See also

IPCB_Net interface

SetState_ConnectsVisible method

(IPCB_Net interface)

Syntax

```
Procedure SetState_ConnectsVisible (Value : Boolean);
```

Description

Example

See also

IPCB_Net interface

SetState_IsHighlighted method

(IPCB_Net interface)

Syntax

```
Procedure SetState_IsHighlighted (Dummy : Boolean);
```

Description

Example

See also

IPCB_Net interface

SetState_Name method

(IPCB_Net interface)

Syntax

```
Procedure SetState_Name (Name : TPCBString);
```

Description

Example

See also

IPCB_Net interface

Methods

CancelGroupWarehouseRegistration method

(IPCB_Net interface)

Syntax

```
Procedure CancelGroupWarehouseRegistration (iPad : IPCB_Pad);
```

Description

Example

See also

IPCB_Net interface

ConnectivelyInValidate method

(IPCB_Net interface)

Syntax

```
Procedure ConnectivelyInValidate;
```

Description

Example

See also

IPCB_Net interface

GetLogicalNet method

(IPCB_Net interface)

Syntax

```
Function GetLogicalNet : IPCB_Group;
```

Description

Example

See also

IPCB_Net interface

HideNetConnects method

(IPCB_Net interface)

Syntax

```
Procedure HideNetConnects;
```

Description

Example

See also

IPCB_Net interface

Rebuild method

(IPCB_Net interface)

Syntax

```
Procedure Rebuild;
```

Description

Example

See also

IPCB_Net interface

ShowNetConnects method

(IPCB_Net interface)

Syntax

```
Procedure ShowNetConnects;
```

Description

Example

See also

IPCB_Net interface

RegisterWithGroupWarehouse method

(IPCB_Net interface)

Syntax

```
Procedure RegisterWithGroupWarehouse (iPad : IPCB_Pad);
```

Description

Example

See also

IPCB_Net interface

Properties

Color property

(IPCB_Net interface)

Syntax

```
Property Color : TColor Read GetState_Color Write SetState_Color;
```

Description

Example

See also

IPCB_Net interface

ConnectivelyInvalid property

(IPCB_Net interface)

Syntax

```
Property ConnectivelyInvalid : Boolean Read GetState_ConnectivelyInvalid;
```

Description

Example

See also

IPCB_Net interface

ConnectsVisible property

(IPCB_Net interface)

Syntax

```
Property ConnectsVisible : Boolean Read GetState_ConnectsVisible Write
SetState_ConnectsVisible;
```

Description

Example

See also

IPCB_Net interface

IsHighlighted property

(IPCB_Net interface)

Syntax

```
Property IsHighlighted : Boolean Read GetState_IsHighlighted Write SetState_IsHighlighted;
```

Description

Example

See also

IPCB_Net interface

Name property

(IPCB_Net interface)

Syntax

```
Property Name : TPCBString Read GetState_Name Write SetState_Name;
```

Description

Example

See also

IPCB_Net interface

PadByName [N property

(IPCB_Net interface)

Syntax

```
Property PadByName [N : TPCBString ] : IPCB_Primitive Read Getstate_PadByName;
```

Description

Example

See also

IPCB_Net interface

PadByPinDescription [N property

(IPCB_Net interface)

Syntax

```
Property PadByPinDescription [N : TPCBString ] : IPCB_Primitive Read
Getstate_PadByPinDescription;
```

Description

Example

See also

IPCB_Net interface

PinCount property

(IPCB_Net interface)

Syntax

```
Property PinCount : Integer Read GetState_PinCount;
```

Description

Example

See also

IPCB_Net interface

RoutedLength property

(IPCB_Net interface)

Syntax

```
Property RoutedLength : TCoord Read GetState_RoutedLength;
```

Description

Example

See also

IPCB_Net interface

ViaCount property

(IPCB_Net interface)

Syntax

```
Property ViaCount : Integer Read GetState_ViaCount;
```

Description

Example

See also

IPCB_Net interface

IPCB_ObjectClass Interface

Overview

A class is defined as a group or set of objects, identified by its unique class name. The PCB editor in the Altium Designer supports Net Classes, Component Classes and From-To Classes.

An object can belong to more than one class. You can create classes (or groups) of objects. Classes of Components, Nets and From-Tos can be created, and multiple membership is permitted. Classes are used to quickly identify a group of objects. For example, you could create a class of components called Surface Mount.

When you set up a paste mask expansion rule for the surface mount components, you simply set the rule scope to Component Class and select the Surface Mount class. Or you may have a set of nets, such as the power nets, which have different clearance requirements from the signal nets. You can create a Net Class which includes all these nets, and then use the Net Class scope when you define the clearance design rule for these nets.

Notes

An ObjectClass object can be created from the PCBClassFactoryByClassMember or PCBObjectFactory methods from the **IPCB_ServerInterface** interface.

The IPCB_ObjectClass hierarchy;

IPCB_Primitive
IPCB_ObjectClass

IPCB_ObjectClass methods

GetState_MemberKind
GetState_Name
GetState_SuperClass
GetState_MemberName

SetState_MemberKind
SetState_Name
SetState_SuperClass

AddMemberByName
AddMember
RemoveMember
RemoveAllMembers
IsMember
IsLayerMember
AddLayerMember
RemoveLayerMember
IsValidObjectKind

IPCB_ObjectClass properties

MemberKind
Name
SuperClass
MemberName [I

Example

```
Var
    Board    : IPCB_Board;
    NetClass : IPCB_ObjectClass;
Begin
    Board := PCBServer.GetCurrentPCBBoard;
    If Board = Nil Then Exit;
    PCBServer.PreProcess;
    NetClass := PCBServer.PCBClassFactoryByClassMember(eClassMemberKind_Net);
    NetClass.SuperClass := False;
    NetClass.Name := 'NetGndClass';
    NetClass.AddMemberByName('GND');
    Board.AddPCBObject(NetClass);
    PCBServer.PostProcess;
End;
```

See also

IPCB_Primitive interface
IPCB_ServerInterface interface
TClassMemberKind enumerated values
PCB Design Objects
Object Class Reporter script from **Example**s\Scripts\Delphiscript Scripts\Pcb\Object Class Report
UnrouteNetClass script from **Example**s\Scripts\Delphiscript Scripts\Pcb\UnRoute Net Class\ folder.
CreateNetClass script from **Example**s\Scripts\Delphiscript Scripts\Pcb\ folder.
ComponentClassInfo script from **Example**s\Scripts\Delphiscript Scripts\Pcb\

GetState and SetState Methods

SetState_SuperClass method

(IPCB_ObjectClass interface)

Syntax

```
Procedure SetState_SuperClass (Value : Boolean);
```

Description

The SuperClass property denotes whether or not the interface contains all members of a particular kind. If this field is set to true, the members of the IPCB_ObjectClass object cannot be edited.

This Setter method is used by the SuperClass property,**Example**

See also

IPCB_ObjectClass interface

SetState_Name method

(IPCB_ObjectClass interface)

Syntax

```
Procedure SetState_Name (Value : TPCBString);
```

Description

This property denotes the name of this Object Class object for the PCB document. This setter method is used by the Name property.

Example

See also

IPCB_ObjectClass interface

SetState_MemberKind method

(IPCB_ObjectClass interface)

Syntax

```
Procedure SetState_MemberKind (Value : TClassMemberKind);
```

Description

This property denotes which particular objects can be stored in the list. This setter method is used by the MemberKind property.

Example

See also

IPCB_ObjectClass interface

GetState_SuperClass method

(IPCB_ObjectClass interface)

Syntax

```
Function GetState_SuperClass : Boolean;
```

Description

The SuperClass property denotes whether or not the interface contains all members of a particular kind. If this field is set to true, the members of the IPCB_ObjectClass object cannot be edited and contains all the names of the objects of the particular kind.

This Getter method is used by the SuperClass property.

Example

See also

IPCB_ObjectClass interface

TClassMemberKind enumerated values

GetState_Name method

(IPCB_ObjectClass interface)

Syntax

```
Function GetState_Name : TPCBString;
```

Description

This property denotes the name of this Object Class object for the PCB document. This getter method is used by the Name property.

Example

See also

IPCB_ObjectClass interface

GetState_MemberName method

(IPCB_ObjectClass interface)

Syntax

```
Function GetState_MemberName (I : Integer) : TPCBString;
```

Description

This property denotes the member name from the list of members in the IPCB_Object class interface. This getter method is used by the MemberName property.

Example

See also

IPCB_ObjectClass interface

GetState_MemberKind method

(IPCB_ObjectClass interface)

Syntax

```
Function GetState_MemberKind : TClassMemberKind;
```

Description

This method denotes which particular objects can be stored in the list. This getstate_MemberKind method is used by the **MemberKind** property.

Example

See also

IPCB_ObjectClass interface

TClassMemberKind type

Methods

AddLayerMember method

(IPCB_ObjectClass interface)

Syntax

```
Procedure AddLayerMember (L : TLayer);
```

Description

This **AddLayerMember** method adds a layer to the object class of eClassMemberKind_Layer type.

Example

See also

IPCB_ObjectClass interface

AddMember method

(IPCB_ObjectClass interface)

Syntax

```
Procedure AddMember (P : IPCB_Primitive);
```

Description

The **AddMember** method adds a primitive that belongs to the same member kind in the Object Class.

Example

See also

IPCB_ObjectClass interface

AddMemberByName method

(IPCB_ObjectClass interface)

Syntax

```
Procedure AddMemberByName (AName : TPCBString);
```

Description

This AddMemberByName adds a member by its name of the member kind in the object class.

Example

```
Var
    Board    : IPCB_Board;
    NetClass : IPCB_ObjectClass;
Begin
    Board := PCBServer.GetCurrentPCBBoard;
    If Board = Nil Then Exit;
    PCBServer.PreProcess;
    NetClass := PCBServer.PCBClassFactoryByClassMember(eClassMemberKind_Net);
    NetClass.SuperClass := False;
    NetClass.Name := 'NetGndClass';
    NetClass.AddMemberByName('GND');
    Board.AddPCBObject(NetClass);
    PCBServer.PostProcess;
End;
```

See also

IPCB_ObjectClass interface

TClassMemberKind enumerated values

IsLayerMember method

(IPCB_ObjectClass interface)

Syntax

```
Function IsLayerMember (L : TLayer) : Boolean;
```

Description

This function checks if this layer is part of the Object Class that is hosting layer classes only (of eClassMemberKind_Layer type).

Example

See also

IPCB_ObjectClass interface

TClassMemberKind enumerated values

IsMember method

(IPCB_ObjectClass interface)

Syntax

```
Function IsMember (S : TPCBString) : Boolean;
```

Description

This function checks if the member (by name) is part of the Object Class.

Example

See also

IPCB_ObjectClass interface

IsValidObjectKind method

(IPCB_ObjectClass interface)

Syntax

```
Function IsValidObjectKind (P : IPCB_Primitive) : Boolean;
```

Description

This function checks if the PCB design object is a valid object kind for this object class.

Example

See also

IPCB_ObjectClass interface

RemoveAllMembers method

(IPCB_ObjectClass interface)

Syntax

```
Procedure RemoveAllMembers;
```

Description

This method removes all the members for this object class.

Example

See also

IPCB_ObjectClass interface

RemoveLayerMember method

(IPCB_ObjectClass interface)

Syntax

```
Procedure RemoveLayerMember (L : TLayer);
```

Description

This method removes the specified layer from the Object Class that hosts the layer classes only.

Example

See also

IPCB_ObjectClass interface

RemoveMember method

(IPCB_ObjectClass interface)

Syntax

```
Procedure RemoveMember (P : IPCB_Primitive);
```

Description

This method removes the specified PCB design object from the list of members in this Object class.

Example

See also

IPCB_ObjectClass interface

Properties

MemberKind property

(IPCB_ObjectClass interface)

Syntax

```
Property MemberKind : TClassMemberKind Read GetState_MemberKind Write SetState_MemberKind;
```

Description

This property denotes which particular objects can be stored in the list.

This property is supported by the GetState_MemberKind and SetState_MemberKind methods.

Example

```
Var
    Board    : IPCB_Board;
    NetClass : IPCB_ObjectClass;
Begin
    Board := PCBServer.GetCurrentPCBBoard;
    If Board = Nil Then Exit;
    PCBServer.PreProcess;
    NetClass := PCBServer.PCBClassFactoryByClassMember(eClassMemberKind_Net);
    NetClass.SuperClass := False;
    NetClass.Name := 'NetGndClass';
    NetClass.AddMemberByName('GND');
    Board.AddPCBObject(NetClass);
    PCBServer.PostProcess;
End;
```

See also

IPCB_ObjectClass interface

TClassMemberKind type

MemberName property

(IPCB_ObjectClass interface)

Syntax

```
Property MemberName [I : Integer] : TPCBString Read GetState_MemberName;
```

Description

This property denotes the member name from the list of members in the IPCB_Object class interface. This read only property is supported by the GetState_MemberName method.

Example

See also

IPCB_ObjectClass interface

Name property

(IPCB_ObjectClass interface)

Syntax

```
Property Name : TPCBString Read GetState_Name Write SetState_Name;
```

Description

This property denotes the name of this Object Class object for the PCB document. This property is supported by the GetState_Name and SetState_Name methods.

Example

See also

IPCB_ObjectClass interface

SuperClass property

(IPCB_ObjectClass interface)

Syntax

```
Property SuperClass : Boolean Read GetState_SuperClass Write SetState_SuperClass;
```

Description

The **SuperClass** property denotes whether or not the interface contains all members of a particular kind. If this field is set to true, the members of the **IPCB_ObjectClass** object cannot be edited.

By default, a super class contains all members of the same member kind - for example, if layer kind is selected, then all layers is included for this Object Class.

This property is supported by the GetState_SuperClass and SetState_SuperClass methods.

Code Snippet **Example**

```
// AObjectClass is a IPCB_ObjectClass interface type
If AObjectClass.SuperClass Then
Begin
    // is a super class!
    Case AObjectClass.MemberKind Of
            eClassMemberKind_Net       : ARpt.Add('All Nets');
            eClassMemberKind_Component : ARpt.Add('All Components');
            eClassMemberKind_FromTo    : ARpt.Add('All FromTos');
            eClassMemberKind_Pad       : ARpt.Add('All Pads');
            eClassMemberKind_Layer     : ARpt.Add('All Layers');
    End;
End;
```

See also

IPCB_ObjectClass interface

TClassMemberKind type

IPCB_Pad Interface

Overview

Pad objects are hole connectors for components and for connection to signal tracks. Pads can be either multilayered or single layered. Pad shapes include circular, rectangular, rounded rectangular or octagonal with X, Y sizes definable from 1 to 10000mils.

Hole size can range from 0 (SMD) to 1000mils.

Pads can be identified with a designator up to four characters long. On a multilayer pad, the Top layer, Mid layer and Bottom layer pad shape and size can be independently assigned to define a pad stack. Note that the surface mount components and edge connectors have single layer pads on the Top and/or Bottom layers.

Altium Designer supports a Full Stack Pad mode for ultimate control over the padstack. This allows different sizes and shapes on all signal layers. Also pads and vias can be selectively tented on the top or bottom side. Altium Designer also supports three types of pad definitions: Simple, Top-Mid-Bottom and Full Stack.

Notes

The Corner radius attribute of rounded pads is represented by the IPCB_Pad2 interface.

A Paste Mask expansion property for a pad object is currently relevant just for pads on top and bottom copper layers.

Vias do not have a paste mask layer. Paste mask layers are used to design stencils which will selectively place solder paste on a blank PCB. Solder paste is only placed on pads where component leads are to be soldered to them. Vias normally don't have anything soldered onto them.

The IPCB_Pad interface hierarchy;

IPCB_Primitive

- IPCB_Pad
- IPCB_Pad2

IPCB_Pad methods	IPCB_Pad properties
GetState_XLocation	X
GetState_YLocation	Y
SetState_XLocation	PinDescriptor
SetState_YLocation	IsConnectedToPlane
GetState_PinDescriptorString	Mode
GetState_IsConnectedToPlane	XSizeOnLayer
SetState_IsConnectedToPlane	YSizeOnLayer
GetState_Mode	ShapeOnLayer
SetState_Mode	XStackSizeOnLayer
GetState_XSizeOnLayer	YStackSizeOnLayer
GetState_YSizeOnLayer	StackShapeOnLayer
GetState_ShapeOnLayer	
GetState_XStackSizeOnLayer	TopXSize
GetState_YStackSizeOnLayer	TopYSize
GetState_StackShapeOnLayer	MidXSize
GetState_TopXSize	MidYSize
GetState_TopYSize	BotXSize
GetState_TopShape	BotYSize
GetState_BotXSize	TopShape
GetState_BotYSize	MidShape
GetState_BotShape	BotShape
GetState_MidXSize	
GetState_MidYSize	HoleSize
GetState_MidShape	Rotation
	Name
GetState_SwapID_Pad	Width
GetState_SwapID_Gate	
GetState_SwappedPadName	SwapID_Pad
GetState_GateID	SwapID_Gate
GetState_OwnerPart_ID	SwappedPadName

SetState_BotShape
SetState_BotXSize
SetState_BotYSize
SetState_MidShape
SetState_MidXSize
SetState_MidYSize
SetState_TopShape
SetState_TopXSize
SetState_TopYSize

SetState_XStackSizeOnLayer
SetState_YStackSizeOnLayer
SetState_StackShapeOnLayer
SetState_SwapID_Pad
SetState_SwapID_Gate
SetState_SwappedPadName
SetState_OwnerPart_ID

GetState_HoleSize
SetState_HoleSize
GetState_Rotation
SetState_Rotation
GetState_Name
SetState_Name
GetState_WidthOnLayer
GetState_Cache
SetState_Cache

GetState_Plated
GetState_DrillType
GetState_HoleType
GetState_HoleWidth
GetState_XPadOffsetOnLayer
GetState_YPadOffsetOnLayer
GetState_HoleRotation
SetState_DrillType
SetState_HoleType
SetState_HoleWidth
SetState_XPadOffsetOnLayer
SetState_YPadOffsetOnLayer
SetState_HoleRotation

BoundingRectangleOnLayer
RotateAroundXY
IsPadStack
IsSurfaceMount

Cache
WidthOnLayer
OwnerPart_ID

Plated

DrillType
HoleType
HoleWidth
XPadOffset
YPadOffset
HoleRotation

PlaneConnectionStyleForLayer

InvalidateSizeShape
ValidateSizeShape
ReValidateSizeShape
UpdateCache
InvalidateCache

Example

This example creates a new pad object and its associated new pad cache and places it on the current PCB document.

```
Procedure PlaceAPCBPad;
Var
    Board         : IPCB_Board;
    WorkSpace     : IWorkSpace;

    Pad           : IPCB_Pad;
    Padcache      : TPadCache;
    TopLayerWidth : TCoord;
Begin
    //Create a new PCB document
    WorkSpace := GetWorkSpace;
    If WorkSpace = Nil Then Exit;
    Workspace.DM_CreateNewDocument('PCB');

    If PCBServer = Nil Then Exit;
    Board := PCBServer.GetCurrentPCBBoard;
    If Board = Nil then exit;

    // Create a Pad object
    Pad := PCBServer.PCBObjectFactory(ePadObject, eNoDimension, eCreate_Default);
    Pad.SetState_XLocation  := MilsToCoord(3000);
    Pad.SetState_YLocation  := MilsToCoord(3000);

    // Setup a pad cache which has common values
    Padcache := Pad.GetState_Cache;
    Padcache.ReliefAirGap := MilsToCoord(11);
    Padcache.PowerPlaneReliefExpansion := MilsToCoord(11);
    Padcache.PowerPlaneClearance       := MilsToCoord(11);
    Padcache.ReliefConductorWidth      := MilsToCoord(11);
    Padcache.SolderMaskExpansion       := MilsToCoord(11);
    Padcache.SolderMaskExpansionValid  := eCacheManual;
    Padcache.PasteMaskExpansion        := MilsToCoord(11);
    Padcache.PasteMaskExpansionValid   := eCacheManual;

    // Assign a new pad cache to the pad
    Pad.SetState_Cache := Padcache;
    TopLayerWidth      := Pad.GetState_WidthOnLayer(eBottomLayer);
    Board.AddPCBObject(Pad);
```

```
    // Refresh PCB document
    ResetParameters;
    AddStringParameter('Action', 'All');
    RunProcess('PCB:Zoom');
End;
```

See also

IPCB_Primitive interface
IPCB_Via interface
TPadName value
TPadCache value
TPadSwapName value
TShape enumerated values
TAngle value
PCB Design Objects
Script examples in **Example**s\Scripts\DelphiScript\PCB\ folder

GetState and SetState Methods

GetState_DrillType method

(IPCB_Pad interface)

Syntax

```
Function  GetState_DrillType : TExtendedDrillType;
```

Description

This function obtains the drill type used for this pad's hole on the PCB.

Example

See also

IPCB_Pad interface

GetState_HoleType method

(IPCB_Pad interface)

Syntax

```
Function  GetState_HoleType : TExtendedHoleType;
```

Description

This function obtains the hole type of the pad's hole.

Example

See also

IPCB_Pad interface

GetState_HoleWidth method

(IPCB_Pad interface)

Syntax

```
Function  GetState_HoleWidth : TCoord;
```

Description

This function obtains the hole width in TCoord units.

Example

See also

IPCB_Pad interface

GetState_XPadOffsetOnLayer method

(IPCB_Pad interface)

Syntax

```
Function  GetState_XPadOffsetOnLayer  (L : TLayer) : TCoord;
```

Description

This function is not implemented.

Example

See also

IPCB_Pad interface

GetState_YPadOffsetOnLayer method

(IPCB_Pad interface)

Syntax

```
Function GetState_YPadOffsetOnLayer  (L : TLayer) : TCoord;
```

Description

This function is not implemented.

Example

See also

IPCB_Pad interface

SetState_DrillType method

(IPCB_Pad interface)

Syntax

```
Procedure SetState_DrillType (DrillType : TExtendedDrillType);
```

Description

This procedure sets the drill type used to drill a hole on the PCB. This attribute is used by the manufacturing output file such as the CAM files.

Example

See also

IPCB_Pad interface

TExtendedDrillType type.

SetState_HoleType method

(IPCB_Pad interface)

Syntax

```
Procedure SetState_HoleType (HoleType  : TExtendedHoleType);
```

Description

This procedure sets the hole type of the pad’s hole. There are three hole types – Round Hole, Square Hole and Slotted Hole.

Example

See also

IPCB_Pad interface

TExtendedHoleType type.

SetState_HoleWidth method

(IPCB_Pad interface)

Syntax

```
Procedure SetState_HoleWidth (HoleWidth : TCoord);
```

Description

This function sets the hole width of a pad's hole on the PCB.

Example

See also

IPCB_Pad interface

SetState_XPadOffsetOnLayer method

(IPCB_Pad interface)

Syntax

```
Procedure SetState_XPadOffsetOnLayer(L       : TLayer;
                                     XOffset : TCoord);
```

Description

This function is not implemented.

Example

See also

IPCB_Pad interface

SetState_YPadOffsetOnLayer method

(IPCB_Pad interface)

Syntax

```
Procedure SetState_YPadOffsetOnLayer(L       : TLayer;
                                     YOffset : TCoord);
```

Description

This function is not implemented.

Example

See also

IPCB_Pad interface

SetState_HoleRotation method

(IPCB_Pad interface)

Syntax

```
Procedure SetState_HoleRotation (HRotation : TAngle);
```

Description

This function sets the rotation property of a pad's hole.

Example

See also

IPCB_Pad interface

TAngle type

SetState_YStackSizeOnLayer method

(IPCB_Pad interface)

Syntax

```
Procedure SetState_YStackSizeOnLayer (L : TLayer;Value : TCoord);
```

Description

This YStackSizeOnLayer procedure determines the size of the pad in Y direction on the specified layer only if the pad has an external stack (ePadMode_ExternalStack type).

This method is used for the YStackSizeOnLayer property.

Example

See also

IPCB_Pad interface

SetState_YLocation method

(IPCB_Pad interface)

Syntax

```
Procedure SetState_YLocation (AY : TCoord);
```

Description

The SetState_XLocation and SetState_YLocation methods set the location of the pad with respect to the PCB document it is on.

These methods are used for the X and Y properties.

Example

See also

IPCB_Pad interface

SetState_XStackSizeOnLayer method

(IPCB_Pad interface)

Syntax

```
Procedure SetState_XStackSizeOnLayer (L : TLayer;Value : TCoord);
```

Description

This XStackSizeOnLayer procedure determines the size of the pad in X direction on the specified layer only if the pad has an external stack (ePadMode_ExternalStack type).

This method is used for the XStackSizeOnLayer property.

Example

See also

IPCB_Pad interface

SetState_XLocation method

(IPCB_Pad interface)

Syntax

```
Procedure SetState_XLocation (AX : TCoord);
```

Description

The SetState_XLocation and SetState_YLocation methods set the location of the pad with respect to the PCB document it is on.

These methods are used for the X and Y properties.

Example

See also

IPCB_Pad interface

SetState_TopYSize method

(IPCB_Pad interface)

Syntax

```
Procedure SetState_TopYSize (Value : TCoord);
```

Description

This procedure determines the top size in U direction of the pad with a top-middle-bottom stack up. This method is used for the TopYSize property.

Example

See also

IPCB_Pad interface

SetState_TopXSize method

(IPCB_Pad interface)

Syntax

```
Procedure SetState_TopXSize (Value : TCoord);
```

Description

This procedure determines the top size in X direction of the pad with a top-middle-bottom stack up. This method is used for the TopXSize property.

Example

See also

IPCB_Pad interface

SetState_TopShape method

(IPCB_Pad interface)

Syntax

```
Procedure SetState_TopShape (Value : TShape);
```

Description

This procedure determines the top shape of the pad with a top-middle-bottom stack up. This method is used for the TopShape property.

Example

See also

IPCB_Pad interface

TShape type

SetState_SwappedPadName method

(IPCB_Pad interface)

Syntax

```
Procedure SetState_SwappedPadName (Value : TPCBString);
```

Description

Example

See also

IPCB_Pad interface

TPCBString type

SetState_SwapID_Pad method

(IPCB_Pad interface)

Syntax

```
Procedure SetState_SwapID_Pad (Value : TPCBString);
```

Description

Example

See also

IPCB_Pad interface

TPCBString type

SetState_SwapID_Gate method

(IPCB_Pad interface)

Syntax

```
Procedure SetState_SwapID_Gate (Value : TPCBString);
```

Description

Example

See also

IPCB_Pad interface

TPCBString type

SetState_StackShapeOnLayer method

(IPCB_Pad interface)

Syntax

```
Procedure SetState_StackShapeOnLayer (L : TLayer;Value : TShape);
```

Description

This procedure determines what shape the pad stack is on that layer. This method is used by the StackShapeOnLayer property.

Example

See also

IPCB_Pad interface

SetState_Rotation method

(IPCB_Pad interface)

Syntax

```
Procedure SetState_Rotation (Value : TAngle);
```

Description

This method sets the rotation of the pad object in degrees (of TAngle type 0 -360 degrees). This method is used for the Rotation property.

Example

See also

IPCB_Pad interface

SetState_Name method

(IPCB_Pad interface)

Syntax

```
Procedure SetState_Name (Value : TPCBString);
```

Description

This method sets the name which is the designator of this pad object. This method is used for the Name property.

Example

See also

IPCB_Pad interface

TPCBString type

SetState_Mode method

(IPCB_Pad interface)

Syntax

```
Procedure SetState_Mode (Mode : TPadMode);
```

Description

The **Mode** property determines what type of pad it is - a simple pad, a pad with three Top, Middle and Bottom layer stack up or a pad with a complex stack up.

If Mode is Simple (ePadMode_Simple) then you only deal with X,Y locations and the TopXSize, TopYSize and TopShape properties.

If Mode is Top-Mid-Bottom stack (ePadMode_LocalStack) then you deal with X,Y Locations, Top.., Mid.. and Bot.. properties.

If Mode is Full Stack (ePadMode_ExternalStack) then you deal with XStackSizeOnLayer, YStackSizeOnLayer and StackShapeOnLayer properties.

The method is used by the Mode property.

Example

See also

IPCB_Pad interface

SetState_MidYSize method

(IPCB_Pad interface)

Syntax

```
Procedure SetState_MidYSize (Value : TCoord);
```

Description

This procedure determines the middle size in Y direction of the pad with a top-middle-bottom stack up. This method is used for the MidYSize property.

Example

See also

IPCB_Pad interface

SetState_MidXSize method

(IPCB_Pad interface)

Syntax

```
Procedure SetState_MidXSize (Value : TCoord);
```

Description

This procedure determines the middle size in X direction of the pad with a top-middle-bottom stack up. This method is used for the MidXSize property.

Example

See also

IPCB_Pad interface

SetState_MidShape method

(IPCB_Pad interface)

Syntax

```
Procedure SetState_MidShape (Value : TShape);
```

Description

This procedure determines the middle shape of the pad with a top-middle-bottom stack up. This method is used for the MidShape property.

Example

See also

IPCB_Pad interface

TShape type

SetState_IsConnectedToPlane method

(IPCB_Pad interface)

Syntax

```
Procedure SetState_IsConnectedToPlane (Layer : TLayer;Value : Boolean);
```

Description

This method sets a boolean value to connect the pad to the specified plane (one of the power internal planes) or not.

This method is used by the IsConnectedToPlane property.

Example

See also

IPCB_Pad interface

SetState_HoleSize method

(IPCB_Pad interface)

Syntax

```
Procedure SetState_HoleSize (Value : TCoord);
```

Description

This method sets the hole size of a pad object where component pins or wires can be passed through and soldered in place.

This method is used by the HoleSize property.

Example

See also

IPCB_Pad interface

SetState_GateID method

(IPCB_Pad interface)

Syntax

```
Procedure SetState_GateID (Value : Integer);
```

Description

Example

See also

IPCB_Pad interface

SetState_Cache method

(IPCB_Pad interface)

Syntax

```
Procedure SetState_Cache (Value : TPadCache);
```

Description

Example

See also

IPCB_Pad interface

SetState_BotYSize method

(IPCB_Pad interface)

Syntax

```
Procedure SetState_BotYSize (Value : TCoord);
```

Description

This procedure determines the bottom size in the Y direction of the pad with a top-middle-bottom stack up. This method is used for the BotYSize property.

Example

See also

IPCB_Pad interface

SetState_BotXSize method

(IPCB_Pad interface)

Syntax

```
Procedure SetState_BotXSize (Value : TCoord);
```

Description

This procedure determines the bottom size in the X direction of the pad with a top-middle-bottom stack up. This method is used for the BotXSize property.

Example

See also

IPCB_Pad interface

SetState_BotShape method

(IPCB_Pad interface)

Syntax

```
Procedure SetState_BotShape (Value : TShape);
```

Description

This procedure determines the bottom shape of the pad with a top-middle-bottom stack up. This method is used for the BotShape property.

Example

See also

IPCB_Pad interface

TShape type

GetState_YStackSizeOnLayer method

(IPCB_Pad interface)

Syntax

```
Function GetState_YStackSizeOnLayer (L : TLayer) : TCoord;
```

Description

This YStackSizeOnLayer function determines the size of the pad in Y direction on the specified layer only if the pad has an external stack (ePadMode_ExternalStack type).

This method is used for the YStackSizeOnLayer property.

Example

See also

IPCB_Pad interface

GetState_YSizeOnLayer method

(IPCB_Pad interface)

Syntax

```
Function GetState_YSizeOnLayer (L : TLayer) : TCoord;
```

Description

This function determines what size in Y direction the pad is on this specified layer. This method is used for the YSizeOnLayer property.

Example

See also

IPCB_Pad interface

GetState_YLocation method

(IPCB_Pad interface)

Syntax

```
Function GetState_YLocation : TCoord;
```

Description

The GetState_XLocation and GetState_YLocation methods retrieves the location of the pad with respect to the PCB document it is on.

These methods are used for the X and Y properties.

Example

See also

IPCB_Pad interface

GetState_XStackSizeOnLayer method

(IPCB_Pad interface)

Syntax

```
Function GetState_XStackSizeOnLayer (L : TLayer) : TCoord;
```

Description

This XStackSizeOnLayer function determines the size of the pad in X direction on the specified layer only if the pad has an external stack (ePadMode_ExternalStack type).

This method is used for the XStackSizeOnLayer property.

Example

See also

IPCB_Pad interface

GetState_XSizeOnLayer method

(IPCB_Pad interface)

Syntax

```
Function GetState_XSizeOnLayer (L : TLayer) : TCoord;
```

Description

This function determines what size in X direction the pad is on this specified layer. This method is used for the XSizeOnLayer property.

Example

See also

IPCB_Pad interface

GetState_XLocation method

(IPCB_Pad interface)

Syntax

```
Function GetState_XLocation : TCoord;
```

Description

The GetState_XLocation and GetState_YLocation methods retrieves the location of the pad with respect to the PCB document it is on.

These methods are used for the X and Y properties.

Example

See also

IPCB_Pad interface

GetState_WidthOnLayer method

(IPCB_Pad interface)

Syntax

```
Function GetState_WidthOnLayer (L : TLayer) : TCoord;
```

Description

This WidthOnLayer function retrieves the width of the pad on the specified layer. This property is used by the WidthOnLayer property.

Example

See also

IPCB_Pad interface

GetState_TopYSize method

(IPCB_Pad interface)

Syntax

```
Function GetState_TopYSize : TCoord;
```

Description

This function determines the top size in Y direction of the pad with a top-middle-bottom stack up. This method is used for the TopYSize property.

Example

See also

IPCB_Pad interface

GetState_TopXSize method

(IPCB_Pad interface)

Syntax

```
Function GetState_TopXSize : TCoord;
```

Description

This function determines the top size in X direction of the pad with a top-middle-bottom stack up. This method is used for the TopXSize property.

Example

See also

IPCB_Pad interface

GetState_TopShape method

(IPCB_Pad interface)

Syntax

```
Function GetState_TopShape : TShape;
```

Description

This function determines the top shape of the pad with a top-middle-bottom stack up. This method is used for the TopShape property.

Example

See also

IPCB_Pad interface

TShape type

GetState_SwappedPadName method

(IPCB_Pad interface)

Syntax

```
Function GetState_SwappedPadName : TPCBString;
```

Description

Example

See also

IPCB_Pad interface

TPCBString type

GetState_SwapID_Pad method

(IPCB_Pad interface)

Syntax

```
Function GetState_SwapID_Pad : TPCBString;
```

Description

Example

See also

IPCB_Pad interface

TPCBString type

GetState_SwapID_Gate method

(IPCB_Pad interface)

Syntax

```
Function GetState_SwapID_Gate : TPCBString;
```

Description

Example

See also

IPCB_Pad interface

TPCBString type

GetState_StackShapeOnLayer method

(IPCB_Pad interface)

Syntax

```
Function GetState_StackShapeOnLayer (L : TLayer) : TShape;
```

Description

This function determines what shape the pad stack is on that layer. This method is used by the StackShapeOnLayer property.

Example

See also

IPCB_Pad interface

TShape type

GetState_ShapeOnLayer method

(IPCB_Pad interface)

Syntax

```
Function GetState_ShapeOnLayer (L : TLayer) : TShape;
```

Description

This property determines what shape the pad stack is on that layer. This method is used by the ShapeOnLayer property.

Example

See also

IPCB_Pad interface

TShape type

GetState_Rotation method

(IPCB_Pad interface)

Syntax

```
Function GetState_Rotation : TAngle;
```

Description

This method retrieves the rotation of the pad object in degrees (of TAngle type 0 -360 degrees).

This method is used for the Rotation property.

Example

See also

IPCB_Pad interface

GetState_PinDescriptorString method

(IPCB_Pad interface)

Syntax

```
Function GetState_PinDescriptorString : TPCBString;
```

Description

This function obtains the description of the pin which represents the pad of a component. This method is used by the PinDescriptorString property.

Example

See also

IPCB_Pad interface

TPCBString type

GetState_Name method

(IPCB_Pad interface)

Syntax

```
Function GetState_Name : TPCBString;
```

Description

This method retrieves the name which is the designator of this pad object.

This method is used for the Name property.

Example

See also

IPCB_Pad interface

TPCBString type

GetState_Mode method

(IPCB_Pad interface)

Syntax

```
Function GetState_Mode : TPadMode;
```

Description

The **Mode** function determines what type of pad it is - a simple pad, a pad with three Top, Middle and Bottom layer stack up or a pad with a complex stack up.

If Mode is Simple (ePadMode_Simple) then you only deal with X,Y locations and the TopXSize, TopYSize and TopShape properties.

If Mode is Top-Mid-Bottom stack (ePadMode_LocalStack) then you deal with X,Y Locations, Top.., Mid.. and Bot.. properties.

If Mode is Full Stack (ePadMode_ExternalStack) then you deal with XStackSizeOnLayer, YStackSizeOnLayer and StackShapeOnLayer properties.

The method is used by the Mode property.

Example

See also

IPCB_Pad interface

GetState_MidYSize method

(IPCB_Pad interface)

Syntax

```
Function GetState_MidYSize : TCoord;
```

Description

This function determines the middle size in Y direction of the pad with a top-middle-bottom stack up. This method is used by the MidYSize property.

Example

See also

IPCB_Pad interface

GetState_MidXSize method

(IPCB_Pad interface)

Syntax

```
Function GetState_MidXSize : TCoord;
```

Description

This function determines the middle size in X direction of the pad with a top-middle-bottom stack up. This method is used for the MidXSize property.

Example

See also

IPCB_Pad interface

GetState_MidShape method

(IPCB_Pad interface)

Syntax

```
Function GetState_MidShape : TShape;
```

Description

This function determines the middle shape of the pad with a top-middle-bottom stack up. This method is used for the MidShape property.

Example

See also

IPCB_Pad interface

TShape type

GetState_IsConnectedToPlane method

(IPCB_Pad interface)

Syntax

```
Function GetState_IsConnectedToPlane (Layer : TLayer) : Boolean;
```

Description

This method retrieves a boolean value whether the pad is connected to the specified plane (one of the power internal planes) or not.

This method is used by the IsConnectedToPlane property.

Example

See also

IPCB_Pad interface

GetState_HoleSize method

(IPCB_Pad interface)

Syntax

```
Function GetState_HoleSize : TCoord;
```

Description

This method retrieves the hole size of a pad object where component pins or wires can be passed through and soldered in place.

This method is used by the HoleSize property.

Example

See also

IPCB_Pad interface

GetState_GateID method

(IPCB_Pad interface)

Syntax

```
Function GetState_GateID : Integer;
```

Description

Example

See also

IPCB_Pad interface

GetState_Cache method

(IPCB_Pad interface)

Syntax

```
Function GetState_Cache : TPadCache;
```

Description

This method retrieves the global cache that stores various design rule settings for pad and via objects.

This method is used for the Cache property.

Example

See also

IPCB_Pad interface

GetState_BotYSize method

(IPCB_Pad interface)

Syntax

```
Function GetState_BotYSize : TCoord;
```

Description

This function determines the bottom size in Y direction of the pad with a top-middle-bottom stack up. This method is used for the BotYSize property.

Example

See also

IPCB_Pad interface

GetState_BotXSize method

(IPCB_Pad interface)

Syntax

```
Function GetState_BotXSize : TCoord;
```

Description

This function determines the bottom size in X direction of the pad with a top-middle-bottom stack up. This method is used for the BotXSize property.

Example

See also

IPCB_Pad interface

GetState_BotShape method

(IPCB_Pad interface)

Syntax

```
Function GetState_BotShape : TShape;
```

Description

This function determines the bottom shape of the pad with a top-middle-bottom stack up. This method is used for the BotShape property.

Example

See also

IPCB_Pad interface

TShape type

GetState_Plated method

(IPCB_PCB interface)

Syntax

```
Function  GetState_Plated : Boolean;
```

Description

This method determines whether the pad is plated or not. This method is used for the Plated property.

Example

See also

IPCB_Pad interface

SetState_Plated method

(IPCB_Pad interface)

Syntax

```
Procedure SetState_Plated (Value  : Boolean);
```

Description

This method determines whether the pad is plated or not. This method is used for the Plated property.

Example

See also

IPCB_Pad interface

Methods

BoundingRectangleOnLayer method

(IPCB_Pad interface)

Syntax

```
Function BoundingRectangleOnLayer (ALayer : TLayer) : TCoordRect;
```

Description

This function retrieves the bounding rectangle (of TCoordRect type) of the component on the specified layer of the PCB document.

Example

See also

IPCB_Pad interface

IsPadStack method

(IPCB_Pad interface)

Syntax

```
Function IsPadStack : Boolean;
```

Description

This function determines whether the pad is a full stack up pad or not. Use this function before you change the properties of a pad stack. You can also use the Mode property to check what type of stack up the pad is.

Example

See also

IPCB_Pad interface

TPadMode property

IsSurfaceMount method

(IPCB_Pad interface)

Syntax

```
Function IsSurfaceMount : Boolean;
```

Description

The pad is a surface mount if the holesize is 0 in size and is on top and/or bottom layers only.

Example

See also

IPCB_Pad interface

PlaneConnectionStyleForLayer method

(IPCB_Pad interface)

Syntax

```
Function PlaneConnectionStyleForLayer(ALayer : TLayer) : TPlaneConnectionStyle;
```

Description

Pads automatically connect to an internal power plane layer that is assigned the same net name. The pad will connect to the plane depending on the applicable Power Plane Connect Style design rule. If you do not want pads to connect to power planes, add another Power Plane Connect Style design rule targeting the specific pads required and with a connection style of No Connect.

The Connect Style defines the style of the connection from a pin of a component, targeted by the scope (Full Query) of the rule, to a power plane.

The following three styles as per the `TPlaneConnectionStyle` type are available:

ePlaneNoConnect - do not connect a component pin to the power plane.

ePlaneReliefConnect - connect using solid copper to the pin.

ePlaneDirectConnect (default) - connect using a thermal relief connection.

Example

See also

IPCB_Pad interface

TPlaneConnectionStyle type

RotateAroundXY method

(IPCB_Pad interface)

Syntax

```
Procedure RotateAroundXY (AX, AY : TCoord;Angle : TAngle);
```

Description

This method rotates a pad object on the PCB document about the AX, AY coordinates with an angle in degrees.

To ensure the pad rotates without moving about, pass in its midpoint (between X1,X2 and Y1, Y2) attributes for the AX,AY parameters or use the Rotation property.

Example

See also

IPCB_Pad interface

Invalidate method

(IPCB_PCB interface)

Syntax

```
Procedure InvalidateSizeShape;
```

Description

Example

See also

IPCB_Pad interface

ValidateSizeShape method

(IPCB_Pad interface)

Syntax

```
Procedure ValidateSizeShape;
```

Description

Example

See also

IPCB_Pad interface

RevalidateSizeShape method

(IPCB_Pad interface)

Syntax

```
Procedure ReValidateSizeShape;
```

Description

Example

See also

IPCB_Pad interface

Properties

BotShape property

(IPCB_Pad interface)

Syntax

```
Property BotShape : TShape Read GetState_BotShape Write SetState_BotShape;
```

Description

This property determines the bottom shape of the pad with a top-middle-bottom stack up. This property is supported by the GetState_BotShape and SetState_BotShape methods.

Example

See also

IPCB_Pad interface

TShape type

TShape type

BotXSize property

(IPCB_Pad interface)

Syntax

```
Property BotXSize : TCoord Read GetState_BotXSize Write SetState_BotXSize;
```

Description

This property determines the bottom X Size of the pad with a top-middle-bottom stack up.

This property is supported by the GetState_BotXSize and SetState_BotXSize methods.

Example

See also

IPCB_Pad interface

BotYSize property

(IPCB_Pad interface)

Syntax

```
Property BotYSize : TCoord Read GetState_BotYSize Write SetState_BotYSize;
```

Description

This property determines the bottom Y Size of the pad with a top-middle-bottom stack up.

This property is supported by the GetState_BotYSize and SetState_BotYSize methods.

Example

See also

IPCB_Pad interface

Cache property

(IPCB_Pad interface)

Syntax

```
Property Cache : TPadCache Read GetState_Cache Write SetState_Cache;
```

Description

This Cache property represents the global cache that stores various design rule settings for pad and via objects. This property is supported by the GetState_Cache and SetState_Cache methods.

Example

```
    (* Create a Pad object*)
    Pad := PCBServer.PCBObjectFactory(ePadObject, eNoDimension, eCreate_Default);
    Pad.X  := MilsToCoord(3000);
    Pad.Y  := MilsToCoord(3000);

    (* Setup a pad cache *)
    Padcache := Pad.Cache;
    Padcache.ReliefAirGap := MilsToCoord(11);
    Padcache.PowerPlaneReliefExpansion := MilsToCoord(11);
    Padcache.PowerPlaneClearance       := MilsToCoord(11);
    Padcache.ReliefConductorWidth      := MilsToCoord(11);
    Padcache.SolderMaskExpansion       := MilsToCoord(11);
    Padcache.SolderMaskExpansionValid  := eCacheManual;
    Padcache.PasteMaskExpansion        := MilsToCoord(11);
    Padcache.PasteMaskExpansionValid   := eCacheManual;

    (* Assign the new pad cache to the pad*)
    Pad.Cache := Padcache;
    Board.AddPCBObject(Pad);
```

See also

IPCB_Pad interface

TPadCache type

PadViaCacheProperties script from **Examples**\Scripts\Delphiscript Scripts\Pcb\ folder.

DrawObjects script from **Examples**\Scripts\DelphiScript Scripts\PCB\ folder.

GateID property

(IPCB_Pad interface)

Syntax

```
Property GateID : Integer Read GetState_GateID Write SetState_GateID;
```

Description

Example

See also

IPCB_Pad interface

HoleSize property

(IPCB_Pad interface)

Syntax

```
Property HoleSize : TCoord Read GetState_HoleSize Write SetState_HoleSize;
```

Description

This property represents the hole size of a pad object where component pins or wires can be passed through and soldered in place.

This property is supported by the GetState_HoleSize and SetState_HoleSize methods.

Example

See also

IPCB_Pad interface

Name property

(IPCB_Pad interface)

Syntax

```
Property Name : TPCBString Read GetState_Name Write SetState_Name;
```

Description

This Name property represents the designator of a pad object.

This method is supported by the GetState_Name and SetState_Name methods.

Example

See also

IPCB_Pad interface

TPCBString type

Rotation property

(IPCB_Pad interface)

Syntax

```
Property Rotation : TAngle Read GetState_Rotation Write SetState_Rotation;
```

Description

This Rotation property deals with the rotation of the pad object in degrees (of TAngle type 0 -360 degrees).

This property is supported by GetState_Rotation and SetState_Rotation methods.

Example

See also

IPCB_Pad interface

TAngle type

SwapID_Gate property

(IPCB_Pad interface)

Syntax

```
Property SwapID_Gate : TPCBString Read GetState_SwapID_Gate Write SetState_SwapID_Gate;
```

Description

Example

See also

IPCB_Pad interface

TPCBString type

SwapID_Pad property

(IPCB_Pad interface)

Syntax

```
Property SwapID_Pad : TPCBString Read GetState_SwapID_Pad Write SetState_SwapID_Pad;
```

Description

Example

See also

IPCB_Pad interface

TPCBString type

SwappedPadName property

(IPCB_Pad interface)

Syntax

```
Property SwappedPadName : TPCBString Read GetState_SwappedPadName Write
SetState_SwappedPadName;
```

Description

Example

See also

IPCB_Pad interface

TPCBString type

Width property

(IPCB_Pad interface)

Syntax

```
Property Width [L : TLayer] : TCoord Read GetState_WidthOnLayer;
```

Description

This read only property is supported by the GetState_WidthOnLayer method and is equivalent to the WidthOnLayer property.

Example

See also

IPCB_Pad interface

WidthOnLayer property

(IPCB_Pad interface)

Syntax

```
Property WidthOnLayer[L : TLayer] : TCoord Read GetState_WidthOnLayer;
```

Description

This property retrieves the width of the pad on the specified layer. This read only property is supported by the GetState_WidthOnLayer method and is equivalent to the Width property.

Example

See also

IPCB_Pad interface

IsConnectedToPlane property

(IPCB_Pad interface)

Syntax

```
Property IsConnectedToPlane[L : TLayer] : Boolean Read GetState_IsConnectedToPlane Write SetState_IsConnectedToPlane;
```

Description

This property determines whether the pad is connected to the specified plane (one of the power internal planes).

This property is supported by GetState_IsConnectedToPlane and SetState_IsConnectedToPlane methods.

Example

See also

IPCB_Pad interface

MidShape property

(IPCB_Pad interface)

Syntax

```
Property MidShape : TShape Read GetState_MidShape Write SetState_MidShape;
```

Description

This property determines the middle shape of the pad with a top-middle-bottom stack up. This property is supported by the GetState_MidShape and SetState_MidShape methods.

Example

See also

IPCB_Pad interface

TShape type

MidXSize property

(IPCB_Pad interface)

Syntax

```
Property MidXSize : TCoord Read GetState_MidXSize Write SetState_MidXSize;
```

Description

This property determines the middle shape of the pad with a top-middle-bottom stack up.

This property is supported by the GetState_MidXSize and SetState_MidXSize methods.

Example

See also

IPCB_Pad interface

MidYSize property

(IPCB_Pad interface)

Syntax

```
Property MidYSize : TCoord Read GetState_MidYSize Write SetState_MidYSize;
```

Description

This property determines the middle Y Size of the pad with a top-middle-bottom stack up.

This property is supported by the GetState_MidYSize and SetState_MidYSize methods.

Example

See also

IPCB_Pad interface

Mode property

(IPCB_Pad interface)

Syntax

```
Property Mode : TPadMode Read GetState_Mode Write SetState_Mode;
```

Description

The **Mode** property determines what type of pad it is - a simple pad, a pad with three Top, Middle and Bottom layer stack up or a pad with a complex stack up.

If Mode is Simple (ePadMode_Simple) then you only deal with X,Y locations and the TopXSize, TopYSize and TopShape properties.

If Mode is Top-Mid-Bottom stack (ePadMode_LocalStack) then you deal with X,Y Locations, Top.., Mid.. and Bot.. properties.

If Mode is Full Stack (ePadMode_ExternalStack) then you deal with XStackSizeOnLayer, YStackSizeOnLayer and StackShapeOnLayer properties.

This property is supported by GetState_mode and SetState_mode methods.

Example

```
PadObject := Board.GetObjectAtCursor(MkSet(ePadObject),
                                     AllLayers,
                                    'Choose a pad');
While PadObject <> 0 Do
Begin
    Ls := 'Pad Designator/Name: ' + PadObject.Name + #13#10;

    // work out the pad stack style
    If PadObject.Mode = ePadMode_Simple Then
        ProcessSimplePad   (PadObject,LS)
    Else If PadObject.Mode = ePadMode_LocalStack    Then
        ProcessTopMidBotPad(PadObject,LS)
    Else If PadObject.Mode = ePadMode_ExternalStack Then
        ProcessFullStackPad(PadObject,LS);

    // Display the results
    ShowInfo(LS);

    // Continue the loop ie user can click on another pad.
    PadObject := Board.GetObjectAtCursor(MkSet(ePadObject), AllLayers, 'Choose a pad');
End;
```

See also

IPCB_Pad interface

TPadMode type

IsPadStack method

PadStackInfo script from **Examples**\Scripts\Delphiscript Scripts\Pcb\ folder

Plated method

(IPCB_Pad interface)

Syntax

```
Property Plated : Boolean Read GetState_Plated Write SetState_Plated;
```

Description

This property denotes whether the pad is plated or not.

Example

See also

IPCB_Pad interface

PinDescriptor property

(IPCB_Pad interface)

Syntax

```
Property PinDescriptor : TPCBString Read GetState_PinDescriptorString;
```

Description

This property obtains the description of the pin which represents the pad of a component. This read only property is supported by the GetState_PinDescriptorString method.

Example

See also

IPCB_Pad interface

TPCBString type

ShapeOnLayer property

(IPCB_Pad interface)

Syntax

```
Property ShapeOnLayer[L : TLayer] : TShape Read GetState_ShapeOnLayer;
```

Description

This property determines what shape the pad is on this specified layer. This read only property is supported by the GetState_ShapeOnlayer method.

Example

See also

IPCB_Pad interface

TShape type

StackShapeOnLayer property

(IPCB_Pad interface)

Syntax

```
Property StackShapeOnLayer[L : TLayer] : TShape Read GetState_StackShapeOnLayer Write
SetState_StackShapeOnLayer;
```

Description

This property determines what shape the pad stack is on that layer. This property is supported by GetState_StackShapeOnLayer and SetState_StackShapeOnLayer methods.

Example

See also

IPCB_Pad interface

TShape type

TopShape property

(IPCB_Pad interface)

Syntax

```
Property TopShape : TShape Read GetState_TopShape Write SetState_TopShape;
```

Description

This property determines the top layer shape of the pad with a top-middle-bottom stack up.

This property is supported by the GetState_TopShape and SetState_TopShape methods.

Example

See also

IPCB_Pad interface

TShape type

TopXSize property

(IPCB_Pad interface)

Syntax

```
Property TopXSize : TCoord Read GetState_TopXSize Write SetState_TopXSize;
```

Description

This property determines the Top layer X Size of the pad with a top-middle-bottom stack up.

This property is supported by the GetState_TopXSize and SetState_TopXSize methods.

Example

See also

IPCB_Pad interface

TopYSize property

(IPCB_Pad interface)

Syntax

```
Property TopYSize : TCoord Read GetState_TopYSize Write SetState_TopYSize;
```

Description

This property determines the Top layer Y Size of the pad with a top-middle-bottom stack up.

This property is supported by the GetState_TopYSize and SetState_TopYSize methods.

Example

See also

IPCB_Pad interface

X property

(IPCB_Pad interface)

Syntax

```
Property X : TCoord Read GetState_XLocation Write SetState_XLocation;
```

Description

The Properties X and Y set the location of the pad with respect to the PCB document it is on.

These properties are supported by GetState_XLocation, GetState_YLocation and SetState_XLocation, SetState_YLocation methods.

Example

See also

IPCB_Pad interface

XSizeOnLayer property

(IPCB_Pad interface)

Syntax

```
Property XSizeOnLayer[L : TLayer] : TCoord Read GetState_XSizeOnLayer;
```

Description

This property determines what size in X direction the pad is on this specified layer. This read only property is supported by the GetState_XSizeOnlayer method.

Example

See also

IPCB_Pad interface

XStackSizeOnLayer property

(IPCB_Pad interface)

Syntax

```
Property XStackSizeOnLayer[L : TLayer] : TCoord Read GetState_XStackSizeOnLayer Write
SetState_XStackSizeOnLayer;
```

Description

This XStackSizeOnLayer property determines the size of the pad in X direction on the specified layer only if the pad has an external stack (ePadMode_ExternalStack type). This property is supported by the GetState_XStackSizeOnLayer and SetState_XStackSizeOnLayer methods.

Example

See also

IPCB_Pad interface

TPadMode type

Y property

(IPCB_Pad interface)

Syntax

```
Property Y : TCoord Read GetState_YLocation Write SetState_YLocation;
```

Description

The Properties X and Y set the location of the pad with respect to the PCB document it is on.

These properties are supported by GetState_XLocation, GetState_YLocation and SetState_XLocation, SetState_YLocation methods.

Example

See also

IPCB_Pad interface

YSizeOnLayer property

(IPCB_Pad interface)

Syntax

```
Property YSizeOnLayer[L : TLayer] : TCoord Read GetState_YSizeOnLayer;
```

Description

This property determines what size in Y direction the pad is on this specified layer. This read only property is supported by the GetState_YSizeOnlayer method.

Example

See also

IPCB_Pad interface

YStackSizeOnLayer property

(IPCB_Pad interface)

Syntax

```
Property YStackSizeOnLayer[L : TLayer] : TCoord Read GetState_YStackSizeOnLayer Write
SetState_YStackSizeOnLayer;
```

Description

This YStackSizeOnLayer property determines the size of the pad in Y direction on the specified layer only if the pad has an external stack (ePadMode_ExternalStack type). This property is supported by the GetState_YStackSizeOnLayer and SetState_YStackSizeOnLayer methods.

Example

See also

IPCB_Pad interface

DrillType property

(IPCB_Pad interface)

Syntax

```
Property DrillType : TExtendedDrillType Read GetState_DrillType Write SetState_DrillType;
```

Description

Example

See also

IPCB_Pad interface

HoleType property

(IPCB_Pad interface)

Syntax

```
Property  HoleType : TExtendedHoleType  Read GetState_HoleType Write SetState_HoleType;
```

Description

Example

See also

IPCB_Pad interface

HoleWidth property

(IPCB_Pad interface)

Syntax

```
Property  HoleWidth : TCoord Read GetState_HoleWidth Write SetState_HoleWidth;
```

Description

Example

See also

IPCB_Pad interface

XPadOffset property

(IPCB_Pad interface)

Syntax

```
Property  XPadOffset[L : TLayer]          : TCoord             Read GetState_XPadOffsetOnLayer
Write SetState_XPadOffsetOnLayer;
```

Description

This property is not implemented.

Example

See also

IPCB_Pad interface

YPadOffset property

(IPCB_Pad interface)

Syntax

```
Property YPadOffset[L : TLayer] : TCoord Read GetState_YPadOffsetOnLayer Write
SetState_YPadOffsetOnLayer;
```

Description

This property is not implemented.

Example

See also

IPCB_Pad interface

HoleRotation property

(IPCB_Pad interface)

Syntax

```
Property  HoleRotation : TAngle Read GetState_HoleRotation Write SetState_HoleRotation;
```

Description

This property defines the rotation attribute of the hole within a pad object. This applies to square and slotted holes. This property is supported by the GetState_HoleRotation and SetState_HoleRotation methods.

Example

See also

IPCB_Pad interface

TAngle type

IPCB_Pad2 Interface

Overview

Pad objects are hole connectors for components and for connection to signal tracks. The IPCB_Pad2 interface represents the extra attributes such as the Corner radius attribute of rounded rectangular pads.

The IPCB_Pad2 interface hierarchy;

IPCB_Primitive

IPCB_Pad

IPCB_Pad2

IPCB_Pad2 methods	IPCB_Pad2 properties
GetState_CornerRadiusOnLayer	CornerRadius
GetState_CRPercentageOnLayer	CRPercentage
GetState_StackCRPctOnLayer	StackCRPctOnLayer
SetState_StackCRPctOnLayer	

Methods

GetState_CornerRadiusOnLayer method

(IPCB_Pad2 interface)

Syntax

```
Function  GetState_CornerRadiusOnLayer(L : TLayer) : TCoord;
```

Description

This function returns the corner radius of a rectangular pad on the specified layer in TCoord units. This function is used by the CornerRadiusOnLayer property.

Example

See also

IPCB_Pad2 interface

GetState_CRPercentageOnLayer method

(IPCB_Pad2 interface)

Syntax

```
Function  GetState_CRPercentageOnLayer(L : TLayer) : Byte;
```

Description

This function returns the percentage of the corner radius of a rectangular pad on the specified layer as a byte value (0-100).

The corner radius percentage of the rounded corners of a pad on the bottom layer and the radius percentage is per layer and is a percentage of half of the shortest side of a pad object. The value of 0% corresponds to a rectangular pad and 100% to a normal rounded pad shape. This value only applies when the Shape field is set to Rounded Rectangle.

Example

See also

IPCB_Pad2 interface

GetState_StackCRPctOnLayer method

(IPCB_Pad2 interface)

Syntax

```
Function GetState_StackCRPctOnLayer (L : TLayer) : Byte;
```

Description

This function returns the percentage of the corner radius of a stack up pad on the specified layer as a byte value (0-100).

The corner radius percentage of the rounded corners of a pad on the bottom layer and the radius percentage is per layer and is a percentage of half of the shortest side of a pad object. The value of 0% corresponds to a rectangular pad and 100% to a normal rounded pad shape. This value only applies when the Shape field is set to Rounded Rectangle.

This function is used by the StackCRPctOnLayer property.

Example

See also

IPCB_Pad2 interface

SetState_StackCRPctOnLayer method

(IPCB_Pad2 interface)

Syntax

```
Procedure SetState_StackCRPctOnLayer (L : TLayer; Value  : Byte);
```

Description

This function sets the percentage of the corner radius of a rectangular pad on the specified layer as a byte value (0-100).

The corner radius percentage of the rounded corners of a pad on the bottom layer and the radius percentage is per layer and is a percentage of half of the shortest side of a pad object. The value of 0% corresponds to a rectangular pad and 100% to a normal rounded pad shape. This value only applies when the Shape field is set to Rounded Rectangle.

Example

See also

IPCB_Pad2 interface

Properties

CornerRadius property

(IPCB_Pad2 interface)

Syntax

```
Property  CornerRadius     [L : TLayer]   : TCoord       Read GetState_CornerRadiusOnLayer;
```

Description

This property returns the corner radius of a rectangular pad on the specified layer in TCoord units. This property is implemented by the GetState_CornerRadiusOnLayer function.

Example

See also

IPCB_Pad2 interface

CRPercentage property

(IPCB_Pad2 interface)

Syntax

```
Property  CRPercentage     [L : TLayer] : Byte Read GetState_CRPercentageOnLayer;
```

Description

This function returns the percentage of the corner radius of a rectangular pad on the specified layer as a byte value (0-100).

The corner radius percentage of the rounded corners of a pad on the bottom layer and the radius percentage is per layer and is a percentage of half of the shortest side of a pad object. The value of 0% corresponds to a rectangular pad and 100% to a normal rounded pad shape. This value only applies when the Shape field is set to Rounded Rectangle.

The Property uses GetState_CRPercentageOnLayer function.

Example

See also

IPCB_Pad2 interface

StackCRPctOnLayer property

(IPCB_Pad2 interface)

Syntax

```
Property  StackCRPctOnLayer[L : TLayer]   : Byte         Read GetState_StackCRPctOnLayer
Write SetState_StackCRPctOnLayer;
```

Description

This property returns the percentage of the corner radius of a stack up pad on the specified layer as a byte value (0-100).

The corner radius percentage of the rounded corners of a pad on the bottom layer and the radius percentage is per layer and is a percentage of half of the shortest side of a pad object. The value of 0% corresponds to a rectangular pad and 100% to a normal rounded pad shape. This value only applies when the Shape field is set to Rounded Rectangle.

This property uses `GetState_StackCRPctOnLayer` and `SetState_StackCRPctOnLayer` methods.

Example

See also

IPCB_Pad2 interface

IPCB_Polygon Interface

Overview

Polygons are similar to area fills, except that they can fill irregular shaped areas of a board and can connect to a specified net as they are poured. By adjusting the grid and track size, a polygon plane can be either solid (copper) areas or a cross hatched lattice. Polygons can be poured on any layer, however if a polygon is placed on a non signal layer, it will not be poured around existing objects.

Polygons are group objects, therefore they have child objects such as tracks and arcs. You can use the **IPCB_GroupIterator** interface with the **GroupIterator_Create** and **GroupIterator_Destroy** methods from the **IPCB_Polygon** to fetch child objects.

The IPCB_Polygon interface hierarchy;

IPCB_Primitive
IPCB_Group
IPCB_Polygon

IPCB_Group methods

`FreePrimitives`
`GetPrimitiveAt`
`GetPrimitiveCount`
`SetState_XSizeYSize`
`FastSetState_XSizeYSize`
`SetState_LayersUsedArray`
`GroupIterator_Create`
`GroupIterator_Destroy`
`AddPCBObject`
`RemovePCBObject`

IPCB_Group properties

`X`
`Y`
`PrimitiveLock`
`LayerUsed`

The **IPCB_Polygon** interface hierarchy is as follows;

IPCB_Polygon methods

GetState_AreaSize
GetState_PolygonType
GetState_RemoveDead

IPCB_Polygon properties

AreaSize
PolygonType
RemoveDead

GetState_UseOctagons
GetState_AvoidObsticles
GetState_PourOver
GetState_Grid
GetState_TrackSize
GetState_MinTrack
GetState_PointCount
GetState_Segments
GetState_PolyHatchStyle
GetState_BorderWidth
GetState_ExpandOutline
GetState_RemoveIslandsByArea
GetState_IslandAreaThreshold
GetState_RemoveNarrowNecks
GetState_NeckWidthThreshold
GetState_ClipAcuteCorners
GetState_MitreCorners
GetState_DrawRemovedNecks
GetState_DrawRemovedIslands
GetState_DrawDeadCopper
GetState_ArcApproximation
SetState_AreaSize
SetState_PolygonType
SetState_RemoveDead
SetState_UseOctagons
SetState_AvoidObsticles
SetState_PourOver
SetState_Grid
SetState_TrackSize
SetState_MinTrack
SetState_PointCount
SetState_Segments
SetState_PolyHatchStyle
SetState_BorderWidth
SetState_ExpandOutline
SetState_RemoveIslandsByArea
SetState_IslandAreaThreshold
SetState_RemoveNarrowNecks
SetState_NeckWidthThreshold
SetState_ClipAcuteCorners
SetState_MitreCorners
SetState_DrawRemovedNecks
SetState_DrawRemovedIslands
SetState_DrawDeadCopper
SetState_ArcApproximation
GetState_HitPrimitive

UseOctagons
AvoidObsticles
PourOver
Grid
TrackSize
MinTrack
PointCount
Segments [I
PolyHatchStyle
BorderWidth
ExpandOutline
RemoveIslandsByArea
IslandAreaThreshold
RemoveNarrowNecks
NeckWidthThreshold
ClipAcuteCorners
MitreCorners
DrawRemovedNecks
DrawRemovedIslands
DrawDeadCopper
ArcApproximation

PrimitiveInsidePoly
Rebuild
SetState_XSizeYSize
SetState_CopperPourInvalid
SetState_CopperPourValid
GetState_CopperPourInvalid
GetState_InRepour
CopperPourValidate
AcceptsLayer
PointInPolygon
xBoundingRectangle
GetState_StrictHitTest
GrowPolyshape
RotateAroundXY

Notes

Polygons can be on internal planes. For example if there are multi layer pads on a PCB document, then all the internal planes are connected to these multi-layer pads as split planes and are called split plane polygons. Check the **PolygonType** property.

The grid property denotes the grid which the tracks within a polygon are placed. Ideally this grid is a fraction of the component pin pitch, to allow the most effective placement of the polygon tracks.

The segments property denotes the array of segments used to construct a polygon. Each segment consists of a record consisting of one group of points in X, Y coordinates as a line (**ePolySegmentline** type) or an arc, a radius and two angles (**ePolySegmentArc** type). Each segment record has a **Kind** field which denotes the type of segment it is.

A segment of a polygon either as an arc or a track is encapsulated as a **TPolySegment** record as shown below;

```
  TPolySegment = Record
      Kind      : TPolySegmentType;

      {Vertex}
      vx,vy     : TCoord;

      {Arc}
      cx,cy     : TCoord;
      Radius    : TCoord;
      Angle1    : TAngle;
      Angle2    : TAngle;
  End;
```

Example

```
Procedure IteratePolygons;
Var
    Board      : IPCB_Board;
    Polygon    : IPCB_Polygon;
    Iterator   : IPCB_BoardIterator;
    PolygonRpt : TStringList;
    FileName   : TPCBString;
    Document   : IServerDocument;
    PolyNo     : Integer;
    I          : Integer;
```

```
Begin
    // Retrieve the current board
    Board := PCBServer.GetCurrentPCBBoard;
    If Board = Nil Then Exit;

    // Search for Polygons and for each polygon found
    // get its attributes and put them in a TStringList object
    // to be saved as a text file.
    Iterator        := Board.BoardIterator_Create;
    Iterator.AddFilter_ObjectSet(MkSet(ePolyObject));
    Iterator.AddFilter_LayerSet(AllLayers);
    Iterator.AddFilter_Method(eProcessAll);

    PolyNo     := 0;
    PolygonRpt := TStringList.Create;

    Polygon := Iterator.FirstPCBObject;
    While (Polygon <> Nil) Do
    Begin
        Inc(PolyNo);
        PolygonRpt.Add('Polygon No : '           + IntToStr(PolyNo));
        //Check if Net exists before getting the Name property.
        If Polygon.Net <> Nil Then
            PolygonRpt.Add(' Polygon Net : '     + Polygon.Net.Name);

        If Polygon.PolygonType = eSignalLayerPolygon Then
            PolygonRpt.Add(' Polygon type : '        + 'Polygon on Signal Layer')
        Else
            PolygonRpt.Add(' Polygon type : '        + 'Split plane polygon')

        PolygonRpt.Add(' Polygon BorderWidth : ' + FloatToStr(Polygon.BorderWidth));
        PolygonRpt.Add('  Area size : '          + FloatToStr(Polygon.AreaSize));

        // Segments of a polygon
        For I := 0 To Polygon.PointCount - 1 Do
        Begin
            If Polygon.Segments[I].Kind = ePolySegmentLine Then
            Begin
                PolygonRpt.Add(' Polygon Segment Line at X: ' +
IntToStr(Polygon.Segments[I].vx));
                PolygonRpt.Add(' Polygon Segment Line at Y: ' +
IntToStr(Polygon.Segments[I].vy));
            End
            Else
            Begin
                PolygonRpt.Add(' Polygon Segment Arc 1  : ' +
FloatToStr(Polygon.Segments[I].Angle1));
                PolygonRpt.Add(' Polygon Segment Arc 2  : ' +
FloatToStr(Polygon.Segments[I].Angle2));
```

```
                PolygonRpt.Add(' Polygon Segment Radius : ' +
FloatToStr(Polygon.Segments[I].Radius));
            End;
        End;
        PolygonRpt.Add('');
        Polygon := Iterator.NextPCBObject;
    End;
    Board.BoardIterator_Destroy(Iterator);

    // The TStringList contains Polygon data and is saved as
    // a text file.
    FileName := ChangeFileExt(Board.FileName,'.pol');
    PolygonRpt.SaveToFile(Filename);
    PolygonRpt.Free;

    // Display the Polygons report
    Document  := Client.OpenDocument('Text', FileName);
    If Document <> Nil Then
        Client.ShowDocument(Document);
End;
```

See also

PCB Design Objects

IPCB_Primitive interface

IPCB_Group interface

IPCB_GroupIterator interface

TPolygonType enumerated values

TPolySegment enumerated values

TPolyHatchStyle enumerated values

IteratePolygons example from the **\Examples\Scripts\DelphiScript\PCB** folder.

OutlinePerimeter example from the **\Examples\Scripts\DelphiScript\PCB** folder.

Methods

AcceptsLayer method

(IPCB_Polygon interface)

Syntax

```
Function AcceptsLayer (Layer : TLayer) : Boolean;
```

Description

Example

See also

IPCB_Polygon interface

CopperPourValidate method

(IPCB_Polygon interface)

Syntax

```
Procedure CopperPourValidate;
```

Description

Example

See also

IPCB_Polygon interface

GetState_CopperPourInvalid method

(IPCB_Polygon interface)

Syntax

```
Function GetState_CopperPourInvalid : Boolean;
```

Description

Example

See also

IPCB_Polygon interface

GetState_StrictHitTest method

(IPCB_Polygon interface)

Syntax

```
Function GetState_StrictHitTest (HitX,HitY : TCoord) : Boolean;
```

Description

Example

See also

IPCB_Polygon interface

GrowPolyshape method

(IPCB_Polygon interface)

Syntax

```
Procedure GrowPolyshape (ADist : TCoord);
```

Description

Example

See also

IPCB_Polygon interface

PointInPolygon method

(IPCB_Polygon interface)

Syntax

```
Function PointInPolygon (HitX,HitY : TCoord) : Boolean;
```

Description

Example

See also

IPCB_Polygon interface

PrimitiveInsidePoly method

(IPCB_Polygon interface)

Syntax

```
Function PrimitiveInsidePoly (APrimitive : IPCB_Primitive) : Boolean;
```

Description

This function determines whether a primitive is indeed part of a polygon or not.

Example

See also

IPCB_Polygon interface

Rebuild method

(IPCB_Polygon interface)

Syntax

```
Procedure Rebuild;
```

Description

This procedure forces a rebuild of the polygon especially after it has been poured.

Example

See also

IPCB_Polygon interface

SetState_CopperPourInvalid method

(IPCB_Polygon interface)

Syntax

```
Procedure SetState_CopperPourInvalid;
```

Description

Example

See also

IPCB_Polygon interface

SetState_CopperPourValid method

(IPCB_Polygon interface)

Syntax

```
Procedure SetState_CopperPourValid;
```

Description

Example

See also

IPCB_Polygon interface

SetState_XSizeYSize method

(IPCB_Polygon interface)

Syntax

```
Function SetState_XSizeYSize : Boolean;
```

Description

This method sets the X and Y size of the polygon.

Example

See also

IPCB_Polygon interface

xBoundingRectangle method

(IPCB_Polygon interface)

Syntax

```
Function xBoundingRectangle : TCoordRect;
```

Description

This function obtains the bounding rectangle of the polygon in TCoordRect.

Example

See also

IPCB_Polygon interface

TCoordRect

RotateAroundXY method

(IPCB_Polygon interface)

Syntax

```
Procedure RotateAroundXY(AX,
                         AY          : TCoord;
                         Angle       : TAngle);
```

Description

This function rotates the polygon about its reference point by an angle.

Example

See also

IPCB_Polygon interface

TCoord type

TAngle type

GetState and SetState Methods

GetState_ArcApproximation method

(IPCB_Polygon interface)

Syntax

```
Function GetState_ArcApproximation : TCoord ;
```

Description

Example

See also

IPCB_Polygon interface

GetState_AreaSize method

(IPCB_Polygon interface)

Syntax

```
Function GetState_AreaSize : Extended;
```

Description

Example

See also

IPCB_Polygon interface

GetState_AvoidObsticles method

(IPCB_Polygon interface)

Syntax

```
Function GetState_AvoidObsticles : Boolean;
```

Description

Example

See also

IPCB_Polygon interface

GetState_BorderWidth method

(IPCB_Polygon interface)

Syntax

```
Function GetState_BorderWidth : TCoord;
```

Description

Example

See also

IPCB_Polygon interface

GetState_ClipAcuteCorners method

(IPCB_Polygon interface)

Syntax

```
Function GetState_ClipAcuteCorners : Boolean ;
```

Description

Example

See also

IPCB_Polygon interface

GetState_DrawDeadCopper method

(IPCB_Polygon interface)

Syntax

```
Function GetState_DrawDeadCopper : Boolean ;
```

Description

Example

See also

IPCB_Polygon interface

GetState_DrawRemovedIslands method

(IPCB_Polygon interface)

Syntax

```
Function GetState_DrawRemovedIslands : Boolean ;
```

Description

Example

See also

IPCB_Polygon interface

GetState_DrawRemovedNecks method

(IPCB_Polygon interface)

Syntax

```
Function GetState_DrawRemovedNecks : Boolean ;
```

Description

Example

See also

IPCB_Polygon interface

GetState_ExpandOutline method

(IPCB_Polygon interface)

Syntax

```
Function GetState_ExpandOutline : Boolean ;
```

Description

Example

See also

IPCB_Polygon interface

GetState_Grid method

(IPCB_Polygon interface)

Syntax

```
Function GetState_Grid : TCoord;
```

Description

Example

See also

IPCB_Polygon interface

GetState_HitPrimitive method

(IPCB_Polygon interface)

Syntax

```
Function GetState_HitPrimitive (APrimitive : IPCB_Primitive) : Boolean;
```

Description

Example

See also

IPCB_Polygon interface

GetState_InRepour method

(IPCB_Polygon interface)

Syntax

```
Function GetState_InRepour : Boolean;
```

Description

Example

See also

IPCB_Polygon interface

GetState_IslandAreaThreshold method

(IPCB_Polygon interface)

Syntax

```
Function GetState_IslandAreaThreshold : Extended ;
```

Description

Example

See also

IPCB_Polygon interface

GetState_MinTrack method

(IPCB_Polygon interface)

Syntax

```
Function GetState_MinTrack : TCoord;
```

Description

Example

See also

IPCB_Polygon interface

GetState_MitreCorners method

(IPCB_Polygon interface)

Syntax

```
Function GetState_MitreCorners : Boolean ;
```

Description

Example

See also

IPCB_Polygon interface

GetState_NeckWidthThreshold method

(IPCB_Polygon interface)

Syntax

```
Function GetState_NeckWidthThreshold : TCoord ;
```

Description

Example

See also

IPCB_Polygon interface

GetState_PointCount method

(IPCB_Polygon interface)

Syntax

```
Function GetState_PointCount : Integer;
```

Description

Example

See also

IPCB_Polygon interface

GetState_PolygonType method

(IPCB_Polygon interface)

Syntax

```
Function GetState_PolygonType : TPolygonType;
```

Description

Example

See also

IPCB_Polygon interface

GetState_PolyHatchStyle method

(IPCB_Polygon interface)

Syntax

```
Function GetState_PolyHatchStyle : TPolyHatchStyle;
```

Description

Example

See also

IPCB_Polygon interface

GetState_PourOver method

(IPCB_Polygon interface)

Syntax

```
Function GetState_PourOver : TPolygonPourOver;
```

Description

Example

See also

IPCB_Polygon interface

GetState_RemoveDead method

(IPCB_Polygon interface)

Syntax

```
Function GetState_RemoveDead : Boolean;
```

Description

Example

See also

IPCB_Polygon interface

GetState_RemoveIslandsByArea method

(IPCB_Polygon interface)

Syntax

```
Function GetState_RemoveIslandsByArea : Boolean ;
```

Description

Example

See also

IPCB_Polygon interface

GetState_RemoveNarrowNecks method

(IPCB_Polygon interface)

Syntax

```
Function GetState_RemoveNarrowNecks : Boolean ;
```

Description

Example

See also

IPCB_Polygon interface

GetState_Segments method

(IPCB_Polygon interface)

Syntax

```
Function GetState_Segments (I : Integer) : TPolySegment;
```

Description

Example

See also

IPCB_Polygon interface

GetState_TrackSize method

(IPCB_Polygon interface)

Syntax

```
Function GetState_TrackSize : TCoord;
```

Description

Example

See also

IPCB_Polygon interface

GetState_UseOctagons method

(IPCB_Polygon interface)

Syntax

```
Function GetState_UseOctagons : Boolean;
```

Description

Example

See also

IPCB_Polygon interface

SetState_ArcApproximation method

(IPCB_Polygon interface)

Syntax

```
Procedure SetState_ArcApproximation (Value : TCoord );
```

Description

Example

See also

IPCB_Polygon interface

SetState_AreaSize method

(IPCB_Polygon interface)

Syntax

```
Procedure SetState_AreaSize (Value : Extended);
```

Description

Example

See also

IPCB_Polygon interface

SetState_AvoidObsticles method

(IPCB_Polygon interface)

Syntax

```
Procedure SetState_AvoidObsticles (Value : Boolean);
```

Description

Example

See also

IPCB_Polygon interface

SetState_BorderWidth method

(IPCB_Polygon interface)

Syntax

```
Procedure SetState_BorderWidth (Value : TCoord);
```

Description

Example

See also

IPCB_Polygon interface

SetState_ClipAcuteCorners method

(IPCB_Polygon interface)

Syntax

```
Procedure SetState_ClipAcuteCorners (Value : Boolean );
```

Description

Example

See also

IPCB_Polygon interface

SetState_DrawDeadCopper method

(IPCB_Polygon interface)

Syntax

```
Procedure SetState_DrawDeadCopper (Value : Boolean );
```

Description

Example

See also

IPCB_Polygon interface

SetState_DrawRemovedIslands method

(IPCB_Polygon interface)

Syntax

```
Procedure SetState_DrawRemovedIslands (Value : Boolean );
```

Description

Example

See also

IPCB_Polygon interface

SetState_DrawRemovedNecks method

(IPCB_Polygon interface)

Syntax

```
Procedure SetState_DrawRemovedNecks (Value : Boolean );
```

Description

Example

See also

IPCB_Polygon interface

SetState_ExpandOutline method

(IPCB_Polygon interface)

Syntax

```
Procedure SetState_ExpandOutline (Value : Boolean );
```

Description

Example

See also

IPCB_Polygon interface

SetState_Grid method

(IPCB_Polygon interface)

Syntax

```
Procedure SetState_Grid (Value : TCoord);
```

Description

Example

See also

IPCB_Polygon interface

SetState_IslandAreaThreshold method

(IPCB_Polygon interface)

Syntax

```
Procedure SetState_IslandAreaThreshold (Value : Extended );
```

Description

Example

See also

IPCB_Polygon interface

SetState_MinTrack method

(IPCB_Polygon interface)

Syntax

```
Procedure SetState_MinTrack (Value : TCoord);
```

Description

Example

See also

IPCB_Polygon interface

SetState_MitreCorners method

(IPCB_Polygon interface)

Syntax

```
Procedure SetState_MitreCorners (Value : Boolean );
```

Description

Example

See also

IPCB_Polygon interface

SetState_NeckWidthThreshold method

(IPCB_Polygon interface)

Syntax

```
Procedure SetState_NeckWidthThreshold (Value : TCoord );
```

Description

Example

See also

IPCB_Polygon interface

SetState_PointCount method

(IPCB_Polygon interface)

Syntax

```
Procedure SetState_PointCount (Value : Integer);
```

Description

Example

See also

IPCB_Polygon interface

SetState_PolygonType method

(IPCB_Polygon interface)

Syntax

```
Procedure SetState_PolygonType (Value : TPolygonType);
```

Description

Example

See also

IPCB_Polygon interface

SetState_PolyHatchStyle method

(IPCB_Polygon interface)

Syntax

```
Procedure SetState_PolyHatchStyle (Value : TPolyHatchStyle);
```

Description

Example

See also

IPCB_Polygon interface

SetState_PourOver method

(IPCB_Polygon interface)

Syntax

```
Procedure SetState_PourOver (Value : TPolygonPourOver);
```

Description

Example

See also

IPCB_Polygon interface

SetState_RemoveDead method

(IPCB_Polygon interface)

Syntax

```
Procedure SetState_RemoveDead (Value : Boolean);
```

Description

Example

See also

IPCB_Polygon interface

SetState_RemoveIslandsByArea method

(IPCB_Polygon interface)

Syntax

```
Procedure SetState_RemoveIslandsByArea (Value : Boolean );
```

Description

Example

See also

IPCB_Polygon interface

SetState_RemoveNarrowNecks method

(IPCB_Polygon interface)

Syntax

```
Procedure SetState_RemoveNarrowNecks (Value : Boolean );
```

Description

Example

See also

IPCB_Polygon interface

SetState_Segments method

(IPCB_Polygon interface)

Syntax

```
Procedure SetState_Segments (I : Integer;Value : TPolySegment);
```

Description

Example

See also

IPCB_Polygon interface

SetState_TrackSize method

(IPCB_Polygon interface)

Syntax

```
Procedure SetState_TrackSize (Value : TCoord);
```

Description

Example

See also

IPCB_Polygon interface

SetState_UseOctagons method

(IPCB_Polygon interface)

Syntax

```
Procedure SetState_UseOctagons (Value : Boolean);
```

Description

Example

See also

IPCB_Polygon interface

Properties

ArcApproximation property

(IPCB_Polygon interface)

Syntax

```
Property ArcApproximation : TCoord Read GetState_ArcApproximation Write
SetState_ArcApproximation ;
```

Description

The polygon drawn around a pad or via is drawn by line segments. The arc resolution value dictates how accurate the polygon is drawn around a pad for example. The segments are drawn between a system defined outer circle and inner circle with a radial distance between these two circles being equal to the arc resolution.

The default value is 0.5mil. The lower the value the more smooth the arc is and the higher the value, the more coarse the arc is with longer line segments.

Example

See also

IPCB_Polygon interface

AreaSize property

(IPCB_Polygon interface)

Syntax

```
Property AreaSize : Extended Read GetState_AreaSize Write SetState_AreaSize;
```

Description

The AreaSize property returns the size of the polygon in Extended type. The GetState_AreaSize and SetState_AreaSize are methods for this property.

Example

See also

IPCB_Polygon interface

AvoidObsticles property

(IPCB_Polygon interface)

Syntax

```
Property  AvoidObsticles : Boolean  Read GetState_AvoidObsticles Write
SetState_AvoidObsticles;
```

Description

Example

See also

IPCB_Polygon interface

BorderWidth property

(IPCB_Polygon interface)

Syntax

```
Property BorderWidth : TCoord Read GetState_BorderWidth Write SetState_BorderWidth;
```

Description

Example

See also

IPCB_Polygon interface

ClipAcuteCorners property

(IPCB_Polygon interface)

Syntax

```
Property ClipAcuteCorners : Boolean Read GetState_ClipAcuteCorners Write
SetState_ClipAcuteCorners ;
```

Description

Example

See also

IPCB_Polygon interface

DrawDeadCopper property

(IPCB_Polygon interface)

Syntax

```
Property DrawDeadCopper : Boolean Read GetState_DrawDeadCopper Write SetState_DrawDeadCopper ;
```

Description

Example

See also

IPCB_Polygon interface

DrawRemovedIslands property

(IPCB_Polygon interface)

Syntax

```
Property DrawRemovedIslands : Boolean Read GetState_DrawRemovedIslands Write
SetState_DrawRemovedIslands ;
```

Description

If this property is true, every time a polygon is created on a PCB document, islands are often created and those islands that are less than the quoted area threshold are not created, otherwise if false, islands are left drawn nonetheless.

Example

See also

IPCB_Polygon interface

DrawRemovedNecks property

(IPCB_Polygon interface)

Syntax

```
Property DrawRemovedNecks : Boolean Read GetState_DrawRemovedNecks Write
SetState_DrawRemovedNecks ;
```

Description

Example

See also

IPCB_Polygon interface

ExpandOutline property

(IPCB_Polygon interface)

Syntax

```
Property ExpandOutline : Boolean Read GetState_ExpandOutline Write SetState_ExpandOutline ;
```

Description

Example

See also

IPCB_Polygon interface

Grid property

(IPCB_Polygon interface)

Syntax

```
Property Grid : TCoord Read GetState_Grid Write SetState_Grid;
```

Description

The Grid property denotes the grid which the tracks within a polygon are placed. Ideally this grid is a fraction of the component pin pitch, to allow the most effective placement of the polygon tracks.

This property is supported by GetState_Grid and SetState_Grid methods.

Example

See also

IPCB_Polygon interface

IslandAreaThreshold property

(IPCB_Polygon interface)

Syntax

```
Property IslandAreaThreshold : Extended Read GetState_IslandAreaThreshold Write
SetState_IslandAreaThreshold;
```

Description

Every time a polygon is created on a PCB document, islands are often created and those islands that are less than the quoted area threshold, these islands are not created.

This property represents a value in mils squared that defines the area of an island and the default value is 2500 mils sq.

Example

See also

IPCB_Polygon interface

MinTrack property

(IPCB_Polygon interface)

Syntax

```
Property MinTrack : TCoord Read GetState_MinTrack Write SetState_MinTrack;
```

Description

Example

See also

IPCB_Polygon interface

MitreCorners property

(IPCB_Polygon interface)

Syntax

```
Property MitreCorners : Boolean Read GetState_MitreCorners Write SetState_MitreCorners ;
```

Description

Example

See also

IPCB_Polygon interface

NeckWidthThreshold property

(IPCB_Polygon interface)

Syntax

```
Property NeckWidthThreshold : TCoord Read GetState_NeckWidthThreshold Write
SetState_NeckWidthThreshold ;
```

Description

The minimum width threshold value for the regions of a polygon. Narrow regions that violate this under width value will be removed by the system. The default value is 5 mils.

Example

See also

IPCB_Polygon interface

PointCount property

(IPCB_Polygon interface)

Syntax

```
Property PointCount : Integer Read GetState_PointCount Write SetState_PointCount;
```

Description

Example

See also

IPCB_Polygon interface

PolygonType property

(IPCB_Board interface)

Syntax

```
Property PolygonType : TPolygonType Read GetState_PolygonType Write SetState_PolygonType;
```

Description

The PolygonType property defines what type the polygon is, whether it is a polygon on a signal layer, or a split plane polygon.

Example

See also

IPCB_Polygon interface

TPolygonType type

PolyHatchStyle property

(IPCB_Polygon interface)

Syntax

```
Property PolyHatchStyle : TPolyHatchStyle Read GetState_PolyHatchStyle Write
SetState_PolyHatchStyle;
```

Description

The property denotes the style of polygon hatching. If the hatching style (**ePolySolid**) is solid, then a region object is used instead.

```
ePolyHatch90, ePolyHatch45, ePolyVHatch, ePolyHHatch,
```

ePolyNoHatch type : the polygon is not filled at all. Only the boundary tracks will be present. You may wish to use this option if you want to place a polygon during the design phase, but do not want it to slow system performance. The polygon can be before re-poured with the desired hatching before generating output.

ePolySolid type: the polygon is filled in solid. You may wish to use this option if you want to place a solid polygon during the design phase. There are further Solid Fill Options to define and control how a solid polygon is drawn on the PCB document.

Example

See also

IPCB_Polygon interface

TPolyHatchStyle type

IPCB_Region interface

PourOver property

(IPCB_Polygon interface)

Syntax

```
Property PourOver : Boolean Read GetState_PourOver Write SetState_PourOver;
```

Description

The pourover property if true will indicate that any existing tracks and arcs within the polygon which are part of the net being connected to will be covered by the polygon.

If this property is false, the polygon will pour around existing tracks on the same net.

Example

See also

IPCB_Polygon interface

RemoveDead property

(IPCB_Polygon interface)

Syntax

```
Property RemoveDead : Boolean Read GetState_RemoveDead Write SetState_RemoveDead;
```

Description

If the RemoveDead property is enabled, any regions of "dead" copper within the polygon will be removed. Dead copper is created when an area of the polygon can not be connected to the selected net. You can view dead copper as unconnected "islands" of copper within the polygon created when existing tracks, pads and vias prevent the plane pouring as one continuous area.

If this property is disabled, any areas of dead copper will not be removed.

Note: The entire polygon is removed if it does not enclose any pads on the selected net, as it is all viewed as dead copper.

Example

See also

IPCB_Polygon interface

RemoveIslandsByArea property

(IPCB_Polygon interface)

Syntax

```
Property RemoveIslandsByArea : Boolean Read GetState_RemoveIslandsByArea Write
SetState_RemoveIslandsByArea;
```

Description

Example

See also

IPCB_Polygon interface

RemoveNarrowNecks property

(IPCB_Polygon interface)

Syntax

```
Property RemoveNarrowNecks : Boolean Read GetState_RemoveNarrowNecks Write
SetState_RemoveNarrowNecks ;
```

Description

If this property is true, thin sections (composing of tracks and arcs for example) are removed from this polygon on the PCB document that violate the minimum width threshold value. If false, narrow necks are left alone.

Example

See also

IPCB_Polygon interface

Segments [I property

(IPCB_Polygon interface)

Syntax

```
Property Segments [I : Integer] : TPolySegment Read GetState_Segments Write SetState_Segments;
```

Description

Example

See also

IPCB_Polygon interface

TrackSize property

(IPCB_Polygon interface)

Syntax

```
Property TrackSize : TCoord Read GetState_TrackSize Write SetState_TrackSize;
```

Description

Example

See also

IPCB_Polygon interface

UseOctagons property

(IPCB_Polygon interface)

Syntax

```
Property UseOctagons : Boolean Read GetState_UseOctagons Write SetState_UseOctagons;
```

Description

The **UseOctagons** property determines that octagons are to surround pads if true. If false, pads are surrounded by arcs. Octagons give smaller Gerber files and faster photoplotting.

This property is supported by GetState_UseOctagons and SetState_UseOctagons methods.

Example

See also

IPCB_Polygon interface

IPCB_RectangularPrimitive Interface

Overview

The **IPCB_RectangularPrimitive** interface is the ancestor interface for **IPCB_Fill** and **IPCB_Text** interfaces and contains the rectangular coordinates as well as the rotation property.

The **IPCB_RectangularPrimitive** interface hierarchy is as follows;

IPCB_Primitive

IPCB_RectangularPrimitive

IPCB_RectangularPrimitive methods	IPCB_RectangularPrimitive properties
GetState_XLocation	XLocation
GetState_YLocation	YLocation
GetState_X1Location	X1Location
GetState_Y1Location	Y1Location
GetState_X2Location	X2Location
GetState_Y2Location	Y2Location
GetState_Rotation	Rotation
SetState_XLocation	
SetState_YLocation	
SetState_X1Location	
SetState_Y1Location	
SetState_X2Location	
SetState_Y2Location	
SetState_Rotation	

RotateAroundXY

IsRedundant

SetState_XSizeYSize

See also

IPCB_Primitive interface

GetState and SetState Methods

SetState_Rotation method

(IPCB_RectangularPrimitive interface)

Syntax

```
Procedure SetState_Rotation (Rotation : TAngle);
```

Description

This SetState_Rotation method deals with the rotation of the rectangular primitive (fill, text, embedded board for example) object in degrees (of TAngle type 0 -360 degrees).

This method is used for the Rotation property.

Example

See also

IPCB_RectangularPrimitive interface

SetState_X1Location method

(IPCB_RectangularPrimitive interface)

Syntax

```
Procedure SetState_X1Location (AX1 : TCoord);
```

Description

The SetState_X1Location method sets the initial X1 location of the rectangular primitive. The X1,Y1 and X2,Y2 coordinates define the boundary of the rectangular primitive.

This method is used for the X1Location property.

Example

See also

IPCB_RectangularPrimitive interface

SetState_X2Location method

(IPCB_RectangularPrimitive interface)

Syntax

```
Procedure SetState_X2Location (AX2 : TCoord);
```

Description

The SetState_X2Location method sets the final X2 location of the rectangular primitive. The X1,Y1 and X2,Y2 coordinates define the boundary of the rectangular primitive.

This method is used for the X2Location property.

Example

See also

IPCB_RectangularPrimitive interface

SetState_XLocation method

(IPCB_RectangularPrimitive interface)

Syntax

```
Procedure SetState_XLocation (AX : TCoord);
```

Description

This method sets the reference X location of the rectangular primitive. The X,Y coordinates define the reference point of the rectangular primitive.

This method is used for the XLocation property.

Example

See also

IPCB_RectangularPrimitive interface

SetState_XSizeYSize method

(IPCB_RectangularPrimitive interface)

Syntax

```
Function SetState_XSizeYSize : Boolean;
```

Description

This method sets the XSize and YSize of the rectangular primitive.

Example

See also

IPCB_RectangularPrimitive interface

SetState_Y1Location method

(IPCB_RectangularPrimitive interface)

Syntax

```
Procedure SetState_Y1Location (AY1 : TCoord);
```

Description

The SetState_Y1Location method sets the initial Y1 location of the rectangular primitive. The X1,Y1 and X2,Y2 coordinates define the boundary of the rectangular primitive.

This method is used for the Y1Location property.

Example

See also

IPCB_RectangularPrimitive interface

SetState_Y2Location method

(IPCB_RectangularPrimitive interface)

Syntax

```
Procedure SetState_Y2Location (AY2 : TCoord);
```

Description

The SetState_Y2Location method sets the initial Y2 location of the rectangular primitive. The X1,Y1 and X2,Y2 coordinates define the boundary of the rectangular primitive.

This method is used for the Y2Location property.

Example

See also

IPCB_RectangularPrimitive interface

SetState_YLocation method

(IPCB_RectangularPrimitive interface)

Syntax

```
Procedure SetState_YLocation (AY : TCoord);
```

Description

This method sets the reference Y location of the rectangular primitive. The X,Y coordinates define the reference point of the rectangular primitive.

This method is used for the YLocation property.

Example

See also

IPCB_RectangularPrimitive interface

GetState_Rotation method

(IPCB_RectangularPrimitive interface)

Syntax

```
Function GetState_Rotation : TAngle;
```

Description

This GetState_Rotation method deals with the rotation of the rectangular primitive (fill, text, embedded board for example) object in degrees (of TAngle type 0 -360 degrees).

This method is used for the Rotation property.

Example

See also

IPCB_RectangularPrimitive interface

GetState_X1Location method

(IPCB_RectangularPrimitive interface)

Syntax

```
Function GetState_X1Location : TCoord;
```

Description

The GetState_X1Location method retrieves the initial X1 location of the rectangular primitive. The X1,Y1 and X2,Y2 coordinates define the boundary of the rectangular primitive.

This method is used for the X1Location property.

Example

See also

IPCB_RectangularPrimitive interface

GetState_X2Location method

(IPCB_RectangularPrimitive interface)

Syntax

```
Function GetState_X2Location : TCoord;
```

Description

The GetState_X1Location method retrieves the final X2 location of the rectangular primitive. The X1,Y1 and X2,Y2 coordinates define the boundary of the rectangular primitive.

This method is used for the X2Location property.

Example

See also

IPCB_RectangularPrimitive interface

GetState_XLocation method

(IPCB_RectangularPrimitive interface)

Syntax

```
Function GetState_XLocation : TCoord;
```

Description

This method obtains the reference X location of the rectangular primitive. The X,Y coordinates define the reference point of the rectangular primitive.

This method is used for the XLocation property.

Example

See also

IPCB_RectangularPrimitive interface

GetState_Y1Location method

(IPCB_RectangularPrimitive interface)

Syntax

```
Function GetState_Y1Location : TCoord;
```

Description

The GetState_Y1Location method retrieves the initial Y1 location of the rectangular primitive. The X1,Y1 and X2,Y2 coordinates define the boundary of the rectangular primitive.

This method is used for the Y1Location property.

Example

See also

IPCB_RectangularPrimitive interface

GetState_Y2Location method

(IPCB_RectangularPrimitive interface)

Syntax

```
Function GetState_Y2Location : TCoord;
```

Description

The GetState_Y2Location method retrieves the final Y2 location of the rectangular primitive. The X1,Y1 and X2,Y2 coordinates define the boundary of the rectangular primitive.

This method is used for the Y2Location property.

Example

See also

IPCB_RectangularPrimitive interface

GetState_YLocation method

(IPCB_RectangularPrimitive interface)

Syntax

```
Function GetState_YLocation : TCoord;
```

Description

This method obtains the reference Y location of the rectangular primitive. The X,Y coordinates define the reference point of the rectangular primitive.

This method is used for the YLocation property.

Example

See also

IPCB_RectangularPrimitive interface

Methods

IsRedundant method

(IPCB_RectangularPrimitive interface)

Syntax

```
Function IsRedundant : Boolean;
```

Description

This method determines whether the object is redundant (unused object) on the PCB document or not.

Example

See also

IPCB_RectangularPrimitive interface

RotateAroundXY method

(IPCB_RectangularPrimitive interface)

Syntax

```
Procedure RotateAroundXY (AX,AY : TCoord;Angle : TAngle);
```

Description

This method rotates a rectangular primitive object such as a fill or a text object on the PCB document about the AX, AY coordinates with an angle in degrees.

To ensure the rectangular primitive rotates without moving about, pass in its midpoint (between X1,X2 and Y1, Y2) attributes for the AX,AY parameters or use the Rotation property.

Example

See also

IPCB_RectangularPrimitive interface

Rotation property

(IPCB_RectangularPrimitive interface)

Syntax

```
Property Rotation : TAngle Read GetState_Rotation Write SetState_Rotation;
```

Description

This Rotation property deals with the rotation of the rectangular primitive (fill, text, embedded board for example) object in degrees (of TAngle type 0 -360 degrees).

This property is supported by GetState_Rotation and SetState_Rotation methods.

Example

See also

IPCB_RectangularPrimitive interface

Properties

X1Location property

(IPCB_RectangularPrimitive interface)

Syntax

```
Property X1Location : TCoord Read GetState_X1Location Write SetState_X1Location;
```

Description

The X1Location property determines the initial X1 location of the rectangular primitive. The X1,Y1 and X2,Y2 coordinates define the boundary of the rectangular primitive.

The property is supported by the GetState_X1Location and SetState_X1Location methods.

Example

See also

IPCB_RectangularPrimitive interface

X2Location property

(IPCB_RectangularPrimitive interface)

Syntax

```
Property X2Location : TCoord Read GetState_X2Location Write SetState_X2Location;
```

Description

The X2Location property determines the final X2 location of the rectangular primitive. The X1,Y1 and X2,Y2 coordinates define the boundary of the rectangular primitive.

The property is supported by the GetState_X2Location and SetState_X2Location methods.

Example

See also

IPCB_RectangularPrimitive interface

XLocation property

(IPCB_RectangularPrimitive interface)

Syntax

```
Property XLocation : TCoord Read GetState_XLocation Write SetState_XLocation;
```

Description

The XLocation property determines the reference X location of the rectangular primitive. The X,Y coordinates define the reference point of the rectangular primitive.

The property is supported by the GetState_XLocation and SetState_XLocation methods.

Example

See also

IPCB_RectangularPrimitive interface

Y1Location property

(IPCB_RectangularPrimitive interface)

Syntax

```
Property Y1Location : TCoord Read GetState_Y1Location Write SetState_Y1Location;
```

Description

The Y1Location property determines the initial Y1 location of the rectangular primitive. The X1,Y1 and X2,Y2 coordinates define the boundary of the rectangular primitive.

The property is supported by the GetState_Y1Location and SetState_Y1Location methods.

Example

See also

IPCB_RectangularPrimitive interface

Y2Location property

(IPCB_RectangularPrimitive interface)

Syntax

```
Property Y2Location : TCoord Read GetState_Y2Location Write SetState_Y2Location;
```

Description

The Y2Location property determines the final Y2 location of the rectangular primitive. The X1,Y1 and X2,Y2 coordinates define the boundary of the rectangular primitive.

The property is supported by the GetState_Y2Location and SetState_Y2Location methods.

Example

See also

IPCB_RectangularPrimitive interface

YLocation property

(IPCB_RectangularPrimitive interface)

Syntax

```
Property YLocation : TCoord Read GetState_YLocation Write SetState_YLocation;
```

Description

The YLocation property determines the reference Y location of the rectangular primitive. The X,Y coordinates define the reference point of the rectangular primitive.

The property is supported by the GetState_YLocation and SetState_YLocation methods.

Example

See also

IPCB_RectangularPrimitive interface

IPCB_Region Interface

Overview

The IPCB_Region interface represents a solid polygon pour as the region object. This region object allows the creation of multi sided polygon regions on the PCB. The region object can also be used to create polygonal shaped fills in PCB footprints.

Notes

You can use **IPCB_Primitive** methods and properties that are relevant to the **IPCB_Region** interface.

The **IPCB_Region** interface hierarchy is as follows;

IPCB_Primitive

IPCB_Region

IPCB_Region methods	IPCB_Region properties
GetState_Kind	Kind
SetState_Kind	Name
GetState_Name	RegionData
SetState_Name	MainContour
GetState_Area	HoleCount

GetRegionData	Holes
GetMainContour	Area
GetHoleCount	
GetHole	
SetOutlineContour	
SetRegionData	

See also

IPCB_Fill Interface

IPCB_Polygon interface

Methods

PCB method

(IPCB_Region interface)

Syntax

```
Procedure SetOutlineContour (Contour : Pgpc_vertex_list)
```

Description

Example

See also

IPCB_Region interface

GetState and SetState Methods

GetHole method

(IPCB_Region interface)

Syntax

```
Function GetHole (I : Integer) : Pgpc_vertex_list;
```

Description

Example

See also

IPCB_Region interface

GetHoleCount method

(IPCB_Region interface)

Syntax

```
Function GetHoleCount : Integer;
```

Description

Example

See also

IPCB_Region interface

GetMainContour method

(IPCB_Region interface)

Syntax

```
Function GetMainContour : Pgpc_vertex_list;
```

Description

Example

See also

IPCB_Region interface

GetRegionData method

(IPCB_Region interface)

Syntax

```
Function GetRegionData : Pgpc_polygon;
```

Description

Example

See also

IPCB_Region interface

GetState_Area method

(IPCB_Region interface)

Syntax

```
Function GetState_Area : Int64;
```

Description

Example

See also

IPCB_Region interface

GetState_Kind method

(IPCB_Region interface)

Syntax

```
Function GetState_Kind : TRegionKind;
```

Description

Example

See also

IPCB_Region interface

GetState_Name method

(IPCB_Region interface)

Syntax

```
Function GetState_Name : TDynamicString;
```

Description

Example

See also

IPCB_Region interface

SetState_Kind method

(IPCB_Region interface)

Syntax

```
Procedure SetState_Kind (Value : TRegionKind);
```

Description

Example

See also

IPCB_Region interface

SetState_Name method

(IPCB_Region interface)

Syntax

```
Procedure SetState_Name (Value : TDynamicString);
```

Description

Example

See also

IPCB_Region interface

Properties

Area property

(IPCB_Region interface)

Syntax

```
Property Area : Int64 Read GetState_Area;
```

Description

Example

See also

IPCB_Region interface

HoleCount property

(IPCB_Region interface)

Syntax

```
Property HoleCount : Integer Read GetHoleCount;
```

Description

Example

See also

IPCB_Region interface

Holes [I property

(IPCB_Region interface)

Syntax

```
Property Holes [I : Integer] : Pgpc_vertex_list Read GetHole;
```

Description

Example

See also

IPCB_Region interface

Kind property

(IPCB_Region interface)

Syntax

```
Property Kind : TRegionKind Read GetState_Kind Write SetState_Kind;
```

Description

Example

See also

IPCB_Region interface

TRegionKind type

MainContour property

(IPCB_Region interface)

Syntax

```
Property MainContour : Pgpc_vertex_list Read GetMainContour;
```

Description

Example

See also

IPCB_Region interface

Name property

(IPCB_Region interface)

Syntax

```
Property Name : TDynamicString Read GetState_Name Write SetState_Name;
```

Description

Example

See also

IPCB_Region interface

RegionData property

(IPCB_Region interface)

Syntax

```
Property RegionData : Pgpc_polygon Read GetRegionData;
```

Description

Example

See also

IPCB_Region interface

IPCB_Text Interface

Overview

Text strings can be placed on any layer with any height. There are two classes of text strings: Free text strings and component text (designators and comments). Free text strings are standalone strings which could be used as descriptors or labels for any application on the workspace. There are two component text objects- designator attribute and comment attribute. Each component must have a unique designator and thus designators are not globally editable. The comment attribute is globally editable though.

The PCB editor includes special strings which are interpreted when output (printing, plotting or generating gerber files) is generated. For example, the string .PRINT_DATE will be replaced by the current date when output is generated.

Notes

The IPCB_Text Interface hierarchy is as follows;

IPCB_Primitive

IPCB_RectangularPrimitive

IPCB_Text

1. Text objects are not inherited from the IPCB_group interface, therefore fetching child objects within a text object is not possible.
2. Text objects are rectangular primitives with rectangular coordinates properties and the rotation property.
3. Text objects can be converted into a series of strokes using the ConvertToStrokeArray method from the IPCB_Text interface.

IPCB_RectangularPrimitive methods	IPCB_RectangularPrimitive properties
	XLocation
	YLocation
RotateAroundXY	X1Location
IsRedundant	Y1Location
SetState_XSizeYSize	X2Location
	Y2Location
	Rotation

IPCB_Text methods	IPCB_Text properties
	Size
GetState_FontID	FontID
GetState_Text	Text
GetState_Width	Width

GetState_Mirror	MirrorFlag
GetState_UnderlyingString	UnderlyingString
GetState_ConvertedString	ConvertedString
GetState_UseTTFonts	
GetState_Bold	UseTTFonts
GetState_Italic	Bold
GetState_FontName	Italic
GetState_Inverted	FontName
GetState_InvertedTTTextBorder	Inverted
GetState_CharSet	InvertedTTTextBorder
	TTTextOutline
SetState_Size	CharSet
SetState_FontID	
SetState_Text	
SetState_Width	
SetState_Mirror	
SetState_UnderlyingString	
SetState_UseTTFonts	
SetState_Bold	
SetState_Italic	
SetState_FontName	
SetState_Inverted	
SetState_InvertedTTTextBorder	
SetState_CharSet	

IsHidden
IsDesignator
IsComment
InAutoDimension
GetDesignatorDisplayString
RotationHandle
ConvertToStrokeArray
GetTrueTypeTextOutline

Example

```
Var
    Board     : IPCB_Board;
    WorkSpace : IWorkSpace;
    TextObj   : IPCB_Text;
Begin
    //create a new pcb document
    WorkSpace := GetWorkSpace;
    If WorkSpace = Nil Then Exit;
    Workspace.DM_CreateNewDocument('PCB');

    Board := PCBServer.GetCurrentPCBBoard;
```

```
    If Board = Nil then exit;

    // Create a text object on a top overlay layer
    Board.LayerIsDisplayed[eTopOverLay] := True;
    TextObj := PCBServer.PCBObjectFactory(eTextObject, eNoDimension, eCreate_Default);
    TextObj.XLocation := MilsToCoord(Board.XOrigin + 4000);
    TextObj.YLocation := MilsToCoord(Board.YOrigin + 2000);
    TextObj.Layer     := eTopOverlay;
    TextObj.Text      := 'Text Object';
    TextObj.Size      := MilsToCoord(90);   // sets the height of the text.
    Board.AddPCBObject(TextObj);
End;
```

See also

PCB Design Objects

IPCB_Primitive interface

IPCB_RectangularPrimitive interface

GetState and SetState Methods

ConvertedString method

(IPCB_ConvertedString interface)

Syntax

```
Function  GetState_ConvertedString : TPCBString;
```

Description

This method is used for the ConvertedString property.

Example

See also

IPCB_Text interface

GetState_FontID method

(IPCB_Text interface)

Syntax

```
Function GetState_FontID : TFontID;
```

Description

This method retrieves the FontID attribute which represents the font used for this Text Object on a PCB document. This method is used for the FontID property.

Example

See also

IPCB_Text interface

TFontID type

GetState_Mirror method

(IPCB_Text interface)

Syntax

```
Function GetState_Mirror : Boolean;
```

Description

This method retrieves the Mirror attribute which represents the mirrored state of this Text Object on a PCB document. This method is used for the Mirror property.

Example

See also

IPCB_Text interface

GetState_Size method

(IPCB_Text interface)

Syntax

```
Function GetState_Size : TCoord;
```

Description

This method retrieves the Size attribute which represents the height of the text used for this Text Object on a PCB document. This method is used for the Size property.

Example

See also

IPCB_Text interface

GetState_Text method

(IPCB_Text interface)

Syntax

```
Function GetState_Text : TPCBString;
```

Description

This method retrieves the Text attribute which represents the text used for this Text Object on a PCB document. This method is used for the Text property.

Example

See also

IPCB_Text interface

GetState_UnderlyingString method

(IPCB_Text interface)

Syntax

```
Function GetState_UnderlyingString : TPCBString;
```

Description

This method retrieves the Text attribute which represents the text used for this Text Object on a PCB document and is equivalent to the GetState_Text method. This method is used for the UnderlyingString property.

Example

See also

IPCB_Text interface

GetState_Width method

(IPCB_Text interface)

Syntax

```
Function GetState_Width : TCoord;
```

Description

This method retrieves the Width attribute which represents the width used for this Text Object on a PCB document. This method is used for the Width property.

Example

See also

IPCB_Text interface

GetState_UseTTFonts method

(IPCB_Text interface)

Syntax

```
Function  GetState_UseTTFonts : Boolean;
```

Description

This property toggles the True Type font for the PCB string text on a PCB document. This property is supported by the GetState_UseTTFonts and SetState_UseTTFonts methods.

Once the UseTTFonts property is enabled, you can use the Bold, Italic, FontName, Inverted, InvertedTTTextBorder and TTTextOutline properties.

Example

See also

IPCB_Text interface

GetState_Bold method

(IPCB_Text interface)

Syntax

```
Function  GetState_UseTTFonts : Boolean;
```

Description

The Bold property sets or gets the bold property of the PCB string True Type text on a PCB document. This property is supported by the GetState_Bold and SetState_Bold methods.

Once the UseTTFonts property is enabled, you can use the Bold, Italic, FontName, Inverted, InvertedTTTextBorder and TTTextOutline properties.

Example

See also

IPCB_Text interface

GetState_Italic method

(IPCB_Text interface)

Syntax

```
Function  GetState_Italic : Boolean;
```

Description

The Italic property sets or gets the italic property of the PCB string True Type text on a PCB document. This property is supported by the GetState_Italic and SetState_Italic methods.

Once the UseTTFonts property is enabled, you can use the Bold, Italic, FontName, Inverted, InvertedTTTextBorder and TTTextOutline properties.

Example

See also

IPCB_Text interface

GetState_FontName method

(IPCB_Text interface)

Syntax

```
Function  GetState_FontName : TPCBString;
```

Description

The FontName property sets or gets the FontName property of the PCB string True Type text on a PCB document. This property is supported by the GetState_Bold and SetState_Bold methods. For example one of the True Type font strings could be 'Arial', 'Arial Narrow', 'Courier New' and 'Verdana'.

Once the UseTTFonts property is enabled, you can use the Bold, Italic, FontName, Inverted, InvertedTTTextBorder and TTTextOutline properties.

Example

See also

IPCB_Text interface

GetState_Inverted method

(IPCB_Text interface)

Syntax

```
Function  GetState_Inverted : Boolean;
```

Description

This property sets or gets the Inverted property of the PCB string True Type text on a PCB document. This property is supported by the GetState_Inverted and SetState_Inverted methods.

The Inverted property inverts the text object and a text boundary is created around the text. The Inverted and InvertedTTTextBorder properties are useful for situations if IPCB_Text object is to be placed on a copper region and create a cutout in the region. The color of the inverted border is the layer color and the text color itself is black.

Once the UseTTFonts property is enabled, you can use the Bold, Italic, FontName, Inverted, InvertedTTTextBorder and TTTextOutline properties.

Example

See also

IPCB_Text interface

GetState_InvertedTTTextBorder method

(IPCB_Text interface)

Syntax

```
Function  GetState_InvertedTTTextBorder : TCoord;
```

Description

This property sets or gets the InvertedTTTextBorder property of the PCB string True Type text on a PCB document. This property is supported by the GetState_InvertedTTTextBorder and SetState_InvertedTTTextBorder methods.

The Inverted property inverts the text object and a text boundary is created around the text. The Inverted and InvertedTTTextBorder properties are useful for situations if IPCB_Text object is to be placed on a copper region and create a cutout in the region. The color of the inverted border is the layer color and the text color itself is black.

Once the UseTTFonts property is enabled, you can use the Bold, Italic, FontName, Inverted, InvertedTTTextBorder and TTTextOutline properties.

Example

See also

IPCB_Text interface

GetTrueTypeTextOutline method

(IPCB_Text interface)

Syntax

```
Property  TTTextOutline : PGPC_Polygon Read GetTrueTypeTextOutline;
```

Description

This property sets or gets the TTTextOutline property of the PCB string True Type text on a PCB document. This property is supported by the GetTrueTypeTextOutline method.

Once the UseTTFonts property is enabled, you can use the Bold, Italic, FontName, Inverted, InvertedTTTextBorder and TTTextOutline properties.

Example

See also

IPCB_Text interface

SetState_FontID method

(IPCB_Text interface)

Syntax

```
Procedure SetState_FontID (FontID : TFontID);
```

Description

This method sets the FontID attribute which represents the font used for this Text Object on a PCB document. This method is used for the FontID property.

Example

See also

IPCB_Text interface

TFontID type

SetState_Mirror method

(IPCB_Text interface)

Syntax

```
Procedure SetState_Mirror (Mirror : Boolean);
```

Description

This method sets the Mirror attribute which represents the mirrored state of this Text Object on a PCB document. This method is used for the Mirror property.

Example

See also

IPCB_Text interface

SetState_Size method

(IPCB_Text interface)

Syntax

```
Procedure SetState_Size (Size : TCoord);
```

Description

This method sets the Size attribute which represents the height of the text used for this Text Object on a PCB document. This method is used for the Size property.

Example

See also

IPCB_Text interface

SetState_Text method

(IPCB_Text interface)

Syntax

```
Procedure SetState_Text (Text : TPCBString);
```

Description

This method sets the Text attribute which represents the text used for this Text Object on a PCB document. This method is used for the Text property.

Example

See also

IPCB_Text interface

SetState_UnderlyingString method

(IPCB_Text interface)

Syntax

```
Procedure SetState_UnderlyingString (Value : TPCBString);
```

Description

This method retrieves the Text attribute which represents the text used for this Text Object on a PCB document and is equivalent to the SetState_Text method. This method is used for the UnderlyingString property.

Example

See also

IPCB_Text interface

SetState_Width method

(IPCB_Text interface)

Syntax

```
Procedure SetState_Width (Width : TCoord);
```

Description

This method sets the Width attribute which represents the width used for this Text Object on a PCB document. This method is used for the Width property.

Example

See also

IPCB_Text interface

SetState_UseTTFonts method

(IPCB_Text interface)

Syntax

```
Procedure SetState_UseTTFonts(UseTTFonts : Boolean);
```

Description

This property toggles the True Type font for the PCB string text on a PCB document. This property is supported by the GetState_UseTTFonts and SetState_UseTTFonts methods.

Once the UseTTFonts property is enabled, you can use the Bold, Italic, FontName, Inverted, InvertedTTTextBorder and TTTextOutline properties.

Example

See also

SetState_Bold method

(IPCB_Text interface)

Syntax

```
Procedure SetState_Bold(Bold : Boolean);
```

Description

The Bold property sets or gets the bold property of the PCB string True Type text on a PCB document. This property is supported by the GetState_Bold and SetState_Bold methods.

Once the UseTTFonts property is enabled, you can use the Bold, Italic, FontName, Inverted, InvertedTTTextBorder and TTTextOutline properties.

Example

See also

IPCB_Text interface

SetState_Italic method

(IPCB_Text interface)

Syntax

```
Procedure SetState_Italic(Italic : Boolean);
```

Description

The Italic property sets or gets the italic property of the PCB string True Type text on a PCB document. This property is supported by the GetState_Italic and SetState_Italic methods.

Once the UseTTFonts property is enabled, you can use the Bold, Italic, FontName, Inverted, InvertedTTTextBorder and TTTextOutline properties.

Example

See also

IPCB_Text interface

SetState_FontName method

(IPCB_Text interface)

Syntax

```
Procedure SetState_FontName(FontName   : TPCBString);
```

Description

The FontName property sets or gets the FontName property of the PCB string True Type text on a PCB document. This property is supported by the GetState_Bold and SetState_Bold methods. For example one of the True Type font strings could be 'Arial', 'Arial Narrow', 'Courier New' and 'Verdana'.

Once the UseTTFonts property is enabled, you can use the Bold, Italic, FontName, Inverted, InvertedTTTextBorder and TTTextOutline properties.

Example

See also

IPCB_Text interface

SetState_Inverted method

(IPCB_Text interface)

Syntax

```
Procedure SetState_Inverted(Inverted : Boolean);
```

Description

This property sets or gets the Inverted property of the PCB string True Type text on a PCB document. This property is supported by the GetState_Inverted and SetState_Inverted methods.

The Inverted property inverts the text object and a text boundary is created around the text. The Inverted and InvertedTTTextBorder properties are useful for situations if IPCB_Text object is to be placed on a copper region and create a cutout in the region. The color of the inverted border is the layer color and the text color itself is black.

Once the UseTTFonts property is enabled, you can use the Bold, Italic, FontName, Inverted, InvertedTTTextBorder and TTTextOutline properties.

Example

See also

IPCB_Text interface

Methods

ConvertToStrokeArray method

(IPCB_Text interface)

Syntax

```
Function ConvertToStrokeArray(Var Count : Integer; Var Strokes : TStrokeArray) : Boolean;
```

Description

Text objects can be converted into a series of strokes using the **ConvertToStrokeArray** method. This is useful for rending text objects as standalone line objects to be used in external programs such as 3D modelling applications.

Example

See also

IPCB_Text interface

TStrokeArray type

GetDesignatorDisplayString method

(IPCB_Text interface)

Syntax

```
Function GetDesignatorDisplayString : TPCBString;
```

Description

This function retrieves the designator string directly from a text object.

Example

See also

IPCB_Text interface

TPCBString type

InAutoDimension method

(IPCB_Text interface)

Syntax

```
Function InAutoDimension : Boolean;
```

Description

This function tests whether this text object is used for the auto dimension object or not.

Example

See also

IPCB_Text interface

IsComment method

(IPCB_Text interface)

Syntax

```
Function IsComment : Boolean;
```

Description

This function tests whether this text object is a comment object associated with a component object for example.

Example

See also

IPCB_Text interface

IsDesignator method

(IPCB_Text interface)

Syntax

```
Function IsDesignator : Boolean;
```

Description

This function tests whether this text object is a designator for a object, for example whether a pad object has a designator.

Example

See also

IPCB_Text interface

IsHidden method

(IPCB_Text interface)

Syntax

```
Function IsHidden : Boolean;
```

Description

This function tests whether the text object is hidden or not.

Example

See also

IPCB_Text interface

RotationHandle method

(IPCB_Text interface)

Syntax

```
Function RotationHandle : TPoint;
```

Description

This function returns the rotation handle of the text object as a record of X and Y coordinates (TPoint).

Note, the TPoint type is a Borland Delphi record consisting of X and Y coordinates.

Example

See also

IPCB_Text interface

GetTrueTypeTextOutline method

(IPCB_Text interface)

Syntax

```
Function  GetTrueTypeTextOutline : Pgpc_polygon;
```

Description

Example

See also

IPCB_Text interface

Properties

FontID property

(IPCB_Text interface)

Syntax

```
Property FontID : TFontID Read GetState_FontID Write SetState_FontID;
```

Description

The **FontID** property denotes which Font the text object is using. The property is supported by **GetState_FontID** and **SetState_FontID** methods.

The **TFontID** type defines the font ID for a text object. It is the index to an entry in the font table in the PCB editor. Each font used in the PCB editor has its own FontID.

Thus when a new font is used (through a Change Font dialog of a Change object dialog), a new FontID is added to the table in the PCB editor. The FontID value can be extracted from PCB text objects.

Example

See also

IPCB_Text interface

TFontID type

MirrorFlag property

(IPCB_Text interface)

Syntax

```
Property MirrorFlag : Boolean Read GetState_Mirror Write SetState_Mirror;
```

Description

This method sets the Mirror attribute which represents the mirrored state of this Text Object on a PCB document. This property supports GetState_Mirror and SetState_Mirror methods.

Example

See also

IPCB_Text interface

Size property

(IPCB_Text interface)

Syntax

```
Property Size : TCoord Read GetState_Size Write SetState_Size;
```

Description

The Size property sets the height of the text. This property is supported by GetState_Size and SetState_Size methods.

Example

See also

IPCB_Text interface

TCoord type

Text property

(IPCB_Text interface)

Syntax

```
Property Text : TPCBString Read GetState_Text Write SetState_Text;
```

Description

The Text property contains the text for the Text object. This property is supported by the GetState_Text and SetState_Text methods.

Note, the PCB editor includes special strings which are interpreted when output (printing, plotting or generating gerber files) is generated. For example, the string .PRINT_DATE will be replaced by the current date when output is generated.

Example

```
Procedure FindSpecialStrings;
Var
    Board         : IPCB_Board;
    SpecialString : IPCB_Text;
    Iterator      : IPCB_BoardIterator;
Begin
    // Retrieve the current board
    Board := PCBServer.GetCurrentPCBBoard;
    If Board = Nil Then Exit;

    // retrieve the iterator
    Iterator        := Board.BoardIterator_Create;
    Iterator.AddFilter_ObjectSet(MkSet(eTextObject));
    Iterator.AddFilter_LayerSet(AllLayers);
    Iterator.AddFilter_Method(eProcessAll);

    // Search special strings
    SpecialString := Iterator.FirstPCBObject;
    While (SpecialString <> Nil) Do
    Begin
        If SpecialString.Text = '.Layer_Name' Then
            ShowMessage(SpecialString.ConvertedString);
        SpecialString := Iterator.NextPCBObject;
    End;
    Board.BoardIterator_Destroy(Iterator);
End;
```

See also

IPCB_Text interface

TPCBString type

UnderlyingString property

(IPCB_Text interface)

Syntax

```
Property UnderlyingString : TPCBString Read GetState_UnderlyingString Write
SetState_UnderlyingString;
```

Description

This UnderlyingString property is equivalent to the Text property. This property is supported by the GetState_UnderlyingString and SetState_UnderlyingString methods.

Note, the PCB editor includes special strings which are interpreted when output (printing, plotting or generating gerber files) is generated. For example, the string .PRINT_DATE will be replaced by the current date when output is generated.

Example

See also

IPCB_Text interface

TPCBString type

Width property

(IPCB_Text interface)

Syntax

```
Property Width : TCoord Read GetState_Width Write SetState_Width;
```

Description

This method sets the Width attribute which represents the width used for this Text Object on a PCB document. This property is supported by the GetState_Width and SetState_Width methods.

Example

See also

IPCB_Text interface

ConvertedString method

(IPCB_Text interface)

Syntax

```
Property  ConvertedString : TPCBString Read GetState_ConvertedString;
```

Description

This property is supported by the GetState_ConvertedString method. This property converts a special string into a text based string. The PCB editor includes special strings which are interpreted when output (printing, plotting or generating gerber files) is generated. For example, the string .PRINT_DATE will be replaced by the current date when the ConvertedString method is invoked.

The available special strings are;

.PRINT_DATE

.PRINT_TIME

.PRINT_SCALE

.LAYER_NAME

.PCB_FILE_NAME

.PCB_FILE_NAME_NO_PATH

.PLOT_FILE_NAME

.ARC_COUNT

.COMPONENT_COUNT

.FILL_COUNT

.HOLE_COUNT

.NET_COUNT

.PAD_COUNT

.STRING_COUNT

.TRACK_COUNT

.VIA_COUNT

.DESIGNATOR

.COMMENT

.LEGEND

.NET_NAMES_ON_LAYER

Example

```
Procedure FindSpecialStrings;
Var
    Board         : IPCB_Board;
    SpecialString : IPCB_Text;
    Iterator      : IPCB_BoardIterator;
Begin
    // Retrieve the current board
    Board := PCBServer.GetCurrentPCBBoard;
    If Board = Nil Then Exit;
    // retrieve the iterator
    Iterator        := Board.BoardIterator_Create;
    Iterator.AddFilter_ObjectSet(MkSet(eTextObject));
    Iterator.AddFilter_LayerSet(AllLayers);
    Iterator.AddFilter_Method(eProcessAll);
    // Search special strings
    SpecialString := Iterator.FirstPCBObject;
    While (SpecialString <> Nil) Do
    Begin
        If SpecialString.Text = '.Layer_Name' Then
            ShowMessage(SpecialString.ConvertedString);
        SpecialString := Iterator.NextPCBObject;
    End;
    Board.BoardIterator_Destroy(Iterator);
End;
```

See also

IPCB_Text interface

IPCB_SpecialStringConverter

IPCB_ServerInterface and its SpecialStringConverter property.

TPCBString type

UseTTFonts property

(IPCB_Text interface)

Syntax

```
Property  UseTTFonts : Boolean Read GetState_UseTTFonts Write SetState_UseTTFonts;
```

Description

This property toggles the True Type font property for the PCB string text on a PCB document. This property is supported by the GetState_UseTTFonts and SetState_UseTTFonts methods.

Once the UseTTFonts property is enabled, you can use the Bold, Italic, FontName, Inverted, InvertedTTTextBorder and TTTextOutline properties.

Delphiscript Script **Example**

```
TextObj := PCBServer.PCBObjectFactory(eTextObject, eNoDimension, eCreate_Default);
// notify that the pcb object is going to be modified
```

```
PCBServer.SendMessageToRobots(TextObj.I_ObjectAddress, c_Broadcast, PCBM_BeginModify,
c_NoEventData);

TextObj.XLocation := Sheet.SheetX + MilsToCoord(1000);
TextObj.YLocation := Sheet.SheetY + MilsToCoord(1000);
TextObj.Layer     := eBottomOverlay;

// Can use Open True Type Fonts...
TextObj.UseTTFonts := True;
TextObj.Italic := True;
TextObj.Bold := False;
TextObj.FontName := 'ARIAL';
TextObj.Inverted := True;
TextObj.InvertedTTTextBorder := MilsToCoord(100);

TextObj.Text      := 'Text with True Type Property enabled.';
TextObj.Size       := MilsToCoord(200);    // sets the height of the text.

Board.AddPCBObject(TextObj);
// notify that the pcb object has been modified
PCBServer.SendMessageToRobots(TextObj.I_ObjectAddress, c_Broadcast, PCBM_EndModify ,
c_NoEventData);
PCBServer.SendMessageToRobots(Board.I_ObjectAddress, c_Broadcast,
PCBM_BoardRegisteration,TextObj.I_ObjectAddress);
```

See also

IPCB_Text interface

Bold property

(IPCB_Text interface)

Syntax

```
Property  Bold : Boolean Read GetState_Bold Write SetState_Bold;
```

Description

This property sets or gets the bold property of the PCB string True Type text on a PCB document. This property is supported by the GetState_Bold and SetState_Bold methods.

Once the UseTTFonts property is enabled, you can use the Bold, Italic, FontName, Inverted, InvertedTTTextBorder and TTTextOutline properties.

Example

See also

IPCB_Text interface

Italic property

(IPCB_Text interface)

Syntax

```
Property Italic : Boolean Read GetState_Italic Write SetState_Italic;
```

Description

The Italic property sets or gets the italic property of the PCB string True Type text on a PCB document. This property is supported by the GetState_Italic and SetState_Italic methods.

Once the UseTTFonts property is enabled, you can use the Bold, Italic, FontName, Inverted, InvertedTTTextBorder and TTTextOutline properties.

Example

See also

IPCB_Text interface

FontName property

(IPCB_Text interface)

Syntax

```
Property FontName : TPCBString Read GetState_FontName Write SetState_FontName;
```

Description

This property sets or gets the FontName property of the PCB string True Type text on a PCB document. For example one of the True Type font strings could be 'Arial', 'Arial Narrow', 'Courier New' and 'Verdana'. This property is supported by the GetState_Bold and SetState_Bold methods.

Once the UseTTFonts property is enabled, you can use the Bold, Italic, FontName, Inverted, InvertedTTTextBorder and TTTextOutline properties.

Delphiscript Script **Example**

```
TextObj := PCBServer.PCBObjectFactory(eTextObject, eNoDimension, eCreate_Default);

// notify that the pcb object is going to be modified
PCBServer.SendMessageToRobots(TextObj.I_ObjectAddress, c_Broadcast, PCBM_BeginModify,
c_NoEventData);
TextObj.XLocation := Sheet.SheetX + MilsToCoord(1000);
TextObj.YLocation := Sheet.SheetY + MilsToCoord(1000);
TextObj.Layer     := eBottomOverlay;
TextObj.UseTTFonts := True;
TextObj.Italic := True;
TextObj.Bold := False;
TextObj.FontName := 'ARIAL';
// inverts the text object and a text boundary is created around the text
// The Inverted and InvertedTTTextBorder properties are useful for situations
// if text is to be placed on a copper region and create a cutout in the region.
// the color of the inverted border is the layer color and the text color itself
// is black.
TextObj.Inverted := True;
// The InvertedTTextBorder property determines the distance between the boundary of the
// the text object itself to the inverted text border boundary.
TextObj.InvertedTTTextBorder := MilsToCoord(100);
TextObj.Text      := 'Text with True Type Property enabled.';
TextObj.Size       := MilsToCoord(200);    // sets the height of the text.
```

See also

IPCB_Text interface

Inverted property

(IPCB_Text interface)

Syntax

```
Property  Inverted : Boolean Read GetState_Inverted Write SetState_Inverted;
```

Description

This property sets or gets the Inverted property of the PCB string True Type text on a PCB document. This property is supported by the GetState_Inverted and SetState_Inverted methods.

The Inverted property inverts the text object and a text boundary is created around the text. The Inverted and InvertedTTTextBorder properties are useful for situations if IPCB_Text object is to be placed on a copper region and create a cutout in the region. The color of the inverted border is the layer color and the text color itself is black.

Once the UseTTFonts property is enabled, you can use the Bold, Italic, FontName, Inverted, InvertedTTTextBorder and TTTextOutline properties.

Example

See also

IPCB_Text interface

InvertedTTTextBorder property

InvertedTTTextBorder property

(IPCB_Text interface)

Syntax

```
Property InvertedTTTextBorder : TCoord Read GetState_InvertedTTTextBorder Write
SetState_InvertedTTTextBorder;
```

Description

This property sets or gets the **InvertedTTTextBorder** property of the PCB string True Type text on a PCB document. This property is supported by the **GetState_InvertedTTTextBorder** and **SetState_InvertedTTTextBorder** methods.

The Inverted property inverts the text object and a text boundary is created around the text. The **Inverted** and **InvertedTTTextBorder** properties are useful for situations if **IPCB_Text** object is to be placed on a copper region and create a cutout in the region. The color of the inverted border is the layer color and the text color itself is black.

Once the **UseTTFonts** property is enabled, you can use the **Bold**, **Italic**, **FontName**, **Inverted**, **InvertedTTTextBorder** and **TTTextOutline** properties.

Example

See also

IPCB_Text interface

Inverted property

TTTextOutline property

(IPCB_Text interface)

Syntax

```
Property TTTextOutline : PGPC_Polygon Read GetTrueTypeTextOutline;
```

Description

This property sets or gets the TTTextOutline property of the PCB string True Type text on a PCB document. This property is supported by the GetState_TTTextOutline and SetState_TTTextOutline methods.

Once the UseTTFonts property is enabled, you can use the Bold, Italic, FontName, Inverted, InvertedTTTextBorder and TTTextOutline properties.

Example

See also

IPCB_Text interface

IPCB_Track Interface

Overview

Tracks can be placed on any layer and their widths can range from 0.001 to 10000 mils wide. Tracks are used to create polygon planes and are also used in coordinates, dimensions and components.

Tracks that carry either signals or power supply can be placed on:

• Top (component side) signal layer.

• Any of the thirty mid signal layers.
• Bottom (solder side) signal layer.
Non-electrical tracks can also be placed on:
• Any of the silk screen overlays (normally used for component package outlines).
• Any of the sixteen internal plane layers (used as voids in these solid copper planes).
• The keep out layer to define the board perimeter for autorouting and auto component placement
• Any of the sixteen mechanical layers for mechanical details.
• Solder or paste mask layers for any special openings required in the masks
The IPCB_Track hierarchy;
IPCB_Primitive
IPCB_Track

IPCB_Track methods	IPCB_Track properties
GetState_X1	X1
GetState_Y1	Y1
GetState_X2	X2
GetState_Y2	Y2
GetState_Width	Width
SetState_X1	
SetState_Y1	
SetState_X2	
SetState_Y2	
SetState_Width	

Example

```
Var
    Board     : IPCB_Board;
    WorkSpace : IWorkSpace;
    Track     : IPCB_Track;
Begin
    //Create a new PCB document
    WorkSpace := GetWorkSpace;
    If WorkSpace = Nil Then Exit;
    Workspace.DM_CreateNewDocument('PCB');

    // Check if the new PCB document exists.
    Board := PCBServer.GetCurrentPCBBoard;
    If Board = Nil then exit;

    // Create a Track object with 'Mils' dimensions
    Track             := PCBServer.PCBObjectFactory(eTrackObject, eNoDimension,
eCreate_Default);
    Track.X1          := MilsToCoord(X1);
    Track.Y1          := MilsToCoord(Y1);
    Track.X2          := MilsToCoord(X2);
    Track.Y2          := MilsToCoord(Y2);
    Track.Layer       := Layer;
```

```
    Track.Width       := MilsToCoord(Width);
    // Add the new track into the PCB document
    Board.AddPCBObject(Track);

    // Refresh the PCB document.
    ResetParameters;
    AddStringParameter('Action', 'All');
    RunProcess('PCB:Zoom');
End;
```

See also

IPCB_Primitive interface

PCB Design Objects

GetState and SetState Methods

GetState_Width method

(IPCB_Track interface)

Syntax

```
Function GetState_Width : TCoord;
```

Description

This method retrieves the width attribute of the track object on a PCB document. This function is used for the Width property.

Example

See also

IPCB_Track interface

GetState_X1 method

(IPCB_Track interface)

Syntax

```
Function GetState_X1 : TCoord;
```

Description

This method retrieves the X1 attribute of the track object on a PCB document. This function is used for the X1 property.

Example

See also

IPCB_Track interface

GetState_X2 method

(IPCB_Track interface)

Syntax

```
Function GetState_X2 : TCoord;
```

Description

This method retrieves the X2 attribute of the track object on a PCB document. This function is used for the X2 property.

Example

See also

IPCB_Track interface

GetState_Y1 method

(IPCB_Track interface)

Syntax

```
Function GetState_Y1 : TCoord;
```

Description

This method retrieves the Y1 attribute of the track object on a PCB document. This function is used for the Y1 property.

Example

See also

IPCB_Track interface

GetState_Y2 method

(IPCB_Track interface)

Syntax

```
Function GetState_Y2 : TCoord;
```

Description

This method retrieves the Y2 attribute of the track object on a PCB document. This function is used for the Y2 property.

Example

See also

IPCB_Track interface

SetState_Width method

(IPCB_Track interface)

Syntax

```
Procedure SetState_Width (Value : TCoord);
```

Description

This method sets the width attribute of the track object on a PCB document. This function is used for the Width property.

Example

See also

IPCB_Track interface

SetState_X1 method

(IPCB_Track interface)

Syntax

```
Procedure SetState_X1 (Value : TCoord);
```

Description

This method sets the X1 attribute of the track object on a PCB document. This function is used for the X1 property.

Example

See also

IPCB_Track interface

SetState_X2 method

(IPCB_Track interface)

Syntax

```
Procedure SetState_X2 (Value : TCoord);
```

Description

This method sets the X2 attribute of the track object on a PCB document. This function is used for the X2 property.

Example

See also

IPCB_Track interface

SetState_Y1 method

(IPCB_Track interface)

Syntax

```
Procedure SetState_Y1 (Value : TCoord);
```

Description

This method sets the Y1 attribute of the track object on a PCB document. This function is used for the Y1 property.

Example

See also

IPCB_Track interface

SetState_Y2 method

(IPCB_Track interface)

Syntax

```
Procedure SetState_Y2 (Value : TCoord);
```

Description

This method sets the Y2 attribute of the track object on a PCB document. This function is used for the Y2 property.

Example

See also

IPCB_Track interface

Properties

Width property

(IPCB_Track interface)

Syntax

```
Property Width : TCoord Read GetState_Width Write SetState_Width;
```

Description

The property represents the width attribute of a track object on the PCB document. This property is supported by the GetState_Width and SetState_Width methods.

Example

See also

IPCB_Track interface

X1 property

(IPCB_Track interface)

Syntax

```
Property X1 : TCoord Read GetState_X1 Write SetState_X1;
```

Description

The property represents the X1 or the initial X coordinate of a track object on the PCB document. This property is supported by the GetState_X1 and SetState_X1 methods.

Example

See also

IPCB_Track interface

X2 property

(IPCB_Track interface)

Syntax

```
Property X2 : TCoord Read GetState_X2 Write SetState_X2;
```

Description

The property represents the X2 or the final X coordinate of a track object on the PCB document. This property is supported by the GetState_X2 and SetState_X2 methods.

Example

See also

IPCB_Track interface

Y1 property

(IPCB_Track interface)

Syntax

```
Property Y1 : TCoord Read GetState_Y1 Write SetState_Y1;
```

Description

The property represents the Y1 or the initial Y coordinate of a track object on the PCB document. This property is supported by the GetState_Y1 and SetState_Y1 methods.

Example

See also

IPCB_Track interface

Y2 property

(IPCB_Track interface)

Syntax

```
Property Y2 : TCoord Read GetState_Y2 Write SetState_Y2;
```

Description

The property represents the Y2 or the final Y coordinate of a track object on the PCB document. This property is supported by the GetState_Y2 and SetState_Y2 methods.

Example

See also

IPCB_Track interface

IPCB_TTFontsCache Interface

Overview

IPCB_TTFontsCache methods	IPCB_TTFontCache properties
I_ObjectAddress	FontsCount
GetState_FontsCount	EmbeddedFontsCount

GetState_EmbeddedFont Font
GetState_Font

AddFont
AddEmbeddedFont
GetFont
GetNextEmbeddedFont
ExportFontsToList
GetLocalizedFontName

Methods

Properties

FontCount property

(IPCB_TTFontsCache interface)

Syntax

```
Property FontsCount : Integer Read GetState_FontsCount;
```

Description

Example

See also

IPCB_TTFontsCache interface

EmbeddedFontCount property

(IPCB_TTFontsCache interface)

Syntax

```
Property  EmbeddedFontsCount [ABoard : Pointer]    : Integer         Read
GetState_EmbeddedFontsCount;
```

Description

Example

See also

IPCB_TTFontsCache interface

Font property

(IPCB_TTFontsCache interface)

Syntax

```
Property  Font [I      : Integer]    : IPCB_TTFontData Read GetState_Font;
```

Description

Example

See also

IPCB_TTFontsCache interface

IPCB_TTFontData Interface

Overview

IPCB_TTFontData methods	IPCB_TTFontData properties
I_ObjectAddress	FontFullName
GetEmbeddedFontData	FontFaceName
IsEmbedded	FontStyleName
IsEmbeddedInDocument	Bold
FontExists	Italic
IsSame	CanEmbed
	EmbeddedFontHandle
GetState_FontFullName	Charset
GetState_FontFaceName	RefCount
GetState_FontStyleName	
GetState_Bold	
GetState_Italic	
GetState_CanEmbed	
GetState_EmbeddedFontHandle	
GetState_Charset	
GetState_RefCount	
AddRef	
vRelease	

Methods

I_ObjectAddress method

(IPCB_TTFontData interface)

Syntax

Description

Example

See also

IPCB_TTFontData interface

GetEmbeddedFontData method

(IPCB_TTFontData interface)

Syntax

Description

Example

See also

IPCB_TTFontData interface

IsEmbedded method

(IPCB_TTFontData interface)

Syntax

Description

Example

See also

IPCB_TTFontData interface

IsEmbeddedInDocument method

(IPCB_TTFontData interface)

Syntax

Description

Example

See also

IPCB_TTFontData interface

IsEmbedded method

(IPCB_TTFontData interface)

Syntax

Description

Example

See also

IPCB_TTFontData interface

FontExists method

(IPCB_TTFontData interface)

Syntax

Description

Example

See also

IPCB_TTFontData interface

IsSame method

(IPCB_TTFontData interface)

Syntax

Description

Example

See also
IPCB_TTFontData interface

GetState and SetState Methods

GetState_Width method
(IPCB_TTFontData interface)
Syntax

Description

Example

See also
IPCB_TTFontData interface

Properties

FontFullName Property
Syntax

Description

Example

See also
IPCB_TTFontData interface

FontFaceName property
Syntax

Description

Example

See also
IPCB_TTFontData interface

FontStyleName property
Syntax

Description

Example

See also

IPCB_TTFontData interface

Bold property

Syntax

Description

Example

See also

IPCB_TTFontData interface

Italic property

Syntax

Description

Example

See also

IPCB_TTFontData interface

CanEmbed property

Syntax

Description

Example

See also

IPCB_TTFontData interface

EmbeddedFontHandle property

Syntax

Description

Example

See also

IPCB_TTFontData interface

Charset property

Syntax

Description

Example

See also

IPCB_TTFontData interface

RefCount

Syntax

Description

Example

See also

IPCB_TTFontData interface

IPCB_Via Interface

Overview

When tracks from two layers need to be connected, vias are placed to carry a signal from one layer to the other. Vias are like round pads, which are drilled and usually through-plated when the board is fabricated. Vias can be multi-layered, blind or buried.

A multi-layer via passes through the board from the Top layer to the Bottom layer and allows connections to all other signal layers.

A blind via connects from the surface of the board to an internal layer, a buried via connects from one internal layer to another internal layer. In Altium Designer, Vias, including blind and buried, can connect to internal planes.

Vias do not have a paste mask layer.

The IPCB_Via hierarchy;

IPCB_Primitive

IPCB_Via

IPCB_Via methods	IPCB_Via properties
GetState_XLocation	X
GetState_YLocation	Y
GetState_IsConnectedToPlane	IsConnectedToPlane
GetState_LowLayer	LowLayer
GetState_HighLayer	HighLayer
GetState_StartLayer	StartLayer
GetState_StopLayer	StopLayer
GetState_HoleSize	HoleSize
GetState_Size	Size
GetState_SizeOnLayer	SizeOnLayer
GetState_ShapeOnLayer	ShapeOnLayer
GetState_Cache	Cache
SetState_XLocation	
SetState_YLocation	
SetState_LowLayer	
SetState_HighLayer	
SetState_IsConnectedToPlane	

SetState_HoleSize
SetState_Size
SetState_Cache
PlaneConnectionStyleForLayer
RotateAroundXY
IntersectLayer

Example

```
Var
    Board     : IPCB_Board;
    WorkSpace : IWorkSpace;
    Via       : IPCB_Via;
    ViaCache  : TPadCache;
Begin
    // Create a new PCB document
    WorkSpace := GetWorkSpace;
    If WorkSpace = Nil Then Exit;
    Workspace.DM_CreateNewDocument('PCB');

    // Check if the new PCB document exists or not.
    Board := PCBServer.GetCurrentPCBBoard;
    If Board = Nil then exit;

    // Create a Via object
    Via           := PCBServer.PCBObjectFactory(eViaObject, eNoDimension, eCreate_Default);
    Via.X         := MilsToCoord(2000);
    Via.Y         := MilsToCoord(2000);
    Via.Size      := MilsToCoord(50);
    Via.HoleSize  := MilsToCoord(20);
    Via.LowLayer  := eTopLayer;
    Via.HighLayer := eBottomLayer;

    // Setup a pad cache
    Viacache := Via.GetState_Cache;
    Viacache.ReliefAirGap := MilsToCoord(11);
    Viacache.PowerPlaneReliefExpansion := MilsToCoord(11);
    Viacache.PowerPlaneClearance       := MilsToCoord(11);
    Viacache.ReliefConductorWidth      := MilsToCoord(11);
    Viacache.SolderMaskExpansion       := MilsToCoord(11);
    Viacache.SolderMaskExpansionValid  := eCacheManual;
    Viacache.PasteMaskExpansion        := MilsToCoord(11);
    Viacache.PasteMaskExpansionValid   := eCacheManual;

    // Assign the new Via cache to the via
    Via.SetState_Cache := Viacache;
    Board.AddPCBObject(Via);

    // Refresh PCB document.
```

```
    ResetParameters;
    AddStringParameter('Action', 'All');
    RunProcess('PCB:Zoom');
End;
```

See also

IPCB_Primitive interface

IPCB_Pad interface

TLayer enumerated values

TPlaneConnectionStyle enumerated values

TCoord value

TAngle value

TPadCache values

GetState and SetState Methods

GetState_Cache method

(IPCB_Via interface)

Syntax

```
Function GetState_Cache : TPadCache;
```

Description

This Cache property represents the global cache that stores various design rule settings for pad and via objects. The method is used by the Cache property.

Example

See also

IPCB_Via interface

GetState_HighLayer method

(IPCB_Via interface)

Syntax

```
Function GetState_HighLayer : TLayer;
```

Description

The HighLayer property denotes the bottom layer. The method is used for the HighLayer property.

Example

See also

IPCB_Via interface

GetState_HoleSize method

(IPCB_Via interface)

Syntax

```
Function GetState_HoleSize : TCoord;
```

Description

This HoleSize property denotes the hole size of the via object. This method is used by the HoleSize property.

Example

See also

IPCB_Via interface

GetState_IsConnectedToPlane method

(IPCB_Via interface)

Syntax

```
Function GetState_IsConnectedToPlane (Layer : TLayer) : Boolean;
```

Description

This property determines whether the via is connected to this specified plane or not by returning a boolean value. This method is used by the IsConnectedToPlane property.

Example

See also

IPCB_Via interface

GetState_LowLayer method

(IPCB_Via interface)

Syntax

```
Function GetState_LowLayer : TLayer;
```

Description

The LowLayer property denotes the bottom layer. The method is used for the LowLayer property.

Example

See also

IPCB_Via interface

GetState_ShapeOnLayer method

(IPCB_Via interface)

Syntax

```
Function GetState_ShapeOnLayer (Layer : TLayer) : TShape;
```

Description

The ShapeOnLayer property determines the shape of the via on the specified layer. This read only property is supported by the GetState_ShapeOnLayer method.

Example

See also

IPCB_Via interface

GetState_Size method

(IPCB_Via interface)

Syntax

```
Function GetState_Size : TCoord;
```

Description

The Size property denotes the size of the via object (the full diameter). The method is used for the Size property.

Example

See also

IPCB_Via interface

GetState_SizeOnLayer method

(IPCB_Via interface)

Syntax

```
Function GetState_SizeOnLayer (Layer : TLayer) : TCoord;
```

Description

This SizeOnLayer property denotes the size of the via on a specified layer. This method is used for the SizeOnLayer property.

Example

See also

IPCB_Via interface

GetState_StartLayer method

(IPCB_Via interface)

Syntax

```
Function GetState_StartLayer : IPCB_LayerObject;
```

Description

This StartLayer property fetches the Start layer of IPCB_LayerObject type that the via is connected to. This method is used for the StartLayer property.

Example

See also

IPCB_Via interface

GetState_StopLayer method

(IPCB_Via interface)

Syntax

```
Function GetState_StopLayer : IPCB_LayerObject;
```

Description

This StartLayer property fetches the Stop layer of IPCB_LayerObject type that the via is connected to. This method is used for the StopLayer property.

Example

See also

IPCB_Via interface

GetState_XLocation method

(IPCB_Via interface)

Syntax

```
Function GetState_XLocation : TCoord;
```

Description

The X and Y properties define the location of the Via object with respect to the PCB document. The GetState_XLocation, GetState_YLocation and SetState_XLocation, SetStateYLocation methods.

Example

See also

IPCB_Via interface

GetState_YLocation method

(IPCB_Via interface)

Syntax

```
Function GetState_YLocation : TCoord;
```

Description

The X and Y properties define the location of the Via object with respect to the PCB document. The GetState_XLocation, GetState_YLocation and SetState_XLocation, SetStateYLocation methods.

Example

See also

IPCB_Via interface

SetState_Cache method

(IPCB_Via interface)

Syntax

```
Procedure SetState_Cache (Value : TPadCache);
```

Description

This Cache property represents the global cache that stores various design rule settings for pad and via objects. The method is used by the Cache property.

Example

See also

IPCB_Via interface

SetState_HighLayer method

(IPCB_Via interface)

Syntax

```
Procedure SetState_HighLayer (L : TLayer);
```

Description

The HighLayer property denotes the bottom layer. The method is used for the HighLayer property.

Example

See also

IPCB_Via interface

SetState_HoleSize method

(IPCB_Via interface)

Syntax

```
Procedure SetState_HoleSize (Value : TCoord);
```

Description

This HoleSize property denotes the hole size of the via object. This method is used by the HoleSize property.

Example

See also

IPCB_Via interface

SetState_IsConnectedToPlane method

(IPCB_Via interface)

Syntax

```
Procedure SetState_IsConnectedToPlane (Layer : TLayer;Value : Boolean);
```

Description

This property determines whether the via is connected to this specified plane or not by returning a boolean value. This method is used by the IsConnectedToPlane property.

Example

See also

IPCB_Via interface

SetState_LowLayer method

(IPCB_Via interface)

Syntax

```
Procedure SetState_LowLayer (L : TLayer);
```

Description

The LowLayer property denotes the bottom layer. The method is used for the LowLayer property.

Example

See also

IPCB_Via interface

SetState_Size method

(IPCB_Via interface)

Syntax

```
Procedure SetState_Size (Size : TCoord);
```

Description

The Size property denotes the size of the via object. The method is used for the Size property.

Example

See also

IPCB_Via interface

SetState_XLocation method

(IPCB_Via interface)

Syntax

```
Procedure SetState_XLocation (AX : TCoord);
```

Description

The X and Y properties define the location of the Via object with respect to the PCB document. The GetState_XLocation, GetState_YLocation and SetState_XLocation, SetStateYLocation methods.

Example

See also

IPCB_Via interface

SetState_YLocation method

(IPCB_Via interface)

Syntax

```
Procedure SetState_YLocation (AY : TCoord);
```

Description

The X and Y properties define the location of the Via object with respect to the PCB document. The GetState_XLocation, GetState_YLocation and SetState_XLocation, SetStateYLocation methods.

Example

See also

IPCB_Via interface

Methods

RotateAroundXY method

(IPCB_Via interface)

Syntax

```
Procedure RotateAroundXY (AX, AY : TCoord;Angle : TAngle);
```

Description

This method rotates a via object on the PCB document about the AX, AY coordinates with an angle in degrees. To ensure the via rotates without moving about, pass in its midpoint (between X1,X2 and Y1, Y2) attributes for the AX,AY parameters.

Example

See also

IPCB_Via interface

PlaneConnectionStyleForLayer method

(IPCB_Via interface)

Syntax

```
Function PlaneConnectionStyleForLayer(ALayer : TLayer) : TPlaneConnectionStyle;
```

Description

Vias automatically connect to an internal power plane layer that is assigned the same net name. The via will connect to the plane depending on the applicable Power Plane Connect Style design rule. If you do not want vias to connect to power planes, add another Power Plane Connect Style design rule targeting the specific vias required and with a connection style of No Connect.

The Connect Style defines the style of the connection from a pin of a component, targeted by the scope (Full Query) of the rule, to a power plane. The following three styles as per the TPlaneConnectionStyle type are available:

No Connect - do not connect a component pin to the power plane.

Direct Connect - connect using solid copper to the pin.

Relief Connect (default) - connect using a thermal relief connection.

Example

See also

IPCB_Via interface

TPlaneConnectionStyle type

IntersectLayer method

(IPCB_Via interface)

Syntax

```
Function IntersectLayer (ALayer : TLayer) : Boolean;
```

Description

Example

See also

IPCB_Via interface

Properties

Cache property

(IPCB_Via interface)

Syntax

```
Property Cache : TPadCache Read GetState_Cache Write SetState_Cache;
```

Description

This Cache property represents the global cache that stores various design rule settings for pad and via objects.

This property is supported by the GetState_Cache and SetState_Cache methods.

Example

```
Var
    PadCache : TPadCache;
    Via      : IPCB_Via;
    Board    : IPCB_Board;
Begin
    (* Create a Via object*)
    Via := PCBServer.PCBObjectFactory(eViaObject, eNoDimension, eCreate_Default);
    Via.X  := MilsToCoord(3000);
    Via.Y  := MilsToCoord(3000);

    (* Setup a pad cache *)
    Padcache := Via.Cache;
    Padcache.ReliefAirGap := MilsToCoord(11);
    Padcache.PowerPlaneReliefExpansion := MilsToCoord(11);
    Padcache.PowerPlaneClearance       := MilsToCoord(11);
    Padcache.ReliefConductorWidth      := MilsToCoord(11);
    Padcache.SolderMaskExpansion       := MilsToCoord(11);
    Padcache.SolderMaskExpansionValid  := eCacheManual;
    Padcache.PasteMaskExpansion        := MilsToCoord(11);
    Padcache.PasteMaskExpansionValid   := eCacheManual;

    (* Assign the new pad cache to the via*)
    Via.Cache := Padcache;
    Board.AddPCBObject(Via);
End;
```

See also

IPCB_Via interface

PadViaCacheProperties script from **Example**s\Scripts\Delphiscript Scripts\Pcb\ folder.

DrawObjects script from **Example**s\Scripts\DelphiScript Scripts\PCB\ folder.

CreateAVia script from **Example**s\Scripts\DelphiScript Scripts\PCB\ folder.

HighLayer property

(IPCB_Via interface)

Syntax

```
Property HighLayer : TLayer Read GetState_HighLayer Write SetState_HighLayer;
```

Description

The HighLayer property denotes the top layer. This property is supported by the GetState_HighLayer and SetState_HighLayer methods.

Example

See also

IPCB_Via interface

HoleSize property

(IPCB_Via interface)

Syntax

```
Property HoleSize : TCoord Read GetState_HoleSize Write SetState_HoleSize;
```

Description

This HoleSize property denotes the hole size of the via object. This property is supported by the GetState_HighLayer and SetState_HighLayer methods.

Example

See also

IPCB_Via interface

IsConnectedToPlane property

(IPCB_Via interface)

Syntax

```
Property IsConnectedToPlane[L : TLayer] : Boolean Read GetState_IsConnectedToPlane Write
SetState_IsConnectedToPlane;
```

Description

This property determines whether the via is connected to this specified plane or not by returning a boolean value.

This property is supported by the GetState_IsConnectedToPlane and SetState_IsConnectedToPlane methods.

Example

See also

IPCB_Via interface

LowLayer property

(IPCB_Via interface)

Syntax

```
Property LowLayer : TLayer Read GetState_LowLayer Write SetState_LowLayer;
```

Description

The LowLayer property denotes the bottom layer. This property is supported by the GetState_LowLayer and SetState_LowLayer methods.

Example

See also

IPCB_Via interface

ShapeOnLayer property

(IPCB_Via interface)

Syntax

```
Property ShapeOnLayer[L : TLayer] : TShape Read GetState_ShapeOnLayer;
```

Description

The via can have different shapes on layers that the via is connected to. This read only property is supported by the GetState_ShapeOnLayer method.

Example

See also

IPCB_Via interface

TShape type

TLayer type

Size property

(IPCB_Via interface)

Syntax

```
Property Size : TCoord Read GetState_Size Write SetState_Size;
```

Description

The Size property denotes the size of the via object (the full diamater of the via). This property is supported by the GetState_Size and SetState_Size methods.

Example

See also

IPCB_Via interface

SizeOnLayer property

(IPCB_Via interface)

Syntax

```
Property SizeOnLayer [L : TLayer] : TCoord Read GetState_SizeOnLayer;
```

Description

This property denotes the size of the via on a specified layer. This read only property is supported by the GetState_SizeOnLayer method.

Example

See also

IPCB_Via interface

StartLayer property

(IPCB_Via interface)

Syntax

```
Property StartLayer : IPCB_LayerObject Read GetState_StartLayer;
```

Description

This property fetches the start layer of IPCB_LayerObject type that the via is connected to.

This read only property is supported by the GetState_StartLayer method.

Example

See also

IPCB_Via interface

IPCB_LayerObject interface

StopLayer property

(IPCB_Via interface)

Syntax

```
Property StopLayer : IPCB_LayerObject Read GetState_StopLayer;
```

Description

This property fetches the last layer of IPCB_LayerObject type that the via is connected to.

This read only property is supported by the GetState_StopLayer method.

Example

See also

IPCB_Via interface

IPCB_LayerObject interface

X property

(IPCB_Via interface)

Syntax

```
Property X : TCoord Read GetState_XLocation Write SetState_XLocation;
```

Description

The X and Y properties define the location of the Via object with respect to the PCB document. This property is supported by the GetState_XLocation and SetState_XLocation methods.

Example

See also

IPCB_Via interface

Y property

(IPCB_Via interface)

Syntax

```
Property Y : TCoord Read GetState_YLocation Write SetState_YLocation;
```

Description

The X and Y properties define the location of the Via object with respect to the PCB document. This property is supported by the GetState_YLocation and SetState_YLocation methods.

Example

See also

IPCB_Via interface

IPCB_Violation Interface

Overview

A Violation object captures the rule that has been violated between two PCB objects that are affected by a binary design rule or a PCB object affected by a unary design rule detected in the PCB editor, with the description of the violation and the type of rule used.

A violation object has a name and its associated description properties, two primitive place holders for binary rules or the first primitive (Primitive1) for unary rules. Check if the second Primitive2 is valid before invoking its methods or properties.

The IPCB_Violation hierarchy;

IPCB_Primitive

IPCB_Violation

IPCB_Violation methods	IPCB_Violation properties
GetState_Name	Name
GetState_Rule	Rule
GetState_Primitive1	Primitive1
GetState_Primitive2	Primitive2
GetState_**Description**	**Description**
GetState_ShortDescriptorString	
IsRedundant	

See also

IPCB_Primitive interface

PCB Design Objects

Violations script in **Example**s\Scripts\DelphiScript\PCB folder.

GetState and SetState Methods

GetState_Description method

(IPCB_Violation interface)

Syntax

```
Function GetState_Description : TPCBString;
```

Description

This method returns the violation description that the violation object is associated with. This method is used for the **Description** property.

The corresponding **GetState_Name** method returns the name of this violation.

Example

```
If Violation <> Nil Then
    ShowMessage('Violation Name : ' + Violation.Name + #13#10 +
                'Description    : ' + Violation.Description);
```

See also

IPCB_Violation interface

GetState_Name method

(IPCB_Violation interface)

Syntax

```
Function GetState_Name : TPCBString;
```

Description

This method returns the violation name that the violation object is associated with. The method is used for the **Name** property.

The corresponding **GetState_Description** method returns the description of this violation.

Example

```
If Violation <> Nil Then
    ShowMessage('Violation Name : ' + Violation.Name + #13#10 +
                'Description    : ' + Violation.Description);
```

See also

IPCB_Violation interface

GetState_Primitive1 method

(IPCB_Violation interface)

Syntax

```
Function GetState_Primitive1 : IPCB_Primitive;
```

Description

A Violation object captures the rule that has been violated between two PCB objects that are affected by a binary design rule or a PCB object affected by a unary design rule detected in the PCB editor, with the description of the violation and the type of rule used.

A violation object that deals with unary rules only has a valid Primitive1 property.

The Primitive2 property is always void for unary rules.

Always check if the second property, Primitive2 is valid before invoking its methods or properties.

Example

See also

IPCB_Violation interface

GetState_Primitive2 method

(IPCB_Violation interface)

Syntax

```
Function GetState_Primitive2 : IPCB_Primitive;
```

Description

A Violation object captures the rule that has been violated between two PCB objects that are affected by a binary design rule or a PCB object affected by a unary design rule detected in the PCB editor, with the description of the violation and the type of rule used.

Note

A violation object that deals with unary rules only has a valid Primitive1 property thus the Primitive2 property is always void for unary rules.

Therefore always check if the second Primitive2 is valid before invoking its methods or properties.

Example

See also

IPCB_Violation interface

GetState_Rule method

(IPCB_Violation interface)

Syntax

```
Function GetState_Rule : IPCB_Primitive;
```

Description

A Violation object captures the rule that has been violated between two PCB objects that are affected by a binary design rule or a PCB object affected by a unary design rule detected in the PCB editor, with the description of the violation and the type of rule used.

However the **IPCB_Primitive** interface actually represents a **IPCB_Rule** ancestor object interface.

Example

See also

IPCB_Violation interface

GetState_ShortDescriptorString method

(IPCB_Violation interface)

Syntax

```
Function GetState_ShortDescriptorString : TPCBString;
```

Description

This method returns the shortened version of the description string.

Example

See also

IPCB_Violation interface

Methods

IsRedundant method

(IPCB_Violation interface)

Syntax

```
Function IsRedundant : Boolean;
```

Description

This method determines whether the object is redundant (unused object) on the PCB document or not.

Example

See also

IPCB_Violation interface

Properties

Rule property

(IPCB_Violation interface)

Syntax

```
Property Rule : IPCB_Primitive Read GetState_Rule;
```

Description

This Rule property returns a rule object encapsulated by the **IPCB_Primitive** interface. However the **IPCB_Primitive** interface actually represents a **IPCB_Rule** ancestor object interface.

Example

```
// Create an iterator to look for violation objects only.
Iterator := Board.BoardIterator_Create;
Iterator.AddFilter_ObjectSet(MkSet(eViolationObject));
Iterator.AddFilter_LayerSet(AllLayers);
Iterator.AddFilter_Method(eProcessAll);

// search for violations
Violation := Iterator.FirstPCBObject;
While Violation <> Nil Do
Begin
    S := 'Violation Name: ' + Violation.Name + '  +#13#10 +
         'Description: '    + Violation.Description);

    //Get design rule associated with the current violation object
    Rule := Violation.Rule;
    If Rule <> Nil Then
        ShowMessage(S + #13#10 + '  Rule Name: ' + Rule.Name);

    S := '';
    Violation := Iterator.NextPCBObject;
End;
Board.BoardIterator_Destroy(Iterator);
```

See also

IPCB_Violation interface

IPCB_Rule interface

Primitive1 property

(IPCB_Violation interface)

Syntax

```
Property Primitive1 : IPCB_Primitive Read GetState_Primitive1;
```

Description

A Violation object captures the rule that has been violated between two PCB objects that are affected by a binary design rule or a PCB object affected by a unary design rule detected in the PCB editor, with the description of the violation and the type of rule used.

A violation object that deals with unary rules only has a valid Primitive1 property.

Notes

The Primitive2 property is always void for unary rules, therefore check if the second Primitive2 is valid before invoking its methods or properties.

A read only property

Example

```
// First pcb object associated with a unary/binary design rule.
PCB1Object := Violation.Primitive1;

// Second pcb object associated with a binary design rule.
// however there are unary and binary rules, thus, for unary rules,
// there will only be one rule object in violation associated with the violation
PCB2Object := Violation.Primitive2;
If PCB2Object <> Nil Then
Begin
    // do what you want with the second object
End;
```

See also

IPCB_Violation interface

Primitive2 property

(IPCB_Violation interface)

Syntax

```
Property Primitive2 : IPCB_Primitive Read GetState_Primitive2;
```

Description

A Violation object captures the rule that has been violated between two PCB objects that are affected by a binary design rule or a PCB object affected by a unary design rule detected in the PCB editor, with the description of the violation and the type of rule used.

A violation object that deals with unary rules only has a valid Primitive1 property.

The Primitive2 property is always void for unary rules.

Check if the second Primitive2 is valid before invoking its methods or properties.

A read only property.

Example

```
// First pcb object associated with a unary/binary design rule.
PCB1Object := Violation.Primitive1;

// Second pcb object associated with a binary design rule.
// however there are unary and binary rules, thus, for unary rules,
// there will only be one rule object in violation associated with the violation
PCB2Object := Violation.Primitive2;
If PCB2Object <> Nil Then
Begin
    // do what you want with the second object
End;
```

See also

IPCB_Violation interface

Name property

(IPCB_Violation interface)

Syntax

```
Property Name : TPCBString Read GetState_Name;
```

Description

This property returns the violation name that the violation object is associated with. The corresponding **Description** property returns the description of this violation (if any).

This is a read only property.

Example

```
If Violation <> Nil Then
    ShowMessage('Violation Name : ' + Violation.Name + #13#10 +
                'Description    : ' + Violation.Description);
```

See also

IPCB_Violation interface

Description property

Description property

(IPCB_Violation interface)

Syntax

```
Property Description : TPCBString Read GetState_Description;
```

Description

This property returns the violation description that the violation object is associated with. The corresponding **Name** property returns the name of this violation. This property is supported by the **GetState_Description** method.

This is a read only property.

Example

```
If Violation <> Nil Then
    ShowMessage('Violation Name : ' + Violation.Name + #13#10 +
                'Description    : ' + Violation.Description);
```

See also

IPCB_Violation interface

Name property

IPCB_ContourPoint Interface

Overview

The **IPCB_ContourPoint** interface hierarchy is as follows;

IPCB_ContourPoint methods	IPCB_ContourPoint properties
GetState_X	X
SetState_X	Y
GetState_Y	
SetState_Y	

See also

Methods

SetState_Y method

(IPCB_ContourPoint interface)

Syntax

```
Procedure SetState_Y (AY : TCoord);
```

Description

Example

See also

IPCB_ContourPoint interface

SetState_X method

(IPCB_ContourPoint interface)

Syntax

```
Procedure SetState_X (AX : TCoord);
```

Description

Example

See also

IPCB_ContourPoint interface

GetState_Y method

(IPCB_ContourPoint interface)

Syntax

```
Function GetState_Y : TCoord;
```

Description

Example

See also

IPCB_ContourPoint interface

GetState_X method

(IPCB_ContourPoint interface)

Syntax

```
Function GetState_X : TCoord;
```

Description

Example

See also

IPCB_ContourPoint interface

Properties

X property

(IPCB_ContourPoint interface)

Syntax

```
Property X : TCoord Read GetState_X Write SetState_X;
```

Description

Example

See also

IPCB_ContourPoint interface

Y property

(IPCB_ContourPoint interface)

Syntax

```
Property Y : TCoord Read GetState_Y Write SetState_Y;
```

Description

Example

See also

IPCB_ContourPoint interface

IPCB_Contour Interface

Overview

The **IPCB_Contour** interface hierarchy is as follows;

IPCB_Contour methods	IPCB_Contour properties
GetState_Rotation	Rotation
SetState_Rotation	CX
GetState_CX	CY
SetState_CX	Points
GetState_CY	Count
SetState_CY	
GetState_Point	
GetState_Count	
Clear	
AddPoint	
InsertPoint	
AddContour	
AddArc	
GetGPCVertexList	
FillGPCVertexList	
I_ObjectAddress	

See also

Methods

Clear method

(IPCB_Contour interface)

Syntax

```
Procedure Clear;
```

Description

Example

See also

IPCB_Contour interface

AddPoint method

(IPCB_Contour interface)

Syntax

```
Procedure AddPoint(x, y : TCoord);
```

Description

Example

See also

IPCB_Contour interface

AddContour method

(IPCB_Contour interface)

Syntax

```
Procedure AddContour(Const C : IPCB_Contour; Const i1, i2 : Integer);
```

Description

Example

See also

IPCB_Contour interface

AddArc method

(IPCB_Contour interface)

Syntax

```
Procedure AddArc(StartAngle, EndAngle : Double; cx, cy : TCoord; Radius : TCoord;AClockwise :
Boolean = False);
```

Description

Example

See also

IPCB_Contour interface

InsertPoint method

(IPCB_Contour interface)

Syntax

```
Procedure InsertPoint(Index : Integer; x, y : TCoord);
```

Description

Example

See also

IPCB_Contour interface

I_ObjectAddress method

(IPCB_Contour interface)

Syntax

```
Function I_ObjectAddress : TPCBObjectHandle;
```

Description

This function returns the true pointer value of the object interface of a design object.

Notes

The IPCB_ServerInterface.SendMessageToRobots method needs the I_ObjectAddress parameter of a design object.

Example

```
//Notify PCB that the fill object is going to be changed.
PCBServer.SendMessageToRobots(
        Fill.I_ObjectAddress,
        c_Broadcast,
        PCBM_BeginModify ,
        c_NoEventData);
```

See also

IPCB_Contour interface

GetState_Point method

(IPCB_Contour interface)

Syntax

```
Function GetState_Point (I : Integer) : IPCB_ContourPoint;
```

Description

Example

See also

IPCB_Contour interface

GetState_Count method

(IPCB_Contour interface)

Syntax

```
Function GetState_Count : Integer;
```

Description

Example

See also

IPCB_Contour interface

GetGPCVertexList method

(IPCB_Contour interface)

Syntax

```
Procedure GetGPCVertexList (Const AContour : Pgpc_vertex_list);
```

Description

Example

See also

IPCB_Contour interface

FillGPCVertexList method

(IPCB_Contour interface)

Syntax

```
Procedure FillGPCVertexList(Const AContour : Pgpc_vertex_list);
```

Description

Example

See also

IPCB_Contour interface

SetState_Rotation method

(IPCB_Contour interface)

Syntax

```
Procedure SetState_Rotation (ARotation : TAngle);
```

Description

Example

See also

IPCB_Contour interface

SetState_CY method

(IPCB_Contour interface)

Syntax

```
Procedure SetState_CY (ACY : TCoord);
```

Description

Example

See also

IPCB_Contour interface

SetState_CX method

(IPCB_Contour interface)

Syntax

```
Procedure SetState_CX (ACX : TCoord);
```

Description

Example

See also

IPCB_Contour interface

GetState_Rotation method

(IPCB_Contour interface)

Syntax

```
Function GetState_Rotation : TAngle;
```

Description

Example

See also

IPCB_Contour interface

GetState_CY method

(IPCB_Contour interface)

Syntax

```
Function GetState_CY : TCoord;
```

Description

Example

See also

IPCB_Contour interface

GetState_CX method

(IPCB_Contour interface)

Syntax

```
Function GetState_CX : TCoord;
```

Description

Example

See also

IPCB_Contour interface

Properties

Rotation property

(IPCB_Contour interface)

Syntax

```
Property Rotation : TAngle Read GetState_Rotation Write SetState_Rotation;
```

Description

Example

See also

IPCB_Contour interface

Points property

(IPCB_Contour interface)

Syntax

```
Property Points[I : Integer] : IPCB_ContourPoint Read GetState_Point;
```

Description

Example

See also

IPCB_Contour interface

CY property

(IPCB_Contour interface)

Syntax

```
Property CY : TCoord Read GetState_CY Write SetState_CY;
```

Description

Example

See also

IPCB_Contour interface

CX property

(IPCB_Contour interface)

Syntax

```
Property CX : TCoord Read GetState_CX Write SetState_CX;
```

Description

Example

See also

IPCB_Contour interface

Count property

(IPCB_Contour interface)

Syntax

```
Property Count : Integer Read GetState_Count;
```

Description

Example

See also

IPCB_Contour interface

IPCB_ContourMaker Interface

Overview

IPCB_ContourMaker methods	IPCB_ContourMaker properties
MakeContour	
DestroyPolygon	

See also

IPCB_Contour interface

Methods

IPCB_MakeContour method

(IPCB_ContourMaker interface)

Syntax

```
Function MakeContour(APrim   : IPCB_Primitive; AExpansion : TCoord; ALayer : TLayer) :
Pgpc_Polygon;
Function MakeContour(ATrack  : IPCB_Track    ; AExpansion : TCoord; ALayer : TLayer) :
Pgpc_Polygon;
Function MakeContour(APad    : IPCB_Pad      ; AExpansion : TCoord; ALayer : TLayer) :
Pgpc_Polygon;
Function MakeContour(AFill   : IPCB_Fill     ; AExpansion : TCoord; ALayer : TLayer) :
Pgpc_Polygon;
Function MakeContour(AVia    : IPCB_Via      ; AExpansion : TCoord; ALayer : TLayer) :
Pgpc_Polygon;
Function MakeContour(AArc    : IPCB_Arc      ; AExpansion : TCoord; ALayer : TLayer) :
Pgpc_Polygon;
Function MakeContour(ARegion : IPCB_Region   ; AExpansion : TCoord; ALayer : TLayer) :
Pgpc_Polygon;
Function MakeContour(AText   : IPCB_Text     ; AExpansion : TCoord; ALayer : TLayer) :
Pgpc_Polygon;
Function MakeContour(APoly   : IPCB_Polygon  ; AExpansion : TCoord; ALayer : TLayer) :
Pgpc_Polygon;
```

Description

Example

See also

IPCB_ContourMaker interface

Dimension Object Interfaces

IPCB_OriginalDimension

Overview

The IPCB_OriginalDimension interface represents the dimensioning information on the current PCB layer. The dimension value is the distance between the start and end markers, measured in the default units. Note that the original dimension object has been superseded by a new set of dimension objects

Notes

The IPCB_OriginalDimension interface hierarchy is as follows;

IPCB_Primitive

IPCB_Group

IPCB_Dimension

IPCB_OriginalDimension

IPCB_OriginalDimension Methods

```
Function  Text    : IPCB_Text;
Function  Track1  : IPCB_Primitive;
Function  Track2  : IPCB_Primitive;
Function  Track3  : IPCB_Primitive;
Function  Track4  : IPCB_Primitive;
Function  Track5  : IPCB_Primitive;
Function  Track6  : IPCB_Primitive;
Function  Track7  : IPCB_Primitive;
Function  Track8  : IPCB_Primitive;
```

See also

IPCB_Dimension interface

PCB Design Objects

IPCB_Dimension

Overview

Dimension objects are used for dimensional details of a PCB board in either imperial or metric units and can be placed on any layer. To create an original Dimension objects, use the IPCB_OriginalDimension class which is used in P99SE and earlier versions.

Altium Designer introduced several new dimension styles - Linear, Angular, Radial, Leader, Datum, Baseline, Center, Linear Diameter and Radial Diameter objects

Notes

The IPCB_Dimension interface is the ancestor interface for IPCB_OriginalDimension, IPCB_LinearDImension, IPCB_AngularDimension, IPCB_RadialDimension, IPCB_LeaderDimension, IPCB_DatumDimension, IPCB_BaselineDimension, IPCB_CenterDimension, IPCB_LinearDiameterDimension, IPCB_RadialDiameterDimension interfaces.

The DimensionKind property determines the type a dimension object is.

A dimension object especially a baseline or a leader dimension has multiple reference points. The references (a reference consists of a record of an object along with its x and y coordinate point, an anchor and is a start or end marker). A reference point is either the start or end marker and the length of two reference points is the dimensional length.

IPCB_Group methods	IPCB_Group properties
FreePrimitives	X
GetPrimitiveAt	Y
GetPrimitiveCount	PrimitiveLock
SetState_XSizeYSize	LayerUsed
FastSetState_XSizeYSize	
SetState_LayersUsedArray	
GroupIterator_Create	
GroupIterator_Destroy	
AddPCBObject	
RemovePCBObject	

IPCB_Dimension Methods

```
Procedure MoveTextByXY (AX,
                        AY    : TCoord);
Procedure MoveTextToXY (AX,
```

```
                        AY    : TCoord);
Procedure RotateAroundXY(AX,
                         AY    : TCoord;
                         Angle : TAngle);
Procedure References_Add(R : TDimensionReference);
Procedure References_Delete(Index : Integer);
Procedure References_DeleteLast;
Function  References_IndexOf(P     : IPCB_Primitive;
                             Index : Integer) : Integer;
Function  References_Validate : Boolean;
Procedure ResetPrefixIfNeeded;
```

IPCB_Dimension Properties

```
DimensionKind      : TDimensionKind
TextX              : TCoord
TextY              : TCoord
X1Location         : TCoord
Y1Location         : TCoord
Size               : TCoord
LineWidth          : TCoord
TextHeight         : TCoord
TextWidth          : TCoord
TextFont           : TFontID
TextLineWidth      : TCoord
TextPosition       : TDimensionTextPosition
TextGap            : TCoord
TextFormat         : TPCBString
TextDimensionUnit  : TDimensionUnit
TextPrecision      : Integer
TextPrefix         : TPCBString
TextSuffix         : TPCBString
TextValue          : TReal
ArrowSize          : TCoord
ArrowLineWidth     : TCoord
ArrowLength        : TCoord
ArrowPosition      : TDimensionArrowPosition
ExtensionOffset    : TCoord
ExtensionLineWidth : TCoord
ExtensionPickGap   : TCoord
Style              : TUnitStyle
References [I : Integer] : TDimensionReference
References_Count         : Integer // Read only
UseTTFonts                        : Boolean
Bold                              : Boolean
Italic                            : Boolean
FontName                          : TPCBString
```

See also

IPCB_Primitive interface

TDimensionTextPosition enumerated values

TDimensionUnit enumerated values

TDimensionArrowPosition enumerated values

TDimensionReference enumerated values

TUnitStyle enumerated values

PCB Design Objects

IPCB_AngularDimension

Overview

The IPCB_AngularDimension object interface allows for the dimensioning of angular distances. There are four references (two reference points associated with two reference objects) which need to be defined and the dimension text is then placed. The references may be tracks, fills, or polygons.

Notes

The IPCB_AngularDimension interface hierarchy is as follows;

IPCB_Primitive

IPCB_Group

IPCB_Dimension

IPCB_AngularDimension

The Radius property denotes the radius size of the IPCB_AngularDimension object.

The Sector property denotes which sector the IPCB_AngularDimension is using. Sector 1 is the angle between 0 – 90 degrees. 2 = 90 – 180 degrees. 3 = 180 =270 degrees. 4 = 270 = 360 or 0 degrees.

IPCB_AngularDimension Methods

```
Function  Text              : IPCB_Text;
Function  Arc1              : IPCB_Arc;
Function  Arc2              : IPCB_Arc;
Function  Arrow1_Track1     : IPCB_Track;
Function  Arrow1_Track2     : IPCB_Track;
Function  Arrow2_Track1     : IPCB_Track;
Function  Arrow2_Track2     : IPCB_Track;
Function  Extension1_Track  : IPCB_Track;
Function  Extension2_Track  : IPCB_Track;
```

IPCB_AngularDimension Properties

```
Property Radius  : TCoord
Property Sector  : Integer
```

See also

IPCB_Dimension interface

IPCB_Track interface

IPCB_Text interface

IPCB_Arc interface

PCB Design Objects

IPCB_BaselineDimension

Overview

The IPCB_BaselineDimension interface allows for the dimensioning of a linear distance of a collection of references, relative to a single reference. The first reference point is the base reference and all the subsequent points are relative to this base reference. The dimension value in each case is the distance between each reference point and the base reference measured in default units. The references may be objects (tracks, arcs, pads, vias, text, fills, polygons or components) or points in free space

Notes

The IPCB_BaselineDimension interface hierarchy is as follows;

IPCB_Primitive

IPCB_Group

IPCB_Dimension

IPCB_BaselineDimension

The angle property denotes the angle or rotation of the IPCB_BaselineDimension object with respect to the horizontal plane.

Since a baseline dimension allows for the dimensioning of a linear distance over a collection of references, thus for each reference relative to the base reference, there is a text location. Use the TextLocationsCount field to obtain the number of dimension labels.

IPCB_Gro**up methods**	IPCB_Group properties
FreePrimitives	X
GetPrimitiveAt	Y
GetPrimitiveCount	PrimitiveLock
SetState_XSizeYSize	LayerUsed
FastSetState_XSizeYSize	
SetState_LayersUsedArray	
GroupIterator_Create	
GroupIterator_Destroy	
AddPCBObject	
RemovePCBObject	

IPCB_BaselineDimension Methods

```
Function  Text       : IPCB_Text;
Function  Texts        (I : Integer) : IPCB_Text;
Function  Arrow1_Track1(I : Integer) : IPCB_Track;
Function  Arrow1_Track2(I : Integer) : IPCB_Track;
Function  Arrow2_Track1(I : Integer) : IPCB_Track;
Function  Arrow2_Track2(I : Integer) : IPCB_Track;
Function  Line_Track1  (I : Integer) : IPCB_Track;
Function  Line_Track2  (I : Integer) : IPCB_Track;
Function  Extension1_Track (I : Integer) : IPCB_Track;
Function  Extension2_Track (I : Integer) : IPCB_Track;
Procedure TextLocations_Add   (Point : TCoordPoint);:
Procedure TextLocations_Delete(Index : Integer);
Procedure TextLocations_DeleteLast;
Procedure TextLocations_Clear;
```

IPCB_BaselineDimension Properties

```
Property Angle                       : TAngle
Property TextLocations [I : Integer] : TCoordPoint
Property TextLocationsCount          : Integer
```

See also

IPCB_Dimension interface

IPCB_Track interface

IPCB_Text interface

PCB Design Objects

IPCB_CenterDimension

Overview

The IPCB_CenterDimension object interface allows for the center of an arc or circle to be marked

Notes

The IPCB_CenterDimension interface hierarchy is as follows;

IPCB_Primitive

IPCB_Group

IPCB_Dimension

IPCB_CenterDimension

The angle property denotes the angle or rotation of the IPCB_CenterDimension object with respect to the horizontal plane.

IPCB_CenterDimension Methods

```
Function  Cross_Vertical_Track   : IPCB_Track;
Function  Cross_Horizontal_Track : IPCB_Track;
```

IPCB_CenterDimension Properties

```
Property Angle : TAngle
```

See also

IPCB_Dimension interface

IPCB_Track interface

PCB Design Objects

IPCB_DatumDimension

Overview

The IPCB_DatumDimension interface references the dimensioning of a linear distance of a collection of objects, relative to a single object. The dimension value is the distance between each reference object and the base object measured in the default units. The references may be tracks, arcs, pads, vias, text, fills, polygons or components.

Notes

The IPCB_DatumDimension interface hierarchy is as follows;

IPCB_Primitive

IPCB_Group

IPCB_Dimension

IPCB_DatumDimension

IPCB_DatumDimension Methods

```
Function  Text                              : IPCB_Text;
Function  Texts           (I     : Integer) : IPCB_Text;
Function  Extension_Track (I     : Integer) : IPCB_Track;
```

IPCB_DatumDimension Properties

```
Property Angle : TAngle
```

See also

IPCB_Dimension interface

IPCB_Track interface

IPCB_Text interface

PCB Design Objects

IPCB_LeaderDimension

Overview

The IPCB_LeaderDimension object interface allows for the labeling of an object, point or area. There are three types of leader dimensions available which reflect the label text either being encapsulated by a circle or square or not at all. The pointer can also be an arrow or a dot which is size -definable.

Notes

The IPCB_LeaderDimension interface hierarchy is as follows;

IPCB_Primitive

IPCB_Group

IPCB_Dimension

IPCB_LeaderDimension

There are three types of leaders available: eNoShape = standard leader which means the dimension text is not enclosed at all. eRectangular the label text is encapsulated by a square, and eRounded – the dimension text is encapsulated by a circle.

The Dot property denotes the dot symbol attached to the pointer of the leader dimension object as a dot or as an arrow.

If the Dot field is enabled, then you can specify the size of the dot as a TCoord value.

IPCB_LeaderDimension Methods

```
Function  Text            : IPCB_Text;
Function  Dot_Arc         : IPCB_Arc;
Function  Circle_Arc      : IPCB_Arc;
Function  Arrow_Track1    : IPCB_Track;
Function  Arrow_Track2    : IPCB_Track;
Function  Square_Track1   : IPCB_Track;
Function  Square_Track2   : IPCB_Track;
Function  Square_Track3   : IPCB_Track;
Function  Square_Track4   : IPCB_Track;
Function  Line_Track (I : Integer) : IPCB_Track;
```

IPCB_LeaderDimension Properties

```
Property Shape    : TShape
Property Dot      : Boolean
Property DotSize  : TCoord
```

See also

IPCB_Dimension interface

IPCB_Track interface

IPCB_Text interface

IPCB_Arc interface

PCB Design Objects

IPCB_LinearDiameterDimension

Overview

The IPCB_LinearDimension interface references the dimensioning information on the current layer with respect to a linear distance. The dimension value is the distance between the start and end markers (reference points) measured in the default units. The references may be objects (tracks, arcs, pads, vias, text fills, polygons or components) or points in free space.

Notes

The IPCB_LinearDiameterDimension interface hierarchy is as follows;

IPCB_Primitive

IPCB_Group

IPCB_Dimension

IPCB_LinearDiameterDimension

Immediate ancestor IPCB_LinearDimension Methods

```
Function  Text              : IPCB_Text;
Function  Arrow1_Track1     : IPCB_Track;
Function  Arrow1_Track2     : IPCB_Track;
Function  Arrow2_Track1     : IPCB_Track;
Function  Arrow2_Track2     : IPCB_Track;
Function  Line_Track1       : IPCB_Track;
Function  Line_Track2       : IPCB_Track;
Function  Extension1_Track  : IPCB_Track;
```

```
Function  Extension2_Track   : IPCB_Track;
```

Immediate ancestor IPCB_LinearDimension Properties

```
Property Angle : TAngle
```

See also

IPCB_Dimension interface

IPCB_Track interface

PCB Design Objects

IPCB_LinearDimension

Overview

The IPCB_LinearDimension object interface places dimensioning information on the current layer with respect to a linear distance. The dimension value is the distance between the start and end markers (reference points) measured in the default units. The references may be objects (tracks, arcs, pads, vias, text fills, polygons or components) or points in free space.

IPCB_LinearDimension object interface has no introduced methods and properties, therefore refer to the IPCB_Dimension interface object entry for details.

Notes

The IPCB_LinearDimension interface hierarchy is as follows;

IPCB_Primitive

IPCB_Group

IPCB_Dimension

IPCB_LinearDimension

The angle property denotes the angle or rotation of the TPCBLinearDimension object with respect to the horizontal plane.

IPCB_LinearDimension Methods

```
Function  Text             : IPCB_Text;
Function  Arrow1_Track1    : IPCB_Track;
Function  Arrow1_Track2    : IPCB_Track;
Function  Arrow2_Track1    : IPCB_Track;
Function  Arrow2_Track2    : IPCB_Track;
Function  Line_Track1      : IPCB_Track;
Function  Line_Track2      : IPCB_Track;
Function  Extension1_Track : IPCB_Track;
Function  Extension2_Track : IPCB_Track;
```

IPCB_LinearDimension Properties

```
Property Angle : TAngle
```

See also

IPCB_Dimension interface

PCB Design Objects

IPCB_RadialDimension

Overview

The IPCB_RadialDimension object interface allows for the dimensioning of a radius with respect to an arc or a circle. The dimension can be placed internally or externally on an arc or a circle.

Notes

The IPCB_RadialDimension interface hierarchy is as follows;

IPCB_Primitive

IPCB_Group

IPCB_Dimension

IPCB_RadialDimension

This field shows the current angular step setting for the dimension. This is the rotation step used when placing the arrow portion of the dimension. Moving the arrow around the circle or arc during placement of the dimension, the number and position of possible places to anchor the dimension are determined by this angular step value.

IPCB_RadialDimension Methods

```
Function  Text         : IPCB_Text;
Function  Arrow_Track1 : IPCB_Track;
Function  Arrow_Track2 : IPCB_Track;
Function  Line1_Track  : IPCB_Track;
Function  Line2_Track  : IPCB_Track;
```

IPCB_RadialDimension Property

```
Property AngleStep : TAngle
```

See also

IPCB_Dimension interface

IPCB_Track interface

IPCB_Text interface

PCB Design Objects

IPCB_RadialDiameterDimension

Overview

The IPCB_RadialDiameterDimension interface references the dimensioning of an arc or circle with respect to the diameter, rather than the radius. The dimension can be placed either internally or externally with respect to the arc or circle

Notes

The IPCB_RadialDiameterDimension interface hierarchy is as follows;

IPCB_Primitive

IPCB_Group

IPCB_Dimension

IPCB_RadialDiameterDimension

IPCB_RadialDiameterDimension Methods

```
Function  Arrow2_Track1  : IPCB_Track;
Function  Arrow2_Track2  : IPCB_Track;
Function  Line3_Track    : IPCB_Track;
```

See also

IPCB_Dimension interface

IPCB_Track interface

PCB Design Objects

PCB Rule Objects Interfaces

The PCB editor incorporates a large set of design rules to help define compliance/constraints regarding the placement of PCB objects, routing methods, and netlists.

These rules include clearances, object geometry, impedance control, routing priority, routing topology and parallelism. Rule scope is the extent of each rule determined. The scope allows you to define the set of target objects that a particular instance of a rule is to be applied to.

See also

Rule ancestor interface

Acute Angle rule interface

Broken Nets rule interface

Clearance rule interface

Confinement Constraint rule interface
Component Clearance rule interface
Component Rotations rule interface
Daisy Chain Stub Length rule interface
Differential Pairs Routing rule interface
Fanout Control rule interface
Layer Pair rule interface
Layer Stack rule interface
Matched Lengths rule interface
Max Min Width rule interface
Max Min Length rule interface
Max Min Hole Size rule interface
Maximum Via Count rule interface
Minimum Annular Ring rule interface
NetsToIgnore rule interface
Parallel Segment rule interface
Paste Mask Expansion rule interface
Power Plane Connect Style rule interface
Power Plane Clearance rule interface
Polygon Connect Style rule interface
Permitted Layers rule interface
Routing Topology rule interface
Routing Priority rule interface
Routing Layers rule interface
Routing Corner Style rule interface
Routing Via Style rule interface
SMD To Plane rule interface
SMD Neck Down rule interface
SMD To Corner rule interface
Solder Mask Expansion rule interface
Short Circuit rule interface
Test Point Style rule interface
Test Point Usage rule interface
Vias Under SMD rule interface
Unconnected Pin rule interface

Signal Integrity Rules
FlightTime_RisingEdge rule interface
FlightTime_FallingEdge rule interface
MaxMinImpedance rule interface
MaxSlope_RisingEdge rule interface
MaxSlope_FallingEdge rule interface
Overshoot_FallingEdge rule interface
Overshoot_RisingEdge rule interface
SignalTopValue rule interface
SignalBaseValue rule interface

SignalStimulus rule interface

SupplyNets rule interface

Undershoot_FallingEdge rule interface

Undershoot_RisingEdge rule interface

IPCB_Rule

Overview

The IPCB_Rule interface object encapsulates an existing PCB design rule in an opened PCB document in Altium Designer. Each design rule has its own Unique ID. To set the scope of a rule, unary or binary scope expressions are defined.

The PCB editor incorporates a large set of design rules to help define compliance/constraints regarding the placement of PCB objects, routing methods, and netlists. These rules include clearances, object geometry, impedance control, routing priority, routing topology and parallelism. Rule scope is the extent of each rule determined. The scope allows you to define the set of target objects that a particular instance of a rule is to be applied to.

IPCB_Rule Methods

```
Function  Priority                  : TRulePrecedence;
Function  ScopeKindIsValid  (AScopeKind  : TScopeKind)          : Boolean;
Function  Scope1Includes    (P           : IPCB_Primitive)      : Boolean;
Function  Scope2Includes    (P           : IPCB_Primitive)      : Boolean;
Function  NetScopeMatches   (P1,
                             P2          : IPCB_Primitive)      : Boolean;
Function  CheckBinaryScope  (P1,
                             P2          : IPCB_Primitive)      : Boolean;
Function  CheckUnaryScope   (P           : IPCB_Primitive)      : Boolean;
Function  GetState_DataSummaryString     : TPCBString;
Function  GetState_ShortDescriptorString : TPCBString;
Function  GetState_ScopeDescriptorString : TPCBString;
Function  ActualCheck               (P1,
                                     P2  : IPCB_Primitive)      : IPCB_Violation;
```

IPCB_Rule Properties

```
Property Scope1Expression : TPCBString
Property Scope2Expression : TPCBString
Property RuleKind         : TRuleKind
Property NetScope         : TNetScope
Property LayerKind        : TRuleLayerKind
Property Comment          : TPCBString
Property Name             : TPCBString
Property DRCEnabled       : Boolean
Property UniqueId         : TPCBString       //Read only
```

Enumerated Types

PCB Design Rules

IPCB_Violation interface

TScopeKind

TNetScope

TRuleKind

TRuleLayerKind

IPCB_AcuteAngle rule

Overview

The IPCB_AcuteAngleRule interface specifies the minimum angle permitted at a track corner.

IPCB_AcuteAngle Properties

```
Minimum : TAngle
```

IPCB_BrokenNetRule rule

Overview

The IPCB_BrokenNetRule rule deals with broken nets in relation to polygons. Polygons that are affected by the broken net rules are highlighted or not.

IPCB_BrokenNetRule Properties

```
HighlightPolygons : Boolean
```

IPCB_ComponentClearanceConstraint rule

Overview

The Component Clearance Constraint PCB Design rule has available Check Mode setting:

Quick Check – uses a components' bounding rectangle to define its shape. The bounding rectangle is the smallest rectangle that encloses all the primitives that make up a component.

Multi Layer Check – also uses a component bounding rectangle, but considers through-hole component pads on a board with components on both sides, allowing surface mount components to be placed under a through-hole component.

Full Check – uses the exact shape that encloses all the primitives that make up each component. Use this option if the design includes a large number of circular or irregular shaped components.

IPCB_ComponentClearanceConstraint Properties

```
Property Gap                : TCoord
Property VerticalGap        : TCoord
Property CollisionCheckMode : TComponentCollisionCheckMode
```

See also

TComponentCollisionCheckMode

IPCB_ComponentRotationsRule rule

Overview

The IPCB_ComponentRotationsRule specifies allowable component orientations. Multiple orientations are permitted, allowing the autoplacer to use any of the enabled orientations. The allowed component orientations are: 0,90,180, 270, or AllRotations. It is possible to have multiple settings, for example setting at 0 and 270 degrees rotations only.

IPCB_ComponentRotationsRule Properties

```
Property AllowedRotations : Integer
```

IPCB_ConfinementConstraint rule

Overview

The IPCB_ConfinementConstraint interface specifies a rectangular region in which a set of objects is either allowed, or not allowed. Use this function to define a region that a class of components must be placed in.

IPCB_ConfinementConstraint Methods

```
Procedure RotateAroundXY (AX,
                          AY    : TCoord;
                          Angle : TAngle);
```

IPCB_ConfinementConstraint Properties

```
Property X            : TCoord
Property Y            : TCoord
Property Kind         : TConfinementStyle
Property Layer        : TLayer
Property BoundingRect : TCoordRect
```

IPCB_ClearanceConstraint Rule

Overview

This interface defines the minimum clearance between any two primitive objects on a copper layer.

Notes

The PrimitivesViolate function checks if two primitives violate the minimum clearance or not.

The Gap property determines the gap size of the track segments.

IPCB_ClearanceConstraint Methods

```
Function  PrimitivesViolate(P1, P2  : IPCB_Primitive) : Boolean;
```

IPCB_ClearanceConstraint Properties

```
Property Gap  : TCoord
```

IPCB_DaisyChainStubLengthConstraint rule

Overview

The daisy chain stub length rule specifies the maximum permissible stub length for a net with a daisy chain topology.

Notes

Limit property for the stub length.

IPCB_DaisyChainStubLengthConstraint Properties

```
Property Limit : TCoord
```

IPCB_ DifferentialPairsRoutingRule Interface

Overview

A differential signaling system is one where a signal is transmitted down a pair of tightly coupled carriers, one of these carrying the signal, the other carrying an equal but opposite image of the signal. Differential signaling was developed to cater for situations where the logic reference ground of the signal source could not be well connected to the logic reference ground of the load. Differential signaling is inherently immune to common mode electrical noise, the most common interference artifact present in an electronic product. Another major advantage of differential signaling is that it minimizes electromagnetic interference (EMI) generated from the signal pair.

Differential pair routing is a design technique employed to create a balanced transmission system able to carry differential (equal and opposite) signals across a printed circuit board. Typically this differential routing will interface to an external differential transmission system, such as a connector and cable.

It is important to note that while the coupling ratio achieved in a twisted pair differential cable may be better than 99%, the coupling achieved in differential pair routing will typically be less than 50%. Current expert opinion is that the PCB routing task is not to try to ensure a specific *differential impedance* is achieved, rather the objective is to maintain the properties required to ensure the differential signal arrives in good condition at the target component as it travels from the external cabling.

Notes

The IPCB_DifferentialPairsRoutingRule Interface hierarchy is as follows;

IPCB_Rule

IPCB_DifferentialPairsRoutingRule

This interface defines the minimum clearance between any two primitive objects on a copper layer.

Notes

The PrimitivesViolate function checks if two primitives violate the minimum clearance or not.

The Gap property determines the gap size of the track segments.

IPCB_DifferentialPairsRoutingRule Methods

```
Function  GetState_MaxGap            (Const L : TLayer) : TCoord;
Function  GetState_MinGap            (Const L : TLayer) : TCoord;
Function  GetState_PreferedGap       (Const L : TLayer) : TCoord;
Function  GetState_MaxUncoupledLength : TCoord;

Procedure SetState_MaxGap            (Const L : TLayer;
```

```
                                        Value   : TCoord);
Procedure SetState_MinGap              (Const L : TLayer;
                                        Value   : TCoord);
Procedure SetState_PreferedGap         (Const L : TLayer;
                                        Value   : TCoord);
Procedure SetState_MaxUncoupledLength(Value   : TCoord);
```

IPCB_DifferentialPairsRoutingRule Properties

```
Property  MaxGap     [Const L : TLayer]  : TCoord  Read GetState_MaxGap Write SetState_MaxGap;
Property  MinGap     [Const L : TLayer]  : TCoord  Read GetState_MinGap Write SetState_MinGap;
Property  PreferedGap[Const L : TLayer]  : TCoord  Read GetState_PreferedGap Write
SetState_PreferedGap;
Property  MaxUncoupledLength             : TCoord  Read GetState_MaxUncoupledLength Write
SetState_MaxUncoupledLength;
```

See also

PCB Design Objects

IPCB_FanoutControlRule rule

Overview

The IPCB_FanoutControl rule determines how BGAs on a PCB document is going to be fanned in respect to vias placement for routing.

IPCB_FanoutControlRule Properties

```
Property FanoutStyle        : TFanoutStyle
Property FanoutDirection    : TFanoutDirection
Property BGAFanoutDirection : TBGAFanoutDirection
Property BGAFanoutViaMode   : TBGAFanoutViaMode
Property ViaGrid            : TCoord
```

IPCB_LayerPairsRule rule

Overview

The IPCB_LayerPairsRule interface deals with whether the layer pairs are going to be enforced or not on the current PCB document.

IPCB_LayerPairsRule Properties

```
Property EnforceLayerPairs : Boolean
```

IPCB_MatchedNetLengthsConstraint rule

Overview

The matched net lengths rule specifies the degree to which nets can have different lengths.

Notes

The 90 degree style is the most compact and the Rounded style is the least compact.

IPCB_MatchedNetLengthsConstraint Methods

```
Function  MatchLengthForFromTo(P1,P2 : IPCB_Primitive) : IPCB_Violation;
Function  MatchLengthForNet   (P1,P2 : IPCB_Primitive) : IPCB_Violation;
```

IPCB_MatchedNetLengthsConstraint Properties

```
Property Amplitude : TCoord
Property Gap       : TCoord
Property Style     : TLengthenerStyle
Property Tolerance : TCoord
```

IPCB_MaxMinHeightConstraint rule

Overview

The IPCB_MaxMinHeightConstraint rule deals with heights of components, and you can set the maximum, minimum and preferred height values for targeted components on a PCB document.

Notes

MaxHeight, MinHeight and PreferedHeight properties.

IPCB_MaxMinHeightConstraint Properties

```
Property MaxHeight      : TCoord
Property MinHeight      : TCoord
Property PreferedHeight : TCoord
```

IPCB_MaxMinHoleSizeConstraint rule

Overview

The IPCB_MaxMinHoleSizeContraint rule deals with the constraints of hole sizes on a PCB document.

IPCB_MaxMinHoleSizeConstraint Properties

```
Property AbsoluteValues : Boolean
Property MaxLimit       : TCoord
Property MinLimit       : TCoord
Property MaxPercent     : TReal
Property MinPercent     : TReal
```

IPCB_MaxMinWidthConstraint rule

Overview

This routing width constraint interface defines the minimum, favored and maximum width of tracks and arcs on copper layers.

IPCB_MaxMinWidth Properties

```
Property  MaxWidth    [Const L : TLayer]  : TCoord
Property  MinWidth    [Const L : TLayer]  : TCoord
Property  FavoredWidth[Const L : TLayer]  : TCoord
Property  ImpedanceDriven                 : Boolean
Property  MinImpedance                    : TDouble
Property  FavoredImpedance                : TDouble
Property  MaxImpedance                    : TDouble
```

IPCB_MaxMinLengthConstraint rule

Overview

This IPCB_MaxMinLengthConstraint rule defines the minimum and maximum lengths of a net.

IPCB_MaxMinLengthConstraint Properties

```
Property MaxLimit : TCoord
Property MinLimit : TCoord
```

IPCB_MinimumAnnularRing rule

Overview

The minimum annular ring rule determines the minimum size of an annular ring.

IPCB_MinimumAnnularRing Properties

```
Property Minimum : TCoord
```

IPCB_MaximumViaCountRule rule

Overview

The maximum via count rule specifies the maximum number of vias permitted on a PCB document.

Notes

Set or return the maximum number of vias for the Limit property

IPCB_MaximumViaCount Properties

```
Property Limit : Integer
```

IPCB_NetsToIgnoreRule rule

Overview

The Nets To Ignore rule determines which nets to ignore during Design Rule Check.

IPCB_NetsToIgnoreRule Methods

No new interface methods

IPCB_NetsToIgnoreRule Properties

No new interface properties

See also

IPCB_Rule interface

IPCB_ParallelSegmentConstraint rule

Overview

This rule specifies the distance two track segments can run in parallel, for a given separation. Note that this rule tests track segments, not collections of track segments. Apply multiple parallel segment constraints to a net to approximate crosstalk characteristics that vary as a function of length and gap.

Notes

The Gap and Limit properties concern the track segments.

IPCB_ParallelSegmentConstraint Properties

```
Property Gap   : TCoord
Property Limit : TCoord
```

IPCB_PasteMaskExpansionRule rule

Overview

The IPCB_PasteMaskExpansionRule function returns or sets values for a paste mask expansion rule object. The Paste Mask Expansion Rule specifies the amount of radial expansion or radial contraction of each pad site.

Notes

The Expansion property sets or returns the radial expansion or contraction value (a negative value denotes contraction).

IPCB_PasteMaskExpansionRule Properties

```
Property Expansion : TCoord
```

IPCB_PermittedLayersRule rule

Overview

The IPCB_PermittedLayersRule function returns or sets the permitted layers rule which specifies the layers components can be placed on during placement with the Cluster Placer. The Cluster Placer does not change the layer a component is on, you must set the component layer prior to running the placer.

IPCB_PermittedLayersRule Properties

```
Property PermittedLayers : TLayerSet
```

IPCB_PowerPlaneClearanceRule rule

Overview

The power plane clearance rule determines the clearance of the power plane.

IPCB_PowerPlaneClearanceRule Properties

```
Property Clearance : TCoord
```

IPCB_PowerPlaneConnectStyleRule rule

Overview

This power plane connect style rule specifies the style of the connection from a component pin to a power plane. There are two connection types - direct connections (the pin to solid copper) or thermal relief connection.

Notes

The **TPlaneConnectStyle** type determines the connection style for a plane. If Thermal Relief connection is used, then the thermal relief conductor width, the relief expansion, the width of the air gap and the number of relief entries need to be determined. If direct connection style is used, then the previous parameters are not needed.

IPCB_PowerPlaneConnectStyleRule Properties

```
Property PlaneConnectStyle    : TPlaneConnectStyle
Property ReliefExpansion      : TCoord
Property ReliefConductorWidth : TCoord
Property ReliefEntries        : Integer
Property ReliefAirGap         : TCoord
```

IPCB_PolygonConnectStyleRule rule

Overview

The Polygon Connect Style Rule returns or sets the polygon connect style rule which specifies how the polygon is connected to the power plane.

Notes

• The **TPlaneConnectStyle** type specifies the polygon connect style rule which is relief connection to a polygon, or direct connection to a polygon from a component pin. That is, the type of connection from a component pin to the polygon.

• The relief conductor width property denotes the width of the conductor between two air gaps.

• The relief entries property specifies the number of relief entries (2 or 4) for the relief connection of the polygon connection. For other types of connection, this field is irrelevant.

• The PolygonReliefAngle type specifies the angle of relief connections in 45 or 90 degrees.

IPCB_PolygonConnectStyleRule Properties

```
Property ConnectStyle         : TPlaneConnectStyle
Property ReliefConductorWidth : TCoord
Property ReliefEntries        : Integer
Property PolygonReliefAngle   : TPolygonReliefAngle
```

IPCB_RoutingCornerStyleRule

Overview

This routing corners rule specifies the corner style to be used during autorouting a PCB document.

Notes

• The TCornerStyle type sets or returns the corner style which can be a 45 degree camfer or rounded using an arc.

• The minsetback and maxsetback properties specify the minimum and maximum distance from the corner location to the start of the corner chamfer or arc.

IPCB_RoutingCornerStyleRule Properties

```
Property Style        TCornerStyle
Property MinSetBack : TCoord
Property MaxSetBack : TCoord
```

IPCB_RoutingLayersRule rule

Overview

This routing layers rule specifies the preferred routing direction for layer to be used during autorouting.

IPCB_RoutingLayersRule Properties

```
Property RoutingLayers [L : TLayer] : Boolean
```

IPCB_RoutingPriorityRule rule

Overview

This routing priority rule function assigns a routing priority which is used to set the order of how the nets will be auto routed.

IPCB_RoutingPriorityRule Properties

```
Property RoutingPriority : Integer
```

IPCB_RoutingTopologyRule rule

Overview

This routing topology rule function specifies the topology of the net. The net compromises a pattern of the pin-to-pin connections. A topology is applied to a net for specific reasons, for example to minimise signal reflections, daisy chain topology is used.

Notes

The Topology property sets or returns the topology of the net. The following topologies can be applied: Shortest, Horizontal, Vertical, Daisy-Simple, Daisy-Mid Driven, Daisy-Balanced, or Star.

IPCB_RoutingTopologyRule Properties

```
Property Topology: TNetTopology
```

IPCB_RoutingViaStyleRule rule

Overview

This routing via style rule specifies the via object to be used during autorouting. Vias can be through-hole, Blind (from a surface layer to an inner layer) or Buried (between two inner layers).

Notes

The ViaStyle property sets or returns the via style. Vias can be thru-hole, blind (from a surface layer to an inner layer) or buried (between two inner layers).

IPCB_RoutingViaStyleRule Properties

```
Property MinHoleWidth      : TCoord
Property MaxHoleWidth      : TCoord
Property PreferedHoleWidth : TCoord
Property MinWidth          : TCoord
Property MaxWidth          : TCoord
Property PreferedWidth     : TCoord
Property ViaStyle          : TRouteVia
```

IPCB_RuleSupplyNets rule

Overview

This IPCB_RuleSupplyNets interface specifies the supply nets on the board. The signal integrity analyzer needs to know each supply net name and voltage.

IPCB_RuleSupplyNets Properties

```
Property Voltage : Double
```

IPCB_ShortCircuitConstraint rule

Overview

The short circuit constraint rule includes a constraint to test for short circuits between primitive objects on the copper layers. A short circuit exists when two objects that have different net names touch.

Notes

The Allowed property sets or returns the boolean value whether or not the short circuit constraint rule is allowed.

IPCB_ShortCircuitConstraint Properties

```
Property Allowed : Boolean
```

IPCB_SMDNeckDownConstraint rule

Overview

IPCB_SMDToPlaneConstraint Properties

```
Property Percent : TReal
```

IPCB_SMDToCornerConstraint rule

Overview

Notes

The Distance property determines the distance between the SMD and a corner.

IPCB_SMDToCornerConstraint Properties

```
Property Distance : TCoord
```

IPCB_SMDToPlaneConstraint rule

Overview

IPCB_SMDToPlaneConstraint Methods

```
Function  IsInternalPlaneNet(Net   : IPCB_Net; Board : IPCb_Board): Boolean;
```

IPCB_SMDToPlaneConstraint Properties

```
Property Distance : TCoord
```

IPCB_SolderMaskExpansionRule rule

Overview

The solder mask expansion rule defines the shape that is created on the solder mask layer at each pad and via site. This shape is expanded or contracted radially by the amount specified by this rule.

Note, Tenting and solder mask are related. A negative value allows the solder mask to be reduced.

IPCB_SolderMaskExpansion Properties

```
Property Expansion : TCoord
```

IPCB_TestPointStyleRule rule

Overview

The auto-router includes a testpoint generator, which can identify existing pads and vias as testpoints, as well as adding testpoint pads to nets which can not be accessed at existing pads and vias. Generally the testpoint types are used in bare board testing or are used for in-circuit testing.

IPCB_TestPointStyleRule Methods

Procedure DoDefaultStyleOrder;

IPCB_TestPointStyleRule Properties

```
Property TestpointUnderComponent       : Boolean
Property MinSize                       : TCoord
Property MaxSize                       : TCoord
Property PreferedSize                  : TCoord
Property MinHoleSize                   : TCoord
Property MaxHoleSize                   : TCoord
Property PreferedHoleSize              : TCoord
Property TestpointGrid                 : TCoord
Property OrderArray [I : Integer]      : TTestPointStyle
Property AllowedSide                   : TTestpointAllowedSideSet
Property AllowedStyleSet               : TTestPointStyleSet
Property Allowed [I : TTestPointStyle] : Boolean
Property TestpointPriority[I : TTestPointStyle] : Integer
```

IPCB_TestPointUsage rule

Overview

Altium Designer's autorouter includes a testpoint generator, which can identify existing pads and vias as testpoints, as well as adding testpoint pads to nets which can not be accessed at existing pads and vias. Generally the testpoint types are used in bare board testing or are used for in-circuit testing.

IPCB_TestPointUsage Properties

```
Property Valid              : TTestpointValid
Property AllowMultipleOnNet : Boolean
```

IPCB_UnConnectedPinRule rule

Overview

This interface deals with unconnected pins on a PCB document.

IPCB_UnConnectedPinRule Properties

No new properties.

See also

IPCB_Rule interface

IPCB_ViasUnderSMDConstraint rule

Overview

The Vias Under SMD constraint rule specifies if vias can be placed under SMD pads during autorouting.

IPCB_ViasUnderSMDConstraint Properties

Property Allowed : Boolean

Signal Integrity Design Rules

IPCB_SignalStimulus rule

Overview

The IPCB_SignalStimulus rule concerns with the definition of a signal for stimulus, such as the stimulus type, signal level, start, stop times and the period of the signal.

IPCB_SignalStimulus Methods

Procedure Export_ToStmFile (AFilename : TString);

IPCB_SignalStimulus Properties

Property Kind : TStimulusType

Property Level : TSignalLevel

Property StartTime : TReal

Property StopTime : TReal

Property PeriodTime : TReal

IPCB_MaxOvershootFall rule

Overview

The IPCB_MaxOvershootFall interface specifies the maximum allowable overshoot (ringing below the base value) on the falling edge of the signal.

IPCB_MaxOvershootFall Properties

Property Maximum : TReal

IPCB_MaxOvershootRise rule

Overview

The IPCB_MaxOvershootRise interface specifies the maximum allowable overshoot (ringing above the base value) on the rising edge of the signal.

IPCB_MaxOvershootRise Properties

Property Maximum : TReal

IPCB_MaxUndershootFall

Overview

The IPCB_MaxUndershootFall interface specifies the maximum allowable undershoot (ringing above the base value) on the falling edge of the signal.

IPCB_MaxUndershootFall Properties

Property Maximum : TReal

IPCB_MaxUndershootRise rule

Overview

The IPCB_MaxUndershootRise function specifies the maximum allowable undershoot (ringing below the top value) on the rising edge of the signal.

IPCB_MaxUndershootRise Properties

Property Maximum : TReal

IPCB_RuleMaxMinImpedance rule

Overview

The IPCB_RuleMaxMinImpedance interface returns or sets values for a MaxMin Impedance rule object depending on the query mode (eGetState or eSetState). This rule specifies the minimum and maximum net impedance allowed. Net impedance is a function of the conductor geometry and conductivity, the surrounding dielectric material (the board base material, multilayer insulation, solder mask, etc) and the physical geometry of the board (distance to other conductors in the z-plane). This function defines the minimum and maximum impedance values allowed for the signal integrity rule.

IPCB_RuleMaxMinImpedance Properties

Property Minimum : TReal

Property Maximum : TReal

IPCB_RuleMinSignalTopValue rule

Overview

The IPCB_RuleMinSignalTopValue function specifies the minimum allowable signal top value. The top value is the voltage that a signal settles into the minimum top state.

IPCB_RuleMinSignalTopValue Properties

Property Minimum : TReal

IPCB_RuleMaxSignalBaseValue rule

Overview

The IPCB_RuleMaxSignalBaseValue function specifies the maximum allowable base value. The base value is the voltage that a signal settles to in the low state.

IPCB_ RuleMaxSignalBaseValue Properties

Property Maximum : TReal

IPCB_RuleFlightTime_RisingEdge rule

Overview

The IPCB_RuleFlightTime_RisingEdge interface returns or sets values for the flight time of the rising edge of a signal. The flight time is the signal delay introduced by the interconnect structure. It is calculated as the time it takes to drive the actual input to the threshold voltage, less the time it would take to drive a reference load (connected directly to the output) to the threshold voltage.

IPCB_RuleFlightTime_RisingEdge Properties

Property MaximumFlightTime : TReal

IPCB_RuleFlightTime_FallingEdge rule

Overview

The IPCB_RuleFlightTime_FallingEdge interface returns or sets values for the flight time of the falling edge of a signal. The flight time is the signal delay introduced by the interconnect structure. It is calculated as the time it takes to drive the actual input to the threshold voltage, less the time it would take to drive a reference load (connected directly to the output) to the threshold voltage.

IPCB_RuleFlightTime_FallingEdge Properties

Property MaximumFlightTime : TReal

IPCB_RuleMaxSlopeRisingEdge rule

Overview

The IPCB_RuleMaxSlope_RisingEdge interface specifies the maximum allowable slope on the rising edge of the signal. The slope is the time it takes for a signal to rise from the threshold voltage to a valid high voltage.

IPCB_RuleMaxSlopeRisingEdge Properties

Property MaxSlope : TReal

IPCB_RuleMaxSlopeFallingEdge rule

Overview

The IPCB_RuleMaxSlope_FallingEdge interface specifies the maximum allowable slope on the falling edge of the signal. The slope is the time it takes for a signal to fall from the threshold voltage to a valid low voltage.

IPCB_RuleMaxSlopeFallingEdge Properties

Property MaxSlope : TReal

PCB Object Iterators

An iterator conducts a search through a PCB document's design database to fetch PCB design objects. With an iterator, you can control which objects on which layers and within specified regions.

There are four different types of iterators; Board Iterator, Library Iterator, Spatial Iterator and Group Iterator. The board iterator is for conducting searches on a PCB document, the library iterator on library documents, spatial iterators conducting searches within a restricted boundary on a document and the group iterator conducting searches for primitives within a group object such as tracks and arcs within a component object.

The scripting system's Delphi Script doesn't support sets, therefore to pass in a set of layers or a set of objects, you need to use the **MkSet** function to create a pseudo set of objects or layers for the **AddFilter_ObjectSet** or **AddFilterLayerSet** procedures.

For example

BoardIterator.AddFilter_ObjectSet(MkSet(eTrackObject,eFillObject));

See also

IPCB_AbstractIterator interface

IPCB_BoardIterator interface

IPCB_LibraryIterator interface

IPCB_SpatialIterator interface

IPCB_GroupIterator interface

IPCB_AbstractIterator

Overview

An abstract iterator object interface which is the ancestor interface for a board, spatial, group and library Iterators.

An iterator object iterates through a PCB database representing the PCB document to fetch specified objects within a specified region on a specified layer if necessary.

Notes

- When using the DelphiScript language set in Scripts, you need to use the MkSet function to specify the object set or the layer set. The **MkSet** function creates a set of objects because the Delphiscript language does not support Object Pascal's sets.

Methods

```
Function  I_ObjectAddress     : TPCBObjectHandle;
Function  FirstPCBObject      : IPCB_Primitive;
Function  NextPCBObject       : IPCB_Primitive

Procedure SetState_FilterAll;

Procedure AddFilter_ObjectSet (AObjectSet  : TObjectSet);
Procedure AddFilter_LayerSet  (ALayerSet   : TLayerSet);
Procedure AddFilter_Area      (X1,
                               Y1,
                               X2,
                               Y2 : TCoord);
Procedure AddFilter_AllLayers;
```

See also

IPCB_BoardIterator interface
IPCB_LibraryIterator interface
IPCB_SpatialIterator interface
IPCB_Primitive interface
TObjectSet set
TObjectId enumerated values
TLayerSet set
TLayer enumerated values
MkSet function

IPCB_BoardIterator

Overview

The **IPCB_BoardIterator** iterates through a PCB document to fetch PCB design objects on this PCB.

With the iterator, you can control which objects on which layers and within specified regions with the **AddFilter_ObjectSet**, **AddFilter_LayerSet** and **AddFilter_Area** methods to be fetched.

The **AddFilter_method** controls how design objects are fetched. The **TIterationMethod** type has three different values; eProcessAll, eProcessFree, eProcessComponents.

Notes

- The Delphiscript language set doesn't support sets, therefore to pass in a set of layers or a set of objects in a function in a script, you need to use the **MkSet** function to create a pseudo set of objects or layers for the **AddFilter_ObjectSet** or **AddFilterLayerSet** procedures. For example **BoardIterator.AddFilter_ObjectSet(MkSet(eTrackObject,eFillObject));**

Methods

```
Function  I_ObjectAddress  : TPCBObjectHandle;

Function  FirstPCBObject   : IPCB_Primitive;
Function  NextPCBObject    : IPCB_Primitive

Procedure SetState_FilterAll;

Procedure AddFilter_ObjectSet (AObjectSet  : TObjectSet);
Procedure AddFilter_LayerSet  (ALayerSet   : TLayerSet);
Procedure AddFilter_Area      (X1,
```

```
                                Y1,
                                X2,
                                Y2          : TCoord);

Procedure AddFilter_AllLayers;
Procedure AddFilter_Method (AMethod : TIterationMethod);
```

Example

```
Var
    BoardHandle : IPCB_Board;
    Pad         : IPCB_Primitive;
    Iterator    : IPCB_BoardIterator;
    PadNumber   : Integer;
Begin
    // Retrieve the current board
    Board := PCBServer.GetCurrentPCBBoard;
    If Board = Nil Then Exit;

    // Setup Board iterator
    Iterator        := Board.BoardIterator_Create;
    Iterator.AddFilter_ObjectSet(MkSet(ePadObject));
    Iterator.AddFilter_LayerSet(AllLayers);
    Iterator.AddFilter_Method(eProcessAll);

    PadNumber       := 0;
    // Search and count pads
    Pad := Iterator.FirstPCBObject;
    While (Pad <> Nil) Do
    Begin
        Inc(PadNumber);
        Pad := Iterator.NextPCBObject;
    End;
    Board.BoardIterator_Destroy(Iterator);

    // Display the count result on a dialog.
    ShowMessage('Pad Count = ' + IntToStr(PadNumber));
```

See also

IPCB_BoardIterator interface

IPCB_LibraryIterator interface

IPCB_SpatialIterator interface

IPCB_Primitive interface

TObjectSet set

TObjectId enumerated values

TLayerSet set

TLayer enumerated values

TIterationMethod enumerated values

MkSet function

IPCB_LibraryIterator

Overview

The **IPCB_LibraryIterator** object interface Iterates through a loaded PCB library in Altium Designer to fetch PCB footprints and its primitives. The library iterator basically retrieves the footprints and to retrieve the child objects of each footprint, you need to employ the group iterator.

The **IPCB_LibraryIterator** object interface iterates through a loaded PCB library in Altium Designer to fetch PCB footprints which are represented by the **IPCB_LibComponent** interfaces. The **IPCB_LibraryIterato**r interface is used in the **IPCB_Library** interface - **LibraryIterator_Create** and **LibraryIterator_Destroy** methods.

The current footprint is a component with an unnamed designator is represented by the **IPCB_LibComponent** interface.

Notes

- The **IPCB_LibraryIterator** interface has only methods inherited from the **IPCB_AbstractIterator** interface and is reproduced here for reference.
- A library is represented by the **IPCB_Library** and the current footprint on a library document is represented by the **IPCB_Board** interface.
- A PCB footprint (from the library) is represented by its **IPCB_LibComponent** interface which is inherited from the **IPCB_Group** object interface.
- A PCB footprint is composed of child objects such as pads and tracks. Therefore the footprint has its own **IPCB_GroupIterator** to fetch its own child objects.
- DelphiScript doesn't support sets, therefore to pass in a set of layers or a set of objects, you need to use the **MkSet** function to create a pseudo set of objects or layers for the **AddFilter_ObjectSet** or **AddFilterLayerSet** procedures. For example LibraryIterator.AddFilter_ObjectSet(**MkSet**(eTrackObject,eFillObject));

Methods

```
Function  I_ObjectAddress  : TPCBObjectHandle;

Function  FirstPCBObject   : IPCB_Primitive;
Function  NextPCBObject    : IPCB_Primitive

Procedure AddFilter_ObjectSet (AObjectSet  : TObjectSet);
Procedure AddFilter_LayerSet  (ALayerSet   : TLayerSet);
Procedure AddFilter_Area      (X1,Y1,X2,Y2 : TCoord);
Procedure AddFilter_AllLayers;

Procedure SetState_FilterAll;
```

Example

```
Procedure LookInsideFootprints;
Var
    CurrentLib        : IPCB_Library;
    AObject           : IPCB_Primitive;
    FootprintIterator : IPCB_LibraryIterator;
    Iterator          : IPCB_GroupIterator;
    Footprint         : IPCB_LibComponent;
    FirstTime         : Boolean;
    NoOfPrims         : Integer;
    S                 : TString;
Begin
    CurrentLib := PCBServer.GetCurrentLibrary;
    If CurrentLib = Nil Then
    Begin
        ShowMessage('This is not a PCB library document');
        Exit;
    End;
```

```
    // For each page of library is a footprint
    FootprintIterator := CurrentLib.LibraryIterator_Create;
    FootprintIterator.SetState_FilterAll;
    S         := '';
    FirstTime := True;
    Try
        // Within each page, fetch primitives of the footprint
        // A footprint is a IPCB_LibComponent inherited from
        // IPCB_Group which is a container object storing primitives.
        Footprint := FootprintIterator.FirstPCBObject; // IPCB_LibComponent
        While Footprint <> Nil Do
        Begin
           If FirstTime Then
           Begin
              S := S + ExtractFileName(Footprint.Board.FileName) + #13;
              S := S + ' Current Footprint : ' +
                   PCBServer.GetCurrentComponent(CurrentLib)+ #13 + #13;
           End;

           S := S + Footprint.Name;

           Iterator := Footprint.GroupIterator_Create;
           Iterator.SetState_FilterAll;
           // Counts number of prims for each Footprint as a IPCB_LibComponent
           // Note that the IPCB_LibComponent has a GetPrimitiveCount method
           NoOfPrims := 0;
           AObject := Iterator.FirstPCBObject;
           While (AObject <> Nil) Do
           Begin
               // counts child objects or primitives
               // for each footprint.
               Inc(NoOfPrims);
               // do what you want with the AObject.
               AObject := Iterator.NextPCBObject;
           End;
           S := S + ' has ' + IntToStr(NoOfPrims) + ' Primitives.' + #13;
           FirstTime := False;
           Footprint.GroupIterator_Destroy(Iterator);
           Footprint := FootprintIterator.NextPCBObject;
        End;
    Finally
        CurrentLib.LibraryIterator_Destroy(FootprintIterator);
    End;
    ShowMessage(S);
End;
```

See also

IPCB_BoardIterator interface

IPCB_SpatialIterator interface

IPCB_GroupIterator interface

IPCB_Primitive interface

TObjectSet set

TObjectId enumerated values

TLayerSet set

TLayer enumerated values

TIterationMethod enumerated values

MkSet function

LibraryIterator example from **\Examples\Scripts\DelphiScript\PCB** folder.

IPCB_SpatialIterator

Overview

The IPCB_SpatialIterator interface iterates through a defined region on the loaded PCB document in Altium Designer to fetch PCB design objects.

You will need to specify the object set, the layer set and the area for the spatial iterator to conduct its search within a defined boundary. The following methods are AddFilter_ObjectSet, AddFilter_LayerSet and AddFilter_Area.

Notes

- **IPCB_SpatialIterator** has only methods inherited from the **IPCB_AbstractIterator** interface and is reproduced here for reference.
- Delphiscript doesn't support sets, therefore to pass in a set of layers or a set of objects, you need to use the **MkSet** function to create a pseudo set of objects or layers for the AddFilter_ObjectSet or AddFilterLayerSet procedures. For example SpatialIterator.AddFilter_ObjectSet(**MkSet**(eTrackObject,eFillObject));

Methods (inherited from IPCB_AbstractIterator)

```
Function  I_ObjectAddress  : TPCBObjectHandle;

Function  FirstPCBObject   : IPCB_Primitive;
Function  NextPCBObject    : IPCB_Primitive

Procedure AddFilter_ObjectSet  (AObjectSet  : TObjectSet);
Procedure AddFilter_LayerSet   (ALayerSet   : TLayerSet);
Procedure AddFilter_Area       (X1,
                                Y1,
                                X2,
                                Y2 : TCoord);

Procedure AddFilter_AllLayers;

Procedure SetState_FilterAll;
```

Example

```
    (* Top/Bottom Layers and Arc/Track objects defined *)
    (* for the Spatial iterator constraints *)
    ASetOfLayers  := MkSet(eTopLayer,eBottomLayer);
    ASetOfObjects := MkSet(eArcObject,eTrackObject);

    Iterator := Board.SpatialIterator_Create;
    Iterator.AddFilter_ObjectSet(ASetOfObjects);
    Iterator.AddFilter_LayerSet(ASetOfLayers);
    Iterator.AddFilter_Area(X1,Y1,X2,Y2);

    (* Iterate for tracks and arcs on bottom/top layers *)
```

```
PCBObject := Iterator.FirstPCBObject;
While PCBObject <> 0 Do
Begin
     PCBObject.Selected := True;
     PCBObject := Iterator.NextPCBObject;
End;
Board.SpatialIterator_Destroy(Iterator);
```

See also

IPCB_BoardIterator interface

IPCB_LibraryIterator interface

IPCB_GroupIterator interface.

IPCB_Primitive interface

TObjectSet set

TObjectId enumerated values

TLayerSet set

TLayer enumerated values

TIterationMethod enumerated values

MkSet function

Spatial iterator script in **\Examples\Scripts\PCB** folder.

IPCB_GroupIterator

Overview

The **IPCB_GroupIterator** interface deals with group objects such as board layouts, polygons, components, footprints in a PCB library, coordinates and dimensions that have child objects within.

When you need to fetch child objects of a group object such as tracks and arcs of a footprint in a PCB library, you need to create a Group Iterator for that group object.

The sequence is basically as follows;

- Set up a board iterator to fetch design objects from the PCB/Library document
- For each design object that is a group object (such as polygons and components), setup a group iterator and fetch child objects for that group object.
- Destroy the group iterator when finished iterating child objects for that group object
- Destroy the board/library iterator when finished iterating

Notes

- IPCB_GroupIterator has methods inherited from the IPCB_AbstractIterator interface and is reproduced here for reference.
- Delphiscript does not support sets, therefore to pass in a set of layers or a set of objects, you need to use the MkSet function to create a pseudo set of objects or layers for the AddFilter_ObjectSet or AddFilterLayerSet procedures.
- For example LibraryIterator.AddFilter_ObjectSet(**MkSet**(eTrackObject,eFillObject));

Methods

```
Function  I_ObjectAddress    : TPCBObjectHandle;
Function  FirstPCBObject     : IPCB_Primitive;
Function  NextPCBObject      : IPCB_Primitive

Procedure AddFilter_ObjectSet  (AObjectSet  : TObjectSet);
Procedure AddFilter_LayerSet   (ALayerSet   : TLayerSet);
Procedure AddFilter_Area       (X1,
                                Y1,
                                X2,
                                Y2 : TCoord);
```

Procedure AddFilter_AllLayers;

Procedure SetState_FilterAll;

Example

```
Procedure CountTracks;
Var
    Track             : IPCB_Track;
    ChildIterator     : IPCB_GroupIterator;
    Component         : IPCB_Component;
    ComponentIterator : IPCB_BoardIterator;
    TrackCount        : Integer;
Begin
    TrackCount     := 0;
    If PCBServer.GetCurrentPCBBoard = Nil Then Exit;

    // Create a board iterator to fetch a component.
    ComponentIteratorHandle := PCBServer.GetCurrentPCBBoard.BoardIterator_Create;
    ComponentIteratorHandle.AddFilter_ObjectSet(MkSet(eComponentObject));

    If Component <> Nil Then
    Begin
        // Create an iterator from the component to fetch
        // its child objects.
        ChildIterator := Component.GroupIterator_Create;
        ChildIterator.AddFilter_ObjectSet(MkSet(eTrackObject));
        ChildIterator.AddFilter_LayerSet(MkSet(eTopOverlay));
        Track := ChildIterator.FirstPCBObject;
        While (Track <> Nil) Do
        Begin
            Inc(TrackCount);
            Track := ChildIterator.NextPCBObject;
        End;

        ShowInfo('This component ' + Component.SourceDesignator  +
                 ' has ' +  IntToStr(TrackCount)  + ' tracks.');
        // When finished iterating component's child objects,
        // destroy the component's        group iterator.
        Component.GroupIterator_Destroy(TrackIterator);
    End;
    // when finished iterating on PCB document, destroy the board iterator.
    PCBServer.GetCurrentPCBBoard.BoardIterator_Destroy(ComponentIterator);
End;
```

See also

IPCB_BoardIterator interface

IPCB_LibraryIterator interface

IPCB_SpatialIterator interface

IPCB_Primitive interface

TObjectSet set

TObjectId enumerated values

TLayerSet set

TLayer enumerated values

TIterationMethod enumerated values

MkSet function

LibraryIterator script example

CountTracksInComponent script example

PCB Enumerated Types

The enumerated types are used for many of the PCB object interfaces methods which are covered in this section.

For example the IPCB_Board interface has a LayerIsUsed [L : TLayer] : Boolean property. You can use this Enumerated Types section below to check what the range is for the TLayer type.

See also

PCB API Reference

TAdvPCBFileFormatVersion

```
TAdvPCBFileFormatVersion  =
(ePCBFileFormatNone,
 eAdvPCBFormat_Binary_V3,
 eAdvPCBFormat_Library_V3,
 eAdvPCBFormat_ASCII_V3,
 eAdvPCBFormat_Binary_V4,
 eAdvPCBFormat_Library_V4,
 eAdvPCBFormat_ASCII_V4,
 eAdvPCBFormat_Binary_V5,
 eAdvPCBFormat_Library_V5,
 eAdvPCBFormat_ASCII_V5);
```

TAngle

```
Double type.
```

TApertureUse

```
TApertureUse = ( eNoApertureUse,
                 eMultiUse,
                 eDrawUse,
                 eFlashUse);
```

TAutoPanMode

```
TAutoPanMode = ( eNoAutoPan
                 eReCentre
                 eFixedJump
                 eShiftAccellerator
                 eShiftDeccellerator
                 eBallistic
                 eAdaptive);
```

TAutoPanUnit

```
TAutoPanUnit  = ( eAutoPanByMils
                  eAutoPanByPixels);
```

TBaud

```
TBaud                = ( eBaud110                              ,
                         eBaud150                              ,
                         eBaud300                              ,
                         eBaud600                              ,
                         eBaud1200                             ,
                         eBaud2400                             ,
                         eBaud4800                             ,
                         eBaud9600                             ,
                         eBaud19200
                       );
```

TBGAFanoutDirection

```
TBGAFanoutDirection  = ( eBGAFanoutDirection_Out               ,
                         eBGAFanoutDirection_NE                ,
                         eBGAFanoutDirection_SE                ,
                         eBGAFanoutDirection_SW                ,
                         eBGAFanoutDirection_NW                ,
                         eBGAFanoutDirection_In
                        );
```

TBGAFanoutViaMode

```
TBGAFanoutViaMode    = ( eBGAFanoutVia_Closest                 ,
                         eBGAFanoutVia_Centered
                       );
```

TBoardSide type

```
TBoardSide = ( eBoardSide_Top, eBoardSide_Bottom);
```

TCacheState

```
TCacheState = ( eCacheInvalid,
                eCacheValid,
                eCacheManual);
```

TChangeScope

```
TChangeScope         = ( eChangeNone                           ,
                         eChangeThisItem                       ,
                         eChangeAllPrimitives                  ,
                         eChangeAllFreePrimitives              ,
                         eChangeComponentDesignators           ,
                         eChangeComponentComments              ,
                         eChangeLibraryAllComponents           ,
                         eChangeCancelled
                       );
```

TClassMemberKind

```
TClassMemberKind = (eClassMemberKind_Net,
                    eClassMemberKind_Component,
                    eClassMemberKind_FromTo,
                    eClassMemberKind_Pad,
                    eClassMemberKind_Layer,
```

```
                    eClassMemberKind_DesignChannel,
                    eClassMemberKind_DifferentialPair
);
```

TComponentCollisionCheckMode

```
TComponentCollisionCheckMode
                     = (eQuickCheck,
                        eMultiLayerCheck,
                        eFullCheck,
                        eComponentBodyCheck,
                       );
```

TComponentMoveKind

```
TComponentMoveKind   = ( eNoComponentMoveNoAction
                         eJumpToComponent
                         eMoveComponentToCursor
                       );
```

TComponentStyle

```
TComponentStyle      = ( eComponentStyle_Unknown                    ,
                         eComponentStyle_Small                      ,
                         eComponentStyle_SmallSMT                   ,
                         eComponentStyle_Edge                       ,
                         eComponentStyle_DIP                        ,
                         eComponentStyle_SIP                        ,
                         eComponentStyle_SMSIP                      ,
                         eComponentStyle_SMDIP                      ,
                         eComponentStyle_LCC                        ,
                         eComponentStyle_BGA                        ,
                         eComponentStyle_PGA
                       );
```

TComponentType

```
TComponentType       = ( eBJT                                       ,
                         eCapactitor                                ,
                         eConnector                                 ,
                         eDiode                                     ,
                         eIC                                        ,
                         eInductor                                  ,
                         eResistor
                       );
```

TConfinementStyle

```
TConfinementStyle    = ( eConfineIn,
                         eConfineOut);
```

TConnectionMode

```
TConnectionMode      = ( eRatsNestConnection
                         eBrokenNetMarker
                       );
```

TCoord

```
TCoord = Integer;
```

TCoordPoint

```
TCoordPoint = Record
        x,
        y : TCoord;
End;
```

TCoordRect

```
TCoordRect   = Record
    Case Integer of
       0 :(left,bottom,right,top : TCoord);
       1 :(x1,y1,x2,y2           : TCoord);
       2 :(Location1,Location2   : TCoordPoint);
End;
```

Note TPoint is a Borland Delphi defined type in the Types.pas unit.

TCopyMode

```
TCopyMode = ( eFullCopy,
              eFieldCopy);
```

TCornerStyle

```
TCornerStyle         = ( eCornerStyle_90,
                         eCornerStyle_45,
                         eCornerStyle_Round);
```

TDaisyChainStyle

```
TDaisyChainStyle     = ( eDaisyChainLoad                          ,
                         eDaisyChainTerminator                    ,
                         eDaisyChainSource
                       );
```

TDataBits

```
TDataBits            = ( eDataBits5                               ,
                         eDataBits6                               ,
                         eDataBits7                               ,
                         eDataBits8
                       );
```

TDielectricType

```
TDielectricType = (eNoDielectric,
                   eCore,
                   ePrePreg,
                   eSurfaceMaterial);
```

TDimensionArrowPosition

```
TDimensionArrowPosition = ( eInside,eOutside);
```

TDimensionReference

```
 TDimensionReference            = Record
     Primitive                  : IPCB_Primitive;
```

```
    Point                       : TCoordPoint;
    Anchor                      : Integer;
End;
```

TDimensionTextPosition

```
TDimensionTextPosition
                    = ( eTextAuto                          ,
                        eTextCenter                        ,
                        eTextTop                           ,
                        eTextBottom                        ,
                        eTextRight                         ,
                        eTextLeft                          ,
                        eTextInsideRight                   ,
                        eTextInsideLeft                    ,
                        eTextUniDirectional                ,
                        eTextManual
                      );
```

TDimensionKind

```
    TDimensionKind      = ( eNoDimension                       ,
                            eLinearDimension                   ,
                            eAngularDimension                  ,
                            eRadialDimension                   ,
                            eLeaderDimension                   ,
                            eDatumDimension                    ,
                            eBaselineDimension                 ,
                            eCenterDimension                   ,
                            eOriginalDimension                 ,
                            eLinearDiameterDimension           ,
                            eRadialDiameterDimension
                          );
```

TDimensionUnit

```
TDimensionUnit = ( eMils,
                   eInches,
                   eMillimeters,
                   eCentimeters,
                   eDegrees,
                   eRadians,
                   eAutomaticUnit);
```

TDirection

```
TDirection =  (eDir_N ,
               eDir_NE,
               eDir_E ,
               eDir_SE,
               eDir_S ,
               eDir_SW,
               eDir_W ,
               eDir_NW);
```

TDirectionSet

```
TDirectionSet = Set Of TDirection;
```

TDisplay

```
TDisplay             = ( eOverWrite
                         eHide
                         eShow
                         eInvert
                         eHighLight
                        );
```

TDrawingOrderArray

```
Type TDrawingOrderArray = Array [0..Ord(MaxLayer)] Of TLayer;
```

TDrawMode

```
TDrawMode = ( eDrawFull,
              eDrawDraft,
              eDrawHidden);
```

TDrillSymbol

```
TDrillSymbol         = ( eSymbols                                ,
                         eNumbers                                ,
                         eLetters
                        ); {Used by gerber and Print/ Plot}
```

TDXColorMode

```
TDXColorMode = ( eDXPrimitive_DisplayNormal      ,
                 eDXPrimitive_DisplayDimm        ,
                 eDXPrimitive_DisplayHighlight   ,
                 eDXPrimitive_DisplayTransparent ,
                 eDXPrimitive_DisplayGrayScale   ,
                 eDXPrimitive_DisplayMonochrome  ,
                 eDXPrimitive_DisplayDRCError             ,
                 eDXPrimitive_DisplayGrayScaleTransparent ,
                 eDXPrimitive_DisplayMonochromeTRansparent,
                 eDXPrimitive_DisplayDimmTransparent      ,
                 eDXPrimitive_DisplayHighlightTransparent
                 );
```

TDynamicString

```
TDynamicString = AnsiString;
```

TEditingAction

```
TEditingAction=(eEditAction_Focus,
                eEditAction_Move,
                eEditAction_Change,
                eEditAction_Delete,
                eEditAction_Select,
                eEditAction_NonGraphicalSelect,
                eEditAction_Measure,
                eEditAction_Dimension);
```

TEmbeddedBoardOriginMode

```
TEmbeddedBoardOriginMode = (eOriginMode_BoardOrigin,
                            eOriginMode_BottomLeft);
```

TEnabledRoutingLayers

```
TEnabledRoutingLayers = Array [eTopLayer..eBottomLayer] Of Boolean;
```

TExtendedDrillType

```
TExtendedDrillType              = ( eDrilledHole,
                                    ePunchedHole,
                                    eLaserDrilledHole,
                                    ePlasmaDrilledHole
                                  );
```

TExtendedHoleType

```
TExtendedHoleType               = ( eRoundHole,
                                    eSquareHole,
                                    eSlotHole
                                  );
```

TTestpointAllowedSide

```
TTestpointAllowedSide
                      = ( eAllowTopSide                          ,
                          eAllowBottomSide                       ,
                          eAllowThruHoleTop                      ,
                          eAllowThruHoleBottom
                        );
```

TFanoutDirection

```
TFanoutDirection = (eFanoutDirection_None,
                    eFanoutDirection_InOnly,
                    eFanoutDirection_OutOnly,
                    eFanoutDirection_InThenOut,
                    eFanoutDirection_OutThenIn,
                    eFanoutDirection_Alternating);
```

TFanoutStyle

```
TFanoutStyle = (eFanoutStyle_Auto,
                eFanoutStyle_Rows,
                eFanoutStyle_Staggered,
                eFanoutStyle_BGA,
                eFanoutStyle_UnderPads);
```

TFontName

```
TFontName             = Array [0..LF_FACESIZE - 1] Of WideChar;
```

TFontID

```
TFontID = SmallInt;
```

TFullFontName

```
TFullFontName         = Array [0..LF_FULLFACESIZE - 1] Of WideChar;
```

TFromToDisplayMode

```
TFromToDisplayMode   = ( eFromToDisplayMode_Automatic,
                         eFromToDisplayMode_Hide,
                         eFromToDisplayMode_Show);
```

TGraphicsCursor

```
TGraphicsCursor = ( eCurShapeCross90,
                    eCurShapeBigCross,
                    eCurShapeCross45);
```

THandshaking

```
THandshaking         = ( eHandshakingNone                          ,
                         eHandshakingXonXOff                       ,
                         eHandshakingHardwire
                       );
```

TInteractiveRouteMode

```
 TInteractiveRouteMode
                     = ( eIgnoreObstacle
                         eAvoidObstacle
                         ePushObstacle
                       );
```

TIterationMethod

TIterationMethod = (eProcessAll, eProcessFree, eProcessComponent);

TLayer

```
TLayer = (eNoLayer          ,
          eTopLayer         ,
          eMidLayer1        ,
          eMidLayer2        ,
          eMidLayer3        ,
          eMidLayer4        ,
          eMidLayer5        ,
          eMidLayer6        ,
          eMidLayer7        ,
          eMidLayer8        ,
          eMidLayer9        ,
          eMidLayer10       ,
          eMidLayer11       ,
          eMidLayer12       ,
          eMidLayer13       ,
          eMidLayer14       ,
          eMidLayer15       ,
          eMidLayer16       ,
          eMidLayer17       ,
          eMidLayer18       ,
          eMidLayer19       ,
          eMidLayer20       ,
          eMidLayer21       ,
```

```
eMidLayer22       ,
eMidLayer23       ,
eMidLayer24       ,
eMidLayer25       ,
eMidLayer26       ,
eMidLayer27       ,
eMidLayer28       ,
eMidLayer29       ,
eMidLayer30       ,
eBottomLayer      ,
eTopOverlay       ,
eBottomOverlay    ,
eTopPaste         ,
eBottomPaste      ,
eTopSolder        ,
eBottomSolder     ,
eInternalPlane1   ,
eInternalPlane2   ,
eInternalPlane3   ,
eInternalPlane4   ,
eInternalPlane5   ,
eInternalPlane6   ,
eInternalPlane7   ,
eInternalPlane8   ,
eInternalPlane9   ,
eInternalPlane10  ,
eInternalPlane11  ,
eInternalPlane12  ,
eInternalPlane13  ,
eInternalPlane14  ,
eInternalPlane15  ,
eInternalPlane16  ,
eDrillGuide       ,
eKeepOutLayer     ,
eMechanical1      ,
eMechanical2      ,
eMechanical3      ,
eMechanical4      ,
eMechanical5      ,
eMechanical6      ,
eMechanical7      ,
eMechanical8      ,
eMechanical9      ,
eMechanical10     ,
eMechanical11     ,
eMechanical12     ,
eMechanical13     ,
eMechanical14     ,
eMechanical15     ,
```

```
          eMechanical16     ,
          eDrillDrawing     ,
          eMultiLayer       ,
          eConnectLayer     ,
          eBackGroundLayer  ,
          eDRCErrorLayer    ,
          eHighlightLayer   ,
          eGridColor1       ,
          eGridColor10      ,
          ePadHoleLayer     ,
          eViaHoleLayer
        );
```

TLayerSet

```
TLayerSet = Set of TLayer;
```

See also

TLayer

TLayerStackStyle

```
TLayerStackStyle     = ( eLayerStack_Pairs       ,
                         eLayerStacks_InsidePairs,
                         eLayerStackBuildup);
```

TLengthenerStyle

```
TLengthenerStyle = (eLengthenerStyle_90,
                    eLengthenerStyle_45,
                    eLengthenerStyle_Round);
```

TLogicalDrawingMode

```
TLogicalDrawingMode  = ( eDisplaySolid                          ,
                         eDisplayHollow                         ,
                         eDisplaySelected                       ,
                         eDisplayDRC                            ,
                         eDisplayFocused                        ,
                         eDisplayMultiFocused                   ,
                         eDisplayHollowDashed
                       );
```

TMechanicalLayerPair

```
TMechanicalLayerPair = Record
    Layer1 : TLayer;
    Layer2 : TLayer;
End;
```

TMirrorOperation

```
TMirrorOperation = (eHMirror,eVMirror);
```

TNetScope

```
TNetScope = (eNetScope_DifferentNetsOnly,
             eNetScope_SameNetOnly,
             eNetScope_AnyNet);
```

TNetTopology

```
TNetTopology         = ( eNetTopology_Shortest                    ,
                         eNetTopology_Horizontal                  ,
                         eNetTopology_Vertical                    ,
                         eNetTopology_DaisyChain_Simple           ,
                         eNetTopology_DaisyChain_MidDriven        ,
                         eNetTopology_DaisyChain_Balanced         ,
                         eNetTopology_Starburst
                       );
```

TObjectCreationMode

```
TObjectCreationMode  = ( eCreate_Default,
                         eCreate_GlobalCopy);
```

TObjectId

```
TObjectId = ( eNoObject           ,
              eArcObject          ,
              ePadObject          ,
              eViaObject          ,
              eTrackObject        ,
              eTextObject         ,
              eFillObject         ,
              eConnectionObject   ,
              eNetObject          ,
              eComponentObject    ,
              ePolyObject         ,
              eRegionObject       ,
              eComponentBodyObject,
              eDimensionObject    ,
              eCoordinateObject   ,
              eClassObject        ,
              eRuleObject         ,
              eFromToObject       ,
              eDifferentialPairObject,
              eViolationObject    ,
              eEmbeddedObject     ,
              eEmbeddedBoardObject,
              eTraceObject        ,
              eSpareViaObject     ,
              eBoardObject        ,
              eBoardOutlineObject,
            );
```

Note, the eTraceObject and eSpareViaObject values are for internal use only and are not used directly with PCB documents (these values are used for Signal Integrity and Situs auto routing modules).

TObjectSet

```
TObjectSet = Set of TObjectId;
```

See also

TObjectId

TOptionsObjectId

```
TOptionsObjectId = (eAbstractOptions,
                    eOutputOptions,
                    ePrinterOptions,
                    eGerberOptions,
                    eAdvancedPlacerOptions,
                    eDesignRuleCheckerOptions,
                    eSpecctraRouterOptions,
                    eAdvancedRouterOptions,
                    eEngineeringChangeOrderOptions,
                    eInteractiveRoutingOptions,
                    eSystemOptions,
                    ePinSwapOptions);
```

TOutputDriverType

```
TOutputDriverType    = ( eUnknownDriver                          ,
                         eProtelGerber                           ,
                         eProtelPlot_Composite                   ,
                         eProtelPlot_Final                       ,
                         eStandardDriver_Composite               ,
                         eStandardDriver_Final
                       );
```

TOutputPort

```
TOutputPort = (eOutputPortCom1,
               eOutputPortCom2,
               eOutputPortCom3,
               eOutputPortCom4,
               eOutputPortFile);
```

TPadCache

```
TPadCache                          = Record
    PlaneConnectionStyle           : TPlaneConnectionStyle;
    ReliefConductorWidth           : TCoord;
    ReliefEntries                  : SmallInt;
    ReliefAirGap                   : TCoord;
    PowerPlaneReliefExpansion      : TCoord;
    PowerPlaneClearance            : TCoord;
    PasteMaskExpansion             : TCoord;
    SolderMaskExpansion            : TCoord;
    Planes                         : Word;
    PlaneConnectionStyleValid      : TCacheState;
    ReliefConductorWidthValid      : TCacheState;
    ReliefEntriesValid             : TCacheState;
    ReliefAirGapValid              : TCacheState;
    PowerPlaneReliefExpansionValid : TCacheState;
    PasteMaskExpansionValid        : TCacheState;
    SolderMaskExpansionValid       : TCacheState;
    PowerPlaneClearanceValid       : TCacheState;
    PlanesValid                    : TCacheState;
```

```
End;
```

TPadMode

```
TPadMode = ( ePadMode_Simple,
             ePadMode_LocalStack,
             ePadMode_ExternalStack);
```

TParity

```
TParity = (eParityNone,
           eParityEven,
           eParityOdd,
           eParityMark,
           eParitySpace);
```

TPCBDragMode

```
TPcbDragMode = ( eDragNone
                 eDragAllTracks
                 eDragConnectedTracks);
```

TPCBObjectHandle

```
TPCBObjectHandle = Pointer;
```

TPCBString

```
TPCBString  = WideString;
```

TPlaceTrackMode

```
TPlaceTrackMode = ( ePlaceTrackNone,
                    ePlaceTrackAny,
                    ePlaceTrack9090,
                    ePlaceTrack4590,
                    ePlaceTrack90Arc);
```

TPlaneConnectionStyle

```
TPlaneConnectionStyle = ( ePlaneNoConnect,
                          ePlaneReliefConnect,
                          ePlaneDirectConnect);
```

TPlaneConnectStyle

```
TPlaneConnectStyle = (eReliefConnectToPlane,
                      eDirectConnectToPlane,
                      eNoConnect);
```

TPlaneDrawMode

```
TPlaneDrawMode       = ( ePlaneDrawoOutlineLayerColoured // <- Protel 99 SE style.
                         ePlaneDrawOutlineNetColoured,
                         ePlaneDrawInvertedNetColoured);
```

TPlotLayer

```
TPlotLayer = (eNullPlot,
              eTopLayerPlot,
              eMidLayer1Plot,
              eMidLayer2Plot,
```

```
eMidLayer3Plot,
eMidLayer4Plot,
eMidLayer5Plot,
eMidLayer6Plot,
eMidLayer7Plot,
eMidLayer8Plot,
eMidLayer9Plot,
eMidLayer10Plot,
eMidLayer11Plot,
eMidLayer12Plot,
eMidLayer13Plot,
eMidLayer14Plot,
eMidLayer15Plot,
eMidLayer16Plot,
eMidLayer17Plot,
eMidLayer18Plot,
eMidLayer19Plot,
eMidLayer20Plot,
eMidLayer21Plot,
eMidLayer22Plot,
eMidLayer23Plot,
eMidLayer24Plot,
eMidLayer25Plot,
eMidLayer26Plot,
eMidLayer27Plot,
eMidLayer28Plot,
eMidLayer29Plot,
eMidLayer30Plot,
eBottomLayerPlot,
eTopOverlayPlot,
eBottomOverlayPlot,
eTopPastePlot,
eBottomPastePlot,
eTopSolderPlot,
eBottomSolderPlot,
eInternalPlane1Plot,
eInternalPlane2Plot,
eInternalPlane3Plot,
eInternalPlane4Plot,
eInternalPlane5Plot,
eInternalPlane6Plot,
eInternalPlane7Plot,
eInternalPlane8Plot,
eInternalPlane9Plot,
eInternalPlane10Plot,
eInternalPlane11Plot,
eInternalPlane12Plot,
eInternalPlane13Plot,
eInternalPlane14Plot,
```

```
eInternalPlane15Plot,
eInternalPlane16Plot,
eDrillGuide_Top_BottomPlot,
eDrillGuide_Top_Mid1Plot,
eDrillGuide_Mid2_Mid3Plot,
eDrillGuide_Mid4_Mid5Plot,
eDrillGuide_Mid6_Mid7Plot,
eDrillGuide_Mid8_Mid9Plot,
eDrillGuide_Mid10_Mid11Plot,
eDrillGuide_Mid12_Mid13Plot,
eDrillGuide_Mid14_Mid15Plot,
eDrillGuide_Mid16_Mid17Plot,
eDrillGuide_Mid18_Mid19Plot,
eDrillGuide_Mid20_Mid21Plot,
eDrillGuide_Mid22_Mid23Plot,
eDrillGuide_Mid24_Mid25Plot,
eDrillGuide_Mid26_Mid27Plot,
eDrillGuide_Mid28_Mid29Plot,
eDrillGuide_Mid30_BottomPlot,
eDrillGuide_SpecialPlot,
eKeepOutLayerPlot,
eMechanical1Plot,
eMechanical2Plot,
eMechanical3Plot,
eMechanical4Plot,
eMechanical5Plot,
eMechanical6Plot,
eMechanical7Plot,
eMechanical8Plot,
eMechanical9Plot,
eMechanical10Plot,
eMechanical11Plot,
eMechanical12Plot,
eMechanical13Plot,
eMechanical14Plot,
eMechanical15Plot,
eMechanical16Plot,
eDrillDrawing_Top_BottomPlot,
eDrillDrawing_Top_Mid1Plot,
eDrillDrawing_Mid2_Mid3Plot,
eDrillDrawing_Mid4_Mid5Plot,
eDrillDrawing_Mid6_Mid7Plot,
eDrillDrawing_Mid8_Mid9Plot,
eDrillDrawing_Mid10_Mid11Plot,
eDrillDrawing_Mid12_Mid13Plot,
eDrillDrawing_Mid14_Mid15Plot,
eDrillDrawing_Mid16_Mid17Plot,
eDrillDrawing_Mid18_Mid19Plot,
eDrillDrawing_Mid20_Mid21Plot,
```

```
                eDrillDrawing_Mid22_Mid23Plot,
                eDrillDrawing_Mid24_Mid25Plot,
                eDrillDrawing_Mid26_Mid27Plot,
                eDrillDrawing_Mid28_Mid29Plot,
                eDrillDrawing_Mid30_BottomPlot,
                eDrillDrawing_SpecialPlot,
                eTopPadMasterPlot,
                eBottomPadMasterPlot);
```

TPlotPolygonProc

```
TPlotPolygonProc  = Procedure(APoly : PGPC_Polygon) Of Object;
```

TPlotterLanguage

```
TPlotterLanguage = ( ePlotterLanguageHPGL,
                     ePlotterLanguageDMPL);
```

TPolygonReliefAngle

```
TPolygonReliefAngle  = ( ePolygonReliefAngle_45,
                         ePolygonReliefAngle_90);
```

TPolygonRepourMode

```
TPolygonRepourMode = ( eNeverRepour
                       eThresholdRepour
                       eAlwaysRepour);
```

TPolygonType

```
TPolygonType = ( eSignalLayerPolygon,
                 eSplitPlanePolygon);
```

TPolyHatchStyle

```
TPolyHatchStyle = ( ePolyHatch90,
                    ePolyHatch45,
                    ePolyVHatch,
                    ePolyHHatch,
                    ePolyNoHatch,
                    ePolySolid);
```

TPolyRegionKind

```
TPolyRegionKind = ( ePolyRegionKind_Copper,
                    ePolyRegionKind_Cutout,
                    ePolyRegionKind_NamedRegion);
```

TPolySegmentType

```
TPolySegmentType = ( ePolySegmentLine,
                     ePolySegmentArc);
```

TPrinterBatch

```
TPrinterBatch = ( ePlotPerSheet,
                  ePanelize);
```

TPrinterComposite

```
TPrinterComposite    = ( eColorComposite,
```

```
                        eMonoComposite);
```

TRouteLayer

```
TRouteLayer = (eRLLayerNotUsed,
               eRLRouteHorizontal,
               eRLRouteVertical,
               eRLRouteSingleLayer,
               eRLRoute_1_OClock,
               eRLRoute_2_OClock,
               eRLRoute_4_OClock,
               eRLRoute_5_OClock,
               eRLRoute_45_Up,
               eRLRoute_45_Down,
               eRLRouteFanout,
               eRLRouteAuto);
```

TRouteVia

```
TRouteVia = (eViaThruHole,
             eViaBlindBuriedPair,
             eViaBlindBuriedAny,
             eViaNone);
TRoutingWidthMode
TRoutingWidthMode =     (eRoutingWidth_Default,
                         eRoutingWidth_Min,
                         eRoutingWidth_Preferred,
                         eRoutingWidth_Max);
```

TRuleKind

```
TRuleKind = ( eRule_Clearance,
              eRule_ParallelSegment,
              eRule_MaxMinWidth,
              eRule_MaxMinLength,
              eRule_MatchedLengths,
              eRule_DaisyChainStubLength,
              eRule_PowerPlaneConnectStyle,
              eRule_RoutingTopology,
              eRule_RoutingPriority,
              eRule_RoutingLayers,
              eRule_RoutingCornerStyle,
              eRule_RoutingViaStyle,
              eRule_PowerPlaneClearance,
              eRule_SolderMaskExpansion,
              eRule_PasteMaskExpansion,
              eRule_ShortCircuit,
              eRule_BrokenNets,
              eRule_ViasUnderSMD,
              eRule_MaximumViaCount,
              eRule_MinimumAnnularRing,
              eRule_PolygonConnectStyle,
              eRule_AcuteAngle,
```

```
                eRule_ConfinementConstraint,
                eRule_SMDToCorner,
                eRule_ComponentClearance,
                eRule_ComponentRotations,
                eRule_PermittedLayers,
                eRule_NetsToIgnore,
                eRule_SignalStimulus,
                eRule_Overshoot_FallingEdge,
                eRule_Overshoot_RisingEdge,
                eRule_Undershoot_FallingEdge,
                eRule_Undershoot_RisingEdge,
                eRule_MaxMinImpedance,
                eRule_SignalTopValue,
                eRule_SignalBaseValue,
                eRule_FlightTime_RisingEdge,
                eRule_FlightTime_FallingEdge,
                eRule_LayerStack,
                eRule_MaxSlope_RisingEdge,
                eRule_MaxSlope_FallingEdge,
                eRule_SupplyNets,
                eRule_MaxMinHoleSize,
                eRule_TestPointStyle,
                eRule_TestPointUsage,
                eRule_UnconnectedPin,
                eRule_SMDToPlane,
                eRule_SMDNeckDown,
                eRule_LayerPair,
                eRule_FanoutControl,
                eRule_MaxMinHeight,
                eRule_DifferentialPairsRouting);
```

TRuleLayerKind

```
TRuleLayerKind = (eRuleLayerKind_SameLayer,
                  eRuleLayerKind_AdjacentLayer);
```

TSameNamePadstackReplacementMode

```
TSameNamePadstackReplacementMode
                    = ( eAskUser
                        eReplaceOne
                        eReplaceAll
                        eRenameOne
                        eRenameAll
                        eKeepOneExisting
                        eKeepAllExisting
                      );
```

TScopeId

```
ScopeId = (eScope1, eScope2);
```

TScopeKind

```
TScopeKind = ( eScopeKindBoard,
               {Lowest Precedence}
               eScopeKindLayerClass,
               eScopeKindLayer,
               eScopeKindObjectKind,
               eScopeKindFootprint,
               eScopeKindComponentClass,
               eScopeKindComponent,
               eScopeKindNetClass,
               eScopeKindNet,
               eScopeKindFromToClass,
               eScopeKindFromTo,
               eScopeKindPadClass,
               eScopeKindPadSpec,
               eScopeKindViaSpec,
               eScopeKindFootprintPad,
               eScopeKindPad,
               eScopeKindRegion
               {Highest Precedence});
```

TScopeObjectId

```
TScopeObjectId = ( eRuleObject_None,
                   eRuleObject_Wire,
                   eRuleObject_Pin,
                   eRuleObject_Smd,
                   eRuleObject_Via,
                   eRuleObject_Fill,
                   eRuleObject_Polygon,
                   eRuleObject_KeepOut);
```

TShape

```
TShape = (eNoShape,
          eRounded,
          eRectangular,
          eOctagonal,
          eCircleShape,
          eArcShape,
          eTerminator,
          eRoundRectShape,
          eRotatedRectShape
          eRoundedRectangular
);
```

TSignalLevel

```
TSignalLevel = ( eLowLevel,
                 eHighLevel);
```

TSortBy

```
TSortBy = (eSortByAXThenAY,
```

```
            eSortByAXThenDY,
            eSortByAYThenAX,
            eSortByDYThenAX,
            eSortByName);
```

TSmartRouteMode

```
TSmartRouteMode = (eSRIgnoreObstacle,
                   eSRAvoidObstacle,
                   eSRWalkAroundObstacle,
                   eSRPushObstacle);
```

TStimulusType

```
TStimulusType = (eConstantLevel,
                 eSinglePulse,
                 ePeriodicPulse);
```

TStopBits

```
TStopBits            = ( eStopBits1                              ,
                         eStopBits1_5                            ,
                         eDataBits2
                       );
```

TString (PCB)

```
TString = ShortString;
```

TStrokeArray

```
TStrokeArray = Array[1..kMaxStrokes] Of TStrokeRecord;
```

TStrokeRecord

```
TStrokeRecord = Record
   X1, Y1, X2, Y2 : TCoord;
End;
```

TTestPointStyle

```
TTestPointStyle = (eExistingSMDBottom,
                   eExistingTHPadBottom,
                   eExistingTHViaBottom,
                   eNewSMDBottom,
                   eNewTHBottom,
                   eExistingSMDTop,
                   eExistingTHPadTop,
                   eExistingTHViaTop,
                   eNewSMDTop,
                   eNewTHTop);
```

TTestpointValid

```
TTestpointValid      = ( eRequire,
                         eInvalid,
                         eIgnore);
```

TTextAlignment

```
TTextAlignment       = ( eNoneAlign                              ,
```

```
                        eCentreAlign                              ,
                        eLeftAlign                                ,
                        eRightAlign                               ,
                        eTopAlign                                 ,
                        eBottomAlign
                      );
```

TTextAutoposition

```
TTextAutoposition    = ( eAutoPos_Manual,
                         eAutoPos_TopLeft,
                         eAutoPos_CenterLeft,
                         eAutoPos_BottomLeft,
                         eAutoPos_TopCenter,
                         eAutoPos_CenterCenter,
                         eAutoPos_BottomCenter,
                         eAutoPos_TopRight,
                         eAutoPos_CenterRight,
                         eAutoPos_BottomRight);
```

TUnit

```
TUnit = (eMetric, eImperial);
```

TUnitStyle

```
TUnitStyle = ( eNoUnits,
               eYesUnits,
               eParenthUnits);
```

TViewableObjectID

```
TViewableObjectID = (eViewableObject_None                      ,
                     eViewableObject_Arc                       ,
                     eViewableObject_Pad                       ,
                     eViewableObject_Via                       ,
                     eViewableObject_Track                     ,
                     eViewableObject_Text                      ,
                     eViewableObject_Fill                      ,
                     eViewableObject_Connection                ,
                     eViewableObject_Net                       ,
                     eViewableObject_Component                 ,
                     eViewableObject_Poly                      ,
                     eViewableObject_LinearDimension           ,
                     eViewableObject_AngularDimension          ,
                     eViewableObject_RadialDimension           ,
                     eViewableObject_LeaderDimension           ,
                     eViewableObject_DatumDimension            ,
                     eViewableObject_BaselineDimension         ,
                     eViewableObject_CenterDimension           ,
                     eViewableObject_OriginalDimension         ,
                     eViewableObject_LinearDiameterDimension   ,
                     eViewableObject_RadialDiameterDimension   ,
                     eViewableObject_Coordinate                ,
```

```
eViewableObject_Class                     ,
eViewableObject_Rule_Clearance            ,
eViewableObject_Rule_ParallelSegment      ,
eViewableObject_Rule_MaxMinWidth          ,
eViewableObject_Rule_MaxMinLength         ,
eViewableObject_Rule_MatchedLengths       ,
eViewableObject_Rule_DaisyChainStubLength ,
eViewableObject_Rule_PowerPlaneConnectStyle,
eViewableObject_Rule_RoutingTopology      ,
eViewableObject_Rule_RoutingPriority      ,
eViewableObject_Rule_RoutingLayers        ,
eViewableObject_Rule_RoutingCornerStyle   ,
eViewableObject_Rule_RoutingViaStyle      ,
eViewableObject_Rule_PowerPlaneClearance  ,
eViewableObject_Rule_SolderMaskExpansion  ,
eViewableObject_Rule_PasteMaskExpansion   ,
eViewableObject_Rule_ShortCircuit         ,
eViewableObject_Rule_BrokenNets           ,
eViewableObject_Rule_ViasUnderSMD         ,
eViewableObject_Rule_MaximumViaCount      ,
eViewableObject_Rule_MinimumAnnularRing   ,
eViewableObject_Rule_PolygonConnectStyle  ,
eViewableObject_Rule_AcuteAngle           ,
eViewableObject_Rule_ConfinementConstraint ,
eViewableObject_Rule_SMDToCorner          ,
eViewableObject_Rule_ComponentClearance   ,
eViewableObject_Rule_ComponentRotations   ,
eViewableObject_Rule_PermittedLayers      ,
eViewableObject_Rule_NetsToIgnore         ,
eViewableObject_Rule_SignalStimulus       ,
eViewableObject_Rule_Overshoot_FallingEdge ,
eViewableObject_Rule_Overshoot_RisingEdge  ,
eViewableObject_Rule_Undershoot_FallingEdge,
eViewableObject_Rule_Undershoot_RisingEdge ,
eViewableObject_Rule_MaxMinImpedance      ,
eViewableObject_Rule_SignalTopValue       ,
eViewableObject_Rule_SignalBaseValue      ,
eViewableObject_Rule_FlightTime_RisingEdge ,
eViewableObject_Rule_FlightTime_FallingEdge,
eViewableObject_Rule_LayerStack           ,
eViewableObject_Rule_MaxSlope_RisingEdge  ,
eViewableObject_Rule_MaxSlope_FallingEdge  ,
eViewableObject_Rule_SupplyNets           ,
eViewableObject_Rule_MaxMinHoleSize       ,
eViewableObject_Rule_TestPointStyle       ,
eViewableObject_Rule_TestPointUsage       ,
eViewableObject_Rule_UnconnectedPin       ,
eViewableObject_Rule_SMDToPlane           ,
eViewableObject_Rule_SMDNeckDown          ,
```

```
    eViewableObject_Rule_LayerPair            ,
    eViewableObject_Rule_FanoutControl        ,
    eViewableObject_Rule_MaxMinHeight         ,
    eViewableObject_Rule_DifferentialPairs    ,
    eViewableObject_FromTo                    ,
    eViewableObject_DifferentialPair          ,
    eViewableObject_Violation                 ,
    eViewableObject_Board                     ,
    eViewableObject_BoardOutline              ,
    eViewableObject_Group                     ,
    eViewableObject_Clipboard                 ,
    eViewableObject_SplitPlane,
    eViewableObject_EmbeddedBoard,
    eViewableObject_Region,
    eViewableObject_ComponentBody);
```

Notes

This TViewableObjectID type is mainly used by the Inspector and List views in Altium Designer and is an extension of TObjectID type.

TWidthArray

TWidthArray = Array[cMinLayer_WidthRule..cMaxLayer_WidthRule] Of TCoord;

PCB Constants

AllLayers

```
AllLayers = [MinLayer..eConnectLayer];
```

AllObjects

```
AllObjects = [FirstObjectId..LastObjectId];
```

AllPrimitives

```
AllPrimitives = [ eArcObject         ,
                  eViaObject         ,
                  eTrackObject       ,
                  eTextObject        ,
                  eFillObject        ,
                  ePadObject         ,
                  eComponentObject   ,
                  eNetObject         ,
                  ePolyObject        ,
                  eDimensionObject   ,
                  eCoordinateObject  ,
                  eEmbeddedObject    ,
                  eEmbeddedBoardObject,
                  eFromToObject      ,
                  eConnectionObject,
                  ePolyRegionObject,
                  eComponentBodyObject
                  ];
```

cAdvPCB

```
cAdvPCB = 'AdvPCB';
```

cBoardSideStrings constant

```
cBoardSideStrings : Array [TBoardSide] Of String[20] =
('Top Side','Bottom Side');
```

cComonentCollisionCheckModeStrings constant

```
cComponentCollisionCheckModeStings : Array [TComponentCollisionCheckMode] Of String[22]=
('Quick Check Mode','Multi-Layer Check Mode','Full Check Mode','Component Body Mode');
```

cDefaultLayerDrawingOrder constant

```
cDefaultLayerDrawingOrder : TDrawingOrderArray = (
eBackGroundLayer,
eMultiLayer,
eTopOverlay,
eBottomOverlay,
eConnectLayer,
eNoLayer,
eTopLayer,
eMidLayer1,
eMidLayer2,
```

```
eMidLayer3,
eMidLayer4,
eMidLayer5,
eMidLayer6,
eMidLayer7,
eMidLayer8,
eMidLayer9,
eMidLayer10,
eMidLayer11,
eMidLayer12,
eMidLayer13,
eMidLayer14,
eMidLayer15,
eMidLayer16,
eMidLayer17,
eMidLayer18,
eMidLayer19,
eMidLayer20,
eMidLayer21,
eMidLayer22,
eMidLayer23,
eMidLayer24,
eMidLayer25,
eMidLayer26,
eMidLayer27,
eMidLayer28,
eMidLayer29,
eMidLayer30,
eBottomLayer,
eTopPaste,
eBottomPaste,
eTopSolder,
eBottomSolder,
eInternalPlane1,
eInternalPlane2,
eInternalPlane3,
eInternalPlane4,
eInternalPlane5,
eInternalPlane6,
eInternalPlane7,
eInternalPlane8,
eInternalPlane9,
eInternalPlane10,
eInternalPlane11,
eInternalPlane12,
eInternalPlane13,
eInternalPlane14,
eInternalPlane15,
eInternalPlane16,
```

```
eDrillGuide,
eKeepOutLayer,
eMechanical1,
eMechanical2,
eMechanical3,
eMechanical4,
eMechanical5,
eMechanical6,
eMechanical7,
eMechanical8,
eMechanical9,
eMechanical10,
eMechanical11,
eMechanical12,
eMechanical13,
eMechanical14,
eMechanical15,
eMechanical16,
eDrillDrawing,
eGridColor1,
eBackGroundLayer,
eBackGroundLayer,
eBackGroundLayer,
eBackGroundLayer,
eBackGroundLayer);
```

cDir_NONE

```
cDir_NONE     = [];
```

cDir_ANY

```
cDir_ANY      = [eDir_N..eDir_NW];
```

cDir_Diagonal

```
cDir_Diagonal = [eDir_NE, eDir_SE, eDir_SW, eDir_NW];
```

cDir_HorVert

```
cDir_HorVert  = cDir_ANY - cDir_Diagonal;
```

cLayerStrings

```
cLayerStrings : Array[TLayer] Of String
              = ( 'NoLayer'        ,
                  'TopLayer'       ,
                  'MidLayer1'      ,
                  'MidLayer2'      ,
                  'MidLayer3'      ,
                  'MidLayer4'      ,
                  'MidLayer5'      ,
                  'MidLayer6'      ,
                  'MidLayer7'      ,
                  'MidLayer8'      ,
                  'MidLayer9'      ,
```

```
'MidLayer10'     ,
'MidLayer11'     ,
'MidLayer12'     ,
'MidLayer13'     ,
'MidLayer14'     ,
'MidLayer15'     ,
'MidLayer16'     ,
'MidLayer17'     ,
'MidLayer18'     ,
'MidLayer19'     ,
'MidLayer20'     ,
'MidLayer21'     ,
'MidLayer22'     ,
'MidLayer23'     ,
'MidLayer24'     ,
'MidLayer25'     ,
'MidLayer26'     ,
'MidLayer27'     ,
'MidLayer28'     ,
'MidLayer29'     ,
'MidLayer30'     ,
'BottomLayer'    ,
'TopOverlay'     ,
'BottomOverlay'  ,
'TopPaste'       ,
'BottomPaste'    ,
'TopSolder'      ,
'BottomSolder'   ,
'InternalPlane1' ,
'InternalPlane2' ,
'InternalPlane3' ,
'InternalPlane4' ,
'InternalPlane5' ,
'InternalPlane6' ,
'InternalPlane7' ,
'InternalPlane8' ,
'InternalPlane9' ,
'InternalPlane10',
'InternalPlane11',
'InternalPlane12',
'InternalPlane13',
'InternalPlane14',
'InternalPlane15',
'InternalPlane16',
'DrillGuide'     ,
'KeepOutLayer'   ,
'Mechanical1'    ,
'Mechanical2'    ,
'Mechanical3'    ,
```

```
                    'Mechanical4'    ,
                    'Mechanical5'    ,
                    'Mechanical6'    ,
                    'Mechanical7'    ,
                    'Mechanical8'    ,
                    'Mechanical9'    ,
                    'Mechanical10'   ,
                    'Mechanical11'   ,
                    'Mechanical12'   ,
                    'Mechanical13'   ,
                    'Mechanical14'   ,
                    'Mechanical15'   ,
                    'Mechanical16'   ,
                    'DrillDrawing'   ,
                    'MultiLayer'     ,
                    'ConnectLayer'   ,
                    'BackGroundLayer',
                    'DRCErrorLayer'  ,
                    'HighlightLayer' ,
                    'GridColor1'     ,
                    'GridColor10'    ,
                    'PadHoleLayer'   ,
                    'ViaHoleLayer');
```

cMaxTestPointStyle

```
cMaxTestPointStyle              = eNewTHTop;
```

cMinTestPointStyle

```
cMinTestPointStyle = eExistingSMDBottom;
```

cMidLayers

```
cMidLayers       : Set Of TLayer = [eMidLayer1 .. eMidLayer30];
```

cMinLayer_WidthRule

```
cMinLayer_WidthRule = eTopLayer;
```

cMaxLayer_WidthRule

```
cMaxLayer_WidthRule = eBottomLayer;
cRoutingWidthModeStrings
cRoutingWidthModeStrings : Array[TRoutingWidthMode] Of String[20]
                         = ('User Choice'   , //eRoutingWidth_Default
                            'Rule Minimum'  , //eRoutingWidth_Min
                            'Rule Preferred', //eRoutingWidth_Preferred
                            'Rule Maximum'    //eRoutingWidth_Max);
```

cRuleIdStrings

```
cRuleIdStrings : Array [TRuleKind] Of String[21]
               = ( 'Clearance'           ,
                   'ParallelSegment'     ,
                   'Width'               ,
                   'Length'              ,
```

```
                    'MatchedLengths'       ,
                    'StubLength'           ,
                    'PlaneConnect'         ,
                    'RoutingTopology'      ,
                    'RoutingPriority'      ,
                    'RoutingLayers'        ,
                    'RoutingCorners'       ,
                    'RoutingVias'          ,
                    'PlaneClearance'       ,
                    'SolderMaskExpansion'  ,
                    'PasteMaskExpansion'   ,
                    'ShortCircuit'         ,
                    'UnRoutedNet'          ,
                    'ViasUnderSMD'         ,
                    'MaximumViaCount'      ,
                    'MinimumAnnularRing'   ,
                    'PolygonConnect'       ,
                    'AcuteAngle'           ,
                    'RoomDefinition'       ,
                    'SMDToCorner'          ,
                    'ComponentClearance'   ,
                    'ComponentOrientations',
                    'PermittedLayers'      ,
                    'NetsToIgnore'         ,
                    'SignalStimulus'       ,
                    'OvershootFalling'     ,
                    'OvershootRising'      ,
                    'UndershootFalling'    ,
                    'UndershootRising'     ,
                    'MaxMinImpedance'      ,
                    'SignalTopValue'       ,
                    'SignalBaseValue'      ,
                    'FlightTimeRising'     ,
                    'FlightTimeFalling'    ,
                    'LayerStack'           ,
                    'SlopeRising'          ,
                    'SlopeFalling'         ,
                    'SupplyNets'           ,
                    'HoleSize'             ,
                    'Testpoint'            ,
                    'TestPointUsage'       ,
                    'UnConnectedPin'       ,
                    'SMDToPlane'           ,
                    'SMDNeckDown'          ,
                    'LayerPairs'           ,
                    'FanoutControl'        ,
                    'Height',
                    'DiffPairsRouting'
);
```

cTextAutopositionStrings

```
cTextAutopositionStrings : Array[TTextAutoPosition] Of String[20]
                         = ( 'Manual'      ,
                             'Left-Above'  ,
                             'Left-Center' ,
                             'Left-Below'  ,
                             'Center-Above',
                             'Center'      ,
                             'Center-Below',
                             'Right-Above' ,
                             'Right-Center',
                             'Right-Below');
```

cTestPointPriorityHigh

```
cTestPointPriorityHigh = Ord(cMinTestPointStyle);
```

cTestPointPriorityLow

```
cTestPointPriorityLow = Ord(cMaxTestPointStyle);
```

cWidthRuleLayers

```
cWidthRuleLayers = [cMinLayer_WidthRule..cMaxLayer_WidthRule];
```

FirstObjectId

```
FirstObjectId = eArcObject;
```

InternalUnits

```
InternalUnits = 10000;
```

InternalPlanes

```
InternalPlanes   : Set Of TLayer = [eInternalPlane1..eInternalPlane16];
```

kDiameterSymbolANSI

```
kDiameterSymbolANSI    = #$F8;
```

kDiameterSymbolUnicode

```
kDiameterSymbolUnicode = #$3A6;
```

kDegreeSymbol

```
kDegreeSymbol          = #$B0;
```

k1Inch

```
k1Inch = 1000 * InternalUnits;
```

Notes

1 mil = 10000 internal units

1 inch = 1000 mils

1 inch = 2.54 cm

1 inch = 25.4 mm and 1 cm = 10 mm

PCB object's coordinates are usually in mils or mm depending on the board's current measurement units.

kDefaultArcResolution

```
kDefaultArcResolution            = k1Mil Div 2;
```

Notes

1 mil = 10000 internal units

1 inch = 1000 mils

1 inch = 2.54 cm

1 inch = 25.4 mm and 1 cm = 10 mm

PCB object's coordinates are usually in mils or mm depending on the board's current measurement units.

k1Mil

```
k1Mil = 1 * InternalUnits;
```

Notes

1 mil = 10000 internal units

1 inch = 1000 mils

1 inch = 2.54 cm

1 inch = 25.4 mm and 1 cm = 10 mm

PCB object's coordinates are usually in mils or mm depending on the board's current measurement units.

kMaxCoord

```
kMaxCoord = 99999  * InternalUnits;
```

kMinCoord

```
kMinCoord = 0 * InternalUnits;
```

kMaxInternalPlane

```
kMaxInternalPlane = eInternalPlane16;
```

kMinInternalPlane

```
kMinInternalPlane = eInternalPlane1;
```

kMaxPolySize

```
kMaxPolySize = 5000;
```

LastObjectId

```
LastObjectId = eEmbeddedBoardObject;
```

kMaxStrokes

```
kMaxStrokes = 2000;
```

MaxLayer

```
MaxLayer = eViaHoleLayer;
```

Notes

Refer to Layer2String and String2Layer functions in the PCB Functions topic.

MaxBoardLayer

```
MaxBoardLayer = eMultiLayer;
```

MaxLogicalTextSize

```
MaxLogicalTextSize = k1Inch;
```

MaxRouteLayer

```
MaxRouteLayer = eBottomLayer;
```

MaxMechanicalLayer constant

```
MaxMechanicalLayer = eMechanical16;
```

MechanicalLayers

```
MechanicalLayers : Set Of TLayer = [eMechanical1..eMechanical16];
```

MinLayer

```
MinLayer = eTopLayer;
```

Notes

Refer to Layer2String and String2Layer functions in the PCB Functions topic.

MinMechanicalLayer constant

MinMechanicalLayer = eMechanical1;

Numbers

```
Numbers          : Set Of Char    = ['0'..'9'];
```

WideStringObjects

```
WideStringObjects = [ eTextObject, eDimensionObject, eCoordinateObject, eComponentObject];
```

PCB Messages

Overview

The PCB Messages are messages that are broadcasted by the PCB Editor server. There are different types of messages that describe a specific action within the PCB server.

Normally the PCB message constants are used for the **IPCB_ServerInterface.SendMessageToRobots** method.

Syntax

PCBM_NullMessage = 0;
PCBM_BeginModify = 1;
PCBM_BoardRegisteration = 2;
PCBM_EndModify = 3;
PCBM_CancelModify = 4;
PCBM_Create = 5;
PCBM_Destroy = 6;
PCBM_ProcessStart = 7;
PCBM_ProcessEnd = 8;
PCBM_ProcessCancel = 9;
PCBM_YieldToRobots = 10;
PCBM_CycleEnd = 11;
PCBM_CycleStart = 12;
PCBM_SystemInvalid = 13;
PCBM_SystemValid = 14;
PCBM_ViewUpdate = 15;
PCBM_UnDoRegister = 16;

c_BroadCast = Nil;
c_NoEventData = Nil;
c_FromSystem = Nil;

See also

SendMessageToRobots method

SignalLayers

SignalLayers : Set Of TLayer = [eTopLayer.. eBottomLayer];

PCB Functions

The major PCB Functions are defined and implemented in the RT_PCBProcs unit.

Unit conversion functions

```
Function  RealToMils      (C : TReal)   : TReal;
Function  RealToMMs       (C : TReal)   : TReal;
Function  CoordToMils     (C : TCoord)  : TReal;
Function  CoordToMMs      (C : TCoord)  : TReal;
Function  MilsToCoord     (M : TReal)   : TCoord;
Function  MMsToCoord      (M : TReal)   : TCoord;
Function  MilsToRealCoord(M : TReal)   : TReal;
Function  MMsToRealCoord  (M : TReal)   : TReal;

Function  MetricString   (Var S        : TString;
                          DefaultUnits : TUnit)   : Boolean;
Function  ImperialString(Var S         : TString;
                          DefaultUnits : TUnit)   : Boolean;

Procedure StringToCoordUnit(S     : TString;
                            Var C : TCoord;
                            Var U : TUnit);

Procedure StringToRealUnit (S     : TString;
                            Var R : TReal;
                            Var U : TUnit);

Function  CoordUnitToString(C : TCoord;
                            U : TUnit) : TString;

Function  RealUnitToString (R : TReal;
                            U : TUnit) : TString;
```

Angle and Trigonometric functions

```
Function  Degrees2Radians       (Angle             : TAngle)        : TReal;
Function  AngleToFormattedString(TextValue         : TReal;
                                  TextFormat        : TString;
                                  TextDimensionUnit : TDimensionUnit;
                                  TextPrecision     : Integer;
                                  TextPrefix        : TString;
                                  TextSuffix        : TString;
                                  UseTTFonts        : Boolean) : TString;

Function  DistanceToFormattedString    (TextValue         : TReal;
                                        TextFormat        : TString;
                                        TextDimensionUnit : TDimensionUnit;
                                        TextPrecision     : Integer;
                                        TextPrefix        : TString;
                                        TextSuffix        : TString;
                                        DisplayUnit       : TUnit;
```

```
                                    DimensionKind     : TDimensionKind;
                                    UseTTFonts        : Boolean)        : TString;

Procedure NormalizeAngle            (Var Angle        : TAngle);

Procedure RotateCoordsAroundXY      (Var x, y         : TCoord;
                                     Xr, Yr           : TCoord;
                                     Angle            : TAngle);

Procedure FindZoomRect(Const FarRect, CloseRect : TCoordRect; Out ZoomRect : TCoordRect);
Overload;
Procedure FindZoomRect(Const FarRect, CloseRect : TCoordRect; Out ZoomRect : TCoordRect; Const
PrecisionFactor : Double); Overload;
```

Object Boundary Functions

```
Function  GetFillBLX                (P                : TPCBObjectHandle;
                                     ExpandBy         : TCoord)         : TCoord;
Function  GetFillBLY                (P                : TPCBObjectHandle;
                                     ExpandBy         : TCoord)         : TCoord;
Function  GetFillTLX                (P                : TPCBObjectHandle;
                                     ExpandBy         : TCoord)         : TCoord;
Function  GetFillTLY                (P                : TPCBObjectHandle;
                                     ExpandBy         : TCoord)         : TCoord;
Function  GetFillTRX                (P                : TPCBObjectHandle;
                                     ExpandBy         : TCoord)         : TCoord;
Function  GetFillTRY                (P                : TPCBObjectHandle;
                                     ExpandBy         : TCoord)         : TCoord;
Function  GetFillBRX                (P                : TPCBObjectHandle;
                                     ExpandBy         : TCoord)         : TCoord;
Function  GetFillBRY                (P                : TPCBObjectHandle;
                                     ExpandBy         : TCoord)         : TCoord;
```

Layer conversion functions

```
Function  Layer2String (Layer : TLayer) : TString;
Function  String2Layer (Layer : TString): TLayer;
```

Font Functions

```
Procedure EnumFontsW                (DC               : HDC;
                                     Const AFontName  : TPCBString;
                                     Items            : TWideStrings);
Procedure EnumFontsA                (DC               : HDC;
                                     Const AFontName  : TPCBString;
                                     Items            : TStrings);
Function  LoadFontNamesW            (Items            : TWideStrings)   : Integer;
Function  LoadFontNamesA            (Items            : TStrings)       : Integer;
Function  GetLocalizedFontName      (Const FontName   : TPCBString)     : TPCBString;
```

Locale Functions

```
Function  GetLocaleData(AID   : LCID; AFlag : DWORD) : TDynamicString;
```

```
Function  IsLocaleLanguageJapanese : Boolean;
Function  IsLocaleLanguageEnglish  : Boolean;
Function  IsLocaleLanguageAsian    : Boolean;
```

General Functions

GetIniFileName

```
Function GetIniFileName : AnsiString;
```

CoordsEqual

```
Function  CoordsEqual (c1, c2 : Double) : Boolean;
```

ConvertEncodedText2WideString

```
Function  ConvertEncodedText2WideString(Const EncodedText : TDynamicString) : TPCBString;
```

ConvertWideString2EncodedText

```
Function  ConvertWideString2EncodedText(Const WString     : TPCBString)     : TDynamicString;
```

StringListCopy

```
Function  StringListCopy               (AWideStringList : TWideStringList;
                                        AAnsiStringList : TStringList)      : Boolean;
```

StringToWideString

```
Function  StringToWideString           (const Str         : string)         : TPCBString;
```

Revision History

Date	Version No.	Revision
22-Nov-2005	V1.0	New product release
15-Feb-2006	V1.1	Updated for latest Altium Designer 6
13-Nov-2006	V1.2	Updated IPCB_Pad interface definition and added IPCB_Pad2 definition for Altium Designer 6.6
28-Feb-2008	V1.3	Updated Page Size to A4 and updated PCB Interfaces.
01-Sep-2011	-	Updated template.