C/PEG™

Portable Embedded GUI

API Reference Manual

C/PEG Library Release 1.60

Second Printing

January 2006

SWELL software

TABLE OF CONTENTS

FORWARD

We at Swell Software thank you for choosing **C/PEG^{TM}**.

This manual details the Application Programming Interface (API) of the C/PEG graphics library. It is intended as a quick reference guide for developers which may already be familiar with how C/PEG works and may need to review details on individual functions.

How This Manual is Organized

This manual is divided into several sections which logically group related functions.

- Object Functions

By far the largest collection of functions, this section deals with every function used by every object in the C/PEG library. If you are looking for the parameters taken by an object's Create function, this section is where to look.

- Structure Functions

This section deals with the structures used in the library. These structures are usually data only and are used widely in function calls. Among the lists of structures contained in the this section are PegRect, PegBrush and PegFont.

- Graphics Functions

This section details the functions used for drawing. It covers not only the drawing primitives such as line and text drawing, it also covers the functions used to begin drawing as well as capturing and moving regions of the screen.

- Library Functions

This section covers functions that do not relate to a single type of object or structure and are not graphics functions. This section includes functions which relate to the run-time environment as well as extracting generic information from any type of object.

- Message Queue Functions

Message passing is such an important part of C/PEG, that the functions covering it have their own section. This section describes every function of the message queue as well as the timers.

- Macros

This section covers all of the macros used in the library. If you were looking for an object or structure function and were unable to find it in any of the proceeding sections, chances are you are looking at a macro. Some macros in C/PEG are very easy to spot since they are all in capital letters. Other macros look a great deal like function calls, but they are simple enough that they are actually implemented as macros to save run-time resources. Most of the latter types are used to dereference an object to call a function or read or write a data member.

- String Library Functions

The C/PEG library implements a subset of the ANSI C string library for use when running in either ANSI or Unicode character encoding. This section details these functions.

- String Table Functions

Working with the string tables produced by the C/PEG WindowBuilder tool requires only a few functions and makes language management very simple. This sections details the functions necessary to work with the string tables including how to switch languages at run-time.

- Memory Management Functions

For systems which have extra video memory or where the compiler may not support the tasking model of the operatins system, C/PEG provides heap memory mangement functions to allow task safe dynamic memory.

- Integration Functions

This sections covers the functions necessary to successfully integrate C/ PEG with an operating system.

CHAPTER 1

C/PEG OBJECT FUNCTIONS

1.1 Overview

The following section lists all of the functions for the PegThing and derived objects as well as functions for working with the various C/PEG structures such as PegRect and PegMessage.

Each of the PegThing derived objects have a set of five related functions that aid application designers and developers in creating these objects at run time as well as defining new custom objects based on an existing object definition. What follows is a short discussion of these functions and how they are used to provide the building blocks for creating objects at run time.

Create

The Create function follows this general form:

```
Object *ObjectCreate(PegRect *pRect, ..., PEGUSHORT usId,
  PEGUSHORT usStyle);
```

Where Object would be the type of object to create. For instance, the PegThing create function is called PegThingCreate, the PegPanel create function is called PegPanelCreate and so forth.

The first parameter is always a pointer to a PegRect structure that defines the size and position of the object. The last two parameters are always the ID to assign the object and the style to assign the object. For some simpler objects, those are the only three parameters in the function call. For more complex objects, there may be as many as four more parameters that govern the text displayed by the object, or to whom the object should report when dismissed.

This function is used to create a fully formed object of a specific type. This function always calls the object type's CreateDefault function to allocate storage for the object and the object type's set function to set the objects data members.

CreateDefault

The CreateDefault function follows this form:

```
    Object *ObjectCreateDefault(void);
```

Where, again, Object would be the type of object to create.

This function is used primarily to allocate storage for a particular type of object and initialize the object to known default values. This function always calls the Init function of the object type to properly initialize the object.

Init

This Init function follows this form:

```
    void ObjectInit(Object *pObject);
```

Where Object would be the type of object to initialize.

This function sets the base type of the object as well as the default colors for the object. This function usually calls PegStatusInit to initialize the status of the object.

This function always calls the object type's SetDefFuncs function to properly set the default function pointers in the object to point to the default functions implemented by the object type.

Set

The Set function follows this general form:

```
    void ObjectSet(Object *pObject, PegRect *pRect, ..., PEGUSHORT
      usId, PEGUSHORT usStyle);
```

This function follows the same rules as the Create function, with the additional parameter of a pointer to an instance of the object type.

This function is always called by the Create function to set the object's data members.

This function usually calls the PegClientInit function to properly initialize the Client and Clip rectangles of pObject after pObject's Real rectangle has been set.

SetDefFuncs

This function has the following form:

```
    void ObjectSetDefFuncs(Object *pObject);
```

This function sets the default function pointers in pObject based on the object type. This function is called by the Init function.

Creation Summary

It may not be immediately apparent why the object creation functions are broke into small sections as they are. It may seem that it would overwhelm the application developer into thinking that they would have to remember each of these functions for each object in order to create any kind of PegThing object at run time.

This is actually not true. For the application developer, the choices are very clear. If the application knows the size, position, ID and style of the object, then the application would call the object's Create function. The object returned by that call would have gone through all of the listed functions in the right order to build a complete object.

Likewise, if the application needs a new object but does not know it's size, position, ID or style, then the application may call the object's CreateDefault function. The object returned from this call is formed and has the correct type, colors, status and function pointers for that type of object filled in properly. The rest of the object's data members are initialized to 0. Thus, it has no position, size, ID or style, or any other attribute associated with that type of object.

For application designers that wish to build new object types, the partitioning of this functionality is very important in that it allows the application designer to use portions of the base object's creation functions without being stuck in the mire of an 'all or nothing' approach.

For instance, if the application developer wished to create a custom object based on PegDecoratedButton and add another bitmap pointer to the object that would point at a bitmap that should be displayed with the object is disabled, the process would work something like this:

```
/* define the new object */
typedef struct
{
PEG_DECORATED_BUTTON_DELCARE,
PegBitmap *pDisabledBitmap;
} MyDecoratedButton;

/* the draw function for the new object */
void MyDecoratedButtonDraw(void *pThing);

/* the create function for the new object */
MyDecoratedButton *MyDecoratedButtonCreate(PegRect *pRect,
const PEGCHAR *pText, PegBitmap *pBitmap, PegBitmap
*pDisabledBitmap, PEGUSHORT usId, PEGUSHORT usStyle)
{
MyDecoratedButton *pmdb = (MyDecoratedButton *)
PEG_ALLOC(sizeof(MyDecoratedButton));
PegDecoratedButtonInit((PegDecoratedButton *)pmdb);
PegDecoratedButtonSet((PegDecoratedButton *)pmdb,
pRect, pText, pBitmap, usId, usStyle);
PegFuncPtrSet(pmdb, PFP_DRAW, MyDecoratedButtonDraw);
pmdb->pDisabledBitmap = pDisabledBitmap;
return(pmdb);
}
```

Table 1 (Custom Object Creation)

This code example demonstrates that custom objects are able to use pieces of it's base object's creation functions in order to effectively get it's job done.

Here, MyDecoratedButton uses the PegDecoratedButton's Init and Set functions to do most of the work of object creation. From this perspective, the application designer must know what they are doing. By calling the PegDecoratedButtonInit function, they are also calling PegDecoratedButtonSetDefFuncs and PegStatusInit implicitly. Likewise, the call to PegDecoratedButtonSet also calls PegClientInit. The implication of this is that with a little knowledge on the inner workings of object creation, it is simple to properly construct new object types and provide the same, reliable, single call object creation that the application developer enjoys with stock C/PEG objects.

There is a more discussion of these principles in the “C/PEG Programmers Manual” in the “The Mighty Thing” section.

Other Considerations

Every PegThing and derivative have a set of function pointer members in their structures. These function pointers aid in simplifying the decision on which function implementation to call for which object type at which time. They also allow the application designers to easily override behavior of a particular object.

C/PEG provides a set of functions and macros to make the process of dealing with these function tables very straight forward and simple. These functions and macros are dealt with in more detail in their respective sections later in this manual.

The point here is that it is never an error to call an object types version of a function. That may seem like a simple statement to make, but there's a little more to the subject than what is first evident.

For instance, take for example PegDecoratedButton. This object derives from PegButton, which derives from PegTextThing with, finally, derives from PegThing. Now, let us suppose that the application designer wished to override the Notify function of an instance of a PegDecoratedButton object. That is simple enough to do using this function:

```
    PegFuncPtrSet(pMyObject, PFP_NOTIFY, MyObjectNotify);
```

That replaces the default version of the Notify function that any PegDecoratedButton uses with the user defined function, MyObjectNotify. So far, so good.

Then, in the MyObjectNotify function, the application designer wishes to catch a user defined message, but leave all other message processing to the default PegDecoratedButton object. Simple enough, the application designer would just make sure to call PegDecoratedButtonNotify from his own MyObjectNotify and pass the return value back to the caller.

But, wait, is there a PegDecoratedButtonNotify function? If not, where is the application designer suppose to pass the default message processing for messages other than the user defined message in which he is interested?

It is true that not every object has a version of each and every function that is available in the function pointers that is specific to that object type. To clarify, not every object has a specific Notify function for that type of object, or a specific Add or AddToEnd function. Therefore, given the above, there may or may not be a PegDecoratedButtonNotify function if there is no reason to have one. It is possible that one of the base objects for a PegDecoratedButton object handles messages just fine and there is no need for a PegDecoratedButton object to override the Notify function.

So, how does the application developer keep track of what object has which function? Easy, they don't have to. It is always correct to defer the call from a custom object to the base object's version of that function when the custom object desires to execute the default implementation of the function for it's base type. Always.

In this case, it is correct for the application designer to call PegDecoratedButtonNotify from within his MyObjectNotify function.

The reason this works is for each C/PEG object, every function in every function pointer in the object is implemented in one of two ways. Either the object implements an actual function that performs some specialized task, or the function call is replaced by a deferral to the base object's version of the function. And, these deferrals are chained together so that each derived object need only call it's base object's implementation of the function in order to keep the chain intact.
“And that means what to me?”

The functions listed in this section only address the functions for each object type where the object actually implements a function. You won't see a listing in here for PegPromptAdd, even though it is okay to call that function from application code. PegPromptAdd is not implemented as a function, it is actually a deferral to PegTextThingAdd, which is, in turn, a deferral to PegThingAdd. PegThingAdd is listed here because it is an actual function.

To summarize, it is always correct to call an immediate base object's implementation of a function when that function is part of the object's function table. If the function is not listed here, it is because there is no actual implementation of that particular function, and that the call is really a deferral to the second generation base object's implementation.

Again, there is more on this subject in the “C/PEG Programmers Manual.”

1.2 Peg2DPolygonCreate

Synopsis:

```
Peg2DPolygon *Peg2DPolygonCreate(PegRect *pRect, PegPoint
  *pPoints, PEGUINT uiNumPoints, PEGUSHORT usId, PEGUSHORT
  usStyle);
```

Arguments:

pRect
: Pointer to a PegRect structure that defines the size and position of the object.

pPoints
: Pointer to an array of PegPoint structures that define the polygon.

uiNumPoints
: Defines the number of points in the array pointed to by pPoints.

usId
: ID for the object.

usStyle
: Style for the object.

Returns:

Upon success, this function will return a pointer to a fully formed Peg2DPolygon object.

Description:

Peg2DPolygon is a derivative of PegThing.

This function allocates memory for a Peg2DPolygon object, and initializes the object.

Errors:

This function may generate an assert if the memory for the new object can not be allocated and the C/PEG library was built with the PEG_USER_ASSERT directive defined.

Related Functions:

Peg2DPolygonCreateDefault, Peg2DPolygonSet, Peg2DPolygonInit

Notes:

The C/PEG library must be compiled with the PEG_FULL_GRAPHICS directive defined for this function to be available.

Polygon coordinates are always relative to a 0 left and 0 top base coordinate. This makes it much easier for the application developer to define complex polygons without the necessity of translating the polygon points to relative screen coordinates that are dependent on the object's position on the screen. When the points are assigned to the object, the object maps the points to it's relative screen address. The object also does this mapping when it is moved through a resize operation.

The style of PF_COPY causes the object to copy the pPoint array into a private array which is allocated by the object during the call to Peg2DPolygonSet. Situations where setting this style flag would be appropriate would be if the original points are declared as `const` or the application wishes to keep the original points from being modified. In normal operation for this object, the point array is modified to match the objects location on the display. If the application developer does not wish the original points to be modified, then setting PF_COPY in the usStyle parameter will prevent the object from using the original points, thus protecting them from modification.

The object also allocates a set of points that are used to hold the base points with the rotation angle applied to them. These are the actual set of points the object uses for drawing. The extra memory associated with this is a trade off for speed at run time. If this array did not exist, the object would be forced to recalculate the rotation points every time the object would drew itself. This would be computationally expensive and would be a greater stress on the system than the extra memory required to hold the rotated data points in an ancillary array.

1.3 Peg2DPolygonCreateDefault

Synopsis:

```
    Peg2DPolygonCreateDefault(void);
```

Arguments:

None

Returns:

A pointer to a Peg2DPolygon object. This object will have been initialized, but does not have size, position, an ID or a style.

Description:

This function allocates memory for a Peg2DPolygon object. The function pointers are filled in, the type is set to PEG_TYPE_2DPOLYGON and the colors for the line and background fill are set to WHITE and BLACK, respectively.

The object does not have a size or position, meaning that it's Real and Client rectangles have their members set to zero. It's ID and style are also zero.

Errors:

This function may generate an assert if it can not allocated the necessary memory for the object and the C/PEG library was compiled with the PEG_USER_ASSERT directive defined.

Related Functions:

Peg2DPolygonCreate, Peg2DPolygonInit, Peg2DPolygonSet

Notes:

The C/PEG library must be compiled with the PEG_FULL_GRAPHICS directive defined for this function to be available.

After creating this object, the easiest way to set up the object is to call Peg2DPolygonSet with the pointer returned from this function.

1.4 Peg2DPolygonDestroy

Synopsis:

```
    void Peg2DPolygonDestroy(void *pThing);
```

Arguments:

pThing

Pointer to a Peg2DPolygon object that is to be destroyed.

Returns:

None

Description:

This function is the proper way to destroy a Peg2DPolygon object. This function ensures that the object's internal PegPoint arrays are properly freed.

Errors:

The function may generate an assert if the pThing pointer is invalid and the C/PEG library was compiled with the PEG_USER_ASSERT directive defined.

Related Functions:

PegDestroy, Peg2DPolygonCreate, Peg2DPolygonCreateDefault

Notes:

The C/PEG library must be compiled with the PEG_FULL_GRAPHICS directive defined for this function to be available.

The object keeps track of it's two point arrays, which it frees during execution of the function. If the object was not created with the PF_COPY style flag, it will not attempt to free the original point array that was passed to it during creation.

Since this object derives from PegThing, it also calls PegThingDestroy to ensure that the object is properly removed from it's parent and the PegMessageQueue is cleared of any messages bound for this object.

1.5 Peg2DPolygonDraw

Synopsis:

```
void Peg2DPolygonDraw(void *pThing);
```

Arguments:

pThing
    Pointer to a Peg2DPolygon object that will be drawing.

Returns:

None

Description:

This function is used to draw a Peg2DPolygon object. The object uses it's internal rotated point array to draw the polygon.

Errors:

This function may generate an assert if the pThing parameter is invalid and the C/PEG library was compiled with the PEG_USER_ASSERT directive defined.

Related Functions:

PegDraw, Peg2DPolygonLineColorSet, Peg2DPolygonFillColorSet

Notes:

The C/PEG library must be compiled with the PEG_FULL_GRAPHICS directive defined for this function to be available.

The background is first filled with either the background color for the object (PCI_NORMAL) or it's parent's background color if the AF_TRANSPARENT style flag is set. The exception to this is if the AF_TRANSPARENT and FF_NONE style flags are both set, in which case, the object will not draw a background at all.

The object then draws the polygon. If the PF_FILLED style flag is set, the polygon will draw filled, otherwise, it will draw as an outline.

1.6 Peg2DPolygonFillColorSet

Synopsis:

```
void Peg2DPolygonFillColorSet(Peg2DPolygon *pdp, PEGCOLORVAL
  c);
```

Arguments:

pdp
: Pointer to a Peg2DPolygon object.

c
: A color value

Returns:

None

Description:

This function sets the fill color used for filling the polygon. The PF_FILLED style flag must be set for the object for this fill color to have any effect when drawing.

This is not the color that is used to draw the background of the object. That color is the object's PCI_NORMAL color index and may be set using PegColorSet.

Errors:

This function may generate an assert if pdp is an invalid pointer and the C/PEG library was compiled with the PEG_USER_ASSERT directive defined.

Related Functions:

Peg2DPolygonLineColorSet, PegColorSet, PegRectRegionInvalidate

Notes:

The C/PEG library must be compiled with the PEG_FULL_GRAPHICS directive defined for this function to be available.

It is important to note that, internally, this function calls PegColorSet. This is important because PegColorSet calls PegRectRegionInvalidate, which, in multi-threaded systems, will lock the C/PEG screen resource.

1.7 Peg2DPolygonInit

Synopsis:

```
void Peg2DPolygonInit(Peg2DPolygon *p2dpoly);
```

Arguments:

p2dpoly
    Pointer to a Peg2DPolygon object.

Returns:

None

Description:

This function initializes a Peg2DPolygon object. In doing so, the object's function pointers are set properly, it's type is set to PEG_TYPE_2DPOLYGON and it's iLineWidth member is set to 1.

Errors:

This function may generate an assert if p2dpoly is invalid and the C/PEG library was compiled with the PEG_USER_ASSERT directive defined.

Related Functions:

Peg2DPolygonCreate, Peg2DPolygonCreateDefault, Peg2DPolygonSet

Notes:

The C/PEG library must be compiled with the PEG_FULL_GRAPHICS directive defined for this function to be available.

This function does not allocate memory for the object.

1.8 Peg2DPolygonLineColorSet

Synopsis:

```
void Peg2DPolygonLineColorSet(Peg2DPolygon *pdp, PEGCOLORVAL
  c);
```

Arguments:

pdp

Pointer to a Peg2DPolygon.

c

A color value.

Returns:

None

Description:

This function set the line color used when drawing the outline of the polygon.

Errors:

This function may generate an assert if the pdp pointer is invalid and the C/PEG library was built with the PEG_USER_ASSERT directive defined.

Notes:

The C/PEG library must be compiled with the PEG_FULL_GRAPHICS directive defined for this function to be available.

It is important to note that, internally, this function calls PegColorSet. This is important because PegColorSet calls PegRectRegionInvalidate, which, in multi-threaded systems, will lock the C/PEG screen resource.

1.9 Peg2DPolygonParentShift

Synopsis:

```
    void Peg2DPolygonParentShift(void *pThing, PEGSHORT x, PEGSHORT
      y);
```

Arguments:

pThing
Pointer to a Peg2DPolygon object.

x
The distance to shift along the x axis.

y
The distance to shift along the y axis.

Returns:

None

Description:

This is an override of the PegThing's function. This is done in order to ensure that the polygon point array is kept up to date with the object's physical position on the screen.

Errors:

This function may generate an assert if the pThing point is invalid and the C/PEG library was built with the PEG_USER_ASSERT directive defined.

Related Functions:

PegThingParentShift

Notes:

The C/PEG library must be compiled with the PEG_FULL_GRAPHICS directive defined for this function to be available.

1.10 Peg2DPolygonResize

Synopsis:

```
void Peg2DPolygonResize(void *pThing, PegRect *pRect);
```

Arguments:

pThing

    Pointer to a Peg2DPolygon object.

pRect

    Pointer to a PegRect structure that contains the new size and position for the object.

Returns:

None

Description:

This function is an override of the PegThing function. This function ensures that the internal point arrays used by the object to draw the polygon are properly updated as the object is physically moved on the screen.

Errors:

This function may generate an assert if either pointer is invalid and the C/PEG library was built with the PEG_USER_ASSERT directive defined.

Related Functions:

PegThingResize

Notes:

The C/PEG library must be compiled with the PEG_FULL_GRAPHICS directive defined for this function to be available.

1.11 Peg2DPolygonSet

Synopsis:

```
    void Peg2DPolygonSet(Peg2DPolygon *p2dpoly, PegRect *pRect,
      PegPoint *pPoints, PEGUINT uiNumPoints, PEGUSHORT usId,
      PEGUSHORT usStyle);
```

Arguments:

p2dpoly
: Pointer to a Peg2DPolygon object.

pRect
: Pointer to a PegRect structure that is used to establish the object's position and size.

pPoints
: Pointer to an array of PegPoints that is used to define the polygon.

uiNumPoints
: The number of PegPoints in the array pointed to by pPoints.

usId
: The ID to assign the object.

usStyle
: The style to assign the object.

Returns:

None

Description:

This function assigns size, position, polygon points, an ID and style to an existing Peg2DPolygon object.

This function does not allocate memory for the Peg2DPolygon itself, but may allocate memory for the point arrays as directed in usStyle. If usStyle contains the PF_COPY flag, the pPoints array is copied locally into the object.

Errors:

This function may generate an assert if any of the parameter pointers are invalid, or if it can not allocated a PegPoint array as directed by usStyle, and the C/PEG library was built with the PEG_USER_ASSERT directive defined.

Related Functions:

Peg2DPolygonCreate, Peg2DPolygonCreateDefault

Notes:

The C/PEG library must be compiled with the PEG_FULL_GRAPHICS directive defined for this function to be available.

This function is called by Peg2DPolygonCreate to properly initialize the new object with the given parameters.

1.12 Peg2DPolygonSetDefFuncs

Synopsis:

```
void Peg2DPolygonSetDefFuncs(Peg2DPolygon *p2dpoly);
```

Arguments:

p2dpoly
    Pointer to a Peg2DPolygon object.

Returns:

None

Description:

This function is an override of the PegThing function and sets up function pointers in the object to point at the correct, overridden Peg2DPolygon functions.

Errors:

This function may generate an assert if the p2dpoly pointer is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegThingSetDefFuncs, Peg2DPolygonCreateDefault

Notes:

The C/PEG library must be compiled with the PEG_FULL_GRAPHICS directive defined for this function to be available.

This function is called by Peg2DPolygonCreateDefault to unitize the function pointers properly.

1.13 Peg2DPolygonThetaSet

Synopsis:

```
void Peg2DPolygonThetaSet(Peg2DPolygon *pdp, PEGINT iTheta);
```

Arguments:

pdp
: Pointer to a Peg2DPolygon object.

iTheta
: Value to use as the new rotation angle of the object. Valid values are from 0 to 359, inclusive.

Returns:

None

Description:

This function sets the rotation angle that is used then drawing the polygon.

The rotation angle is a value from 0 to 359, inclusive, that describes the rotation of the object relative to an angle of 0 being straight horizontal, facing right, and continuing around in a counter-clockwise direction, with a value of 90 defining straight up, a value of 180 straight horizontal, facing left, and a value of 270 defining straight down.

Errors:

This function may generate an assert if the pdp pointer is invalid and the C/PEG library was built with the PEG_USER_ASSERT directive defined.

Related Functions:

PegSinCosLookup, PegRectRegionInvalidate

Notes:

The C/PEG library must be compiled with the PEG_FULL_GRAPHICS directive defined for this function to be available.

Internally, the object uses it's base point array, rotates the points and stores the rotated points in it's internal rotated point array.

It is important to note this function call will call PegRectRegionInvalidate if the object is visible.

1.14 PegBitmapButtonCreate

Synopsis:

```
    PegBitmapButton *PegBitmapButtonCreate(PegRect *pRect,
      PegBitmap *pBitmap, PEGUSHORT usId, PEGUSHORT usStyle);
```

Arguments:

pRect
: Pointer to a PegRect structure that determines the size and position of the object.

pBitmap
: Pointer to a PegBitmap structure that is used as the display bitmap on the button.

usId
: ID of the object.

usStyle
: Style of the object.

Returns:

Upon success, returns a pointer to a newly allocated PegBitmapButton object.

Description:

This function allocates memory for a PegBitmapButton structure and sets the new object's data members with the values passed in the parameter list.

Errors:

This function may generate an assert if the memory for the PegBitmapButton can not be allocated and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegBitmapButtonCreateDefault, PegDestroy

Notes:

As with any PegThing derived object, the memory returned to the caller by this function must be released by calling the object's destory function and should not be directly freed by the application code.
It is not an error for the pBitmap pointer to be NULL.

1.15 PegBitmapButtonCreateDefault

Synopsis:

```
    PegBitmapButton *PegBitmapButtonCreateDefault(void);
```

Arguments:

None

Returns:

Returns a pointer to a newly allocated PegBitmapButton object.

Description:

The object returned by this function is initialized, in that it's function pointers are set, it's type is set to PEG_TYPE_BM_BUTTON, it has the PSF_CLICKED signal set and it's default colors are set. All other object members are set to zero.

Errors:

This function may generate an assert if the memory for the PegBitmapButton object can not be allocated and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegBitmapButtonCreate, PegBitmapButtonSet, PegDestroy

Notes:

The data members of the object may be easily updated by calling PegBitmapButtonSet for this object.

1.16 PegBitmapButtonDraw

Synopsis:

```
void PegBitmapButtonDraw(void *pThing);
```

Arguments:

pThing

Pointer to a PegBitmapButton object that is used for drawing.

Returns:

None

Description:

This function is an override of the PegButton draw function. This draw function first draws the button, then the associated bitmap, if available.

Errors:

This function may generate an assert if the pThing pointer is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegButtonDraw, PegThingDraw

Notes:

This function calls PegButtonDraw to draw the basic button object.

1.17 PegBitmapButtonEraseFocus

Synopsis:

```
    void PegBitmapButtonEraseFocus(void *pThing);
```

Arguments:

pThing

A pointer to a PegBitmapButton object used for drawing.

Returns:

None

Description:

This function is an override of the PegButtonEraseFocus function. In this version, the entire button area is invalidated and redrawn.

Errors:

This function may generate an assert if the pThing pointer is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegButtonEraseFocus

Notes:

The function is simple in it's approach to erasing the focus indicator, it simple redraws itself.

1.18 PegBitmapButtonInit

Synopsis:

```
    void PegBitmapButtonInit(PegBitmapButton *pBitmapButton);
```

Arguments:

pBitmapButton
    Pointer to a PegBitmapButton object that is to be initialized.

Returns:

None

Description:

This function initialized a PegBitmapButton object. The function sets the function pointers for the object, set it's type to PEG_TYPE_BM_BUTTON, sets the PSF_CLICKED signal and sets the object's default colors.

Errors:

This function may generate an assert if the pBitmapButton pointer is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegBitmapButtonCreate, PegBitmapButtonCreateDefault

Notes:

This function is called by PegBitmapButtonCreateDefault to properly initialize the object.

1.19 PegBitmapButtonSet

Synopsis:

```
void PegBitmapButtonSet(PegBitmapButton *pBitmapButton, PegRect
  *pRect, PegBitmap *pBitmap, PEGUSHORT usId, PEGUSHORT
  usStyle);
```

Arguments:

pBitmapButton	Pointer to a PegBitmapButton object.
pRect	Pointer to a PegRect structure that determines the size and position of the object.
pBitmap	Pointer to a PegBitmap structure that is used for drawing.
usId	The ID of the object.
usStyle	The style of the object.

Returns:

None

Description:

This function sets the data members of the pBitmapButton object.

Related Functions:

PegBitmapButtonCreate

Notes:

This function does not allocate any memory, it merely sets the data members of the PegBitmapButton object.

This function is called by PegBitmapButtonCreate to properly set the data members of the object.

1.20 PegBitmapButtonSetDefFuncs

Synopsis:

```
    void PegBitmapButtonSetDefFuncs(PegBitmapButton
      *pBitmapButton);
```

Arguments:

pBitmapButton
: Pointer to a PegBitmapButton object.

Returns:

None

Description:

This function assigns the default PegBitmapButton function pointers in the pBitmapButton object.

Errors:

This function may generate an assert if pBitmapButton is invalid and the C/ PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegBitmapButtonCreate, PegBitmapButtonCreateDefault, PegBitmapButtonInit

Notes:

This function is called by PegBitmapButtonInit to set the default function pointers for the PegBitmapButton objects.

1.21 PegButtonCreate

Synopsis:

```
    PegButton *PegButtonCreate(PegRect *pRect, const PEGSTRING
      String, PEGUBYTE ubFont, PEGUSHORT usId, PEGUSHORT usStyle);
```

Arguments:

pRect
: Pointer to a PegRect structure that determines the size and position of the object.

String
: a NULL terminated string used as the text on the object.

ubFont
: Value to use in the C/PEG font lookup table to assign a font to this object.

usId
: ID for the object.

usStyle
: Style of the object.

Returns:

Upon success, returns a pointer to a newly allocated PegButton object.

Description:

This function allocates memory for a PegButton object and initializes all of the appropriate function pointers and data members.

The main purpose of this object is to provide a common base from which the other PegButton derivatives extend. This object is usually the starting point for custom objects to use in applications that need button behavior but wish to draw themselves differently.

Errors:

This function may generate an assert if the memory for the PegButton object can not be allocated from the system and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegButtonCreateDefault

Notes:

The PegButton object does not draw the text. Instead, it provides basic drawing, like the frame, for other PegButton objects that are derived from this object. If the application developer needs a button that will display it's text, use the PegTextButton object instead of just PegButton.

In order for the button object to send a PSF_CLICKED signal to it's parent object, usId can not be 0.

1.22 PegButtonCreateDefault

Synopsis:

```
PegButton *PegButtonCreateDefault(void);
```

Arguments:

None

Returns:

A pointer to a newly allocated PegButton object.

Description:

This function allocates memory for a PegButton object then call PegButtonInit to set the type, function pointers and default colors of the object.

Errors:

This function may generate an assert if the memory for the object can not be allocated from the system and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegButtonInit, PegButtonSet

Notes:

To set the date members of the object, the application may call PegButtonSet using the object returned from this call.

1.23 PegButtonDraw

Synopsis:

```
void PegButtonDraw(void *pThing);
```

Arguments:

pThing

Pointer to a PegButton object that is used as the drawing context.

Returns:

None

Description:

This function is an override of the PegTextThing draw function. The function draws the background and frame appropriate for the state of the button (normal or pushed down). This function is usually called by objects that derive from PegButton before they do their own drawing of text or bitmaps.

Errors:

This function may generate an assert if the pThing pointer is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegTextThingDraw, PegThingDraw

Notes:

This function is often replaced by custom application objects that wish to use the basic functionality of a PegButton, but need a custom representation on the screen.

1.24 PegButtonDrawFocus

Synopsis:

```
void PegButtonDrawFocus(void *pThing, PEGBOOL bDraw);
```

Arguments:

pThing

Pointer to a PegButton object.

bDraw

Instructs the object to redraw itself in it's entirety when drawing it's focus indicator.

Returns:

None

Description:

This function is on override of the PegTextThing draw focus function. This function draws a rectangle on the border of the object's client area using PCLR_FOCUS_INDICATOR for the line color.

Errors:

This function may generate an assert if the pThing pointer is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegButtonEraseFocus

Notes:

Most PegButton derivatives use this function to indicate focus.

1.25 PegButtonEnable

Synopsis:

```
    void PegButtonEnable(PegButton *pButton, PEGBOOL bEnable);
```

Arguments:

pButton
: Pointer to a PegButton object.

bEnable
: Boolean value to determine if the button should be enabled (TRUE) or disabled (FALSE).

Returns:

None

Description:

It is often necessary for the application code to enable and disable button type objects during the normal course of execution depending on the state of the application. This function allows the application developer to easily enable or disable a given PegButton, or any of it's derivatives.

Errors:

This function may generate an assert if pButton is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegButtonDraw, PegButtonNotify

Notes:

All PegButton and derivative objects draw themselves differently if they are enabled or disabled. For instance, the PegTextButton object draws it's text in a different color, usually grayed out, on target systems that support color, or with a line through the text on monochrome display. This informs the user that the object can not be selected.

To that end, when a PegButton object is disabled, it no longer responds to user input such as being clicked on by the mouse of selected by the keyboard.

1.26 PegButtonEraseFocus

Synopsis:

```
    void PegButtonEraseFocus(void *pThing);
```

Arguments:

pThing

Pointer to a PegButton object.

Returns:

None

Description:

This function erases the focus indicator drawn with PegButtonDrawFocus. If the object has the style AF_TRANSPARENT set, then it redraws itself completely. Otherwise, it draws a rectangle along the edge of it's client area in it's own background color.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegButtonDrawFocus

Notes:

Most PegButton derived objects use this function to erase the focus indicator.

1.27 PegButtonInit

Synopsis:

```
    void PegButtonInit(PegButton *pButton);
```

Arguments:

pButton
Pointer to a PegButton object.

Returns:

None

Description:

This function initializes the pButton object with the default PegButton function pointers, colors and styles. It also sets the object's type to PEG_TYPE_BUTTON and adds the PSF_CLICKED signal.

Errors:

This function may generate an assert if pButton is invalid and the C/PEG library was compiled with the PEG_USE_ASSERT directive defined.

Related Functions:

PegButtonCreateDefault

Notes:

This function is called by PegButtonCreateDefault to properly initialize the object before use.

1.28 PegButtonNotify

Synopsis:

```
PEGINT PegButtonNotify(void *pThing, const PegMessage *pMesg);
```

Arguments:

pThing
: Pointer to a PegButton object.

pMesg
: Pointer to a PegMessage structure that contains the message being processed.

Returns:

Various, depending on the message type and state of the object.

Description:

This function is an override of the PegTextThing notify function. There is extensive user input handling in this function. Most PegButton derived objects defer the majority of messages to this function for processing.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegTextThingNotify, PegThingNotify

Notes:

This function handles most mouse and keyboard input. It also handles starting a timer and continually sending it's parent PSF_CLICKED messages if the objects style flag contains the BF_REPEAT flag.

1.29 PegButtonSet

Synopsis:

```
void PegButtonSet(PegButton *pButton, PegRect *pRect, const
  PEGSTRING String, PEGUBYTE ubFont, PEGUSHORT usId, PEGUSHORT
  usStyle);
```

Arguments:

pButton
: Pointer to a PegButton object.

pRect
: Pointer to a PegRect structure that defines the size and location of the object.

String
: A NULL terminated string used for drawing text on the object.

ubFont
: A value which is used as an index into the C/PEG font lookup table to determine the font the object will use when drawing text.

usId
: ID of the object.

usStyle
: Style of the object.

Returns:

None

Description:

This function sets the data members of a PegButton object.

Errors:

This function may generate an assert if pButton is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegButtonCreate

Notes:

This function is called by PegButtonCreate to set the data members of the object. It is also called by most of the PegButton derived objects to set the common data members of the derived object.

1.30 PegButtonSetDefColors

Synopsis:

```
void PegButtonSetDefColors(PegButton *pButton);
```

Arguments:

pButton

Pointer to a PegButton object.

Returns:

None

Description:

This function sets the default colors for the pButton object.

Errors:

This function may generate an assert if pButton is invalid and the C/PEG library was compiled with the PEG_USE_ASSERT directive defined.

Related Functions:

PegButtonInit

Notes:

This function is called by PegButtonInit to set the default colors for the PegButton object. Most PegButton derived objects call this function at some point to also set their base colors.

1.31 PegButtonSetDefFuncs

Synopsis:

```
    void PegButtonSetDefFuncs(PegButton *pButton);
```

Arguments:

pButton
    Pointer to a PegButton object.

Returns:

None

Description:

This function sets the basic PegButton function pointers in the object pointed to by pButton.

Errors:

This function may generate an assert if pButton is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegButtonCreateDefault, PegButtonInit

Notes:

This function is called by PegButtonInit to properly set the default function pointers for the PegButton object. This function is also called by many of the PegButton derived objects to set the base function pointers for their object types.

1.32 PegCheckBoxCreate

Synopsis:

```
    PegCheckBox *PegCheckBoxCreate(PegRect *pRect, const PEGSTRING
      String, PEGUSHORT usId, PEGUSHORT usStyle);
```

Arguments:

pRect
: Pointer to a PegRect struct which defines the size and position of the object.

String
: a NULL terminated string which appears next to the check box.

usId
: ID of the object.

usStyle
: Style of the object.

Returns:

Upon success, returns a newly allocated PegCheckBox object.

Description:

This function allocates memory for a PegCheckBox object by calling PegCheckBoxCreateDefault, then sets the objects data members by calling PegCheckBoxInit with it's parameter list.

Errors:

This function may generate an assert if the necessary memory can not be allocated from the system and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegCheckBoxCreateDefault, PegCheckBoxInit, PegCheckBoxSet

Notes:

In order for the check box to send messages to it's parent object that the box was checked or unchecked, usId can not be 0.

1.33 PegCheckBoxCreateDefault

Synopsis:

```
    PegCheckBox *PegCheckBoxCreateDefault(void);
```

Arguments:

None

Returns:

Upon success, returns a pointer to the newly allocated PegCheckBox object.

Description:

This function allocated memory for a PegCheckBox object, it then calls PegCheckBoxInit to properly initialize the object before returning the pointer back to the caller.

Errors:

This function may generate an assert if the necessary memory can not be allocated by the system and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegCheckBoxCreate, PegCheckBoxInit

Notes:

1.34 PegCheckBoxDraw

Synopsis:

```
void PegCheckBoxDraw(void *pThing);
```

Arguments:

pThing

Pointer to a PegCheckBox object that is used for the drawing context.

Returns:

None

Description:

The PegCheckBox object overrides the PegButton draw function in order to properly display the correct check bitmap (checked, unchecked, enabled, disabled, etc.) as well as the user assigned text.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegButtonDraw

Notes:

The application designer may wish to replace the default bitmaps used to draw the check box. But, care must be taken to ensure that the bitmaps are appropriate for the target display constraints.

1.35 PegCheckBoxNotify

Synopsis:

```
    PEGINT PegCheckBoxNotify(void *pThing, const PegMessage
      *pMesg);
```

Arguments:

pThing
: Pointer to a PegCheckBox object.

pMesg
: Pointer to a PegMessage structure that contains the message data for processing.

Returns:

None

Description:

This function is an override of the PegButton notify function. The PegCheckBox has to do some special handling of mouse and keyboard input in order to display the box as checked or not.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegButtonNotify

Notes:

1.36 PegCheckBoxInit

Synopsis:

```
void PegCheckBoxInit(PegCheckBox *pCheckBox);
```

Arguments:

pCheckBox
Pointer to a PegCheckBox object.

Returns:

None

Description:

This function sets the type, default function pointers, default colors and the signals PSF_CHECK_ON and PSF_CHECK_OFF in the pCheckBox object.

Errors:

This function may generate an assert if pCheckBox is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegCheckBoxCreateDefault, PegButtonSetDefColors

Notes:

This function is usually called by PegCheckBoxCreateDefault to properly initialize the new PegCheckBox object. This function also uses the PegButtonSetDefColors function to set it's own default colors.

1.37 PegCheckBoxIsChecked

Synopsis:

```
    PEGBOOL PegCheckBoxIsChecked(PegCheckBox *pCheckBox);
```

Arguments:

pCheckBox
    Pointer to a PegCheckBox object.

Returns:

Returns TRUE if pCheckBox is checked, otherwise, it returns FALSE.

Description:

This function is a convenient way to determine the checked state of any PegCheckBox or derivative. The function queries it's internal style for the BF_SELECTED flag. If this is present, then the function returns TRUE, otherwise, it returns FALSE.

Errors:

This function may generate an assert if pCheckBox is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegCheckBoxSetChecked

Notes:

1.38 PegCheckBoxSet

Synopsis:

```
void PegCheckBoxSet(PegCheckBox *pCheckBox, PegRect *pRect,
  const PEGSTRING String, PEGUSHORT usId, PEGUSHORT usStyle);
```

Arguments:

pCheckBox
: Pointer to a PegCheckBox object.

pRect
: Pointer to a PegRect structure that defines the size and position of the object.

String
: a NULL terminated string that defines the text that appears beside the check box bitmap.

usId
: ID of the object.

usStyle
: Style of the object.

Returns:

None

Description:

This function sets the data members of pCheckBox.

Errors:

This function may generate an assert if pCheckBox is invalid and the C/ PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegCheckBoxCreate

Notes:

This function is called by PegCheckBoxCreate to properly set the data members of the object.

1.39 PegCheckBoxSetChecked

Synopsis:

```
void PegCheckBoxSetChecked(PegCheckBox *pCheckBox, PEGBOOL
  bChecked);
```

Arguments:

pCheckBox

Pointer to a PegCheckBox object.

bChecked

Boolean value to determine if the check box should be checked (TRUE) or unchecked (FALSE).

Returns:

None

Description:

There are occasions during the normal course of application execution where it is appropriate for the system to programmatically set the checked state of a check box object. This function allows the application developer the flexibility to do this with one function call.

Errors:

This function may generate an assert if pCheckBox is not valid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegCheckBoxIsChecked

Notes:

Use PegCheckBoxIsChecked to check the status of the check box.

1.40 PegCheckBoxSetDefFuncs

Synopsis:

```
    void PegCheckBoxSetDefFuncs(PegCheckBox *pCheckBox);
```

Arguments:

pCheckBox
	Pointer to a PegCheckBox object.

Returns:

None

Description:

This function sets the default PegCheckBox function pointers in pCheckBox.

Errors:

This function may generate an assert if pCheckBox is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegCheckBoxCreateDefault

Notes:

This function is called by PegCheckBoxCreateDefault to properly initialize the functions pointers of the new PegCheckBox object.

1.41 PegComboBoxAdd

Synopsis:

```
void PegComboBoxAdd(void *pThing, void *pAdd, PEGBOOL bRedraw);
```

Arguments:

pThing
Pointer to the parent object to which pAdd will be added.
pAdd
Pointer to the child object that will be added to pThing
bRedraw
Controls if the parent object should redraw (TRUE) or not (FALSE).

Returns:

None

Description:

PegComboBox overrides the PegThing add function in order to properly add the child object to the combo box's embedded list object.

Errors:

This function may generate an assert if either pointer parameter is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegComboBoxAddToEnd, PegComboBoxRemove, PegThingAdd

Notes:

1.42 PegComboBoxAddToEnd

Synopsis:

```
void PegComboBoxAddToEnd(void *pThing, void *pAdd, PEGBOOL
  bRedraw);
```

Arguments:

pThing
: Pointer to the parent object to which pAdd will be added.

pAdd
: Pointer to the child object that will be added to pThing

bRedraw
: Controls if the parent object should redraw (TRUE) or not (FALSE).

Returns:

None

Description:

PegComboBox overrides the PegThing add to end function in order to properly add the child object to the combo box's embedded list object.

Errors:

This function may generate an assert if either pointer parameter is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegComboBoxAdd, PegComboBoxRemove, PegThingAddToEnd

Notes:

1.43 PegComboBoxClear

Synopsis:

```
PEGINT PegComboBoxClear(void *pComboBox);
```

Arguments:

pComboBox
    Pointer to a PegComboBox object.

Returns:

The number of items removed and destroyed from the embedded list object.

Description:

This function removes and destroys all of the child objects contained in the embedded list object.

Errors:

This function may generate an assert if the list member of pComboBox is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegComboBoxRemove

Notes:

1.44 PegComboBoxCreate

Synopsis:

```
PegComboBox *PegComboBoxCreate(PegRect *pRect, PEGUSHORT usId,
  PEGUSHORT usStyle);
```

Arguments:

pRect

Pointer to a PegRect structure that is used to defined the size and position of the combo box. This rectangle defines the size of the combo box when it is opened. The combo box determines it's closed height based on the size of the children in the list.

usId

ID of the object.

usStyle

Style of the object

Returns:

Upon success, returns a newly allocated PegComboBox object.

Description:

This function allocates a new PegComboBox object by calling PegComboBoxCreateDefault then sets the member data variables by calling PegComboBoxSet.

Errors:

This function may generate an assert if the memory for the new object can not be allocated by the system and the C/PEG library was compiled with the PEG_USE_ASSERT directive defined.

Related Functions:

PegComboBoxCreateDefault, PegComboBoxSet, PegComboBoxInit, PegComboBoxNotify

Notes:

The PegComboBox calculates it's own closed height when it receives a PM_SHOW message in it's notify function.
If usId is 0, then the combo box will not send PSF_LIST_SELECT messages to it's parent.

1.45 PegComboBoxCreateDefault

Synopsis:

```
    PegComboBox *PegComboBoxCreateDefault(void);
```

Arguments:

None

Returns:

Upon success, a newly allocated PegComboBox is returned.

Description:

This function allocates memory for the PegComboBox object. It then calls PegComboBoxInit to properly initialize the function pointers, type and PSF_LIST_SELECT signal to itself.

It then creates and adds PegBitmapButton to itself that is used for opening the list. It then creates and adds a PegComboBoxList object to itself. This is the object that actually contains the user defined objects.

Errors:

This function may generate an assert if any of the object allocation fails and the C/PEG library was compiled with the PEG_USE_ASSERT directive defined.

Related Functions:

PegComboBoxCreate, PegBitmapButtonCreate, PegComboBoxListCreate

Notes:

1.46 PegComboBoxDestroy

Synopsis:

```
void PegComboBoxDestroy(void *pThing);
```

Arguments:

pThing

    Pointer to a PegComboBox object.

Returns:

None

Description:

This is an override of the PegThingDestroy function. The PegComboBox object properly cleans up the list object and drop button.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegThingDestroy, PegComboBoxCreate, PegComboBoxCreateDefault

Notes:

It is important the application code does not manually free the memory associated with this object. Doing so will result in a memory leak. A PegComboBox object must be freed using this function.

1.47 PegComboBoxDraw

Synopsis:

```
    void PegComboBoxDraw(void *pThing);
```

Arguments:

pThing

Pointer to a PegComboBox object.

Returns:

None

Description:

PegComboBox overrides the PegThing draw function in order to properly draw itself as current or not.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

RelatedFunctions:

PegThingDraw

Notes:

Even if the combo box is open, it does not cause the list object to be redrawn. This is because the list object, when visible, is added to the PegPresentation in order to avoid clipping issues when the list is showing against smaller “parent” objects.

1.48 PegComboBoxIndexGetFromPtr

Synopsis:

```
    PEGINT PegComboBoxIndexGetFromPtr(void *pComboBox, void
      *pChild);
```

Arguments:

pComboBox
    Pointer to a PegComboBox object.
pChild
    Pointer to a child object whose position index the caller is querying.

Returns:

Upon success, the position of the child in the list object owned by pComboBox. If pChild is NULL or is not found in the list, -1 is returned.

Description:

This function allows the application developer to query the list object of pComboBox to determine the position of pChild. Position index numbers begin at 0, therefore if pChild were to point at the first child object in the list, a value of 0 would be returned.

Errors:

This function may generate an assert if the list object of pComboBox is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegListIndexGetFromPtr, PegComboBoxSelectedSetFromIndex, PegComboBoxInsert

Notes:

This function is useful when the application needs to insert a child object into the combo box's list object in a specific place. The value returned from this function call may be used as the positional argument in PegComboBoxInsert.

1.49 PegComboBoxInit

Synopsis:

```
    void PegComboBoxInit(PegComboBox *pComboBox);
```

Arguments:

pComboBox
  Pointer to a PegComboBox object.

Returns:

None

Description:

This function initializes pComboBox by setting it's default function pointers, type and adding the PSF_LIST_SELECT signal.

Errors:

This function may generate an assert if pComboBox is invalid and the C/PEG library was compiled with the PEG_USE_ASSERT directive defined.

Related Functions:

PegComboBoxCreate, PegComboBoxCreateDefault

Notes:

This function is called by PegComboBoxCreateDefault to properly initialize pComboBox.

1.50 PegComboBoxInsert

Synopsis:

```
    void PegComboBoxInsert(void *pComboBox, void *pThing, PEGINT
      iWhere, PEGBOOL bSelect, PEGBOOL bRedraw);
```

Arguments:

pComboBox
    Pointer to a PegComboBox object.
pThing
    Pointer to a PegThing derived object that is to be inserted.
iWhere
    Positional index were pThing will be inserted.
bSelect
    Optionally select pThing after it is inserted.
bRedraw
    Optionally redraw pComboBox after pThing is inserted.

Returns:

None

Description:

This function allows the application code to insert a child object into the list object of pComboBox at a given location defined by iWhere. If iWhere is 0, then pThing is put in the list at the head. If iWhere is greater than the number of objects in the list, pThing is added to the end.

Errors:

This function may generate an assert if the list object of pComboBox is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegListInsert, PegComboBoxAdd, PegComboBoxAddToEnd

Notes:

When inserting more than one object into the combo box, it is good practice to not redraw the combo box with every child insertion, and to redraw the combo box when all insertions have completed.

1.51 PegComboBoxListClose

Synopsis:

```
    void PegComboBoxListClose(void *pCombo);
```

Arguments:

pCombo
Pointer to a PegComboBox object

Returns:

None

Description:

It is sometimes necessary for the application to programmatically close a combo box. This function allows the application developer to easily close a combo box.

Errors:

This function may generate an assert if pCombo is invalid and the C/PEG library was compiled with the PEG_USE_ASSERT directive defined.

Related Functions:

None

Notes:

1.52 PegComboBoxNotify

Synopsis:

```
    PEGINT PegComboBoxNotify(void *pThing, const PegMessage
      *pMesg);
```

Arguments:

pThing
  Pointer to a PegComboBox object.

pMesg
  Pointer to a PegMessage structure that contains the message data.

Returns:

Various returns values based on the type of message.

Description:

This function is an override of the PegThingNotify function.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegThingNotify

Notes:

The combo box determines it's closed size when it receives a PM_SHOW message based on the height of the children in the combo box's embedded list object. The combo box also draws itself differently when it is current.

1.53 PegComboBoxNumItemsGet

Synopsis:

```
PEGINT PegComboBoxNumItemsGet(void *pComboBox);
```

Arguments:

pComboBox
    Pointer to a PegComboBox object.

Returns:

Returns the number of child items in list object contained by pComboBox.

Description:

This function calls PegListNumItemsGet and passes pComboBox's embedded list object as the argument.

Errors:

This function may generate an assert if the list object in pComboBox is invalid and the library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegListNumItemsGet

Notes:

1.54 PegComboBoxPageDown

Synopsis:

```
    void *PegComboBoxPageDown(void *pComboBox);
```

Arguments:

pComboBox
    Pointer to a PegComboBox object.

Returns:

A pointer to the currently selected list object.

Description:

This function is usually called when a PegComboBox object receives a message, through PegComboBoxNotify, with a type of PM_KEY and a key type of PK_PGDN.

The object that is returned is the currently selected object in the combo box, which may be different than the object that was selected prior to this call. If the list is at the end of it's children, then the selected object will not change.

Errors:

This function may generate an assert if pComboBox's embedded list is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegListPageDown

Notes:

Internally, this function calls PegListPageDown and passes it's embedded list object to the call.

1.55 PegComboBoxPageUp

Synopsis:

```
    void *PegComboBoxPageUp(void *pComboBox);
```

Arguments:

pComboBox

Pointer to a PegComboBox object.

Returns:

Pointer to the currently selected object in the list.

Description:

This function is usually called when a PegComboBox object receives a message, through PegComboBoxNotify, with a type of PM_KEY and a key type of PK_PGUP.

The object that is returned is the currently selected object in the combo box, which may be different than the object that was selected prior to this call. If the list is at the head of it's children, then the selected object will not change.

Errors:

This function may generate an assert if pComboBox's embedded list is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegListPageUp

Notes:

Internally, this function calls PegListPageUp and passes it's embedded list object to the call.

1.56 PegComboBoxPtrGetFromIndex

Synopsis:

```
    void *PegComboBoxPtrGetFromIndex(void *pComboBox, PEGINT
      iIndex);
```

Arguments:

pComboBox

    Pointer to a PegComboBox object.

iIndex

    Zero based index of the child object.

Returns:

Upon success, a pointer to the list's child object that corresponds to the iIndex position in the list. If iIndex is out of bounds of the list's current number of child objects, then NULL is returned.

Description:

This function is used to attain a pointer to the child object in the index position described by iIndex from pComboBox's embedded list object.

Errors:

This function may generate an assert if the list object is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegListPtrGetFromIndex, PegComboBoxIndexGetFromPtr

Notes:

Internally, this function calls PegListPtrGetFromIndex and passes the PegComboBox's embedded list object to the function.

1.57 PegComboBoxRemove

Synopsis:

```
    void *PegComboBoxRemove(void *pThing, void *pRemove, PEGBOOL
      bRedraw);
```

Arguments:

pThing
: Pointer to a PegComboBox object.

pRemove
: Pointer to a PegThing derived object that is being removed.

bRedraw
: Optional directive to cause pThing to redraw after the removal is complete.

Returns:

Upon success, returns a pointer to the object that was removed. If pRemove is not a child of pThing, NULL is returned.

Description:

This function is an override of the PegThingRemove function. The object actually has to be removed for the PegComboBox's embedded list object, not the PegComboBox itself.

Errors:

This function may generate an assert if the embedded list object is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

RelatedFunctions:

PegThingRemove, PegListRemove

Notes:

The list objects are not immediate children of a PegComboBox, they are instead children of the PegComboBox's list object. Therefore, this function passes the call to the PegListRemove function.

1.58 PegComboBoxResize

Synopsis:

```
    void PegComboBoxResize(void *pThing, PegRect *pRect);
```

Arguments:

pThing
    Pointer to a PegComboBox object.
pRect
    Pointer to a PegRect structures that described the new size and position of pThing.

Returns:

None

Description:

This function is an override of PegThingResize. The size specified by pRect should be the desired size of the object when the list portion of the PegComboBox is opened. The PegComboBox calculates it's closed height based on the size of the children in the embedded list object.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegThingResize

Notes:

1.59 PegComboBoxSelectNext

Synopsis:

```
    void *PegComboBoxSelectNext(void *pComboBox);
```

Arguments:

pComboBox

Pointer to a PegComboBox object.

Returns:

A pointer to pComboBox's list's child object that is currently selected.

Description:

This function is usually called from PegComboBoxNotify when the object receives a message of type PM_KEY and a key value of PK_LNDN.

This function may modify the currently selected object in the list.

If the previously selected child is at the end of the list, and the PegComboBox object has the style of LS_WRAP_SELECT, then the first child will be selected. If LS_WRAP_SELECT is not set, the list will not wrap and the selected object will not change.

Errors:

This function may generate an assert if the list object is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegListSelectNext

Notes:

Internally, this function call PegListSelectNext and passes the combo box's embedded list object.

1.60 PegComboBoxSelectPrevious

Synopsis:

```
    void *PegComboBoxSelectPrevious(void *pComboBox);
```

Arguments:

pComboBox
    Pointer to a PegComboBox object.

Returns:

A pointer to pComboBox's list's child object that is currently selected.

Description:

This function is usually called from PegComboBoxNotify when the object receives a message of type PM_KEY and a key value of PK_LNUP.

This function may modify the currently selected object in the list.

If the previously selected child is at the head of the list, and the PegComboBox object has the style of LS_WRAP_SELECT, then the last child will be selected. If LS_WRAP_SELECT is not set, the list will not wrap and the selected object will not change.

Errors:

This function may generate an assert if the list object is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegListSelectPrevious

Notes:

Internally, this function call PegListSelectPrevious and passes the combo box's embedded list object.

1.61 PegComboBoxSelectedIndexGet

Synopsis:

```
    PEGINT PegComboBoxSelectedIndexGet(void *pComboBox);
```

Arguments:

pComboBox
: Pointer to a PegComboBox object

Returns:

Returns the index of the list's currently selected object. If the list does not have a selected child object, then -1 is returned.

Description:

This function returns the zero based index of the currently selected child in the PegComboBox's embedded list object.

Errors:

This function may generate an assert if the list object is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegListSelectedIndexGet

Notes:

Internally, this function calls PegListSelectedIndexGet using the PegComboBox's embedded list object as the subject of the call.

1.62 PegComboBoxSelectedPtrGet

Synopsis:

```
void *PegComboBoxSelectedPtrGet(void *pComboBox);
```

Arguments:

pComboBox
    Pointer to a PegComboBox object.

Returns:

Upon success, a pointer to the PegCombBox's embedded list's selected object is returned. If the list has no selected child, then NULL is returned.

Description:

This function returns a pointer to the combo box's list's selected child object.

Errors:

This function may generate an assert if the embedded list is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegListSelectedPtrGet

Notes:

Internally, this function calls PegListSelectedPtrGet and passes the combo box's embedded list object to the call.

1.63 PegComboBoxSelectedSetFromIndex

Synopsis:

```
    void *PegComboBoxSelectedSetFromIndex(void *pComboBox, PEGINT
      iIndex);
```

Arguments:

pComboBox
    Pointer to a PegComboBox object.

iIndex
    Index of the child to set as selected.

Returns:

None

Description:

This function attempts to set the selected item in pComboBox's embedded list to the object whose index matches iIndex. If the child object can not be found, the selected item in the list is set to NULL.

Errors:

This function may generate an assert if the combo box's embedded list object is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegListSelectedSetFromIndex, PegComboBoxSelectedSetFromPtr

Notes:

It is legal to pass -1 in iIndex to force the list to select nothing.

1.64 PegComboBoxSelectedSetFromPtr

Synopsis:

```
void PegComboBoxSelectedSetFromPtr(void *pComboBox, void
  *pThing);
```

Arguments:

pComboBox
: Pointer to a PegComboBox object.

pThing
: Pointer to a PegThing derivative.

Returns:

None

Description:

This function attempts to set the selected item in pComboBox's embedded list to the object passed in pThing. If the child object can not be found, the selected item in the list is set to NULL.

Errors:

This function may generate an assert if the combo box's embedded list object is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegListSelectedSetFromPtr, PegComboBoxSelectedSetFromIndex

Notes:

It is legal to pass NULL in pThing to force the list to select nothing.

1.65 PegComboBoxSet

Synopsis:

```
    void PegComboBoxSet(PegComboBox *pComboBox, PegRect *pRect,
      PEGUSHORT usId, PEGUSHORT usStyle);
```

Arguments:

pComboBox
: Pointer to a PegComboBox object.

pRect
: Pointer to a PegRect structure used to determine the size and position of pComboBox.

usId
: ID of the object.

usStyle
: Style of the object.

Returns:

None

Description:

This function sets the data members of pComboBox to the incoming arguments.

Errors:

This function may generate an assert if pComboBox is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegComboBoxCreate

Notes:

This function is called by PegComboBoxCreate to properly set up the object after it has been initialized.

1.66 PegComboBoxSetDefFuncs

Synopsis:

```
void PegComboBoxSetDefFuncs(PegComboBox *pComboBox);
```

Arguments:

pComboBox
Pointer to a PegComboBox object.

Returns:

None

Description:

This function sets the function pointers in pComboBox to the default functions used by PegComboBox.

Errors:

This function may generate an assert if pComboBox is invalid and the C/PEG library was built with the PEG_USER_ASSERT directive defined.

Related Functions:

PegComboBoxInit

Notes:

This function is called by PegComboBoxInit to properly set up the function pointers in PegComboBox objects.

1.67 PegDecoratedButtonCreate

Synopsis:

```
PegDecoratedButton *PegDecoratedButtonCreate(PegRect *pRect,
    const PEGSTRING String, PegBitmap *pBitmap, PEGBOOL
    bFlyOver, PEGUSHORT usId, PEGUSHORT usStyle);
```

Arguments:

pRect
: Pointer to a PegRect structure that determines the size and position of the object.

String
: a NULL terminated string that is used to draw text on the object.

pBitmap
: Pointer to a PegBitmap structure that is used to draw a bitmap on the object.

bFlyOver
: Optional draw style.

usId
: ID assigned to the object.

usStyle
: Style assigned to the object.

Returns:

Upon success, returns a pointer to a newly allocated PegDecoratedButton object.

Description:

This function allocates the memory necessary for a new instance of a PegDecoratedButton object. It then initializes the object and assigns the object's data member from the parameters list.

Errors:

This function may generate an assert if the memory for the object can not be allocated from the system and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Function:

PegDecoratedButtonCreateDefault, PegDecoratedButtonInit, PegDecoratedButtonSet

Notes:

This function calls PegDecoratedButtonCreateDefault to allocate the storage for the object and PegDecoratedButtonSet to set up the object with the passed in parameter list.

It is not an error for either pBitmap or pText or both to be NULL. But, if the application is not using both the text and bitmap, then it should be considered to use either PegBitmapButton or PegTextButton instead of this object to save a bit of drawing overhead.

The bFlyOver style, when set as TRUE, causes the object to draw itself flat when it is not selected, then to draw itself raised when the mouse pointer is over it's screen region. This is well known behavior with many popular desktop GUI libraries. This does create more over head at run time because the object must redraw itself when the mouse pointer enters it's screen region and must redraw itself when the mouse pointer exits it's screen region. For most systems, this redrawing will not have a great impact, but, if it does, it is easily turned off by passing FALSE in bFlyOver.

The usId must be greater than 0 in order for the object to send PSF_CLICKED signals to it's parent.

Along with the normal styles supported by the PegButton object, from which PegDecoratedButton is derived, the PegDecoratedButton supports two more styles that are used to determine the position of the bitmap and text that it will draw. The following table lists these styles and describes their effect in the visual presentation of the text and bitmap. Note that these flags are only effective is the if the object has a valid string and valid bitmap to draw.

Flags	Text and Bitmap Position
BF_ORIENT_TR && !BF_ORIENT_BR	The bitmap and text are horizontally centered. The bitmap is drawn above the text.
!BF_ORIENT_TR && BF_ORIENT_BR	The bitmap and text are horizontally centered. The bitmap is drawn below the text.
BF_ORIENT_TR && BR_ORIENT_BR	The bitmap and text are vertically centered. The bitmap is drawn on the right side of the text.
!BR_ORIENT_TR && !BF_ORIENT_BR	The bitmap and text are vertically centered. The bitmap is drawn on the left side of the text.

1.68 PegDecoratedButtonCreateDefault

Synopsis:

```
    PegDecoratedButton *PegDecoratedButtonCreateDefault(void);
```

Arguments:

None

Returns:

Upon success, a pointer to a newly allocated PegDecoratedButton object is returned.

Description:

This function allocates memory for a PegDecoratedButton object. It also initializes the object's function pointers, type and assigns itself the PSF_CLICKED signal.

Errors:

This function may generate an assert if the memory for the object cannot be allocated from the system and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegDecoratedButtonInit, PegDecoratedButtonCreate

Notes:

After allocating the memory for this object, this function will call PegDecoratedButtonInit to properly initialize the object.

1.69 PegDecoratedButtonDraw

Synopsis:

```
    void PegDecoratedButtonDraw(void *pThing);
```

Arguments:

pThing

Pointer to a PegDecoratedButton object.

Returns:

None

Description:

This function is an override of the PegButton draw function. The PegDecoratedButton may draw itself flat, with no frame, if optionally instructed to do so. It must also draw the text and bitmap, it any, associated with the object.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegButtonDraw

Notes:

Please see the Notes section for PegDecoratedButtonCreate that describes the four different ways that the object's text and bitmap may be oriented when drawn.

1.70 PegDecoratedButtonInit

Synopsis:

```
    void PegDecoratedButtonInit(PegDecoratedButton *pDecButton);
```

Arguments:

pDecButton
Pointer to a PegDecoratedButton object.

Returns:

None

Description:

This function assigns the default values associated with the PegDecoratedButton to pDecButton.

Errors:

This function may generate an assert if pDecButton is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegDecoratedButtonCreateDefault

Notes:

This function called PegDecoratedButtonSetDefFuncs to set up the function pointers for the object.

1.71 PegDecoratedButtonNotify

Synopsis:

```
    PEGINT PegDecoratedButtonNotify(void *pThing, const PegMessage
      *pMesg);
```

Arguments:

pThing
: Pointer to a PegDecoratedButton object.

pMesg
: Pointer to a PegMessage structure that contains the data for the message.

Returns:

Various depending on the message being processed.

Description:

This function is an override of the PegButtonNotify function.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegButtonNotify

Notes:

This function handles PM_POINTER_ENTER and PM_POINTER_EXIT messages to draw itself differently if the object is optionally set to do so.

1.72 PegDecoratedButtonSet

Synopsis:

```
void PegDecoratedButtonSet(PegDecoratedButton *pDecButton,
  PegRect *pRect, const PEGSTRING String, PegBitmap *pBitmap,
  PEGBOOL bFlyOver, PEGUSHORT usId, PEGUSHORT usStyle);
```

Arguments:

pDecButton
: Pointer to a PegDecoratedButton object.

pRect
: Pointer to a PegRect structure that determines the size and position of the object.

String
: a NULL terminated string that is used to draw text on the object.

pBitmap
: Pointer to a PegBitmap structure that is used to draw a bitmap on the object.

bFlyOver
: Optional draw style.

usId
: ID assigned to the object.

usStyle
: Style assigned to the object.

Returns:

None

Description:

This function sets pDecButton's data members to the corresponding values from the parameter list.

Errors:

This function may generate an assert if pDecButton is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegDecoratedButtonCreate

Notes:

1.73 PegDecoratedButtonSetDefFuncs

Synopsis:

```
void PegDecoratedButtonSetDefFuncs(PegDecoratedButton
  *pDecButton);
```

Arguments:

pDecButton
    Pointer to a PegDecoratedButton object.

Returns:

None

Description:

This function sets the default PegDecoratedButton function pointers in pDecButton.

Errors:

This function may generate an assert if pDecButton is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegDecoratedButtonInit

Notes:

This function is called by PegDecoratedButtonInit.

1.74 PegEditFieldCreate

Synopsis:

```
    PegEditField *PegEditFieldCreate(PegRect *pRect, const
      PEGSTRING String, PEGUSHORT usId, PEGUSHORT usStyle);
```

Arguments:

pRect	Pointer to a PegRect structure that defines the size and position of the object.
String	a NULL terminated string that will be assigned to the object.
usId	ID assigned to the object.
usStyle	Style assigned to the object.

Returns:

Upon success, returns a newly allocated PegEditField object.

Description:

This function allocates storage for a PegEditField object, initializes the object and sets the object's data members from the passed in parameter list.

Errors:

This function may generate an assert if the necessary memory for the object can not be allocated from the system and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegEditFieldCreateDefault, PegEditFieldInit, PegEditFieldSet, PegEditFieldSetDefFuncs

Notes:

1.75 PegEditFieldCreateDefault

Synopsis:

```
    PegEditField *PegEditFieldCreateDefault(void);
```

Arguments:

None

Returns:

Upon success, this function returns a pointer to a newly allocated PegEditField object.

Description:

This function allocates storage for a PegEditField object, initializes the object's default members and returns a pointer to the new object.

Errors:

This function may generate an assert if the necessary memory can not be allocated from the system and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegEditFieldCreate, PegEditFieldInit

Notes:

This function calls PegEditFieldInit to set the state of the object.

1.76 PegEditFieldDestroy

Synopsis:

```
    void PegEditFieldDestroy(void *pThing);
```

Arguments:

pThing
    Pointer to a PegEditField object.

Returns:

None

Description:

This function is an override of PegTextThingDestroy. The PegEditField object may, optionally, have a backup copy of the original string before any user edits were invoked. The storage for this backup string copy is released during the destroy process.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegTextThingDestroy

Notes:

1.77 PegEditFieldDraw

Synopsis:

```
void PegEditFieldDraw(void *pThing);
```

Arguments:

pThing

Pointer to a PegEditField object.

Returns:

None

Description:

This function is an override of PegTextThingDraw.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegTextThingDraw

Notes:

The PegEditField object draws selected text using a different color background and foreground color than non-selected text. It must also horizontally scroll the text when the text string becomes too wide to be completely visible within the bounds of it's size.

1.78 PegEditFieldEditEnable

Synopsis:

```
void PegEditFieldEditEnable(PegEditField *pEditField, PEGBOOL
  bEnable);
```

Arguments:

pEditField
: Pointer to a PegEditField object.

bEnable
: Pass TRUE to enable editing in the object, or FALSE to disable editing.

Returns:

None

Description:

This function allows the application to enable or disable editing in the object. By default, all PegEditField objects are created with editing enabled.

Errors:

This function may generate an assert if pEditField is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

None

Notes:

1.79 PegEditFieldInit

Synopsis:

```
    void PegEditFieldInit(PegEditField *pEditField);
```

Arguments:

pEditField
: Pointer to a PegEditField object.

Returns:

None

Description:

This function initializes pEditField to the default values associated with the PegEditField object.

Errors:

This function may generate an assert if pEditField is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegEditFieldCreateDefault

Notes:

This function is called by PegEditFieldCreateDefault to properly initialize the object.

1.80 PegEditFieldMarkedTextCopy

Synopsis:

```
PEGBOOL PegEditFieldMarkedTextCopy(PegEditField *pEditField);
```

Arguments:

pEditField
    Pointer to a PegEditField object.

Returns:

Upon success, TRUE, otherwise, FALSE.

Description:

This function is used to copy the marked text from pEditField to the clipboard maintained by PegPresentation.

Errors:

This function may generate an assert if pEditField is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegEditFieldMarkedTextCut, PegEditFieldMarkedTextDelete, PegEditFieldTextPaste, PegScratchPadSet, PegScratchPadGet

Notes:

If pEditField has no text or no text that is marked, then this function returns FALSE. Only the marked text in pEditField is copied.

1.81 PegEditFieldMarkedTextCut

Synopsis:

```
PEGBOOL PegEditFieldMarkedTextCut(PegEditField *pEditField);
```

Arguments:

pEditField
    Pointer to a PegEditField object.

Returns:

On success, TRUE, otherwise, FALSE.

Description:

This function is used to cut the marked text from pEditField to the clipboard maintained by PegPresentation.

Errors:

This function may generate an assert if pEditField is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegEditFieldMarkedTextCopy, PegEditFieldMarkedTextDelete, PegEditFieldTextPaste, PegScratchPadSet, PegScratchPadGet

Notes:

If pEditField has no text or no text that is marked, then this function returns FALSE. Only the marked text in pEditField is cut.

1.82 PegEditFieldMarkedTextDelete

Synopsis:

```
PEGBOOL PegEditFieldMarkedTextDelete(PegEditField *pEditField);
```

Arguments:

pEditField
    Pointer to a PegEditField object.

Returns:

On success, TRUE, otherwise, FALSE.

Description:

This function is used to delete the marked text from pEditField.

Errors:

This function may generate an assert if pEditField is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegEditFieldMarkedTextCopy, PegEditFieldMarkedTextCut, PegEditFieldTextPaste

Notes:

If pEditField has no text or no text that is marked, then this function returns FALSE. Only the marked text in pEditField is deleted.

1.83 PegEditFieldTextPaste

Synopsis:

```
    PEGBOOL PegEditFieldTextPaste(PegEditField *pEditField);
```

Arguments:

pEditField
    Pointer to a PegEditField object.

Returns:

Upon success, TRUE, otherwise, FALSE.

Description:

This function is used to paste the contents of the clipboard into pEditField at the current cursor position. If any of the text in pEditField is marked, the marked text is first deleted, and the cursor is placed at the after the last character before the marked text.

Errors:

This function may generate an assert if pEditField is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegEditFieldMarkedTextCopy, PegEditFieldMarkedTextDelete, PegEditFieldMarkedTextCut, PegScratchPadSet, PegScratchPadGet

Notes:

If there is no text on the clipboard, this function returns FALSE.

1.84 PegEditFieldSet

Synopsis:

```
    void PegEditFieldSet(PegEditField *pEditField, PegRect *pRect,
      const PEGSTRING String, PEGUSHORT usId, PEGUSHORT usStyle);
```

Arguments:

pEditField
    Pointer to a PegEditField object.
pRect
    Pointer to a PegRect structure that determines the size and position of the object.
String
    a NULL terminated string to assign to the object.
usId
    ID assigned to the object.
usStyle
    Style assigned to the object.

Returns:

None

Description:

This functions sets the data members of pEditField with the parameters passed in the parameter list.

Errors:

This function may generate an assert if pEditField is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegEditFieldCreate

Notes:

This function is called by PegEditFieldCreate to assign values to the object's data members.

1.85 PegEditFieldSetDefColors

Synopsis:

```
void PegEditFieldSetDefColors(PegEditField *pEditField);
```

Arguments:

pEditField
    Pointer to a PegEditField object.

Returns:

None

Description:

This function sets the internal colors for the background (normal and selected) and the text (normal and selected) to their default values.

Errors:

This function may generate an assert if pEditField is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegEditFieldCreate, PegEditFieldCreateDefault, PegEditFieldInit

Notes:

The application is free to modify the default colors used for this object.

1.86 PegEditFieldSetDefFuncs

Synopsis:

```
    void PegEditFieldSetDefFuncs(PegEditField *pEditField);
```

Arguments:

pEditField
    Pointer to a PegEditField object.

Returns:

None

Description:

This function sets the function pointers in pEditField to the default PegEditField functions.

Errors:

This function may generate an assert if pEditField is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegEditFieldInit

Notes:

1.87 PegEditFieldTextSet

Synopsis:

```
void PegEditFieldTextSet(void *pThing, const PEGCHAR *pText);
```

Arguments:

pThing

  Pointer to a PegEditField object.

pText

  Pointer to a NULL terminated string.

Returns:

None

Description:

This function is an override of PegTextThingTextSet.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegTextThingTextSet

Notes:

If the length of pText is greater than the maximum allowable string length of pThing, and pThing has style TT_COPY, only the first maximum allowable string length – 1 characters are copied fro pText into pThing's internal buffer.

This function calls PegRectRegionInvalidate, so it is advised to call pThing's draw function when the caller has completed updating the object.

1.88 PegGroupCreate

Synopsis:

```
    PegGroup *PegGroupCreate(PegRect *pRect, const PEGSTRING
      String, PEGUSHORT usId, PEGUSHORT usStyle);
```

Arguments:

pRect
: Pointer to a PegRect structure that determines the size and position of the object.

String
: a NULL terminated string.

usId
: ID to assign the object.

usStyle
: Style to assign the object.

Returns:

Upon success, returns a pointer to a newly allocated PegGroup object.

Description:

This function allocates storage for a PegGroup object, initializes the object and assigns it's member variables the values passed in the parameter list.

Errors:

This function may generate an assert if the necessary memory for the object can not be allocated from the system and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegGroupCreateDefault, PegGroupInit, PegGroupSet

Notes:

This function calls PegGroupCreateDefault to allocate the storage for the object and PegGroupSet to assign the member variables.

1.89 PegGroupCreateDefault

Synopsis:

```
    PegGroup *PegGroupCreateDefault(void);
```

Arguments:

None

Returns:

Returns a pointer to a newly allocated PegGroup object.

Description:

This function allocates storage for a new PegGroup object and initializes the object with default function pointers and type.

Errors:

This function may generate an assert if the necessary memory could not be allocated from the system and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegGroupCreate, PegGroupInit

Notes:

This function calls PegGroupInit to initialize the object with the PegGroup defaults.

1.90 PegGroupDraw

Synopsis:

```
    void PegGroupDraw(void *pThing);
```

Arguments:

pThing

Pointer to a PegGroup object.

Returns:

None

Description:

This function overrides the PegTextThingDraw function. It draws the text, if any, and a recessed border around it's client region.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegTextThingDraw

Notes:

The PegGroup object may or may not have text that it displays in the upper left corner of it's client area.

1.91 PegGroupEnable

Synopsis:

```
void PegGroupEnable(PegGroup *pGroup, PEGBOOL bEnable);
```

Arguments:

pGroup
    Pointer to a PegGroup object.
bEnable
    Passing TRUE will enable the object, passing FALSE will disable it.

Returns:

None

Description:

This function enables or disables a PegGroup object. When the PegGroup object is disabled, any button type of objects that are children of the PegGroup object are also disabled. Likewise, if the PegGroup object is enabled, then all of it's children are enabled.

Error:

It is not an error to enable a PegGroup object that is already enabled. It is not an error to disable a PegGroup object that is already disabled.

This function may generate an assert if pGroup is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegButtonEnable

Notes:

When the object is disabled, the text, if any, is redrawn grayed out, or with a strike out style on monochrome screens, to visually alert the system user that the group and it's children are disabled. Any button type children also redraw their text in a disabled manner.

1.92 PegGroupInit

Synopsis:

```
    void PegGroupInit(PegGroup *pGroup);
```

Arguments:

pGroup
	Pointer to a PegGroup object.

Returns:

None

Description:

This function initializes pGroup to the defaults of the PegGroup type.

Errors:

This function may generate an assert if pGroup is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegGroupCreateDefault, PegGroupSetDefFuncs

Notes:

This function is called by PegGroupCreateDefault to properly initialize the object.

1.93 PegGroupNotify

Synopsis:

```
PEGINT PegGroupNotify(void *pThing, const PegMessage *pMesg);
```

Arguments:

pThing

Pointer to a PegGroup object.

pMesg

Pointer to a PegMessage structure that contains the message data.

Returns:

Various, depending on the message type.

Description:

This function is an override of the PegTextThingNotify function.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegTextThingNotify

Notes:

The PegGroup object has to handle key handling to properly give focus to it's child objects when it receives focus. It also must handle mouse messages.

1.94 PegGroupRemove

Synopsis:

```
void *PegGroupRemove(void *pThing, void *pRemove, PEGBOOL
  bRedraw);
```

Arguments:

pThing
    Pointer to a PegGroup object.
pRemove
    Pointer to a PegThing or derivative that is to be removed.
bRedraw
    Optional redraw of pThing after pRemove has been removed.

Returns:

A pointer to the object that was removed. If pRemove is not a child of pThing, returns NULL.

Description:

This function is an override of PegTextThingRemove. If bRedraw is TRUE, then the PegGroup object calls it's parent's draw function so the client area of the group will be redrawn properly.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegTextThingRemove

Notes:

1.95 PegGroupSet

Synopsis:

```
    void PegGroupSet(PegGroup *pGroup, PegRect *pRect, const
      PEGSTRING String, PEGUSHORT usId, PEGUSHORT usStyle);
```

Arguments:

pGroup
: Pointer to a PegGroup object.

pRect
: Pointer to a PegRect structure that defines the size and position of the object.

String
: a NULL terminated string.

usId
: ID to assign the object.

usStyle
: Style to assign the object.

Returns:

None

Description:

This function assigns the data members of pGroup with the data passed in on the parameter list.

Errors:

This function may generate an assert if pGroup is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegTextThingSet

Notes:

1.96 PegGroupSetDefFuncs

Synopsis:

```
void PegGroupSetDefFuncs(PegGroup *pGroup);
```

Arguments:

pGroup
Pointer to a PegGroup object.

Returns:

None

Description:

This function sets the function pointers in pGroup to the default functions of PegGroup.

Errors:

This function may generate an assert if pGroup is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegTextThingSetDefFuncs

Notes:

PegGroupCreateInit calls this function to properly set the default function pointers in pGroup.

1.97 PegHorzListCreate

Synopsis:

```
    PegHorzList *PegHorzListCreate(PegRect *pRect, PEGUSHORT usId,
      PEGUSHORT usStyle);
```

Arguments:

pRect
: Pointer to a PegRect structure that defines the size and position of the object.

usId
: ID to assign to the object.

usStyle
: Style to assign to the object.

Returns:

Upon success, returns a pointer to a newly allocated PegVertList object.

Description:

This function allocates storage for a PegVertList object, initializes the object and set the object's data members.

Errors:

This function may generate an assert if the necessary memory for the object can not be allocated from the system and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegHorzListCreateDefault, PegHorzListSet

Notes:

This function calls PegHorzListCreateDefault to allocate storage for the object, then calls PegHorzListSet to set the object's data members.

1.98 PegHorzListCreateDefault

Synopsis:

```
    PegHorzList *PegHorzListCreateDefault(void);
```

Arguments:

None

Returns:

Upon success, returns a pointer to a newly allocated PegHorzList object.

Description:

This function allocates storage for a PegHorzList object and initializes the object to the default associated with a PegHorzList object type.

Errors:

This function may generate an assert if the necessary memory for the object can not be allocated from the system and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegHorzListCreate, PegHorzListInit

Notes:

This function is called by PegHorzList create to allocate the storage for the object. This function then calls PegHorzListInit to properly initialize the object.

1.99 PegHorzListInit

Synopsis:

```
    void PegHorzListInit(PegHorzList *pHList);
```

Arguments:

pHList

Pointer to a PegHorzList object.

Returns:

None

Description:

This function initializes pHList with the defaults associated with a PegHorzList object type.

Errors:

This function may generate an assert if pHList is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegHorzListCreateDefault, PegHorzListSetDefFuncs

Notes:

This function is called by PegHorzListCreateDefault to set the default in the object. This function calls PegHorzListSetDefFuncs to set the function pointers in the object.

1.100 PegHorzListSet

Synopsis:

```
    void PegHorzListSet(PegHorzList *pHList, PegRect *pRect,
      PEGUSHORT usId, PEGUSHORT usStyle);
```

Arguments:

pHList
: Pointer to a PegHorzList object.

pRect
: Pointer to a PegRect structure that defines the size and position of the object.

usId
: ID to assign to the object.

usStyle
: Style to assign to the object.

Returns:

None

Description:

This function assigns the data members of pHList with the values in the passed in parameter list.

Errors:

This function may generate an assert if pHList is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegHorzListCreate

Notes:

This function is called by PegHorzListCreate to set the data members in the object.

1.101 PegHorzListSetDefFuncs

Synopsis:

```
    void PegHorzListSetDefFuncs(PegHorzList *pHList);
```

Arguments:

pHList

    Pointer to a PegHorzList object.

Returns:

None

Description:

This function sets the function pointers in pHList to the default functions associated with a PegHorzList object type.

Errors:

This function may generate an assert if pHList is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegHorzListInit

Notes:

This function is called by the PegHorzListInit function to properly set the function pointers in the object.

1.102 PegHorzScrollCreate

Synopsis:

```
    PegHorzScroll *PegHorzScrollCreate(PegRect *pRect, PEGUSHORT
      usId, PEGUSHORT usStyle);
```

Arguments:

pRect
: Pointer to a PegRect structure that defines the size and position of the object.

usId
: ID to assign to the object.

usStyle
: Style to assign to the object.

Returns:

Upon success, returns a pointer to a newly allocated PegHorzScroll object.

Description:

This function allocates storage for a PegHorzScroll object, initializes the object, then sets the object's data members.

Errors:

This function may generate an assert if the necessary memory for the object can not be allocated from the system and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegHorzScrollCreateDefault, PegHorzScrollSet

Notes:

This function calls PegHorzScrollCreateDefault to allocate the object, then calls PegHorzScrollSet to set the data members of the object.

The thumb button and scroll bar buttons are added to the object at this point. If the application designer wishes to use different scroll bar buttons or a different thumb button, then the application designer may implement a different Create function to create instances of custom scroll bar buttons and a thumb button.

The define PEG_SCROLLBAR_SUPPORT must be turned on in order for this function to be available.

1.103 PegHorzScrollCreateDefault

Synopsis:

```
    PegHorzScroll *PegHorzScrollCreateDefault(void);
```

Arguments:

None

Returns:

Upon success, returns a pointer to the newly allocated PegHorzScroll object.

Description:

This function allocates storage for a PegHorzScroll object. It then initializes the object with the default associated with a PegHorzScroll object type.

Errors:

This function may generate an assert if the necessary memory for the object can not be allocated from the system and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegHorzScrollCreate, PegHorzScrollInit

Notes:

This function is called by PegHorzScrollCreate to allocate the object. This function calls PegHorzScrollInit to properly initialize the object.

The define PEG_SCROLLBAR_SUPPORT must be turned on in order for this function to be available.

1.104 PegHorzScrollDraw

Synopsis:

```
void PegHorzScrollDraw(void *pThing);
```

Arguments:

pThing

    Pointer to a PegHorzScroll object.

Returns:

None

Description:

This function is an override of the PegThingDraw function.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegThingDraw

Notes:

This function draws the background stipple of the scroll bar, and also causes the child buttons to draw.

The define PEG_SCROLLBAR_SUPPORT must be turned on in order for this function to be available.

1.105 PegHorzScrollInit

Synopsis:

```
    void PegHorzScrollInit(PegHorzScroll *pHScroll);
```

Arguments:

pHScroll
Pointer to a PegHorzScroll object.

Returns:

None

Description:

This function initializes pHScroll with the defaults associated with a PegHorzScroll object type.

Errors:

This function may generate an assert if pHScroll is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegHorzScrollCreateDefault, PegHorzScrollSetDefFuncs

Notes:

This function is called by PegHorzScrollCreateDefault to initialize the object. This function calls PegHorzScrollSetDefFuncs to set the function pointers in the object.

The define PEG_SCROLLBAR_SUPPORT must be turned on in order for this function to be available.

1.106 PegHorzScrollNotify

Synopsis:

```
    PEGINT PegHorzScrollNotify(void *pThing, const PegMessage
      *pMesg);
```

Arguments:

pThing
: Pointer to a PegHorzScroll object.

pMesg
: Pointer to a PegMessage structure that contains the contents of the message.

Returns:

Various, depending on the type of message.

Description:

This function is an override of the PegThingNotify function.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegThingNotify

Notes:

This function allows the scroll bar to react properly to user and application input.

The define PEG_SCROLLBAR_SUPPORT must be turned on in order for this function to be available.

1.107 PegHorzScrollReset

Synopsis:

```
    void PegHorzScrollReset(PegHorzScroll *pHScroll, PegScrollInfo
      *pScrollInfo);
```

Arguments:

pHScroll
    Pointer to a PegHorzScroll object.
pScrollInfo
    Pointer to a PegScrollInfo structure.

Returns:

None

Description:

This function resets the scrolling info in pHScroll with the values contained in pScrollInfo. If pScrollInfo is NULL, the scroll info of pHScroll is reset to default values.

Errors:

This function may generate an assert if pHScroll is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegVertScrollReset

Notes:

The PegScrollInfo structure defines how the scroll bar actually works. See the section on scrolling in the “C/PEG Programmers Manual” for a discussion of how scrolling works in C/PEG.

The define PEG_SCROLLBAR_SUPPORT must be turned on in order for this function to be available.

1.108 PegHorzScrollResize

Synopsis:

```
void PegHorzScrollResize(void *pThing, PegRect *pRect);
```

Arguments:

pThing
: Pointer to a PegHorzScroll object.

pRect
: Pointer to a PegRect structure that defines the new size and position of the object.

Returns:

None

Description:

This function is an override of the PegThingResize function.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegThingResize

Notes:

When a PegHorzScroll object is resized, it must calculate the new size and position of the thumb button as well as the step for scrolling.

The define PEG_SCROLLBAR_SUPPORT must be turned on in order for this function to be available.

1.109 PegHorzScrollSet

Synopsis:

```
    void PegHorzScrollSet(PegHorzScroll *pHScroll, PegRect *pRect,
      PEGUSHORT usId, PEGUSHORT usStyle);
```

Arguments:

pHScroll	Pointer to a PegHorzScroll object.
pRect	Pointer to a PegRect structure that defines the size and position of the object.
usId	ID to assign the object.
usStyle	Style to assign the object.

Returns:

None

Description:

This function sets the data members of pHScroll.

Errors:

This function may generate an assert if pHScroll is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegHorzScrollCreate

Notes:

This function is called by PegHorzScrollCreate to set the data members of the object.

The define PEG_SCROLLBAR_SUPPORT must be turned on in order for this function to be available.

1.110 PegHorzScrollSetDefFuncs

Synopsis:

```
void PegHorzScrollSetDefFuncs(PegHorzScroll *pHScroll);
```

Arguments:

pHScroll
    Pointer to a PegHorzScroll object.

Returns:

None

Description:

This function sets the function pointers in pHScroll to the default functions associated with a PegHorzScroll object type.

Errors:

This function may generate an assert if pHScroll is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegHorzScrollInit

Notes:

This function is called by PegHorzScrollInit to set the function pointers in the object.

The define PEG_SCROLLBAR_SUPPORT must be turned on in order for this function to be available.

1.111 PegIconCreate

Synopsis:

```
    PegIcon *PegIconCreate(PegRect *pRect, const PegBitmap
      *pBitmap, PEGUSHORT usId, PEGUSHORT usStyle);
```

Arguments:

pRect
: Pointer to a PegRect structure that determines the size and position of the object.

pBitmap
: Pointer to a PegBitmap structure.

usId
: ID to assign the object.

usStyle
: Style to assign the object.

Returns:

Upon success, returns a pointer to a newly allocated PegIcon object.

Description:

This function allocates storage for a PegIcon structure, assigns it's data members to the parameter list and returns a pointer to the object.

Errors:

This function may generate an assert if the memory for the object can not be allocated from the system and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegIconCreateDefault, PegIconInit, PegIconSetDefFuncs, PegIconSet

Notes:

This function calls PegIconCreateDefault to allocate storage and setup the default function pointers for the object. It then calls PegIconSet with the parameter list to properly initialize the object.

1.112 PegIconCreateDefault

Synopsis:

```
    PegIcon *PegIconCreateDefault(void);
```

Arguments:

None

Returns:

Upon success, returns a pointer to a newly allocated PegIcon object.

Description:

This function allocates storage for a PegIcon object, initializes the object, and returns a pointer to the object.

Errors:

This function may generate an assert if the necessary memory for the object can not be allocated from the system and the C/PEG library was built with the PEG_USER_ASSERT directive defined.

Related Functions:

PegIconCreate, PegIconInit, PegIconSetDefFuncs

Notes:

This function calls PegIconInit to properly initialize the object.

1.113 PegIconDraw

Synopsis:

```
    void PegIconDraw(void *pThing);
```

Arguments:

pThing

Pointer to a PegIcon object.

Returns:

None

Description:

This function is an override of PegThingDraw. If the object's bitmap pointer is valid, then the object draws it's associated bitmap. Otherwise, it draws a solid color rectangle covering it's client area.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was build with the PEG_USE_ASSERT directive defined.

Related Functions:

PegThingDraw

Notes:

1.114 PegIconInit

Synopsis:

```
void PegIconInit(PegIcon *pIcon);
```

Arguments:

pIcon
    Pointer to a PegIcon object.

Returns:

None

Description:

This function initializes the default PegIcon values in pIcon for type, colors and the function pointers.

Errors:

This function may generate an assert if pIcon is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegIconCreateDefault

Notes:

This function is called by PegIconCreateDefault to properly initialize pIcon to the default PegIcon values.

1.115 PegIconSet

Synopsis:

```
    void PegIconSet(PegIcon *pIcon, PegRect *pRect, const PegBitmap
      *pBitmap, PEGUSHORT usId, PEGUSHORT usStyle);
```

Arguments:

pIcon
: Pointer to a PegIcon object.

pRect
: Pointer to a PegRect structure that defines the size and position of the object.

pBitmap
: Pointer to a PegBitmap structure.

usId
: ID to assign the object.

usStyle
: Style to assign the object.

Returns:

None

Description:

This function assigns the data members of pIcon with the parameter list values.

Errors:

This function may generate an assert if pIcon is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegIconCreate

Notes:

This function is called by PegIconCreate to properly assign the object's data members.

1.116 PegIconSetDefFuncs

Synopsis:

```
void PegIconSetDefFuncs(PegIcon *pIcon);
```

Arguments:

pIcon
: Pointer to a PegIcon object.

Returns:

None

Description:

This function sets the default PegIcon function pointers to pIcon.

Errors:

This function may generate an assert if pIcon is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegIconInit

Notes:

This function is called by PegIconInit to properly set the function pointers in pIcon.

1.117 PegListAdd

Synopsis:

```
    void PegListAdd(void *pThing, void *pAdd, PEGBOOL bRedraw);
```

Arguments:

pThing
    Pointer to a PegList or derived object.
pAdd
    Pointer to a PegThing or derived object.
bRedraw
    Optionally redraw pThing when the add operation is complete.

Returns:

None

Description:

This function is an override of the PegThingAdd function.

Errors:

This function may generate an assert if either pointer is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegListAddToEnd, PegThingAdd

Notes:

When adding objects to a list, it is important to remember that the adding procedure adds the object to the front of the list. In effect, the last object added to a list will be the first object in the list.

When adding several objects to a list at one time, it is best to pass FALSE in bRedraw during the addition process, then calling the list's Draw function after all of the objects have been added.

1.118 PegListAddToEnd

Synopsis:

```
    void PegListAddToEnd(void *pThing, void *pAdd, PEGBOOL
      bRedraw);
```

Arguments:

pThing
: Pointer to a PegList or derived object.

pAdd
: Pointer to a PegThing or derived object.

bRedraw
: Optionally redraw pThing when the add operation is complete.

Returns:

None

Description:

This function is an override of the PegThingAddToEnd function.

Errors:

This function may generate an assert if either pointer is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegListAdd, PegThingAddToEnd

Notes:

When adding several objects to a list at one time, it is best to pass FALSE in bRedraw during the addition process, then calling the list's Draw function after all of the objects have been added.

1.119 PegListClear

Synopsis:

```
    PEGINT PegListClear(void *pList);
```

Arguments:

pList

    Pointer to a PegList or derived object.

Returns:

The number of children removed.

Description:

This function removes and destroys all of the client children from a PegList or derived object.

Errors:

This function may generate an assert if pList is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegListRemove, PegDestroy

Notes:

If a child of the list has a status of PSF_NONCLIENT, then the object is not removed or destroyed.

1.120 PegListCreate

Synopsis:

```
    PegList *PegListCreate(PegRect *pRect, PEGUSHORT usId,
      PEGUSHORT usStyle);
```

Arguments:

pRect
: Pointer to a PegRect structure that defines the size and position of the object.

usId
: ID to assign the object.

usStyle
: Style to assign the object.

Returns:

Upon success, returns a pointer to a newly allocated PegList object.

Description:

This function allocates storage for a PegList object, initializes the object and sets the data members of the object.

Errors:

This function may generate an assert if the necessary memory for the object can not be allocated from the system and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegListCreateDefault, PegListSet

Notes:

This function calls PegListCreateDefault to allocate storage for a new object, then calls PegListSet to set the object's data members.

1.121 PegListCreateDefault

Synopsis:

```
    PegList *PegListCreateDefault(void);
```

Arguments:

None

Returns:

Upon success, returns a pointer to a newly allocated PegList object.

Description:

This function allocates storage for a PegList object and initializes the object using the defaults associated with a PegList object type.

Errors:

This function may generate an assert if the necessary memory for the object can not be allocated from the system and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegListCreate, PegListInit

Notes:

This function is called by PegListCreate to allocate storage for the new object. This function then calls PegListInit to properly initialize the object.

1.122 PegListIndexGetFromPtr

Synopsis:

```
PEGINT PegListIndexGetFromPtr(void *pList, void *pChild);
```

Arguments:

pList

    Pointer to a PegList or derived object.

pChild

    Pointer to a PegThing or derived object.

Returns:

The index of pChild. If pChild is NULL or not a child of pList, returns -1.

Description:

This function returns the index number of pChild as it is positioned in the PegList pointed to by pList. The index is 0 based, therefore, the first child object is index 0.

Errors:

This function may generate an assert if pList is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegListPtrGetFromIndex, PegListSelectedIndexGet

Notes:

This function only counts non-client objects. If a child object has a status of PSF_NONCLIENT, then it is not counted as part of the child list of pList.

1.123 PegListInit

Synopsis:

```
    void PegListInit(PegList *pList);
```

Arguments:

pList
    Pointer to a PegList object.

Returns:

None

Description:

This function initializes pList with the default associated with a PegList object type.

Errors:

This function may generate an assert if pList is invalid and the C/PEG library was compiled with the PEG_USE_ASSERT directive defined.

Related Functions:

PegListCreateDefault, PegListSetDefFuncs

Notes:

This function is called by PegListCreateDefault to initialize the object. This function calls PegListSetDefFuncs to set the function pointers in the object.

1.124 PegListInsert

Synopsis:

```
    void PegListInsert(void *pList, void *pInsert, PEGINT iWhere,
      PEGBOOL bSelect, PEGBOOL bRedraw);
```

Arguments:

pList
: Pointer to a PegList or derived object.

pInsert
: Pointer to a PegThing or derived object.

iWhere
: Index denoting where the insertion will take place.

bSelect
: Optionally select pInsert after the insertion process completes.

bRedraw
: Optionally redraw pList after the insertion process completes.

Returns:

None

Description:

This function inserts pInsert into the list pointed to by pList at index iWhere.

If iWhere is less than or equal to 0, then pInsert is put at the head of the list. If iWhere is greater than the number of child objects in the list, then pInsert is added to the tail of the list.

Otherwise, pInsert is inserted into the list at position iWhere. The object that previously occupied that index position, plus any objects with an index higher than iWhere, is moved down on.

Errors:

This function may generate an assert if pList is invalid and the C/PEG library was compiled with the PEG_USE_ASSERT directive defined.

Related Functions:

PegListAdd, PegListAddToEnd

Notes:

This is a fairly intense operation since the child members at indices below iWhere must be moved. This could add up to a fair amount of swapping going on. If the object is to be added to the head or the tail of the list, use PegListAdd and PegListAddToEnd, respectively, instead.

1.125 PegListNotify

Synopsis:

```
    PEGINT PegListNotify(void *pThing, const PegMessage *pMesg);
```

Arguments:

pThing

    Pointer to a PegList object.

pMesg

    Pointer to a PegMessage structure that contains the data of the message.

Returns:

Various, depending on the type of message.

Description:

This function is an override of the PegThingNotify function.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegThingNotify

Notes:

The PegList object must properly handle notifications from it's child objects and send those message to it's own parent object for processing.

1.126 PegListNumItemsGet

Synopsis:

```
    PEGINT PegListNumItemsGet(void *pList);
```

Arguments:

pList

Pointer to a PegList or derived object.

Returns:

Returns the number of client children in pList.

Description:

This function returns the number of child object which do not have PSF_NONCLIENT status in pList.

Errors:

This function may generate an assert if pList is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegListAdd, PegListAddToEnd, PegListInsert

Notes:

1.127 PegListPageDown

Synopsis:

```
void *PegListPageDown(void *pList);
```

Arguments:

pList

Pointer to a PegList or derived object.

Returns:

None

Description:

This function causes the contents of pList to scroll up one full page. If the list is already at the bottom of it's client children, this call does nothing.

Errors:

This function may generate an assert if pList is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegListPageUp, PegListSelectNext, PegListSelectPrevious

Notes:

This function allows the application code to scroll the contents of the pList object. This aids in connecting user input keys to a 'page down' key when there is not one present on the system.

This also aids in adding a scroll bar type of object to the list to allow mouse events to scroll the contents of pList.

1.128 PegListPageUp

Synopsis:

```
    void *PegListPageUp(void *pList);
```

Arguments:

pList

  Pointer to a PegList or derived object.

Returns:

None

Description:

This function causes the contents of pList to scroll down one full page. If the list is already at the top of it's client children, this call does nothing.

Errors:

This function may generate an assert if pList is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegListPageDown, PegListSelectNext, PegListSelectPrevious

Notes:

This function allows the application code to scroll the contents of the pList object. This aids in connecting user input keys to a 'page up' key when there is not one present on the system.

This also aids in adding a scroll bar type of object to the list to allow mouse events to scroll the contents of pList.

1.129 PegListPtrGetFromIndex

Synopsis:

```
    PegThing *PegListPtrGetFromIndex(void *pList, PEGINT iIndex);
```

Arguments:

pList

Pointer to a PegList or derived object.

iIndex

Index of the object to get.

Returns:

Upon success, returns a pointer to the PegThing or derived object that occupies index position iIndex. If iIndex is out of bounds, then NULL is returned.

Description:

This function returns a pointer to the child at index iIndex.

The list index is 0 based, so to returns a pointer to the first object in the list, pass 0 as the index.

Errors:

This function may generate an assert if pList is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegListIndexGetFromPtr

Notes:

1.130 PegListRemove

Synopsis:

```
    void *PegListRemove(void *pThing, void *pRemove, PEGBOOL
      bRedraw);
```

Arguments:

pThing
: Pointer to a PegList object.

pRemove
: Pointer to a PegThing or derived object.

bRedraw
: Optionally redraw pThing after the remove operation is complete.

Returns:

Upon success, returns a pointer to the removed object. If pRemove is not a child of pThing, then NULL is returned.

Description:

This function is an override of the PegThingRemove function.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegThingRemove

Notes:

The PegList object overrides this function in order to properly reposition it's children after the remove operation is complete.

As with any adding and removing, if there are several children to remove at the same time, it is best to pass FALSE in bRedraw to save drawing operations. Then calling the list's Draw function after all of the children have been removed.

1.131 PegListSelectNext

Synopsis:

```
void *PegListSelectNext(void *pList);
```

Arguments:

pList

Pointer to a PegList or derived object.

Returns:

A pointer to the newly selected child object. If at the time of this call, no child object is selected, this function returns NULL.

Description:

This function selects the next child object in pList. The child object must be a client object and have selectable status in order to be selected.

Errors:

This function may generate an assert if pList is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegListSelectPrevious

Notes:

If the currently selected object is at the end of the list, and pList has LS_WRAP_SELECT in it's style mask, then the list wraps to the front of the list and selects the first child that is a client object and is selectable.

This function allows the application to manually select objects in a PegList object.

1.132 PegListSelectPrevious

Synopsis:

```
    void *PegListSelectPrevious(void *pList);
```

Arguments:

pList

Pointer to a PegList or derived object.

Returns:

A pointer to the newly selected child object. If at the time of this call, no child object is selected, this function returns NULL.

Description:

This function selects the previous child object in pList. The child object must be a client object and have selectable status in order to be selected.

Errors:

This function may generate an assert if pList is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegListSelectNext

Notes:

If the currently selected object is at the end of the head, and pList has LS_WRAP_SELECT in it's style mask, then the list wraps to the end of the list and selects the first child that is a client object and is selectable.

This function allows the application to manually select objects in a PegList object.

1.133 PegListSelectedIndexGet

Synopsis:

```
PEGINT PegListSelectedIndexGet(void *pList);
```

Arguments:

pList
    Pointer to a PegList or derived object.

Returns:

The index of the currently selected child object in pList. If pList does not have a selected child object, then -1 is returned.

Description:

Returns the index of the currently selected child object in pList.

Errors:

This function may generate an assert if pList is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegListSelectedPtrGet

Notes:

It is not an error for a PegList object to not have a selected child. Therefore, it is important to always check the return value of this function.

1.134 PegListSelectedPtrGet

Synopsis:

```
    void *PegListSelectedPtrGet(void *pList);
```

Arguments:

pList

Pointer to a PegList or derived object.

Returns:

Upon success, pointer to the PegThing or derived child object that is selected in pList. If no child object is selected, NULL is returned.

Description:

This function returns a pointer to the currently selected child object in pList.

Errors:

This function may generate an assert if pList is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegListSelectedIndexGet

Notes:

It is not an error for a PegList object to not have a selected child. Therefore, it is important to always check the return value of this function.

1.135 PegListSelectedSetFromIndex

Synopsis:

```
    void *PegListSelectedSetFromIndex(void *pList, PEGINT iIndex);
```

Arguments:

pList

    Pointer to a PegList or derived object.

iIndex

    Index of the object to select.

Returns:

Upon success, a pointer to the selected child object. If iIndex is out of bounds, then NULL is returned.

Description:

This function allows application code to set the currently selected child object in pList. If this entails selecting a new object in the list, a message to the list's parent is generate in the same manner as if the user had selected to new object in the list.

Errors:

This function may generate an assert if pList is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegListSelectedSetFromPtr

Notes:

The index of a list is zero based, therefore to select the first child object in the list, use index 0. The child must be a client object and able to be selected in order for the child object to actually be selected.

1.136 PegListSelectedSetFromPtr

Synopsis:

```
void PegListSelectedSetFromPtr(void *pList, void *pThing);
```

Arguments:

pList
    Pointer to a PegList or derived object.
pThing
    Pointer to a PegThing or derived object.

Returns:

None

Description:

This function allows application code to set the currently selected child object in pList. If this entails selecting a new object in the list, a message to the list's parent is generate in the same manner as if the user had selected to new object in the list.

Errors:

This function may generate an assert if pList is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegListSelectedSetFromIndex

Notes:

The child must be a client object and able to be selected in order for the child object to actually be selected.

1.137 PegListSet

Synopsis:

```
    PegList *PegListCreate(PpegList *pList, PegRect *pRect,
      PEGUSHORT usId, PEGUSHORT usStyle);
```

Arguments:

pList
: Pointer to a PegList object.

pRect
: Pointer to a PegRect structure that defines the size and position of the object.

usId
: ID to assign the object.

usStyle
: Style to assign the object.

Returns:

None

Description:

This function sets the data members of the pList.

Errors:

This function may generate an assert if pList is invalid and the C/PEG library was built with the PEG_USE_ASSSERT directive defined.

Related Functions:

PegListCreate

Notes:

This function is called by PegListCreate to set the data members in the object.

1.138 PegListSetDefFuncs

Synopsis:

```
    void PegListSetDefFuncs(PegList *pList);
```

Arguments:

pList

Pointer to a PegList object.

Returns:

None

Description:

This function sets the function pointers in pList to the default functions associated with a PegList object type.

Errors:

This function may generate an assert if pList is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegListInit

Notes:

This function is called by PegListInit to properly set the function pointers in the object.

1.139 PegMessagePanelCreate

Synopsis:

```
PegMessagePanel *PegMessagePanelCreate(PegRect *pRect, const
  PEGSTRING Title, const PEGSTRING Message, const PegBitmap
  *pBitmap, void *pReportTo, PEGUSHORT usId, PEGUSHORT
  usStyle);
```

Arguments:

pRect
: Pointer to a PegRect structure that defines the size and position of the object.

Title
: a NULL terminated string that is used for the title text. This parameter may be NULL if a title is not desired.

Message
: a NULL terminated string used as the text of the message. This parameter may be NULL if the message text is not desired.

pBitmap
: Pointer to a PegBitmap structure used as an icon in the message panel body. This parameter my be NULL if no icon is desired.

pReportTo
: Pointer to a PegThing or derived object that will receive the status of the message panel when the message panel is closed. If this parameter is NULL, the status message is sent to the parent of the message panel object.

usId
: ID to assign the object.

usStyle
: Style to assign the object.

Returns:

Upon success, returns a pointer to the newly allocated PegMessagePanel object.

Description:

This function allocates storage for a PegMessagePanel object and returns a pointer to it.

Errors:

This function may generate an assert if the necessary memory can not be allocated by the system and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegMessagePanelCreateDefault, PegMessagePanelSet

Notes:

This function call PegMessagePanelCreateDefault to allocate and initialize a new object. It then calls PegMessagePanelSet with the passed in parameter list to assign the data members of the object.

Aside from the PegPanel supported style types, the PegMessagePanel uses style flags to add buttons to itself. In other words, the proper way to add an “OK” button at the bottom of the panel is to pass in the MP_OK style in the usStyle parameter.

Listed in the table below are the valid style flags, the ID and the constant text pointer assigned to the button. These flags may be logically OR'd together to add any of the buttons in any combination as the application sees fit.

Style Flag	Button ID	ButtonText
MP_OK	PEG_IDB_OK	PEGSC_OK
MP_YES	PEG_IDB_YES	PEGSC_YES
MP_NO	PEG_IDB_NO	PEGSC_NO
MP_ABORT	PEG_IDB_ABORT	PEGSC_ABORT
MP_RETRY	PEG_IDB_RETRY	PEGSC_RETRY
MP_CANCEL	PEG_IDB_CANCEL	PEGSC_CANCEL

The ID given to the button is important in that the ID comprises the return value of calling the PegMessagePanelExecute function. This is the way for the caller to determine which button was used by the user to dismiss the message panel.

For illustrative purposes, here is a code snippet showing the use of the style flags to put the OK and Cancel buttons on the message panel, and how to determine the button that was used to dismiss the panel.

```
PegMessagePanel *pMesgPanel = PegMessagePanelCreate(&r,
    “Title Text”, “Message Text”, NULL, NULL, ID_MESG_PANEL,
  MP_OK | MP_CANCEL);
  if (PegExecute(pMesgPanel) == PEG_IDB_OK)
{
    /* the user pressed the OK button */
}
else
{
    /* the user pressed the cancel button */
}
```

This should clarify that the button ID makes up the return value from the PegExecute call.

The buttons are always added to the bottom of the panel in the following order:

1) MP_OK
2) MP_YES
3) MP_NO
4) MP_ABORT
5) MP_RETRY
6) MP_CANCEL

Likewise, the message text is always centered in the panel. If there is a bitmap icon present, then the bitmap is presented on the left, vertical center, of the panel, and the text is located on the right, vertical center of the panel. The title is always on top and spans the width of the client region of the panel.

1.140 PegMessagePanelCreateDefault

Synopsis:

```
    PegMessagePanel *PegMessagePanelCreateDefault(void);
```

Arguments:

None

Returns:

Upon success, returns a pointer to a newly allocated PegMessagePanel object.

Description:

This function allocates storage for a new PegMessagePanel object. It also initializes the object to the default PegMessagePanel settings for type, colors and function pointers.

Errors:

This function may generate an assert if the necessary memory can not be allocated by the system and the C/PEG library was defined with the PEG_USE_ASSERT directive defined.

Related Functions:

PegMessagePanelCreate, PegMessagePanelInit

Notes:

This function is called by PegMessagePanelCreate to allocate and initialize a new PegMessagePanel object.

1.141 PegMessagePanelFontSet

Synopsis:

```
    void PegMessagePanelFontSet(void *pThing, PegFont *pFont);
```

Arguments:

pThing
    Pointer to a PegMessagePanel object.
pFont
    Pointer to a PegFont structure.

Returns:

None

Description:

This function is an override of PegPanelFontSet.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegPanelFontSet, PegTextThingFontSet

Notes:

This function is used to set the font in the object that does the actual drawing of the message text on the object.

1.142 PegMessagePanelInit

Synopsis:

```
    void PegMessagePanelInit(PegMessagePanel *pMesgPanel);
```

Arguments:

pMesgPanel
: Pointer to a PegMessagePanel object.

Returns:

None

Description:

This function initializes pMesgPanel with the default type, function pointers, colors and status of a PegMessagePanel object.

Errors:

This function may generate an assert if pMesgPanel is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegMessagePanelCreateDefault

Notes:

This function is called by PegMessagePanelCreateDefault to properly initialize the new object.

1.143 PegMessagePanelNotify

Synopsis:

```
    PEGINT PegMessagePanelNotify(void *pThing, const PegMessage
      *pMesg);
```

Arguments:

pThing
    Pointer to a PegMessagePanel object.
pMesg
    Pointer to a PegMessage structure that contains the message data.

Returns:

Various, depending on the contents of the message.

Description:

This is an override of PegPanelNotify.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was defined with the PEG_USE_ASSERT directive defined.

Related Functions:

PegPanelNotify

Notes:

The PegMessagePanel does some extra work when it is current and non-current to make sure that the panel's title, if available, draws itself properly to indicate that the panel has focus.

1.144 PegMessagePanelSet

Synopsis:

```
void PegMessagePanelSet(PegMessage *pMesgPanel, PegRect *pRect,
  const PEGSTRING Title, const PEGSTRING Message, const
  PegBimap *pBitmap, void *pReportTo, PEGUSHORT usId,
  PEGUSHORT usStyle);
```

Arguments:

pMesgPanel
: Pointer to a PegMessagePanel object.

pRect
: Pointer to a PegRect structure that defines the size and position of the object.

Title
: Pointer to a NULL terminated string that is used for the title text. This parameter may be NULL if a title is not desired.

Message
: Pointer to a NULL terminated string used as the text of the message. This parameter may be NULL if the message text is not desired.

pBitmap
: Pointer to a PegBitmap structure used as an icon in the message panel body. This parameter my be NULL if no icon is desired.

pReportTo
: Pointer to a PegThing or derived object that will receive the status of the message panel when the message panel is closed. If this parameter is NULL, the status message is sent to the parent of the message panel object.

usId
: ID to assign the object.

usStyle
: Style to assign the object.

Returns:

None

Description:

This function is used to set the data members of a PegMessagePanel object.

Errors:

This function may generate an assert if pMesgPanel is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegMessagePanelCreate

Notes:

This function is called by PegMessagePanelCreate to properly assign the object's data members.

See the Notes section of PegMessagePanelCreate for an explanation of the style flags available for this object.

1.145 PegMessagePanelSetDefFuncs

Synopsis:

```
void PegMessagePanelSetDefFuncs(PegMessagePanel *pMesgPanel);
```

Arguments:

pMesgPanel
Pointer to a PegMessagePanel object.

Returns:

None

Description:

This function set the function pointers in pMesgPanel to the default function panels for a PegMessagePanel object.

Errors:

This function may generate an assert if pMesgPanel is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegMessagePanelInit

Notes:

This function is called by PegMessagePanelInit to properly set the function pointers in the object.

1.146 PegMessagePanelTextSet

Synopsis:

```
    void PegMessagePanelTextSet(void *pThing, const PEGCHAR
      *pText);
```

Arguments:

pThing
: Pointer to a PegMessagePanel object.

pText
: Pointer to a NULL terminated string that is used for the message text on the pThing.

Returns:

None

Description:

This function is an override of PegPanelTextSet and is used to set the text in the object in the PegMessagePanel that displays the message text.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegPanelTextSet, PegTextThingTextSet, PegTextSet

Notes:

1.147 PegMLPromptCreate

Synopsis:

```
    PegMLPrompt *PegMLPromptCreate(PegRect *pRect, const PEGSTRING
      String, PEGUSHORT usId, PEGUSHORT usStyle);
```

Arguments:

pRect
: Pointer to a PegRect structure that defines the size and position of the object.

String
: a NULL terminated string.

usId
: ID to assign the object.

usStyle
: Style to assign the object.

Returns:

Upon success, returns a pointer to a newly allocated PegMLPrompt object.

Description:

This function creates storage for a PegMLPrompt object and returns a pointer to it.

Errors:

This function may generate an assert if the necessary memory for the object can not be allocated from the system and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegMLPromptCreateDefault, PegMLPromptSet

Notes:

This function calls PegMLPromptCreateDefault to create a new object, then call PegMLPromptSet with the new object and parameter list to assign the object's data members.

If the style flag AF_DRAW_SELECTED is passed in with usStyle, the object will draw itself in selected colors when the object is selected.

1.148 PegMLPromptCreateDefault

Synopsis:

```
    PegMLPrompt *PegMLPromptCreateDefault(void);
```

Arguments:

None

Returns:

Upon success, returns a pointer to a newly allocated PegMLPrompt object.

Description:

This function allocates storage for a PegMLPrompt object and sets the object's defaults for type, function pointers and colors to the defaults for a PegMLPrompt object.

Errors:

This function may generate an assert if the necessary memory for the object can not be allocated from the system and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegMLPromptCreate, PegMLPromptInit

Notes:

This function calls PegMLPromptInit to properly initialize the object.

1.149 PegMLPromptDestroy

Synopsis:

```
void PegMLPromptDestroy(void *pThing);
```

Arguments:

pThing

Pointer to a PegMLPrompt object.

Returns:

None

Description:

This function is an override of the PegTextThingDestroy function. Since the PegMLPrompt object deals with multiple lines of text, the object allocates pointers that coincide with the start of each line in it's text string. This array of pointers must be freed when the object is destroyed.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegTextThingDestroy, PegThingDestroy

Notes:

1.150 PegMLPromptDraw

Synopsis:

```
void PegMLPromptDraw(void *pThing);
```

Arguments:

pThing
    Pointer to a PegMLPrompt object.

Returns:

None

Description:

This function is an override of the PegTextThingDraw function.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was defined with the PEG_USE_ASSERT directive defined.

Related Functions:

PegTextThingDraw, PegMLPromptDrawText

Notes:

This function calls PegMLPromptDrawText to do the actual text drawing. This allows application designers the ability to override how the PegMLPromptObject draws it's frame and background, without the needs to replace the text drawing.

1.151 PegMLPromptDrawText

Synopsis:

```
    void PegMLPromptDrawText(PegMLPrompt *pMLPrompt);
```

Arguments:

pMLPrompt
    Pointer to a PegMLPrompt object.

Returns:

None

Description:

This function does the actual text rendering for a PegMLPrompt object.

Errors:

This function may generate an assert if pMLPrompt is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegMLPromptDraw

Notes:

The text rendering for PegMLPrompt is separate from the way the object draws it's background and frame to give the application designer flexibility in creating custom objects based on PegMLPrompt that may draw their background and frame in a custom manner, but still need to implement the default text rendering.

1.152 PegMLPromptFontSet

Synopsis:

```
void PegMLPromptFontSet(void *pThing, PegFont *pFont);
```

Arguments:

pThing
    Pointer to a PegMLPrompt object.
pFont
    Pointer to a PegFont structure.

Returns:

None

Description:

This function is an override of the PegTextThingFontSet function. When the font is set on a PegMLPrompt object, it must recalculate the line start array used to keep track of where to break it's internal text into individual lines.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegTextThingFontSet, PegMLPromptTextSet

Notes:

1.153 PegMLPromptInit

Synopsis:

```
    void PegMLPromptInit(PegMLPrompt *pMLPrompt);
```

Arguments:

pMLPrompt
    Pointer to a PegMLPrompt object.

Returns:

None

Description:

This function sets pMLPrompt to the defaults for a PegMLPrompt object including it's type, function pointers and colors.

Errors:

This function may generate an assert if pMLPrompt is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegMLPromptCreateDefault, PegMLPromptSetDefFuncs

Notes:

This function is called by PegMLPromptCreateDefault when creating a new default PegMLPrompt object. This function also calls PegMLPromptSetDefFuncs to set the default function pointers in the object.

1.154 PegMLPromptLineDown

Synopsis:

```
    void PegMLPromptLineDown(PegMLPrompt *pMLPrompt, PEGBOOL
      bRedraw);
```

Arguments:

pMLPrompt
    Pointer to a PegMLPrompt object.
bRedraw
    Optionally redraw pMLPrompt after the line down operation.

Returns:

None

Description:

If the number of visible lines in pMLPrompt is less than the total number of lines associated with pMLPrompt's text, then this function will cause the text to scroll up one line, exposing a new line of text at the bottom. If the bottom visible line of text is the last line of text in the object, then this call is ignored.

Errors:

This function may generate an assert if pMLPrompt is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegMLPromptLineUp, PegMLPromptPageDown, PegMLPromptPageUp

Notes:

This function provides an easy way for application code to manually scroll the contents of a PegMLPrompt object.

When keyboard support is enabled in the C/PEG library, this function is called when the object processes a PK_LNDN key. The PK_LNDN key usually corresponds to a down arrow key on AT style keyboards, but may be optionally configured by the system designer to map to a soft key or some other arbitrary input mechanism.

1.155 PegMLPromptLineUp

Synopsis:

```
    void PegMLPromptLineUp(PegMLPrompt *pMLPrompt, PEGBOOL
      bRedraw);
```

Arguments:

pMLPrompt
 Pointer to a PegMLPrompt object.

bRedraw
 Optionally redraw pMLPrompt after the line down operation.

Returns:

None

Description:

If the number of visible lines in pMLPrompt is less than the total number of lines associated with pMLPrompt's text, then this function will cause the text to scroll down one line, exposing a new line of text at the top. If the top visible line of text is the first line of text in the object, then this call is ignored.

Errors:

This function may generate an assert if pMLPrompt is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegMLPromptLineDown, PegMLPromptPageDown, PegMLPromptPageUp

Notes:

This function provides an easy way for application code to manually scroll the contents of a PegMLPrompt object.

When keyboard support is enabled in the C/PEG library, this function is called when the object processes a PK_LNUP key. The PK_LNUP key usually corresponds to a up arrow key on AT style keyboards, but may be optionally configured by the system designer to map to a soft key or some other arbitrary input mechanism.

1.156 PegMLPromptNotify

Synopsis:

```
PEGINT PegMLPromptNotify(void *pThing, const PegMessage
  *pMesg);
```

Arguments:

pThing
Pointer to a PegMLPrompt object.

pMesg
Pointer to a PegMessage structure that contains the message data.

Returns:

Various, depending on the contents of the message.

Description:

This function is an override of the PegTextThingNotify function. The PegMLPrompt object handles key processing when keyboard support is enabled in the library as well as determining how to draw itself when it is selected or not selected.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegTextThingNotify

Notes:

The PegMLPrompt object will only draw itself in different colors if the AF_DRAW_SELECTED flag is part of it's style data.

1.157 PegMLPromptPageDown

Synopsis:

```
    void PegMLPromptPageDown(PegMLPrompt *pMLPrompt, PEGBOOL
      bRedraw);
```

Arguments:

pMLPrompt
    Pointer to a PegMLPrompt object.
bRedraw
    Optionally redraw pMLPrompt after the page down operation has completed.

Returns:

None

Description:

This function allows application code to easily scroll the contents of pMLPrompt a full page down. A page is defined as the number of lines of text that are visible on the object at one time.

If the total number of lines exceeds the number of visible lines in the object, then this function will scroll a page of lines down, thus moving the current visible lines up one page of lines and exposing an entirely new page of lines. If the number of lines below the currently visible bottom line is less than an entire page of lines, the number of lines scrolled is equal to the remaining number of lines in the object. If the currently visible bottom line is the last line of text, this function does nothing.

Errors:

This function may generate an assert if pMLPrompt is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegMLPromptPageUp, PegMLPromptLineDown, PegMLPromptLineUp

Notes:

1.158 PegMLPromptPageUp

Synopsis:

```
    void PegMLPromptPageUp(PegMLPrompt *pMLPrompt, PEGBOOL
      bRedraw);
```

Arguments:

pMLPrompt
    Pointer to a PegMLPrompt object.
bRedraw
    Optionally redraw pMLPrompt after the page down operation has completed.

Returns:

None

Description:

This function allows application code to easily scroll the contents of pMLPrompt a full page up. A page is defined as the number of lines of text that are visible on the object at one time.

If the total number of lines exceeds the number of visible lines in the object, then this function will scroll a page of lines up, thus moving the current visible lines down one page of lines and exposing an entirely new page of lines. If the number of lines above the currently visible top line is less than an entire page of lines, the number of lines scrolled is equal to the top number of lines in the object minus the current top line. If the currently visible top line is the first line of text, this function does nothing.

Errors:

This function may generate an assert if pMLPrompt is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegMLPromptPageDown, PegMLPromptLineDown, PegMLPromptLineUp

Notes:

1.159 PegMLPromptResize

Synopsis:

```
void PegMLPromptResize(void *pThing, PegRect *pRect);
```

Arguments:

pThing
: Pointer to a PegMLPrompt object.

pRect
: Pointer to a PegRect structure that determines the new size and position of pThing.

Returns:

None

Description:

This function is an override of the PegTextThingResize function. For the PegMLPrompt object, it is necessary to determine if the new size of the object will affect the number of total lines in the object, the number of visible lines in the object and where the line breaks occur.

Errors:

Thing function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegTextThingResize

Notes:

1.160 PegMLPromptSet

Synopsis:

```
    void PegMLPromptSet(PegMLPrompt *pMLPrompt, PegRect *pRect,
      const PEGSTRING String, PEGUSHORT usId, PEGUSHORT usStyle);
```

Arguments:

pMLPrompt
: Pointer to a PegMLPrompt object.

pRect
: Pointer to a PegRect structure that determines the size and position of pMLPrompt.

String
: a NULL terminated string used to set the text of pMLPrompt.

usId
: ID to assign the object.

usStyle
: Style to assign the object.

Returns:

None

Description:

This function assigns the data members of pMLPrompt with the passed over parameter list.

Errors:

This function may generate an assert if pMLPrompt is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegMLPromptCreate, PegTextThingSet

Notes:

This function is called by PegMLPromptCreate to set the data members of the new object.

1.161 PegMLPromptSetDefFuncs

Synopsis:

```
void PegMLPromptSetDefFuncs(PegMLPrompt *pMLPrompt);
```

Arguments:

pMLPrompt
Pointer to a PegMLPrompt object.

Returns:

None

Description:

This function set the function pointers in pMLPrompt to the default functions of a PegMLPrompt object.

Errors:

This function may generate an assert if pMLPrompt is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegMLPromptInit

Notes:

This function is called by PegMLPromptCreateDefault to properly set the function pointers in pMLPrompt.

1.162 PegMLPromptTextSet

Synopsis:

```
void PegMLPromptTextSet(void *pThing, const PEGCHAR *pText);
```

Arguments:

pThing

Pointer to a PegMLPrompt object.

pText

Pointer to a NULL terminated string to assign to pThing.

Returns:

None

Description:

This function is an override of the PegTextThingTextSet function. The PegMLPrompt object determines the line breaks for pText after it has been assigned to the object.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegTextThingTextSet

Notes:

1.163 PegMLTextButtonCreate

Synopsis:

```
    PegMLTextButton *PegMLTextButtonCreate(PegRect *pRect, const
      PEGSTRING String, PEGUSHORT usId, PEGUSHORT usStyle);
```

Arguments:

pRect
: Pointer to a PegRect structure that defines the size and position of the object.

String
: a NULL terminated string to set in the object.

usId
: ID to assign to the object.

usStyle
: Style to assign to the object.

Returns:

Upon success, returns a pointer to a newly allocated PegMLTextButton object.

Description:

This function allocates storage for a PegMLTextButton object, initializes the object and sets the object's data members to the parameters passed on the argument list.

Errors:

This function may generate an assert if the necessary memory can not be allocated by the system and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegMLTextButtonCreateDefault, PegMLTextButtonSet

Notes:

This function first calls PegMLTextButtonCreateDefault to allocate and initialize a new object, then calls PegMLTextButtonSet to set the data members.

The string pointed to by pText may include hard line breaks set by the user of the object. The hard line break is mapped to the define constant

PEG_ML_TEXT_BUTTON_BREAK. In the default implementation, this is defined as 0x7c, which is the pipe character. Therefore, when this character appears in the text, the logic in the button assumes that this a hard line break and begins a new line at the first non-space character following the line break symbol.

If the application designer wishes to use a line break character other than 0x7c, the define in cpeg/include/pconfig.h may be modified and the C/PEG library rebuilt.

1.164 PegMLTextButtonCreateDefault

Synopsis:

```
    PegMLTextButton *PegMLTextButtonCreateDefault(void);
```

Arguments:

None

Returns:

Upon success, returns a pointer to a newly allocated PegMLTextButton object.

Description:

This function allocates storage for a PegMLTextButton object. It then initializes the object.

Errors:

This function may generate an assert if the necessary memory for the object can not be allocated by the system and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegMLTextButtonCreate, PegMLTextButtonInit

Notes:

This function calls PegMLTextButtonInit to properly initialize the object.

1.165 PegMLTextButtonDestroy

Synopsis:

```
    void PegMLTextButtonDestroy(void *pThing);
```

Arguments:

pThing

Pointer to a PegMLTextButton object.

Returns:

None

Description:

This function is an override of the PegButtonDestroy function.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegButtonDestroy

Notes:

1.166 PegMLTextButtonDraw

Synopsis:

```
void PegMLTextButtonDraw(void *pThing);
```

Arguments:

pThing
Pointer to a PegMLTextButton object.

Returns:

None

Description:

This function is an override of the PegButtonDraw function. The PegMLTextButton draws it's associated text on any number of lines, determined by the size of the object as well as the number of embedded hard line breaks of the text string.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegButtonDraw

Notes:

See the Notes section of PegMLTextButtonCreate for a discussion on embedded hard line breaks in the text string for PegMLTextButtons.

1.167 PegMLTextButtonFontSet

Synopsis:

```
void PegMLTextButtonFontSet(void *pThing, PegFont *pFont);
```

Arguments:

pThing
    Pointer to a PegMLTextButton object.
pFont
    Pointer to a PegFont structure.

Returns:

None

Description:

This function is an override of the PegButtonFontSet function. The PegMLTextButton function includes logic for determining the line starts of it's text string based on the size of the font.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegButtonFontSet, PegTextThingFontSet

Notes:

1.168 PegMLTextButtonInit

Synopsis:

```
void PegMLTextButtonInit(PegMLTextButton *pMLTButton);
```

Arguments:

pMLTButton
    Pointer to a PegMLTextButton object.

Returns:

None

Description:

This function initializes pMLTButton with the correct type, function pointers and color used for the default PegMLTextButton object.

Errors:

This function may generate an assert if pMLTButton is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegMLTextButtonCreateDefault, PegMLTextButtonSetDefFuncs

Notes:

This function is called by PegMLTextButtonCreateDefault to properly initialize the object.

1.169 PegMLTextButtonSet

Synopsis:

```
void PegMLTextButtonSet(PegMLTextButton *pMLTButton, PegRect
  *pRect, const PEGSTRING String, PEGUSHORT usId, PEGUSHORT
  usStyle);
```

Arguments:

pMLTButton	Pointer to a PegMLTextButton object.
pRect	Pointer to a PegRect structure that defines the size and position of the object.
String	a NULL terminated string to set in the object.
usId	ID to assign to the object.
usStyle	Style to assign to the object.

Returns:

None

Description:

This function set the data members of pMLTButton with the arguments in the parameter list.

Errors:

This function may generate an assert if pMLTButton is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegMLTextButtonCreate

Notes:

This function is called by PegMLTextButtonCreate to properly set the data members of the object.

1.170 PegMLTextButtonSetDefFuncs

Synopsis:

```
    void PegMLTextButtonSetDefFuncs(PegMLTextButton *pMLTButton);
```

Arguments:

pMLTButton
    Pointer to a PegMLTextButton object.

Returns:

None

Description:

This function sets the function pointers in pMLTButton to the default functions associated with a PegMLTextButton object.

Errors:

This function may generate an assert if pMLTButton is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegMLTextButtonCreateDefault, PegMLTextButtonInit

Notes:

This function is called by PegMLTextButtonInit to properly set the function pointers of the object.

1.171 PegMLTextButtonTextSet

Synopsis:

```
    void PegMLTextButtonTextSet(void *pThing, const PEGCHAR
      *pText);
```

Arguments:

pThing
    Pointer to a PegMLTextButton object.
pText
    Pointer to a NULL terminated string.

Returns:

None

Description:

This function is an override of the PegButtonTextSet function. When the text is set on a PegMLTextButton object, the object must determine the number of lines in the text and where the line breaks lie within the string.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegButtonTextSet, PegTextThingTextSet

Notes:

See the Notes section of PegMLTextButtonCreate for a discussion on embedding hard line breaks in a string.

1.172 PegPanelCreate

Synopsis:

```
    PegPanel *PegPanelCreate(PegRect *pRect, PEGUSHORT usId,
        PEGUSHORT usStyle);
```

Arguments:

pRect
: Pointer to a PegRect structure that determines the size and position of the panel object. Pass NULL in this parameter to default the panel to occupy the entire screen.

usId
: ID to assign to the object.

usStyle
: Style to assign to the object.

Returns:

Upon success, returns a pointer to a newly allocated PegPanel object.

Description:

This function allocates storage for a PegPanel object. It then initializes the object and sets it's data members.

Errors:

This function may generate and assert if the memory for the object can not be allocated by the system and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegPanelCreateDefault, PegPanelSet

Notes:

PegPanel objects are generally used as top level objects (an object whose parent is the PegPresentation object) that contain many child objects. Given this, the common use for a PegPanel object is container for a group of objects that present the user with information of and options for defining the state of the system. Most frequently, the panel will occupy the entire screen. Thus, by passing NULL in pRect, the logic of creating a new PegPanel object sets the area occupied by the object as the entire screen. This saves the application developer from the task of creating a PegRect structure and populating it's members with known coordinates.

This function calls PegPanelCreateDefault to allocate the storage for the object and PegPanelSet to set it's data members.

The PegPanel introduces a new style flag regarding how the object's border is draw: FF_THICK. This is in addition to the frame style flags supported by PegThing.

The PegPanel object also introduces a new function pointer to the function pointers in PegThing and PegTextThing: Execute. The execute function is used to modally execute a panel object, either locally or globally. What this means is that a panel object may appear on the screen and capture all of the mouse and keyboard input, thus, disallowing any other object on the screen, that is not a child of the modally executing dialog, from gaining focus or responding to user interaction.

The caveat to this is that if the panel is executing locally, then other objects that are executing within the context of other tasks or threads on the system may gain keyboard and mouse control. Usually, this is the desired behavior. On the other hand, if a PegPanel object executes modally on a global scope, then no other object may receive mouse or keyboard input, regardless of the context in which they are running.

See the Notes section of PegPanelExecute for a further discussion on this subject.

1.173 PegPanelCreateDefault

Synopsis:

```
    PegPanel *PegPanelCreatDefault(void);
```

Arguments:

None

Returns:

Upon success, returns a pointer to a newly allocated PegPanel object.

Description:

This function allocates storage for a PegPanel object and initializes the object.

Errors:

This function may generate and assert if the memory for the object can not be allocated by the system and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegPanelCreatet, PegPanelInit

Notes:

This function calls PegPanelInit to properly initialize the object.

1.174 PegPanelDraw

Synopsis:

```
    void PegPanelDraw(void *pThing);
```

Arguments:

pThing
  Pointer to a PegPanel object.

Returns:

None

Description:

This function is an override of the PegTextThingDraw function. The PegPanel object supports background wallpapers, and so must override the draw function to properly draw the wallpaper bitmap.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegPanelDrawBorder, PegTextThingDraw, PegThingDraw

Notes:

1.175 PegPanelDrawBorder

Synopsis:

```
void PegPanelDrawBorder(void *pThing, PEGCOLORVAL c);
```

Arguments:

pThing

    Pointer to a PegPanel object.

c

    Color value used to fill in the client area of the object.

Returns:

None

Description:

This is an override of the PegTextThingDrawBorder function. The PegPanel object introduces the FF_THICK frame style which draws a thick, 3D frame around the client area of the object.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegPanelDraw, PegTextThingDrawBorder, PegThingDrawBorder

Notes:

For target systems that support less than four colors, the FF_THICK frame is drawn wide, but not 3D looking, since there are not enough colors to work with to accurately display this effect.

1.176 PegPanelExecute

Synopsis:

```
PEGINT PegPanelExecute(void *pPanel, PEGUSHORT usMode);
```

Arguments:

pPanel
    Pointer to a PegPanel object.
usMode
    Defines the execution operation.

Returns:

Various. Will generally return the same value as the PegPanelNotify function for various message types.

Description:

This function provides an optional way to add an object to PegPresentation and have the panel object display on the screen. Instead of only appearing on the screen, the panel may also, optionally, depending on the value passed in the usMode parameter, capture all user input messages from the mouse and keyboard, thus not allowing any other object who are direct children of PegPresentation to gain focus.

This technique is used to display dialog messages that the user must explicitly dismiss.

Errors:

This function may generate an assert if pPanel is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegSubTaskExecuteBegin, PegMessageRouteToTask, PegMessageDispatch

Notes:

The usMode parameter may be exactly one of the following as outlined in the following table.

Argument	Meaning
PEF_NORMAL	Normal execution. Works the same regardless of threading model.
PEF_MODAL	Modal execution. In stand alone mode, this is essentially equivalent to PEF_GLOBAL_MODAL. When running in a multi-tasking environment, a new message queue is created for the process in which the panel is executing. Any other panel object running within the same context as the modally executing panel will not receive input focus or input messages from PegPresentation until this panel is removed or closed.
PEG_GLOBAL_MODAL	Global modal execution. In stand alone mode, this is equivalent to PEF_MODAL. In a multi-tasking environment, a new message queue is created for the process in which the panel is running. Any other panel object, regardless of process association, will not receive input focus or input messages from PegPresentation until this panel is removed or closed.

1.177 PegPanelInit

Synopsis:

```
void PegPanelInit(PegPanel *pPanel);
```

Arguments:

pPanel
  Pointer to a PegPanel object.

Returns:

None

Description:

This function initializes pPanel with the default type, function pointers, status and colors associated with a PegPanel object.

Errors:

This function may generate an assert if pPanel is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegPanelCreateDefault, PegPanelSetDefFuncs

Notes:

This function is called by PegPanelCreateDefault to properly initialize the object. This function also calls PegPanelSetDefFuncs to properly set the object's function pointers.

1.178 PegPanelNotify

Synopsis:

```
PEGINT PegPanelNotify(void *pThing, const PegMessage *pMesg);
```

Arguments:

pThing
: Pointer to a PegPanel object.

pMesg
: Pointer to a PegMessage structure which contains the message data.

Returns:

Various, depending on the contents of the message.

Description:

This function is an override of the PegTextThingNotify function.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegTextThingNotify, PegThingNotify

Notes:

This function has logic for handling child objects and returning an exit code if the panel is closed.

1.179 PegPanelResize

Synopsis:

```
void PegPanelResize(void *pThing, PegRect *pRect);
```

Arguments:

pThing

  Pointer to a PegPanel object.

pRect

  Pointer to a PegRect structure that determines the new size and position of the object.

Returns:

None

Description:

This function is an override of the PegTextThingResize function.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegTextThingResize, PegThingResize

Notes:

1.180 PegPanelSet

Synopsis:

```
    void PegPanelSet(PegPanel *pPanel, PegRect *pRect, PEGUSHORT
      usId, PEGUSHORT usStyle);
```

Arguments:

pPanel
: Pointer to a PegPanel object.

pRect
: Pointer to a PegRect structure that determines the size and position of the panel object. Pass NULL in this parameter to default the panel to occupy the entire screen.

usId
: ID to assign to the object.

usStyle
: Style to assign to the object.

Returns:

Upon success, returns a pointer to a newly allocated PegPanel object.
Description:

This function assigns the data members of pPanel with the parameter list.

Errors:

This function may generate an assert if pPanel is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegPanelCreate

Notes:

See the Notes section of PegPanelCreate for a description of the valid parameter values.

1.181 PegPanelSetDefFuncs

Synopsis:

```
    void PegPanelSetDefFuncs(PegPanel *pPanel);
```

Arguments:

pPanel
Pointer to a PegPanel object.

Returns:

None

Description:

This function sets the function pointers in pPanel to point to the default functions used by the PegPanel type.

Errors:

This function may generate an assert if pPanel is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegPanelInit

Notes:

This function is called by PegPanelInit to properly set up the function pointers in the object.

1.182 PegPresentationAdd

Synopsis:

```
void PegPresentationAdd(void *pThing, void *pAdd, PEGBOOL
  bRedraw);
```

Arguments:

pThing
: Pointer to the PegPresentation object.

pAdd
: Pointer to a PegThing or derived object to add.

bRedraw
: Optionally redraw after the addition is complete.

Returns:

None

Description:

This function is an override of the PegPanelAdd function. There is extra house keeping that the PegPresentation object must do when top level objects are added.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegPanelAdd

Notes:

As with any add function, the child being added becomes the first child of the object to which it is being added. For the PegPresentation object, this may mean adjusting which object has focus and which objects may draw.

1.183 PegPresentationCreate

Synopsis:

```
    PegPresentation *PegPresentationCreate(PegRect *pRect);
```

Arguments:

pRect

Pointer to a PegRect structure that defines the size of the PegPresentation object. The top, left coordinates in this rectangle will always be 0, 0. The bottom coordinate will always be the vertical size of the screen minus one. The right coordinate will always be the horizontal size of the screen minus one.

Returns:

Upon success, returns a pointer to the PegPresentation object.

Description:

This function creates the PegPresentation object for the system. This function is called during C/PEG's initialization process and does not need to be called by application code.

Errors:

This function may generate an assert if the necessary memory for the object can not be allocated from the system and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegPresentationInit

Notes:

This function is called when the C/PEG library is initializing, before PegAppInitialize is called. If the application needs to implement a custom PegPresentation object, then it is proper to delete the default version and install the custom version in PegAppInitialize.

1.184 PegPresentationExecute

Synopsis:

```
PEGINT PegPresentationExecute(void *pThing, PEGUSHORT usMode);
```

Arguments:

pThing
    Pointer to the PegPresentation object.
usMode
    Mode at which to execute.

Returns:

Various, depending on how the object was dismissed.

Description:

This function is an override of the PegPanelExecute function, and is the main loop of any C/PEG application.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegPanelExecute

Notes:

This function is called during the C/PEG initialization process immediately after PegAppInitialize. This function constitutes the main message loop for any C/PEG application.

1.185 PegPresentationInit

Synopsis:

```
void PegPresentationInit(PegPresentation *pPresent);
```

Arguments:

pPresent
    Pointer to the PegPresentation object.

Returns:

None

Description:

This function initializes pPresent with the default type, colors and function pointers for a PegPresentation object.

Errors:

This function may generate and assert if pPresent is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegPanelInit, PegPresentationCreate

Notes:

If the application requires a custom PegPresentation object, it is usually good practice to call this function from the creation function of the custom PegPresentation object to ensure that the custom version is properly initialized.

1.186 PegPresentationNotify

Synopsis:

```
    PEGINT PegPresentationNotify(void *pThing, const PegMessage
      *pMesg);
```

Arguments:

pThing

Pointer to a PegPresentation object.

pMesg

Pointer to a PegMessage structure that contains the message data.

Returns:

Various, depending on the type of message.

Description:

This function is an override of PegPanelNotify.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegPanelNotify

Notes:

This function has some additional logic for moving focus between top level objects as well as handling shutting itself down.

1.187 PegPresentationRemove

Synopsis:

```
    void *PegPresentationRemove(void *pThing, void *pRemove,
      PEGBOOL bRedraw);
```

Arguments:

pThing
: Pointer to the PegPresentation object.

pRemove
: Pointer to a PegThing or derived object to remove.

bRedraw
: Optionally redraw pThing when pRemove has been removed.

Returns:

If pRemove is a child of pThing, a pointer to pRemove is returned, otherwise, NULL is returned.

Description:

This function is an override of PegPanelRemove.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegPanelRemove

Notes:

This function has logic in it to move focus around when a top level object is removed.

1.188 PegPresentationSetDefFuncs

Synopsis:

```
    void PegPresentationSetDefFuncs(PegPresentation *pPresent);
```

Arguments:

pPresent

Pointer to the PegPresentation object.

Returns:

None

Description:

This function sets the function pointers in pPresent to the default functions used by the PegPresentation object.

Errors:

This function may generate an assert if pPresent is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegPresentationInit

Notes:

This function is called by PegPresentationInit to properly set up the function pointers in pPresent.

1.189 PegProgressBarCreate

Synopsis:

```
PegProgressBar *PegProgressBarCreate(PegRect *pRect, PEGINT
  iMin, PEGINT iMax, PEGINT iCur, PEGUSHORT usId, PEGUSHORT
  usStyle);
```

Arguments:

pRect
: Pointer to a PegRect structure that defines the size and position of the object.

iMin
: The integer value that corresponds to 0 percent.

iMax
: The integer value that corresponds to 100 percent.

iCur
: The current integer value.

usId
: ID to assign the object.

usStyle
: Style to assign the object.

Returns:

Upon success, returns a pointer to a newly allocated PegProgressBar object.

Description:

This function creates a PegProgessBar object, initializes it, the assigns it's data members from the passed in parameters.

Errors:

This function may generate an assert if the necessary memory can not be allocated from the system and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegProgressBarCreateDefault, PegProgressBarSet

Notes:

The PegProgressBar object displays it's current value as a percentage, between 0 and 100, of the minimum value and the maximum value. It is very common to assign 0 to as the minimum value and 100 as the maximum value as this directly corresponds to displayed percentage. This percentage is always drawn as a bar of some type (solid or dashed), and, optionally with text that gives the exact percentage.

There are several style specific to the PegProgressBar object that controls how the object displays the output value. The table below lists the style flags along with their meanings.

Style Flag	Meaning
PS_SHOW_VAL	Controls whether or not the current percentage value is displayed as text.
PS_LED	Controls how the bar output is displayed. With this turned on, the bar will be dashed. With this turned off, the bar will be solid.
PS_VERTICAL	Controls the orientation of the progress. If this is turned on, the minimum is mapped to the bottom of the object, and the maximum is mapped to the top. With this turned off, the minimum is mapped to the left side of the object, and the maximum is mapped to the right side of the object.
PS_PERCENT	Controls whether or not the text displaying the current value adds a percent (%) symbol after the text. This flag has no effect if PS_SHOW_VAL is not set.

The colors for the object are also very important. The following table describes which color index is mapped to which display characteristic.

Color Index	Display
PCI_NORMAL	Normal background color.
PCI_SELECTED	Color of the progress bar.
PCI_TEXT	Color of the text that is not over the progress bar.
PCI_STEXT	Color of the text that is over the progress bar.

Having the two different text colors allows the application to direct the object to draw the output text, if PS_SHOW_VAL is turned on, in two different colors depending on whether or not it is being drawn over the progress bar rectangle. If the edge of the progress bar rectangle is under the area of the screen occupied by the text, the portion of the text that is over the progress bar is drawn in the selected color, and the portion that is not is drawn in the normal color.

If the normal text color and the selected text color are the same, then, obviously, the text is only drawn in one color.

This function calls PegProgressBarCreateDefault to allocate and initialize the object, then calls PegProgessBarSet with the data in the parameter list to properly set the data members of the object.

1.190 PegProgressBarCreateDefault

Synopsis:

```
    PegProgressBar *PegProgressBarCreateDefault(void);
```

Arguments:

None

Returns:

Upon success, returns a pointer to a newly allocated PegProgressBar object.

Description:

This function allocates storage for a PegProgressBar object, and initializes it with the default settings for a PegProgressBar.

Errors:

This function may generate an assert if the necessary memory can not be allocated from the system and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegProgressBarCreate, PegProgressBarInit

Notes:

This function is called by PegProgressBar create to allocated and initialize the object. This function calls PegProgressBarInit to properly initialize the object.

1.191 PegProgressBarDraw

Synopsis:

```
    void PegProgressBarDraw(void *pThing);
```

Arguments:

pThing

Pointer to a PegProgressBar object.

Returns:

None

Description:

This function is an override of PegTextThing draw. This function draws the object's background, frame, progress bar and text.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegTextThingDraw

Notes:

See the Notes section in PegProgressBarCreate for a discussion of controlling how the PegProgressBar object draws itself.

1.192 PegProgressBarInit

Synopsis:

```
    void PegProgressBarInit(PegProgressBar *pProgBar);
```

Arguments:

pProgBar
    Pointer to a PegProgressBar object.

Returns:

None

Description:

This function initializes pProgBar with the defaults for the PegProgressBar type.

Errors:

This function may generate an assert if pProgBar is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegProgressBarCreateDefault, PegProgressBarSetDefFuncs

Notes:

This function is called by PegProgressBarCreateDefault to properly initialize the object. This function calls PegProgressBarSetDefFuncs to set the function pointers in pProgBar.

1.193 PegProgressBarParentShift

Synopsis:

```
    void PegProgressBarParentShift(void *pThing, PEGSHORT xshift,
      PEGSHORT yshift);
```

Arguments:

pThing
    Pointer to a PegProgressBar object.
xshift
    Amount to shift the object along the x axis.
yshift
    Amount to shift the object along the y axis.

Returns:

None

Description:

This function is an override of the PegTextThingParentShift functions.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegProgressBarResize, PegTextThingParentShift, PegThingParentShift

Notes:

The PegProgressBar keeps track of the size and position of the progress bar that is drawn and the object must update this position when it is moved around.

1.194 PegProgressBarProgressSet

Synopsis:

```
void PegProgressBarProgressSet(PegProgressBar *pProgBar, PEGINT
  iCur, PEGBOOL bRedraw);
```

Arguments:

pProgBar	Pointer to a PegProgressBar object.
iCur	The current value to set in pProgBar.
bRedraw	Optionally redraw the object after the current value has been updated.

Returns:

None

Description:

This function provides the application with a reliable way to update the current value of a PegProgressBar or derived object.

Errors:

This function may generate an assert if pProgBar is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegProgressBarCreate

Notes:

This is the proper way to update the current value in a PegProgressBar object. The application should not just set the current value member in the PegProgressBar structure and expect the bar to update itself.

1.195 PegProgressBarResize

Synopsis:

```
void PegProgressBarResize(void *pThing, PegRect *pRect);
```

Arguments:

pThing

    Pointer to a PegProgressBar object.

pRect

    Pointer to a PegRect structure that defines the size and position of pThing.

Returns:

None

Description:

This function is an override of the PegTextThingResize function.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegProgressBarParentShift

Notes:

The PegProgressBar object must calculate the area covered by the progress bar when it is resized.

1.196 PegProgressBarSet

Synopsis:

```
void PegProgressBarSet(PegProgressBar *pProgBar, PegRect
  *pRect, PEGINT iMin, PEGINT iMax, PEGINT iCur, PEGUSHORT
  usId, PEGUSHORT usStyle);
```

Arguments:

pProgBar
: Pointer to a PegProgressBar object.

pRect
: Pointer to a PegRect structure that defines the size and position of the object.

iMin
: The integer value that corresponds to 0 percent.

iMax
: The integer value that corresponds to 100 percent.

iCur
: The current integer value.

usId
: ID to assign the object.

usStyle
: Style to assign the object.

Returns:

None

Description:

This function assigns the data members in pProgBar from the passed in parameters.

Errors:

This function may generate an assert if pProgBar is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegProgressBarCreate

Notes:

This function is called by PegProgressBarCreate to set the data members in the object.

1.197 PegProgressBarSetDefColors

Synopsis:

```
void PegProgressBarSetDefColors(PegProgressBar *pProgBar);
```

Arguments:

pProgBar

Pointer to a PegProgressBar object.

Returns:

None

Description:

This function sets up the default PegProgressBar colors in pProgBar.

Errors:

This function may generate an assert if pProgBar is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegProgressBarInit

Notes:

This function is called by PegProgressBarInit to set the default colors in the object.

1.198 PegProgressBarSetDefFuncs

Synopsis:

```
void PegProgressBarSetDefFuncs(PegProgressBar *pProgBar);
```

Arguments:

pProgBar
    Pointer to a PegProgressBar object.

Returns:

None

Description:

This function sets the function pointers in pProgBar to point to the default PegProgressBar functions.

Errors:

This function may generate an assert if pProgBar is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegProgressBarInit

Notes:

This function is called by PegProgressBarInit to properly set the function pointers in the object.

1.199 PegPromptCreate

Synopsis:

```
PegPrompt *PegPromptCreate(PegRect *pRect, const PEGSTRING
  String, PEGUSHORT usId, PEGUSHORT usStyle);
```

Arguments:

pRect
: Pointer to a PegRect structure that defines the size and position of the object.

String
: a NULL terminated string.

usId
: ID assigned to the object.

usStyle
: Style to assign to the object.

Returns:

Upon success, returns a pointer to a newly allocated PegPrompt object.

Description:

This function allocates storage for a PegPrompt object, initializes it and assigns it data members from the passed in parameter list.

Errors:

This function may generate an assert if the necessary memory for the object can not be allocated from the system and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegPromptCreateDefault, PegPromptSet

Notes:

This function calls PegPromptCreateDefault to allocate and initialize the object, then calls PegPromptSet to set the data members of the object.

1.200 PegPromptCreateDefault

Synopsis:

```
    PegPrompt *PegPromptCreateDefault(void);
```

Arguments:

None

Returns:

Upon success, returns a pointer to a newly allocated PegPrompt object.

Description:

This function allocates storage for a PegPrompt object and initializes the object with the default values for a PegPrompt object type.

Errors:

This function may generate an assert if the necessary memory for the object can not be allocated from the system and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegPromptCreate, PegPromptInit

Notes:

This function calls PegPromptInit to initialize the object. This function is called by PegPromptCreate to allocate storage for a new PegPrompt object.

1.201 PegPromptDraw

Synopsis:

```
    void PegPromptDraw(void *pThing);
```

Arguments:

pThing
Pointer to a PegPrompt object.

Returns:

None

Description:

This function is an override of the PegTextThingDraw function.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegTextThingDraw

Notes:

If the prompt object has a style of AF_ENABLED, it will draw itself using it's selected colors when it has input focus. For example, when this type of object is in a list, it draws itself in selected colors when it is selected by the user.

1.202 PegPromptInit

Synopsis:

```
    void PegPromptInit(PegPrompt *pPrompt);
```

Arguments:

pPrompt
    Pointer to a PegPrompt object.

Returns:

None

Description:

This function initializes the pPrompt object with the default PegPrompt type settings.

Errors:

This function may generate an assert if pPrompt is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegPromptCreateDefault, PegPromptSetDefFuncs

Notes:

This function is called by PegPromptCreateDefault to properly set up the object. This function also calls PegPromptSetDefFuncs to set the function pointers in the object.

1.203 PegPromptNotify

Synopsis:

```
    PEGINT PegPromptNotify(void *pThing, const PegMessage *pMesg);
```

Arguments:

pThing

Pointer to a PegPrompt object.

pMesg

Pointer to a PegMessage structure that contains the message data.

Returns:

Various, depending on the type of message.

Description:

This function is an override of the PegTextThingNotify function.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegTextThingNotify

Notes:

1.204 PegPromptSet

Synopsis:

```
    void PegPromptSet(PegPrompt *pPrompt, PegRect *pRect, const
      PEGSTRING String, PEGUSHORT usId, PEGUSHORT usStyle);
```

Arguments:

pPrompt
: Pointer to a PegPrompt object.

pRect
: Pointer to a PegRect structure that defines the size and position of the object.

String
: a NULL terminated string.

usId
: ID assigned to the object.

usStyle
: Style to assign to the object.

Returns:

None

Description:

This function assigns pPrompt's data members from the passed in parameter list.

Errors:

This function may generate an assert pPrompt is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegPromptCreate

Notes:

This function is called by PegPromptCreate to set the object's data members.

1.205 PegPromptSetDefFuncs

Synopsis:

```
    void PegPromptSetDefFuncs(PegPrompt *pPrompt);
```

Arguments:

pPrompt
	Pointer to a PegPrompt object.

Returns:

None

Description:

This function sets the function pointers in pPrompt to the default functions for the PegPrompt type.

Errors:

This function may generate an assert if pPrompt is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegPromptInit

Notes:

This function is called by PegPromptInit to set the function pointers in the object.

1.206 PegRadioButtonCreate

Synopsis:

```
    PegRadioButton *PegRadioButtonCreate(PegRect *pRect, const
      PEGSTRING String, PEGUSHORT usId, PEGUSHORT usStyle);
```

Arguments:

pRect
: Pointer to a PegRect structure that defines the size and position of the object.

String
: Pointer to a NULL terminated string that is used as the text on the button.

usId
: ID to assign to the object.

usStyle
: Style to assign to the object.

Returns:

Upon success, returns a pointer to a newly allocated PegRadioButton object.

Description:

This function allocates storage for a PegRadioButton object, initializes it, and set's the object's data members from the passed in parameter list.

Errors:

This function may generate an assert if the necessary memory for the object can not be allocated from the system and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegRadioButtonCreateDefault, PegRadioButtonSet

Notes:

This function calls PegRadioButtonCreateDefault to allocate the storage for the object, then calls PegRadioButtonSet to set the object's data members.

1.207 PegRadioButtonCreateDefault

Synopsis:

```
PegRadioButton *PegRadioButtonCreateDefault(void);
```

Arguments:

None

Returns:

Upon success, returns a pointer to a newly allocated PegRadioButton object.

Description:

This function allocates storage for a PegRadioButton object, then initializes the object with the default values associated with a PegRadioButton type.

Errors:

This function may generate an assert if the necessary memory for the object can not be allocated from the system and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegRadioButtonCreate, PegRadioButtonInit

Notes:

This function is called by PegRadioButtonCreate to allocate object storage. This function calls PegRadioButtonInit to properly initialize the object.

1.208 PegRadioButtonInit

Synopsis:

```
    void PegRadioButtonInit(PegRadioButton *pRadButton);
```

Arguments:

pRadButton
    Pointer to a PegRadioButton object.

Returns:

None

Description:

This function initializes pRadButton with the defaults for a PegRadioButton object.

Errors:

This function may generate an assert if pRadButton is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegRadioButtonCreateDefault, PegRadioButtonSeDefFuncs

Notes:

This function is called by PegRadioButtonCreateDefault to initialize the object. This function calls PegRadioButtonSetDefFuncs to set the function pointers in the object.

1.209 PegRadioButtonIsDotted

Synopsis:

```
    PEGBOOL PegRadioButtonIsDotted(PegRadioButton *pRadButton);
```

Arguments:

pRadButton
    Pointer to a PegRadioButton object.

Returns:

If the pRadButton is dotted, TRUE. Otherwise, FALSE.

Description:

This function queries the status of pRadButton regarding it's dotted state.

Errors:

This function may generate an assert if pRadButton is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegRadioButtonSetDotted

Notes:

1.210 PegRadioButtonNotify

Synopsis:

```
    PEGINT PegRadioButtonNotify(void *pThing, const PegMessage
      *pMesg);
```

Arguments:

pThing
: Pointer to a PegRadioButton object.

pMesg
: Pointer to a PegMessage structure that contains the message data.

Returns:

Various, depending on the type of message.

Description:

This function is an override of the PegButtonNotify function.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegButtonNotify

Notes:

1.211 PegRadioButtonSet

Synopsis:

```
void PegRadioButtonSet(PegRadioButton *pRadButton, PegRect
  *pRect, const PEGSTRING String, PEGUSHORT usId, PEGUSHORT
  usStyle);
```

Arguments:

pRadButton	Pointer to a PegRadioButton object.
pRect	Pointer to a PegRect structure that defines the size and position of the object.
String	a NULL terminated string that is used as the text on the button.
usId	ID to assign to the object.
usStyle	Style to assign to the object.

Returns:

None

Description:

This function assigns pRadButton with the passed in parameters.

Errors:

This function may generate an assert if pRadButton is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegRadioButtonCreate

Notes:

This function is called by the PegRadioButtonCreate function to set the data members of the object.

1.212 PegRadioButtonSetDefFuncs

Synopsis:

```
void PegRadioButtonSetDefFuncs(PegRadioButton *pRadButton);
```

Arguments:

pRadButton

Pointer to a PegRadioButton object.

Returns:

None

Description:

This function sets the function pointers in pRadButton to the default PegRadioButton functions.

Errors:

This function may generate an assert if pRadButton is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegRadioButtonInit

Notes:

This function is called by PegRadioButtonInit to properly set the function pointers in the object.

1.213 PegRadioButtonSetDotted

Synopsis:

```
    void PegRadioButtonSetDotted(PegRadioButton *pRadButton,
      PEGBOOL bDotted);
```

Arguments:

pRadButton
: Pointer to a PegRadioButton object.

bDotted
: If TRUE, sets the dotted status of pRadButton to on. If FALSE, sets the dotted status of pRadButton to off.

Returns:

None

Description:

This function sets the dotted status of pRadButton.

Errors:

This function may generate an assert if pRadButton is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegRadioButtonIsDotted

Notes:

1.214 PegScrollButtonCreateDefault

Synopsis:

```
    PegScrollButton *PegScrollButtonCreateDefault(void);
```

Arguments:

None

Returns:

Upon success, returns a pointer to a newly allocated PegScrollButton object.

Description:

This function allocates memory for a PegScrollButton object. The new object's type is set to PEG_TYPE_SCROLL_BUTTON, it's function pointers are assigned and it's default colors are assigned.

Errors:

This function may generate an assert if the memory for the new object can not be allocated from the system and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegHScrollCreateDefault, PegVScrollCreateDefault

Notes:

This object is of interest to the C/PEG scroll bar objects. It is a modified PegThing object that is used as the scroll bar button on scroll bars. The application designer is free to replace this object on the scroll bars with a custom object to achieve a different look for the application.

1.215 PegScrollButtonDraw

Synopsis:

```
    void PegScrollButtonDraw(void *pThing);
```

Arguments:

pThing

Pointer to a PegScrollButton object that is used for the drawing context.

Returns:

None

Description:

This function is an override of the PegThing draw function that allows the PegScrollButton object to draw itself within the context of it's scroll bar parent.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegThingDraw

Notes:

If the application designer wishes to draw the scroll bar buttons in a different manner, then this function is the place to start.

1.216 PegScrollButtonInit

Synopsis:

```
    void PegScrollButtonInit(PegScrollButton *pScrollButton);
```

Arguments:

pScrollButton
    Pointer to a PegScrollButton object.

Returns:

None

Description:

This function initializes the PegScrollButton passed in pScrollButton. The object's type is set to PEG_TYPE_SCROLL_BUTTON, it's function pointers and default colors are also assigned.

Errors:

This function may generate an assert if pScrollButton is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegScrollBarCreateDefault

Notes:

This function is called by PegScrollBarCreateDefault to properly initialize the object.

1.217 PegScrollButtonNotify

Synopsis:

```
    PEGINT PegScrollButtonNotify(void *pThing, const PegMessage
      *pMesg);
```

Arguments:

pThing
Pointer to a PegScrollButton object.
pMesg
Pointer to a PegMessage structure containing the message.

Returns:

Returns various values depending on the passed in message.

Description:

This function is an override of the PegThing notify function. This is necessary so the PegScrollButton object can catch messages from it's scroll bar parent and move itself properly.

Errors:

This function may generate an assert if pThing or pMesg is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegThingNotify

Notes:

This override catches mouse and keyboard messages from the user and updates it position accordingly. It then notifies it's parent scroll bar object that it has changed positions. This, in turn, will usually cause the scroll bar to notify it's parent that the scrolling in the parent needs to be updated.

1.218 PegScrollButtonSetDefFuncs

Synopsis:

```
    void PegScrollButtonSetDefFuncs(PegScrollButton
      *pScrollButton);
```

Arguments:

pScrollButton
    Pointer to a PegScrollButton object.

Returns:

None

Description:

This function assigns the default functions for a PegScrollButton object to pScrollButton.

Errors:

This function may generate an assert if pScrollButton is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegScrollButtonCreateDefault

Notes:

This function is called by PegScrollButtonCreateDefault to properly set up the object's function pointers.

1.219 PegSliderCalcPointFromVal

Synopsis:

```
    void PegSliderCalcPointFromVal(PegSlider *pSlider, PEGINT iVal,
      PegPoint *pPoint);
```

Arguments:

pSlider
: Pointer to a PegSlider object.

iVal
: Integer value to evaluate.

pPoint
: Pointer to a PegPoint structure.

Returns:

Returns the absolute position of the thumb button in pPoint. If iVal is out of bounds, then -1 is returned in both members of pPoint.

Description:

This function calculates the point along the travel of the thumb button for pSlider that maps to iVal.

Errors:

If iVal falls outside of the minimum and maximum allowable values for pSlider, -1 is returned in pPoint.

Related Functions:

PegSliderCalcValFromPoint

Notes:

This function is used internally by the PegSlider object to calculate where to draw the thumb button based on the object's current value. Normally, a function such as this would not be exposed to the application level code. But, the drawing operation for the thumb button may be overridden by the application designer, who would, in turn, also need access to this function to determine the point along the line of travel of the object to place the thumb button.
If the object is oriented vertically, the y member of pPoint holds the relevant data. Likewise, if the object is oriented horizontally, the x member of pPoint holds the relevant data.

1.220 PegSliderCalcValFromPoint

Synopsis:

```
    PEGINT PegSliderCalcValFromPoint(PegSlider *pSlider, PegPoint
      *pPoint, PEGBOOL bConstrain);
```

Arguments:

pSlider
: Pointer to a PegSlider object.

pPoint
: Pointer to a PegPoint structure.

bConstrain
: Optionally constrain the return value to a value that between the minimum and maximum allowable values of pSlider.

Returns:

Returns the value mapped from the coordinates presented in pPoint.

Description:

This function calculates a value that is a function of the coordinates in pPoint relative to the travel coordinates of pSlider and returns the value to the caller.

If bConstrain is true, the return value will always be a valid value for the pSlider object. In other words, the return value will always be between the minimum and maximum allowable values as dictated by pSlider, regardless of the coordinates presented in pPoint.

The only exception to this is if the travel distance for the slider is zero and the maximum value minus the minimum value sums to zero. In this case, the return value will be zero.

Errors:

If the travel distance is zero and the sum of the maximum value minus the minimum value is zero, the function returns zero.

Related Functions:

PegSliderCalcPointFromVal

Notes:

See the Notes section of PegSliderCalcPointFromVal for a further discussion of these two functions.

1.221 PegSliderCreate

Synopsis:

```
PegSlider *PegSliderCreate(PegRect *pRect, PEGINT iMin, PEGINT
  iMax, PEGINT iScale, PEGINT iCur, PEGUSHORT usId, PEGUSHORT
  usStyle);
```

Arguments:

pRect
: Pointer to a PegRect structure that defines the size and position of the object.

iMin
: Minimum valid value for the object.

iMax
: Maximum valid value for the object.

iScale
: Value to determine how often to draw a tic mark on the face of the object.

iCur
: Current value to set in the object.

usId
: ID to assign to the object.

usStyle
: Style to assign to the object.

Returns:

Upon success, returns a pointer to a newly allocated PegSlider object.

Description:

This function allocates storage for a PegSlider object, initializes the object, then sets the data members of the object with the values in the parameter list.

Errors:

This function may generate an assert if the necessary memory for the object can not be allocated by the system and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegSliderCreateDefault, PegSliderSet

Notes:

This function calls PegSliderCreateDefault to allocate the storage for the object, then calls PegSliderSet to set the object's data members.

1.222 PegSliderCreateDefault

Synopsis:

```
PegSlider *PegSliderCreateDefault(void);
```

Arguments:

None

Returns:

Upon success, returns a pointer to a newly allocated PegSlider object.

Description:

This function allocates storage for a PegSlider object and initializes the object using the defaults for a PegSlider type.

Errors:

This function may generate an assert if the necessary memory can not be allocated from the system and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegSliderCreate, PegSliderInit

Notes:

This function is called by PegSliderCreate to allocate storage for a new PegSlider object. This function also calls PegSliderInit to properly initialize the object.

1.223 PegSliderDraw

Synopsis:

```
void PegSliderDraw(void *pThing);
```

Arguments:

pThing

Pointer to a PegSlider object.

Returns:

None

Description:

This function is an override of the PegThingDraw function.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegThingDraw, PegSliderDrawThumbButton, PegSliderDrawScale

Notes:

This function draws the border and background of the object. The thumb button and scale are drawn by the PegSliderDrawThumbButton and PegSliderDrawScale functions, respectively.

Please see the Notes section for each of these functions for a discussion on how a PegSlider object is drawn.

1.224 PegSliderDrawScale

Synopsis:

```
    void PegSliderDrawScale(void *pThing);
```

Arguments:

pThing

Pointer to a PegSlider object.

Returns:

None

Description:

This is the default implementation for a PegSlider drawing it's scale. If pThing does not have a scale set, then this function does not draw any scale tic marks.

This function also draws the travel line for the thumb button.

The color used to draw the tic marks and the normal color for the travel line is controlled by the PCI_NTEXT color index.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegSliderDraw, PegSliderDrawThumbButton, PegFuncPtrSet

Notes:

This function introduces a new function pointer to the PegSlider object, and as such, this function may be overridden in custom objects that are derived from PegSlider.

To set the function pointer in a PegSlider object to a custom scale drawing function, use the PegFuncPtrSet function and pass PFP_DRAWSCALE as the function ID.

1.225 PegSliderDrawThumbButton

Synopsis:

```
    void PegSliderDrawThumbButton(void *pThing);
```

Arguments:

pThing
    Pointer to a PegSlider object.

Returns:

None

Description:

This function is the default PegSlider implementation for drawing the thumb button on the slider object.

The default implementation draws a beveled rectangle that is centered over the point where the current value for the slider is mapped.

For normal drawing, the color index at PEG_STEXT is used to fill the rectangle.

If the PEG_DRAW_FOCUS directive is defined, then the rectangle is filled with the color at color index PCI_SELECTED when the object is current.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegSliderDraw, PegSliderDrawScale, PegFuncPtrSet

Notes:

This function introduces a new function pointer to the PegSlider object, and as such, this function may be easily overridden by the application designer.

To set the function pointer in a PegSlider object to point to a custom thumb button drawing function, call the PegFuncPtrSet function with the function ID of PFP_DRAWTHUMBBUTTON.

1.226 PegSliderInit

Synopsis:

```
    void PegSliderInit(PegSlider *pSlider);
```

Arguments:

pSlider
    Pointer to a PegSlider object.

Returns:

None

Description:

This function initializes pSlider with the defaults relating to a PegSlider object type.

Errors:

This function may generate an assert if pSlider is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegSliderCreateDefault, PegSliderSetDefFuncs

Notes:

This function is called by the PegSliderCreateDefault function to initialize the object. This function also calls PegSliderSetDefFuncs to properly set the function pointers in the object.

1.227 PegSliderNotify

Synopsis:

```
PEGINT PegSliderNotify(void *pThing, const PegMessage *pMesg);
```

Arguments:

pThing
: Pointer to a PegSlider object.

pMesg
: Pointer to a PegMessage structure that contains the message data.

Returns:

Various, depending on the type of message received.

Description:

This function is an override of the PegThingNotify function. The PegSlider object responds to mouse and keyboard user input to move the thumb button along the line of travel dictated by the minimum and maximum valid values of the object.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegThingNotify

Notes:

1.228 PegSliderParentShift

Synopsis:

```
void PegSliderParentShift(void *pThing, PEGSHORT xShift,
  PEGSHORT yShift);
```

Arguments:

pThing
: Pointer to a PegSlider object.

xShift
: Amount to shift along the x axis.

yShift
: Amount to shift along the y axis.

Returns:

None

Description:

This is an override of the PegThingParentShift function.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegThingParentShift

Notes:

The PegSlider object must map the current value to a screen location. When the object is moved, it must update this location.

1.229 PegSliderResize

Synopsis:

```
void PegSliderResize(void *pThing, PegRect *pRect);
```

Arguments:

pThing
: Pointer to a PegSlider object.

pRect
: Pointer to a PegRect structure that defines the new size and position of the object.

Returns:

None

Description:

This function is an override of the PegThingResize function.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegThingResize

Notes:

The PegSlider object tracks the current value mapped to a screen coordinate. When the object is moved or resized, it must recalculate this value mapping.

1.230 PegSliderSet

Synopsis

```
    PegSlider *PegSliderSet(PegSlider *pSlider, PegRect *pRect,
      PEGINT iMin, PEGINT iMax, PEGINT iScale, PEGINT iCur,
      PEGUSHORT usId, PEGUSHORT usStyle);
```

Arguments:

pSlider
: Pointer to a PegSlider object.

pRect
: Pointer to a PegRect structure that defines the size and position of the object.

iMin
: Minimum valid value for the object.

iMax
: Maximum valid value for the object.

iScale
: Value to determine how often to draw a tic mark on the face of the object.

iCur
: Current value to set in the object.

usId
: ID to assign to the object.

usStyle
: Style to assign to the object.

Returns:

None

Description:

This function sets the data members of pSlider with the values in the parameter list.

Errors:

This function may generate an assert if pSlider is invalid and the C/PEG library was compiled with the PEG_USE_ASSERT directive defined.

Related Functions:

PegSliderCreate

Notes:

This function is called by PegSliderCreate to properly assign the data members the passed in values.

1.231 PegSliderSetDefColors

Synopsis:

```
void PegSliderSetDefColors(PegSlider *pSlider);
```

Arguments:

pSlider

Pointer to a PegSlider object.

Returns:

None

Description:

This function sets the default colors for a PegSlider object.

Errors:

This function may generate an assert if pSlider is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegSliderInit

Notes:

The following table illustrates which color index is related to which drawing operation.

Color Index	Used to Draw
PCI_NORMAL	The background fill color
PCI_NTEXT	The tic marks and travel line
PCI_SELECTED	The thumb button if the object is currently selected and the PEG_DRAW_FOCUS directive is defined.
PCI_STEXT	The thumb button when the object is not current, and the thumb button when the object is current and PEG_DRAW_FOCUS is not defined.

1.232 PegSliderSetDefFuncs

Synopsis:

```
    void PegSliderSetDefFuncs(PegSlider *pSlider);
```

Arguments:

pSlider
    Pointer to a PegSlider object.

Returns:

None

Description:

This function sets the function pointers in pSlider to point to the default PegSlider functions.

Errors:

This function may generate an assert if pSlider is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegSliderInit

Notes:

This function is called by PegSliderInit to properly set the function pointers in the object.

1.233 PegSliderValueSet

Synopsis:

```
    void PegSliderValueSet(PegSlider *pSlider, PEGINT iVal, PEGBOOL
      bNotifyParent, PEGBOOL bRedraw);
```

Arguments:

pSlider
: Pointer to a PegSlider object.

iVal
: The value to set in pSlider.

bNotifyParent
: Optionally notify pSlider's parent that the object's value has changed.

bRedraw
: Optionally redraw pSlider after the value has been updated.

Returns:

None

Description:

This function allows application code to manually set the current value of pSlider. If bNotifyParent is true, the object will send a signal to it's parent just as if the value had been changed by user input.

Errors:

This function may generate an assert if pSlider is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

None

Notes:

If iVal is outside the boundaries of the minimum and maximum values allowed by pSlider, this object immediately returns without updating.

If pSlider does not have a parent, in which case it would not be visible, the value is set, and the bNotifyParent argument is ignored.
If the pSlider object does not have an ID, no signal will be sent, regardless of the status of bNotifyParent.

1.234 PegSpinButtonCreate

Synopsis:

```
PegSpinButton *PegSpinButtonCreate(PegRect *pRect, PegTextThing
  *pDisplay, PEGLONG lMin, PEGLONG lMax, PEGINT iStep, PEGINT
  iDisplayWidth, PEGUSHORT usId, PEGUSHORT usStyle);
```

Arguments:

pRect
: Pointer to a PegRect structure that defines the size and position of the object.

pDisplay
: Pointer to a PegTextThing or derived object that acts as the output of the spin button.

lMin
: Minimum valid value of the object.

lMax
: Maximum valid value of the object.

iStep
: The value to step up or down.

iDisplayWidth
: The number of characters to display in pDisplay.

usId
: ID to assign the object.

usStyle
: Style to assign to the object.

Returns:

Upon success, returns a pointer to a newly created PegSpinButton object.

Description:

This function allocates storage for a PegSpinButton object, initializes the object, then sets the object's data members from the passed in parameter list.

Errors:

This function may generate an assert if the necessary memory for the object can not be allocated by the system and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Notes:

The object to which pDisplay points is used to display the current value of the PegSpinButton object. pDisplay must be a PegTextThing derived object. PegTextThing derived objects include PegPrompt, PegEditField, PegTextButton, PegPanel and others. When the value of the PegSpinButton is modified, the object will put's it's current value into a string and call pDisplay's TextSet function, then call pDisplay's Draw function. Thus, updating the display output. This display output object should have the TT_COPY flag set in it's style member, otherwise, the display may become corrupt during ancillary draw operations.

The iStep parameter refers to a value describing how much the PegSpinButton object will increment it's current value either up or down when the user selects on of the arrow buttons.

To zero fill the display output to a specific size, use iDisplayWidth. If iDisplayWidth is 0, then the output display will simple be the current value of the object. On the other hand, if iDisplayWidth is greater than 0, the output display is zero padded on the left the difference between the width of the current value and the requested width in iDisplayWidth.

The SB_VERTICAL style controls the orientation of the object. If this is passed in usStyle, then the object will orient itself vertically and present two bitmapped buttons stacked on top of each other, displaying an up arrow on the top button and a down arrow on the bottom button. If SB_VERTICAL is not set, the object orients itself horizontally, and presents two bitmapped buttons side by side, with a left arrow on the left bitmap button and a right arrow on the right bitmap button.

The PegSpinButton object is a compound object, meaning that within this object, there are instances of other PegThing derived objects. For the PegSpinButton, there are two instances of PegBitmapButton that provide the interface to the user. These buttons are added to the PegSpinButton object during this function, and are oriented as described in the previous paragraph. These objects are not added to the PegSpinButton object in the call to PegSpinButtonCreateDefault.

If the application designer wishes to replace the buttons that the PegSpinButton uses, it is a simple process to do so. Since the PegSpinButtonCreateDefault does not create these buttons, the custom object creation would call PegSpinButtonCreateDefault, then add the custom buttons to the returned object, instead of calling

PegSpinButtonCreate. The custom creation function could then call PegSpinButtonSet to set up the data members of the object, and that would be it.

1.235 PegSpinButtonCreateDefault

Synopsis:

```
PegSpinButton *PegSpinButtonCreateDefault(void);
```

Arguments:

None

Returns:

Upon success, returns a pointer to a newly allocated PegSpinButton object.

Description:

This function allocates storage for a PegSpinButton object, and initializes the object to use the default for a PegSpinButton type.

Errors:

This function may generate an assert if the necessary memory for the object can not be allocated from the system and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegSpinButtonCreate, PegSpinButtonInit

Notes:

This function is called by PegSpinButtonCreate to allocate the object. This function calls PegSpinButtonInit to properly initialize the object.

1.236 PegSpinButtonInit

Synopsis:

```
void PegSpinButtonInit(PegSpinButton *pSpin);
```

Arguments:

pSpin
  Pointer to a PegSpinButton object.

Returns:

None

Description:

This function initializes pSpin with the defaults associated with the PegSpinButton object type.

Errors:

This function may generate an assert if pSpin is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegSpinButtonCreateDefault, PegSpinButtonSetDefFuncs

Notes:

This function is called by PegSpinButtonCreateDefault to initialize the object. This function calls PegSpinButtonSetDefFuncs to properly set the function pointers in the object.

1.237 PegSpinButtonNotify

Synopsis:

```
PEGINT PegSpinButtonNotify(void *pThing, const PegMessage
  *pMesg);
```

Arguments:

pThing
: Pointer to a PegSpinButton object.

pMesg
: Pointer to a PegMessage structure that contains the data of the message.

Returns:

Various, depending on the message type.

Description:

This function is an override of the PegThingNotify function.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegThingNotify

Notes:

The PegSpinButtonNotify function catches the signals sent from it's child buttons to spin more or less.

1.238 PegSpinButtonSet

Synopsis:

```
void PegSpinButtonSet(PegSpinButton *pSpin, PegRect *pRect,
  void *pDisplay, PEGLONG lMin, PEGLONG lMax, PEGINT iStep,
  PEGINT iDisplayWidth, PEGUSHORT usId, PEGUSHORT usStyle);
```

Arguments:

pSpin
: Pointer to a PegSpinButton object.

pRect
: Pointer to a PegRect structure that defines the size and position of the object.

pDisplay
: Pointer to a void or derived object that acts as the output of the spin button.

lMin
: Minimum valid value of the object.

lMax
: Maximum valid value of the object.

iStep
: The value to step up or down.

iDisplayWidth
: The number of characters to display in pDisplay.

usId
: ID to assign the object.

usStyle
: Style to assign to the object.

Returns:

None

Description:

This function sets the data members in pSpin to the passed in parameter list.

Errors:

This function may generate an assert if pSpin is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegSpinButtonCreate

Notes:

See the Notes section PegSpinButtonCreate for an explanation of the parameters.

1.239 PegSpinButtonSetDefFuncs

Synopsis:

```
void PegSpinButtonSetDefFuncs(PegSpinButton *pSpin);
```

Arguments:

pSpin
    Pointer to a PegSpinButton object.

Returns:

None

Description:

This function sets the function pointers in pSpin to the default functions associated with the PegSpinButton object type.

Errors:

This function may generate an assert if pSpin is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegSpinButtonInit

Notes:

This function is called by PegSpinButtonInit to set the default functions in the object.

1.240 PegTextButtonCreate

Synopsis:

```
    PegTextButton *PegTextButtonCreate(PegRect *pRect, const
      PEGSTRING String, PEGUSHORT usId, PEGUSHORT usStyle);
```

Arguments:

pRect
: Pointer to a PegRect structure that defines the size and position of the object.

String
: a NULL terminated string.

usId
: ID to assign the object.

usStyle
: Style to assign to the object.

Returns:

Upon success, returns a pointer to a newly allocated PegTextButton object.

Description:

This function allocates storage for a PegTextButton object, initializes the object, then sets the object's data member with the values passed in the parameter list.

Errors:

This function may generate an assert if the necessary memory for the object can not be allocated from the system and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegTextButtonCreateDefault, PegTextButtonSet

Notes:

This function call PegTextButtonCreateDefault to allocate and initialize the object, then calls PegTextButtonSet to set the data members.

1.241 PegTextButtonCreateDefault

Synopsis:

```
    PegTextButton *PegTextButtonCreateDefault(void);
```

Arguments:

None

Returns:

Upon success, returns a pointer to a newly allocated PegTextButton object.

Description:

This function allocates storage for a PegTextButton object and then initializes the object.

Errors:

This function may generate an assert if the necessary memory for the object can not be allocated from the system and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegTextButtonCreate, PegTextButtonInit

Notes:

This function is called by PegTextButtonCreate to allocate and initialize the object.

1.242 PegTextButtonDraw

Synopsis:

```
    void PegTextButtonDraw(void *pThing);
```

Arguments:

pThing

Pointer to a PegTextButton object.

Returns:

None

Description:

This function is an override of the PegButtonDraw function.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegButtonDraw

Notes:

This function make use of PegButtonDraw to draw the frame and background of the object, then draws it's own text.

1.243 PegTextButtonInit

Synopsis:

```
    void PegTextButtonInit(PegTextButton *pButton);
```

Arguments:

pButton
    Pointer to a PegTextButton object.

Returns:

None

Description:

This function initializes pButton to the default associated with the PegTextButton object type.

Errors:

This function may generate an assert if pButton is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegTextButtonCreateDefault, PegTextButtonSetDefFuncs

Notes:

This function is called by the PegTextButtonCreateDefault function to properly initialize the object. This function then calls PegTextButtonSetDefFuncs to set the function pointers in the object.

1.244 PegTextButtonSet

Synopsis:

```
    void PegTextButtonSet(PegTextButton *pButton, PegRect *pRect,
      const PEGSTRING String, PEGUSHORT usId, PEGUSHORT usStyle);
```

Arguments:

pButton

Pointer to a PegTextButton object.

pRect

Pointer to a PegRect structure that defines the size and position of the object.

String

a NULL terminated string.

usId

ID to assign the object.

usStyle

Style to assign to the object.

Returns:

None

Description:

This function sets the object's data member with the values passed in the parameter list.

Errors:

This function may generate an assert if pButton is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegTextButtonCreate

Notes:

This function is called by PegTextButtonCreate to set the data members in the object.

1.245 PegTextButtonSetDefFuncs

Synopsis:

```
    void PegTextButtonSetDefFuncs(PegTextButton *pButton);
```

Arguments:

pButton

Pointer to a PegTextButton object.

Returns:

None

Description:

This function sets the function pointers in pButton to the default functions associated with the PegTextButton object type.

Errors:

This function may generate an assert if pButton is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegTextButtonInit

Notes:

This function is called by the PegTextButtonInit function to set the function pointers in the object.

1.246 PegTextThingCreate

Synopsis:

```
PegTextThing *PegTextThingCreate(PegRect *pRect, const
  PEGSTRING String, PEGUBYTE ubFontIndex, PEGUSHORT usId,
  PEGUSHORT usStyle);
```

Arguments:

pRect
: Pointer to a PegRect structure that defines the size and position of the object.

String
: a NULL terminated string.

ubFontIndex
: Index value to set the font for the object.

usId
: ID to assign to the object.

usStyle
: Style to assign to the object.

Returns:

Upon success, returns a pointer to a newly allocated PegTextThing object.

Description:

This function allocates storage for a new PegTextThing object, initializes the object, then sets it's data members from the values in the parameter list.

Errors:

This function may generate an assert if the necessary memory for the object can not be allocated from the system and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegTextThingCreateDefault, PegTextThingSet, PegTextMaxLenSet

Notes:

See the “C/PEG Programmers Manual” for a discussion on the font index mechanism.

Every object in the C/PEG library that derives from PegTextThing knows it's own font index, and, as such, this mechanism is mostly transparent to the application level code. For instance, when the application creates a PegTextButton, the application does not need to know what font index a PegTextButton uses, the PegTextButton creation function handles this.

Every PegTextThing or derived object has a define that correlates to the maximum number of characters in a string, including the NULL terminator, that the object will accept when the TT_COPY style flag is turned on. This aids in controlling unexpected buffer overruns; but, obviously, does not completely prevent them.

In the Create functions for these objects, the object instance's maximum text length is set in the object. This may be, optionally, overridden by the application developer by calling PegTextMaxLen for the object. The define PEG_TEXT_MAX_IGNORE is used to relieve the object from enforcing a text maximum length at all.

1.247 PegTextThingCreateDefault

Synopsis:

```
PegTextThing *PegTextThingCreateDefault(void);
```

Arguments:

None

Returns:

Upon success, this function returns a pointer to a newly allocated PegTextButton object.

Description:

This function allocates storage for a PegTextThing object and initializes the object.

Errors:

This function may generate an assert if the necessary memory for the object can not be allocated from the system and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegTextThingCreate, PegTextThingInit

Notes:

This function is called by PegTextThingCreate to allocate the object. This function calls PegTextThingInit to properly initialize the object.

1.248 PegTextThingDestroy

Synopsis:

```
void PegTextThingDestroy(void *pThing);
```

Arguments:

pThing

Pointer to a PegTextThing object.

Returns:

None

Description:

This function is an override of the PegThingDestroy function. The PegTextThing object needs to override this function in order to properly free it's internal text string.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegThingDestroy, PegTextThingCopyModeOn

Notes:

If the PegTextThing object was created with the TT_COPY flag enabled, or the PegTextThingCopyModeOn function was called with this object instance as an argument, then pThing will have allocated an internal buffer to hold it's text string. This buffer must be freed when the object is destroyed.

1.249 PegTextThingFontSet

Synopsis:

```
    void PegTextThingFontSet(void *pTextThing, PegFont *pFont);
```

Arguments:

pTextThing
    Pointer to a PegTextThing or derived object.
pFont
    Pointer to a PegFont structure.

Returns:

None

Description:

This function is the default implementation for setting a font for any object that derives from PegTextThing.

The PegTextThing object introduces a new function pointer into the object structure for setting the font used by the object. This function may be overridden by objects that derive from PegTextThing.

To override this function, use the PegFuncPtrSet function and pass PFP_FONT_SET as the function ID.

Errors:

This function may generate an assert if pTextThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegFontSet, PegFontGet, PegFuncPtrSet

Notes:

1.250 PegTextThingInit

Synopsis:

```
    void PegTextThingInit(PegTextThing *pTextThing);
```

Arguments:

pTextThing
    Pointer to a PegTextThing object.

Returns:

None

Description:

This function initializes pTextThing with the defaults associated with a PegTextThing object type.

Errors:

This function may generate an assert if pTextThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegTextThingCreateDefault, PegTextThingSetDefFuncs

Notes:

This function is called by PegTextThingCreateDefault to initialize the object. This function calls PegTextThingSetDefFuncs to set the functions pointers in pTextThing.

1.251 PegTextThingSet

Synopsis:

```
    void PegTextThingSet(PegTextThing *pTextThing, PegRect *pRect,
      const PEGSTRING String, PEGUBYTE ubFontIndex, PEGUSHORT
      usId, PEGUSHORT usStyle);
```

Arguments:

pTextThing
: Pointer to a PegTextThing object.

pRect
: Pointer to a PegRect structure that defines the size and position of the object.

String
: a NULL terminated string.

ubFontIndex
: Index value to set the font for the object.

usId
: ID to assign to the object.

usStyle
: Style to assign to the object.

Returns:

None

Description:

This function sets the data members of pTextThing from the values in the parameter list. This function may also allocated storage for an internal string buffer if TT_COPY is specified as a flag in usStyle.

Errors:

This function may generate an assert if pTextThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegTextThingCreate

Notes:

This function is called by PegTextThingCreate to set the data members of the object.

1.252 PegTextThingSetDefFuncs

Synopsis:

```
    void PegTextThingSetDefFuncs(PegTextThing *pTextThing);
```

Arguments:

pTextThing
    Pointer to a PegTextThing object.

Returns:

None

Description:

This function sets the function pointers in pTextThing to point to the default functions associated with a PegTextThing object type.

Errors:

This function may generate an assert if pTextThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegTextThingInit

Notes:

This function is called by PegTextThingInit to set the object's function pointers.

1.253 PegTextThingTextSet

Synopsis:

```
void PegTextThingTextSet(void *pTextThing, const PEGCHAR
  *pText);
```

Arguments:

pTextThing
    Pointer to a PegTextThing or derived object.
pText
    Pointer to a NULL terminated string to assign to pTextThing.

Returns:

None

Description:

This function is the default implementation for assigning a string to a PegTextThing or derived object.

The PegTextThing object introduces a function pointer to this function in the structure. This function may be overridden by objects that derive from PegTextThing.

Errors:

This function may generate an assert if pTextThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegTextSet, PegTextGet, PegTextThingCopyModeOn

Notes:

If the object pointed to by pTextThing was created with the TT_COPY flag in it's style mask or the PegTextThingCopyModeOn function was called for this instance of pTextThing, then the string pointed to by pText is copied into an internal buffer maintained by pTextThing.

1.254 PegThingAdd

Synopsis:

```
void PegThingAdd(void *pThing, void *pAdd, PEGBOOL bRedraw);
```

Arguments:

pThing
: Pointer to a PegThing or derived object that will be the parent

pAdd
: Pointer to a PegThing or derived object that will be the child to pThing

bRedraw
: If the child object should be told to draw itself after being added

Returns:

None

Description:

This is default way of adding a child object to a parent object. PegThing and most PegThing derived objects eventually call this function when adding children to a parent object.

The object being added to the parent will always be placed at the front of other child objects. To place children at the end of the child list, use PegThingAddToEnd.

Errors:

This function may generate an assert if either object is NULL and the C/PEG library was compiled with the PEG_USE_ASSERT directive defined.

Related Functions:

PegThingAddToEnd, PegThingRemove

PegAdd, PegAddToEnd, PegRemove

All PegThing derived objects in the C/PEG library also have a Add procedure.

Notes:

If an object is never added to a parent object that is itself an ancestor of the PegPresentation, then the object will never be visible.

When adding several children to a parent, it is often more efficient to pass FALSE for bRedraw on each call. After you are finished adding all of the children, invalidate the parent object then call the parents draw function.

1.255 PegThingAddToEnd

Synopsis:

```
    void PegThingAddToEnd(void *pThing, void *pAdd, PEGBOOL
      bRedraw);
```

Arguments:

pThing
: Pointer to a PegThing or derived object that will be the parent

pAdd
: Pointer to a PegThing or derived object that will be the child to pThing

bRedraw
: If the child object should be told to draw itself after being added

Returns:

None

Description:

This function adds a child object to a parent object much the same way as PegThingAdd, with the exception that the child object is placed at the end of the child object list if there are other children of the parent object.

Errors:

This function may generate an assert if either object is NULL and the C/PEG library was compiled with the PEG_USE_ASSERT directive defined.

Related Functions:

PegThingAdd, PegThingRemove,

PegAdd, PegAddToEnd, PegRemove

All PegThing derived objects in the C/PEG library also have a AddToEnd procedure.

Notes:

See the Notes section of PegThingAdd.

1.256 PegThingCreate

Synopsis:

```
PegThing *PegThingCreate(PegRect *pRect, PEGUSHORT usId,
  PEGUSHORT usStyle);
```

Arguments:

pRect
: Pointer to a PegRect structure that defines the size and position of the object.

usId
: Unique numerical identification assigned to the object. These should be unique for all child objects that share a common parent.

usStyle
: Style to assign to the object. For PegThing objects, this is generally one of several frame styles.

Returns:

Upon success, this function returns a pointer to a newly allocated PegThing structure. There is no error code return. If the C/PEG library was compiled with PEG_ASSERT turned on, then this function will generate an assert if the memory for the object can not be allocated.

Description:

This function allocates a default PegThing object by calling PegThingCreateDefault. It then calls PegThingSet with the passed over parameters to initialize the objects internal size and position as well as it's ID and style.

Errors:

This function may generate an assert if the PegThing object can not be allocated on the heap and the C/PEG library was compiled with the PEG_USE_ASSERT directive defined.

Related Functions:

PegThingCreateDefault

Notes:

It is the responsibility of the caller to free the memory associated with the object.

1.257 PegThingCreateDefault

Synopsis:

```
    PegThing *PegThingCreateDefault(void);
```

Arguments:

None

Returns:

Upon success, this function returns a pointer to a newly allocated PegThing structure. There is no error code return. If the C/PEG library was compiled with PEG_ASSERT turned on, then this function will generate an assert if the memory for the object can not be allocated.

Description:

This function allocates a default PegThing object. The memory for the object is first initialized to 0. The function table for the object is then filled in and the type is set to PEG_TYPE_THING.

The object returned from this call may then be sized and placed by the caller.
This function is most often used by PegThingCreate to allocate the memory for a new object. Objects that are based upon PegThing may also wish to call this function to allocate their own objects, provided the derived objects are the same size as a PegThing object.

Errors:

This function may generate an assert if the PegThing object can not be allocated on the heap and the C/PEG library was compiled with the PEG_USE_ASSERT directive defined.

Related Functions:

PegThingCreate

Notes:

It is the responsibility of the caller to free the memory associated with the object.

1.258 PegThingDestroy

Synopsis:

```
void PegThingDestroy(void *pThing);
```

Arguments:

pThing

Pointer to a PegThing object to destroy

Returns:

None

Description:

This is the preferred way to delete a PegThing or derivative object.

If the object is visible, it is first removed from it's parent. The main C/PEG message queue is then cleared of any timers and pending messages that were associated with the object. If the object has any children, then the children are also recursively destroyed. Finally, the memory associated with the object is released.

Errors:

This function may generate an assert if pThing is NULL and the C/PEG library was compiled with the PEG_USE_ASSERT directive defined.

Related Functions:

PegDestroy

PegThingRemove, PegRemove

Notes:

It can be dangerous if the application developer frees the object without calling this function. If there are any messages or timers associated with the object, then these resources will not be purged properly and may very easily cause a run-time memory fault. Also, if the object has any children, the children will be orphaned and could possibly lead to large memory leaks.

Therefore, never free a PegThing or derivative object. When the application is done with an object, call PegDestroy.

1.259 PegThingDraw

Synopsis:

```
    void PegThingDraw(void *pThing);
```

Arguments:

pThing
    Pointer the a PegThing or derived object

Returns:

None

Description:

Since the PegThing object does not actually do any drawing for itself, this function calls PegDrawChildren for the passed in object.

Errors:

This function may generate an assert if the PegThing object is NULL and the C/PEG library was compiled with the PEG_USE_ASSERT directive defined.

Related Functions:

PegDraw
All PegThing derived objects in the C/PEG library also have a Draw procedure.

Notes:

For objects that derive from PegThing, calling this function during their own draw procedure ensures that the child objects will also draw. Usually, this is done after the calling object has drawn it's own frame and background, otherwise the child objects will not show up on top of the caller.

1.260 PegThingDrawBorder

Synopsis:

```
void PegThingDrawBorder(void *pThing, PEGCOLORVAL FillColor);
```

Arguments:

pThing
    Object to use to draw the border
FillColor
    Background fill color

Returns:

None

Description:

This function first fills the region occupied by the object with the background fill color. It then draws a border around the object using one of the following border style flags:

FF_RAISED
FF_RECESSED
FF_THIN
FF_NONE

Most derivatives of PegThing call this function to draw their border.

Errors:

This function may generate an assert if pThing is NULL and the C/PEG library was compiled with the PEG_USE_ASSERT directive defined.

Related Functions:

PegDrawBorder

Notes:

If you are drawing a custom object, it is standard practice to call this function before drawing any foreground elements of your object.

1.261 PegThingDrawFocus

Synopsis:

```
    void PegThingDrawFocus(void *pThing, PEGBOOL bDraw);
```

Arguments:

pThing
: Pointer to the object that will be used to draw

bDraw
: Determines whether the object should completely redraw itself, or only draw the focus indicator

Returns:

None

Description:

The default version of this function does not do any drawing.

Errors:

This function may generate an assert if pThing is NULL and the C/PEG library was compiled with the PEG_USE_ASSERT directive defined.

Related Functions:

PegDrawFocus, PegEraseFocus, PegThingEraseFocus

Notes:

1.262 PegThingEraseFocus

Synopsis:

```
    void PegThingEraseFocus(void *pThing);
```

Arguments:

pThing
    Pointer to a PegThing object

Returns:

None

Description:

The default version of the function does not do any drawing.

Errors:

This function may generate an assert if pThing is NULL and the C/PEG library was compiled with the PEG_USE_ASSERT directive defined.

Related Functions:

PegThingDrawFocus, PegEraseFocus, PegDrawFocus

Notes:

1.263 PegThingInit

Synopsis:

```
void PegThingInit(PegThing *pThing);
```

Arguments:

pThing
: A pointer to a PegThing object

Returns:

None

Description:

This function is called by PegThingCreateDefault to set up the function table for the passed over PegThing object. Objects that are based on PegThing may also wish to call this function to ensure that the objects function table is set up properly. The type of the object, PEG_TYPE_THING, is also assigned to the object.

Errors:

This function may generate an assert if the PegThing object is NULL and the C/PEG library was compiled with the PEG_USE_ASSERT directive defined.

Related Functions:

PegThingCreateDefault

Notes:

1.264 PegThingNotify

Synopsis:

```
PEGINT PegThingNotify(void *pThing, const PegMessage *pMesg);
```

Arguments:

pThing
    Pointer to an object receiving the message
pMesg
    Pointer to a PegMessage structure

Returns:

For all message types handled by this function, the return value will be 0.

Description:

This is the default message handling function for any PegThing or derived object.

To alleviate the caller from casting their object type, the pointer to the object is defined as void. The function casts the pointer appropriately when necessary to access members of the PegThing type.

Errors:

This function may generate an assert if the PegThing object or the PegMessage pointer is NULL and the C/PEG library was compiled with the PEG_USE_ASSERT directive defined.

Related Functions:

PegNotify

All PegThing derived objects in the C/PEG library also have a Notify procedure.

Notes:

Every object that is derived from PegThing and overrides the default Notify function should call this function to handle message types that the derived object is not interested in handling. This ensures proper operation for the object.

1.265 PegThingParentShift

Synopsis:

```
    void PegThingParentShift(void *pThing, PEGSHORT x, PEGSHORT y);
```

Arguments:

pThing

    Pointer to a PegThing object to shift

x

    value to shift along the x axis

y

    value to shift along the y axis

Returns:

None

Description:

This function is commonly called from PegThingResize to recursively move child objects when pThing has itself been moved or resized.

Errors:

This function may generate an assert if pThing is NULL and the C/PEG library was compiled with the PEG_USE_ASSERT directive defined.

Related Functions:

PegThingResize, PegResize, PegParentShift

Notes:

When developing a custom object, it may be necessary to call this function during a resize or reposition event to ensure that the child objects are properly relocated.

1.266 PegThingRemove

Synopsis:

```
void *PegThingRemove(void *pThing, void *pRemove, PEGBOOL
  bRedraw);
```

Arguments:

pThing
    Pointer to the parent object from which the child is to be removed
pRemove
    Pointer to the child object that is to be removed
bRedraw
    Directive to draw the parent object once the child is removed

Returns:

Upon success, a pointer to the child object that was removed is returned. If the child was not a child of pThing, then NULL is returned.

Description:

Use this function to remove children that were added to a parent object using PegAdd or PegAddToEnd. Once the child object has been removed from it's parent, it is no longer visible.

It is important to keep in mind that when a child object is removed from a parent, the child object is no longer a part of the object tree that the C/PEG library maintains. Typically, when an object is destroyed, the object also recursively destroys all of it's children. This ensures that memory is freed up in a systematic fashion. But, when an object is removed from it's parent, it is not destroyed and it's associated memory is not freed. Therefore, it is the responsibility of the application programmer to keep track of this object and either add it back into the C/PEG tree, or destroying it on their own.

In short, if the application programmer removes a child from it's parent and does not keep track of the object, then the memory is never freed and this is a memory leak.

If the child object is visible, it will receive two messages during the removal process. The first will be of type PM_HIDE followed by a PM_NONDRAWABLE.

Errors:

This function may generate an assert if either pThing or pRemove are NULL and the C/PEG library was compiled with the PEG_USE_ASSERT macro defined.

Related Functions:

PegThingAdd, PegThingAddToEnd, PegThingDestroy

PegAdd, PegAddToEnd, PegDestroy

Notes:

Take care to do house keeping when removing children.

1.267 PegThingResize

Synopsis:

```
    void PegThingResize(void *pThing, PegRect *pRect);
```

Arguments:

pThing
    Pointer to a PegThing or derivative that will be resized
pRect
    Pointer to a PegRect that will be used to resize pThing

Returns:

None

Description:

This function properly resizes and/or repositions an existing PegThing object. It properly updates the real, client and clipping rectangles associated with the object. If the object is being repositioned, it also shifts it's child objects relative to it's own new coordinates.

Errors:

This function may generate an assert if pThing or pRect is NULL and the C/PEG library was compiled with the PEG_USE_ASSERT directive defined.

Related Functions:

PegResize

Notes:

Application developers should use this function instead of manually changing any of the object's rectangles directly.

1.268 PegThingSet

Synopsis:

```
    void PegThingSet(PegThing *pThing, PegRect *pRect, PEGUSHORT
      usId, PEGUSHORT usStyle);
```

Arguments:

pThing
: Pointer to a PegThing object

pRect
: Pointer to a PegRect structure that will be assigned to the pThing object to determine the PegThing objects size and position.

usId
: ID to assign the PegThing object

usStyle
: Style to assign the PegThing object

Returns:

None

Description:

This function is used to assign a size, position, ID and style to an existing PegThing object. This function is called by PegThingCreate and may also be called by any object that is derived from PegThing.

The Real, Client and Clip rectangles of the object are initialized and set appropriate for the frame style imposed in the usStyle parameter by calling the PegClientInit function.

Errors:

This function may generate an assert if the PegThing object or the pointer to the PegRect structure is NULL and the C/PEG library was compiled with the PEG_USE_ASSERT directive defined.

Related Functions:

PegThingCreate

Every PegThing derived object also has an initialization function.

Notes:

This function may be called at any time for any PegThing derived object, although it is usually only called during the creation process. Calling this function for a PegThing object that is already visible may have undesirable effects.

1.269 PegThingSetDefFuncs

Synopsis:

```
    void PegThingSetDefFuncs(PegThing *pThing);
```

Arguments:

pThing
: Pointer to a PegThing object

Returns:

None

Description:

This function initializes the function table for every function pointer in a PegThing object. PegThingCreateDefault calls this function. Every PegThing derived object also calls this function during it's initialization process to set up the base function pointers before replacing selected function pointers with their own.

Errors:

This function may generate an assert if the PegThing object is NULL and the C/PEG library was compiled with the PEG_USE_ASSERT directive defined.

Related Functions:

PegThingInit

Most PegThing derived objects in the C/PEG library also have a procedure to set their own function table.

Notes:

This function may be called at any time for any PegThing derived object, although it is usually only called during the creation process. Calling this function for a PegThing object that is already visible may have undesirable effects.

1.270 PegTitleCreate

Synopsis:

```
    PegTitle *PegTitleCreate(const PEGSTRING String, PEGUSHORT
      usId, PEGUSHORT usStyle);
```

Arguments:

String
: a NULL terminated string.

usId
: ID to assign the object.

usStyle
: Style to assign to the object.

Returns:

Upon success, returns a pointer to a newly allocated PegTitle object.

Description:

This function allocates storage for a PegTitle object, initializes the objects and sets it's data members from the passed in parameter list.

Errors:

This function may generate an assert if the necessary memory for the object can not be allocated from the system and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegTitleCreateDefault, PegTitleSet

Notes:

This is one of the few PegThing derived objects that does not take a PegRect structure in the parameter list for it's create function. This is because the PegTitle sizes itself to it's parent object when it receives the PM_SHOW message.

To have a close button appear on the title, pass TF_CLOSE_BUTTON in usStyle.

This function calls PegTitleCreateDefault to allocate storage for the object and PegTitleSet to set the data members of the object.

1.271 PegTitleCreateDefault

Synopsis:

```
    PegTitle *PegTitleCreateDefault(void);
```

Arguments:

None

Returns:

Upon success, returns a pointer to a newly allocated PegTitle object.

Description:

This function allocates storage for a PegTitle object and initializes the object with the default associated with the PegTitle object type.

Errors:

This function may generate an assert if the necessary memory for the object can not be allocated from the system and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegTitleCreate, PegTitleInit

Notes:

This function calls PegTitleInit to initialize the object.

1.272 PegTitleDraw

Synopsis:

```
    void PegTitleDraw(void *pThing);
```

Arguments:

pThing

Pointer to a PegTitle object.

Returns:

None

Description:

This function is an override of the PegTextThingDraw function. The PegTitle object draws itself in normal colors when it's parent window is not current, and in selected colors when it's parent object is current.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegTextThingDraw

Notes:

This function also, optionally, draws a bitmap button on the far right side of the object that acts as a close button.

1.273 PegTitleFontSet

Synopsis:

```
    void PegTitleFontSet(void *pThing, PegFont *pFont);
```

Arguments:

pThing
    Pointer to a PegTitle object.
pFont
    Pointer to a PegFont structure.

Returns:

None

Description:

This function is an override of the PegTextThingFontSet function.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegTextThingFontSet

Notes:

When the font is set on a PegTitle object, the object may have to resize itself in order to properly display it's text.

1.274 PegTitleInit

Synopsis:

```
    void PegTitleInit(PegTitle *pTitle);
```

Arguments:

pTitle

Pointer to a PegTitle object.

Returns:

None

Description:

This function initializes pTitle with the default for a PegTitle object type.

Errors:

This function may generate an assert if pTitle is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegTitleCreateDefault, PegTitleSetDefFuncs

Notes:

This function is called by PegTitleCreateDefault to properly initialize the object. This function calls PegTitleSetDefFuncs to set up the function pointers in the object.

1.275 PegTitleNotify

Synopsis:

```
PEGINT PegTitleNotify(void *pThing, const PegMessage *pMesg);
```

Arguments:

pThing

Pointer to a PegTitle object.

pMesg

Pointer to a PegMessage structure that contains the message data.

Returns:

Various, depending on the type of message.

Description:

This function is an override of the PegTextThingNotify function.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegTextThingNotify

Notes:

1.276 PegTitleSet

Synopsis:

```
    void PegTitleSet(PegTitle *pTitle, PegRect *pRect, const
      PEGSTRING String, PEGUSHORT usId, PEGUSHORT usStyle);
```

Arguments:

pTitle
: Pointer to a PegTitle object.

pRect
: Pointer to a PegRect structure that defines the size and position of the object.

String
: Pointer to a NULL terminated string.

usId
: ID to assign the object.

usStyle
: Style to assign to the object.

Returns:

None

Description:

This function sets the data members in pTitle to the values in the passed in parameter list.

Errors:

This function may generate an assert if pTitle is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegTitleCreate

Notes:

This function is called by the PegTitleCreate function to set up the data members in the object.

1.277 PegTitleSetDefFuncs

Synopsis:

```
    void PegTitleSetDefFuncs(PegTitle *pTitle);
```

Arguments:

pTitle

Pointer to a PegTitle object.

Returns:

None

Description:

This function set the function pointers in pTitle to the default functions for a PegTitle object type.

Errors:

This function may generate an assert if pTitle is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegTitleInit

Notes:

This function is called by PegTitle init to properly set the function pointers in the object.

1.278 PegVertListCreate

Synopsis:

```
    PegVertList *PegVertListCreate(PegRect *pRect, PEGUSHORT usId,
      PEGUSHORT usStyle);
```

Arguments:

pRect
    Pointer to a PegRect structure that defines the size and position of the object.
usId
    ID to assign to the object.
usStyle
    Style to assign to the object.

Returns:

Upon success, returns a pointer to a newly allocated PegVertList object.

Description:

This function allocates storage for a PegVertList object, initializes the object and sets the object's data members.

Errors:

This function may generate an assert if the necessary memory for the object can not be allocated from the system and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegVertListCreateDefault, PegVertListSet

Notes:

This function calls PegVertListCreateDefault to allocate storage for the object, then calls PegVertListSet to set the object's data members.

1.279 PegVertListCreateDefault

Synopsis:

```
    PegVertList *PegVertListCreateDefault(void);
```

Arguments:

None

Returns:

Upon success, returns a pointer to a newly allocated PegVertList object.

Description:

This function allocates storage for a PegVertList object and initializes the object to the default associated with a PegVertList object type.

Errors:

This function may generate an assert if the necessary memory for the object can not be allocated from the system and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegVertListCreate, PegVertListInit

Notes:

This function is called by PegVertList create to allocate the storage for the object. This function then calls PegVertListInit to properly initialize the object.

1.280 PegVertListInit

Synopsis:

```
void PegVertListInit(PegVertList *pVList);
```

Arguments:

pVList

Pointer to a PegVertList object.

Returns:

None

Description:

This function initializes pVList with the defaults associated with a PegVertList object type.

Errors:

This function may generate an assert if pVList is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegVertListCreateDefault, PegVertListSetDefFuncs

Notes:

This function is called by PegVertListCreateDefault to set the default in the object. This function calls PegVertListSetDefFuncs to set the function pointers in the object.

1.281 PegVertListSet

Synopsis:

```
    void PegVertListSet(PegVertList *pVList, PegRect *pRect,
      PEGUSHORT usId, PEGUSHORT usStyle);
```

Arguments:

pVList
: Pointer to a PegVertList object.

pRect
: Pointer to a PegRect structure that defines the size and position of the object.

usId
: ID to assign to the object.

usStyle
: Style to assign to the object.

Returns:

None

Description:

This function assigns the data members of pVList with the values in the passed in parameter list.

Errors:

This function may generate an assert if pVList is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegVertListCreate

Notes:

This function is called by PegVertListCreate to set the data members in the object.

1.282 PegVertListSetDefFuncs

Synopsis:

```
    void PegVertListSetDefFuncs(PegVertList *pVList);
```

Arguments:

pVList
    Pointer to a PegVertList object.

Returns:

None

Description:

This function sets the function pointers in pVList to the default functions associated with a PegVertList object type.

Errors:

This function may generate an assert if pVList is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegVertListInit

Notes:

This function is called by the PegVertListInit function to properly set the function pointers in the object.

1.283 PegVertPromptCreate

Synopsis:

```
    PegVertPrompt *PegVertPromptCreate(PegRect *pRect, const
      PEGSTRING String, PEGUSHORT usId, PEGUSHORT usStyle);
```

Arguments:

pRect
: Pointer to a PegRect structure that defines the size and position of the object.

String
: a NULL terminated string to assign to the object.

usId
: ID to assign to the object.

usStyle
: Style to assign to the object.

Returns:

Upon success, returns a pointer to a newly allocated PegVertPrompt object.

Description:

This function allocates storage for a PegVertPrompt object, initializes the object then sets the object's data members.

Errors:

This function may generate an assert if the necessary memory for the object can not be allocated from the system and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegVertPromptCreateDefault, PegVertPromptSet

Notes:

This function calls PegVertPromptCreateDefault to allocate storage for and initialize the object. Then calls PegVertPromptSet to set the object's data members.

1.284 PegVertPromptCreateDefault

Synopsis:

```
    PegVertPrompt *PegVertPromptCreateDefault(void);
```

Arguments:

None

Returns:

Upon success, returns a pointer to a newly allocated PegVertPrompt object.

Description:

This function allocates storage for a PegVertPrompt object and initializes the objects with the defaults associated with a PegVertPrompt object type.

Errors:

This function may generate an assert if the necessary memory for the object can not be allocated from the system and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegVertPromptCreate, PegVertPromptInit

Notes:

This function is called by PegVertPromptCreate to allocate the object. This function calls PegVertPromptInit to properly initialize the object's default function pointers and data members.

1.285 PegVertPromptDraw

Synopsis:

```
    void PegVertPromptDraw(void *pThing);
```

Arguments:

pThing
    Pointer to a PegVertPrompt object.

Returns:

None

Description:

This function is an override of the PegPromptDraw function. The PegVertPrompt draws it's text vertically.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegPromptDraw

Notes:

The PegVertPrompt draws it's text vertically. The justification flags for a text object also apply to a PegVertPrompt object. The only difference being is that the text output is displayed vertically.

1.286 PegVertPromptInit

Synopsis:

```
void PegVertPromptInit(PegVertPrompt *pVPrompt);
```

Arguments:

pVPrompt
    Pointer to a PegVertPrompt object.

Returns:

None

Description:

This function sets pVPrompt with the defaults associated with a PegVertPrompt object.

Errors:

This function may generate an assert if pVPrompt is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegVertPromptCreateDefault, PegVertPromptSetDefFuncs

Notes:

This function is called by PegVertPromptCreateDefault to initialize the pVPrompt object. This function calls PegVertPromptSetDefFuncs to set the function pointers in the object.

1.287 PegVertPromptSet

Synopsis:

```
    void PegVertPromptSet(PegVertPrompt *pVPrompt, PegRect *pRect,
      const PEGSTRING String, PEGUSHORT usId, PEGUSHORT usStyle);
```

Arguments:

pVPrompt
: Pointer to a PegVertPrompt object.

pRect
: Pointer to a PegRect structure that defines the size and position of the object.

String
: a NULL terminated string to assign to the object.

usId
: ID to assign to the object.

usStyle
: Style to assign to the object.

Returns:

None

Description:

This function sets the data members in pVPrompt.

Errors:

This function may generate an assert if pVPrompt is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegVertPromptCreate

Notes:

This function is called by PegVertPromptCreate to set the data members in the object.

1.288 PegVertPromptSetDefFuncs

Synopsis:

```
void PegVertPromptSetDefFuncs(PegVertPrompt *pVPrompt);
```

Arguments:

pVPrompt
    Pointer to a PegVertPrompt object.

Returns:

None

Description:

This function sets the function pointers in pVPrompt to the default functions associated with a PegVertPrompt object type.

Errors:

This function may generate an assert if pVPrompt is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegVertPromptInit

Notes:

This function is called by PegVertPromptInit to set the function pointers in the object.

1.289 PegVertScrollCreate

Synopsis:

```
    PegVertScroll *PegVertScrollCreate(PegRect *pRect, PEGUSHORT
      usId, PEGUSHORT usStyle);
```

Arguments:

pRect
: Pointer to a PegRect structure that defines the size and position of the object.

usId
: ID to assign to the object.

usStyle
: Style to assign to the object.

Returns:

Upon success, returns a pointer to a newly allocated PegVertScroll object.

Description:

This function allocates storage for a PegVertScroll object, initializes the object, then sets the object's data members.

Errors:

This function may generate an assert if the necessary memory for the object can not be allocated from the system and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegVertScrollCreateDefault, PegVertScrollSet

Notes:

This function calls PegVertScrollCreateDefault to allocate the object, then calls PegVertScrollSet to set the data members of the object.

The thumb button and scroll bar buttons are added to the object at this point. If the application designer wishes to use different scroll bar buttons or a different thumb button, then the application designer may implement a different Create function to that creates instances of custom scroll bar buttons and a thumb button.

The define PEG_SCROLLBAR_SUPPORT must be turned on in order for this function to be available.

1.290 PegVertScrollCreateDefault

Synopsis:

```
    PegVertScroll *PegVertScrollCreateDefault(void);
```

Arguments:

None

Returns:

Upon success, returns a pointer to the newly allocated PegVertScroll object.

Description:

This function allocates storage for a PegVertScroll object. It then initializes the object with the default associated with a PegVertScroll object type.

Errors:

This function may generate an assert if the necessary memory for the object can not be allocated from the system and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegVertScrollCreate, PegVertScrollInit

Notes:

This function is called by PegVertScrollCreate to allocate the object. This function calls PegVertScrollInit to properly initialize the object.

The define PEG_SCROLLBAR_SUPPORT must be turned on in order for this function to be available.

1.291 PegVertScrollDraw

Synopsis:

```
    void PegVertScrollDraw(void *pThing);
```

Arguments:

pThing
Pointer to a PegVertScroll object.

Returns:

None

Description:

This function is an override of the PegThingDraw function.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegThingDraw

Notes:

This function draws the background stipple of the scroll bar, and also causes the child buttons to draw.

The define PEG_SCROLLBAR_SUPPORT must be turned on in order for this function to be available.

1.292 PegVertScrollInit

Synopsis:

```
    void PegVertScrollInit(PegVertScroll *pVScroll);
```

Arguments:

pVScroll
    Pointer to a PegVertScroll object.

Returns:

None

Description:

This function initializes pVScroll with the defaults associated with a PegVertScroll object type.

Errors:

This function may generate an assert if pVScroll is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegVertScrollCreateDefault, PegVertScrollSetDefFuncs

Notes:

This function is called by PegVertScrollCreateDefault to initialize the object. This function calls PegVertScrollSetDefFuncs to set the function pointers in the object.

The define PEG_SCROLLBAR_SUPPORT must be turned on in order for this function to be available.

1.293 PegVertScrollNotify

Synopsis:

```
PEGINT PegVertScrollNotify(void *pThing, const PegMessage
  *pMesg);
```

Arguments:

pThing
: Pointer to a PegVertScroll object.

pMesg
: Pointer to a PegMessage structure that contains the contents of the message.

Returns:

Various, depending on the type of message.

Description:

This function is an override of the PegThingNotify function.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegThingNotify

Notes:

This function allows the scroll bar to react properly to user and application input.

The define PEG_SCROLLBAR_SUPPORT must be turned on in order for this function to be available.

1.294 PegVertScrollReset

Synopsis:

```
void PegVertScrollReset(PegVertScroll *pVScroll, PegScrollInfo
  *pScrollInfo);
```

Arguments:

pVScroll
    Pointer to a PegVertScroll object.

pScrollInfo
    Pointer to a PegScrollInfo structure.

Returns:

None

Description:

This function resets the scrolling info in pVScroll with the values contained in pScrollInfo. If pScrollInfo is NULL, the scroll info of pVScroll is reset to default values.

Errors:

This function may generate an assert if pVScroll is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegVertScrollReset

Notes:

The PegScrollInfo structure defines how the scroll bar actually works. See the section on scrolling in the “C/PEG Programmers Manual” for a discussion of how scrolling works in C/PEG.

The define PEG_SCROLLBAR_SUPPORT must be turned on in order for this function to be available.

1.295 PegVertScrollResize

Synopsis:

```
void PegVertScrollResize(void *pThing, PegRect *pRect);
```

Arguments:

pThing
: Pointer to a PegVertScroll object.

pRect
: Pointer to a PegRect structure that defines the new size and position of the object.

Returns:

None

Description:

This function is an override of the PegThingResize function.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegThingResize

Notes:

When a PegVertScroll object is resized, it must calculate the new size and position of the thumb button as well as the step for scrolling.

The define PEG_SCROLLBAR_SUPPORT must be turned on in order for this function to be available.

1.296 PegVertScrollSet

Synopsis:

```
    void PegVertScrollSet(PegVertScroll *pVScroll, PegRect *pRect,
      PEGUSHORT usId, PEGUSHORT usStyle);
```

Arguments:

pVScroll
: Pointer to a PegVertScroll object.

pRect
: Pointer to a PegRect structure that defines the size and position of the object.

usId
: ID to assign the object.

usStyle
: Style to assign the object.

Returns:

None

Description:

This function sets the data members of pVScroll.

Errors:

This function may generate an assert if pVScroll is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegVertScrollCreate

Notes:

This function is called by PegVertScrollCreate to set the data members of the object.

The define PEG_SCROLLBAR_SUPPORT must be turned on in order for this function to be available.

1.297 PegVertScrollSetDefFuncs

Synopsis:

```
void PegVertScrollSetDefFuncs(PegVertScroll *pVScroll);
```

Arguments:

pVScroll
    Pointer to a PegVertScroll object.

Returns:

None

Description:

This function sets the function pointers in pVScroll to the default functions associated with a PegVertScroll object type.

Errors:

This function may generate an assert if pVScroll is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegVertScrollInit

Notes:

This function is called by PegVertScrollInit to set the function pointers in the object.

The define PEG_SCROLLBAR_SUPPORT must be turned on in order for this function to be available.

CHAPTER 2

C/PEG STRUCTURE FUNCTIONS

2.1 Overview

The following functions deal with handling some of the structures defined in C/PEG.

In C/PEG a structure is a data collection that does not contain function pointers and does not derive from the PegThing object. As such, they usually do not have functions to create, initialize and destroy objects related to them. The structures are used primarily within the context of a single function and are thus usually created on the stack.

Examples of structures are PegMessage, PegPoint and PegFont. Each of these structures have complimentary functions that deal with setting the members of the structure. Some also have functions that are the equivalent of logical operations, like equals or not equals, as well as functions that would act as a unary operator, such as adding and subtracting.

The PegRect function has the greatest number of associated functions simply because so much of a graphical user interface deals with rectangular shapes.

For a list of the structure in C/PEG and their associated meanings, see the structures section in the “C/PEG Programmers Manual”.

2.2 PegBitmapCreate

Synopsis:

```
PegBitmap *PegBitmapCreate(PEGSHORT sWidth, PEGSHORT sHeight,
  PEGUBYTE ubBitsPix, PEGUBYTE ubFlags, PEGULONG ulTransColor,
  PEGBOOL bAllocData);
```

Arguments:

sWidth
: The width of the bitmap, expressed in pixels.

sHeight
: The height of the bitmap, expressed in pixels.

ubBitsPix
: The bits per pixel in which the image data is formatted.

ubFlags
: Modifier flags.

ulTransColor
: The color value to use as the transparent color for the bitmap.

bAllocData
: If TRUE, directs the function to allocate memory for the bitmap data.

Returns:

A pointer to a newly allocated PegBitmap structure.

Description:

This function allocates a PegBitmap structure in memory, sets the structure members from the parameters list and returns a pointer to the structure.

Errors:

This function may generate an assert if the memory for the bitmap structure can not be allocated and the C/PEG library was compiled with the PEG_USER_ASSERT directive defined.

Related Functions:

PegBitmapCreateDefault, PegBitmapDestroy

Notes:

Valid flag values are listed in the following table:

Flag	Attribute
BMF_RAW	Bitmap data is in raw format
BMF_RLE	Bitmap data is in RLE format
BMF_NATIVE	Bitmap data is in native video format
BMF_ROTATED	Bitmap data is rotated 90 or 270 degrees
BMF_GRAYSCALE	Bitmap data is in grayscale format
BMF_HAS_TRANS	Bitmap has transparency
BMF_SPRITE	Bitmap data resides in video hardware memory
BMF_RGB	Bitmap data is in RGB not BGR format
BMF_ALPHA	For 16 or 24 bpp bitmaps, bitmap data has an alpha channel.

If bAllocData is TRUE, the function attempts to allocate the necessary amount of memory to hold the bitmap data based on the height, width and color depth of the image. If PEG_USE_VID_MEM_MANAGER is defined, the function attempts to allocate the memory from a memory segment on the video hardware. If not, then the function allocates the memory from local system memory.

This structure does not derive from PegThing, and, as such, does not fall under the same rules of memory ownership as PegThing derived objects. The application is always responsible for keeping track of this structure and associated memory. This structure should be destroyed using the PegBitmapDestroy function, and not by manually freeing it.

2.3 PegBitmapCreateDefault

Synopsis:

```
PegBitmap *PegBitmapCreateDefault(void);
```

Arguments:

None

Returns:

Returns a pointer to a newly allocated PegBitmap structure.

Description:

This function allocates a PegBitmap structure and returns a pointer to it. All of the structure members are initialized to 0. The application code may use this structure in a call to PegBitmapSet to easily set all of the members of the structure.

Errors:

This function may generate an assert if the PegBitmap structure memory can not be allocated and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegBitmapCreate, PegBitmapDestroy

Notes:

The PegBitmap structure is not a derivative of PegThing, and therefore does not fall under the same memory ownership rules as a PegThing derivative. The caller is responsible for the memory returned from this call at all times. To destory this structure, PegBitmapDestroy should be called to properly release the memory for this structure.

2.4 PegBitmapDestroy

Synopsis:

```
    void PegBitmapDestroy(PegBitmap *pBitmap);
```

Arguments:

pBitmap
    Pointer to a PegBitmap structure to destroy.

Returns:

None

Description:

This function releases the memory for the bitmap data as well as the memory associated with the pointer to the structure.

Errors:

This function may generate an assert if the pBitmap pointer is invalid and the C/PEG library was built with the PEG_USER_ASSERT directive defined.

Related Functions:

PegBitmapCreate, PegBitmapCreateDefault

Notes:

This is the proper way to release the memory of a PegBitmap that was created using PegBitmapCreate or PegBitmapCreateDefault.

2.5 PegBitmapSet

Synopsis:

```
    void PegBitmapSet(PegBitmap *pBitmap, PEGSHORT sWidth, PEGSHORT
      sHeight, PEGUBYTE ubBitsPix, PEGUBYTE ubFlags, PEGULONG
      ulTransColor, PEGUBYTE PEGFAR *pData);
```

Arguments:

pBitmap	Pointer to a PegBitmap structure.
sWidth	The width of the bitmap, expressed in pixels.
sHeight	The height of the bitmap, expressed in pixels.
ubBitsPix	The color depth of the bitmap.
ubFlags	Flags that determine the layout and attributes of the bitmap data
ulTransColor	A color value used for mapping the transparent color of the bitmap data.
pData	Pointer for the actual bitmap image data.

Returns:

None

Description:

This function sets the data members of the pBitmap structure.

Errors:

This function may generate an assert if the pBitmap pointer is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

It is not an error to pass NULL for the pData parameter.

Related Functions:

PegBitmapCreate, PegBitmapCreateDefault

Notes:

This function does not allocate any new memory.

2.6 PegBrushCopy

Synopsis:

```
void PegBrushCopy(PegBrush *pTarget, PegBrush *pSource);
```

Arguments:

pTarget
: Pointer to a PegBrush structure.

pSource
: Pointer to a PegBrush structure.

Returns:

None

Description:

This function copies the values from pSource to pTarget. This is a convenient way to copy PegBrush structures.

Errors:

This function may generate an assert if the either PegBrush pointer is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegBrushCreate, PegBrushSet

Notes:

2.7 PegBrushCreate

Synopsis:

```
    PegBrush *PegBrushCreate(PEGCOLORVAL fore, PEGCOLORVAL back,
      PEGUBYTE ubFlags, PEGINT iWidth);
```

Arguments:

fore
    Color value used for foreground drawing.
back
    Color value used for background drawing
ubFlags
    Modifier flag.
iWidth
    Width to use for line drawing.

Returns:

None

Description:

This function allocates memory for a PegBrush object, sets it's member data from the parameters and returns a pointer to the new structure.

Errors:

This function may generate an assert if the memory for the new PegBrush object can not be allocated and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegBrushCreateDefault

Notes:

This structure does not derive from PegThing and, therefore, does not fall under the same rules of ownership as a PegThing derivative. The application is responsible for calling PegBrushDestroy for this object when the application is finished with the object.

2.8 PegBrushCreateDefault

Synopsis:

```
    PegBrush *PegBrushCreateDefault(void);
```

Arguments:

None

Returns:

A pointer to a newly allocated PegBrush structure.

Description:

This function allocates memory for a PegBrush structure. The structure members are initialized as follows:

- Foreground: BLACK
- Background: WHITE
- iWidth: 1
- ulPattern: PEG_DEF_FILL_PATTERN
- Clip: All members set to -1

The remaining members are initialized to zero.

Errors:

This function may generate an assert if the memory for the structure can not be allocated and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegBrushCreate

Notes:

See the Notes section in PegBrushCreate.

2.9 PegBrushDestroy

Synopsis:

```
    void PegBrushDestroy(PegBrush *pBrush);
```

Arguments:

pBrush
    Pointer to a PegBrush object that is to be destroyed.

Returns:

None

Description:

This function releases the memory associated with pBrush. This memory must have been allocated by PegBrushCreate or PegBrushCreateDefault.

Errors:

This function may generate an assert if pBrush is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegBrushCreate, PegBrushCreateDefault

Notes:

This function does not release the memory associated with the internal PegBitmap pointer. It is the responsibility of the application code to destroy this bitmap.

2.10 PegBrushSet

Synopsis:

```
    void PegBrushSet(PegBrush* pBrush, PEGCOLORVAL fore,
        PEGCOLORVAL back, PEGUBYTE ubFlags, PEGINT iWidth);
```

Arguments:

pBrush
    Pointer to a PegBrush structure.
fore
    Foreground color.
back
    Background color.
ubFlags
    Modifier flags.
iWidth
    Width to use in line drawing operations.

Returns:

None

Description:

This function sets the Foreground, Background, ubFlags and iWidth members of the pBrush structure.

Errors:

This function may generate an assert if pBrush is invalid and the C/PEG library was compiled with the PEG_USE_ASSERT directive defined.

Related Functions:

PegBrushColorsSet, PegBrushFlagsSet, PegBrushPatternSet, PegBrushBitmapSet, PegBrushClipRectSet

Notes:

2.11 PegCaptureRealloc

Synopsis:

```
    void PegCaptureRealloc(PegCapture *pCapture, PEGLONG lSize);
```

Arguments:

pCapture
    Pointer to a PegCapture structure.
lSize
    Size of the bitmap data array to allocate.

Returns:

None

Description:

This function deals with the bitmap data associated with pCapture. If the data block has already been allocated, the data is first freed.

To release all of the memory associated with the bitmap block, set lSize to 0.

Errors:

This function may generate an assert if either pCapture is invalid and the C/PEG library was compiled with the PEG_USE_ASSERT directive defined.

Related Functions:

PegCaptureMoveTo, PegCaptureShift,
PegCaptureSetPos,PegCaptureSet.

Notes:

2.12 PegCaptureSetPos

Synopsis:

```
    void PegCaptureSetPos(PegCapture *pCapture, PegRect *pRect);
```

Arguments:

pCapture
	Pointer to a PegCapture structure.
pRect
	Pointer to a PegRect structure.

Returns:

None

Description:

This function sets the capture rectangle in pCapture to pRect. It also sets the internal bitmap width and height to the width and height of the rectangle.

Errors:

This function may generate an assert if either pCapture or pRect are invalid and the C/PEG library was compiled with the PEG_USE_ASSERT directive defined.

Related Functions:

PegCaptureRealloc, PegCaptureSet.

Notes:

2.13 PegCaptureSet

Synopsis:

```
    void PegCaptureSet(PegCapture *pCapture, PegRect *pRect);
```

Arguments:

pCapture
Pointer to a PegCapture structure.
pRect
Pointer to a PegRect structure.

Returns:

None

Description:

This function sets the internal capture rectangle of pCapture to pRect and resets the valid flag to false.

Errors:

This function may generate an assert if either pCapture or pRect are invalid and the C/PEG library was compiled with the PEG_USE_ASSERT directive defined.

Related Functions:

PegCaptureRealloc, PegCaptureSetPos.

Notes:

Be careful not to call this function for a PegCapture structure which may have already allocated bitmap data.

2.14 PegMessageSet

Synopsis:

```
    void PegMessageSet(PegMessage *pMesg, PEGUINT uiType, PEGINT
      iData, void *pTarget, void *pSource);
```

Arguments:

pMesg
: Pointer to a PegMessage structure.

uiType
: Type to assign the message.

iData
: Value to put into pMesg's sData member.

pTarget
: Pointer to a PegThing or derivative that is the target of the message.

pSource
: Pointer to a PegThing or derivative that is the source of the message.

Returns:

None

Description:

This function is a convenient way to set the data members of a PegMessage structure.

Errors:

This function may generate an assert if pMesg is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

None

Notes:

See the “C/PEG Programmers Manual” for an explanation of the data members of the PegMessage structure.

2.15 PegPointAdd

Synopsis:

```
    void PegPointAdd(PegPoint *pTarget, const PegPoint *pSource);
```

Arguments:

pTarget
    Pointer to a PegPoint structure.
pSource
    Pointer to a PegPoint structure.

Returns:

None

Description:

This function adds the coordinates of pSource to the respective coordinates of pTarget.

Errors:

This function may generate an assert if either pTarget or pSource are invalid and the C/PEG library was compiled with the PEG_USE_ASSERT directive defined.

Related Functions:

PegPointCopy, PegPointDistance

Notes:

2.16 PegPointCopy

Synopsis:

```
void PegPointCopy(PegPoint *pTarget, PegPoint *pSource);
```

Arguments:

pTarget
Pointer to a PegPoint structure.
pSource
Pointer to a PegPoint structure.

Returns:

None

Description:

This function copies the contents of pSource into pTarget.

Errors:

This function may generate an assert if either pointer is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegPointAdd

Notes:

2.17 PegPointDistance

Synopsis:

```
    PEGLONG PegPointDistance(PegPoint *pPoint1, PegPoint *pPoint2);
```

Arguments:

pPoint1
    Pointer to a PegPoint structure.
pPoint2
    Pointer to a PegPoint structure.

Returns:

The square of the distance between the points.

Description:

This function is primarily used by the C/PEG library to determine keyboard navigation within PegPanel objects.

Errors:

This function may generate an assert if either pointer is invalid and the C/PEG library was compiled with the PEG_USE_ASSERT directive defined.

Related Functions:

PegPointCopy

Notes:

The algorithm for determining this distance is very simple. It is the square of the difference between the x coordinates plus the square of the difference between the y coordinates. For simple keyboard navigation, this provides a good enough approach to determining which sibling object is closest to any given sibling object.

This function is only available if PEG_KEYBOARD_SUPPORT and PEG_ARROW_KEY_SUPPORT is defined in the C/PEG library.

2.18 PegPointEquals

Synopsis:

```
    PEGBOOL PegPointEquals(const PegPoint *pPoint1, const PegPoint
      *pPoint2);
```

Arguments:

pPoint1
: Pointer to a PegPoint structure.

pPoint2
: Pointer to a PegPoint structure.

Returns:

TRUE if the coordinates in pPoint1 match the coordinates in pPoint2, otherwise, FALSE.

Description:

This function compares the coordinates in pPoint1 with the coordinates in pPoint2.

Errors:

This function may generate and assert if either pointer is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegPointNotEquals

Notes:

2.19 PegPointNotEquals

Synopsis:

```
    PEGBOOL PegPointNotEquals(const PegPoint *pPoint1, const
      PegPoint pPoint2);
```

Arguments:

pPoint1
    Pointer to a PegPoint structure.
pPoint2
    Pointer to a PegPoint structure.

Returns:

TRUE if the coordinates in pPoint1 do not match the coordinates in pPoint2, otherwise, FALSE.

Description:

This function compares the coordinates in pPoint1 with the coordinates in pPoint2.

Errors:

This function may generate and assert if either pointer is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegPointEquals

Notes:

2.20 PegRGBSet

Synopsis:

```
    void PegRGBSet(PegRGB* pRGB, PEGUBYTE r, PEGUBYTE g, PEGUBYTE
      b, PEGUBYTE alpha);
```

Arguments:

pRGB
: Pointer to a PegRGB structure.

r
: Value to assign the red member.

g
: Value to assign the green member.

b
: Value to assign the blue member.

alpha
: Value to assign the alpha channel member.

Returns:

None

Description:

This function sets the data members of a PegRGB structure.

Errors:

This function may generate an assert if pRGB is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

None

Notes:

2.21 PegRectAddPoint

Synopsis:

```
    void PegRectAddPoint(PegRect *pRect, PegPoint *pPoint);
```

Arguments:

pRect
    Pointer to a PegRect structure.
pPoint
    Pointer to a PegPoint structure.

Returns:

None

Description:

This function adds the x coordinate of pPoint to the left and right members of pRect and adds the y coordinate of pPoint to the top and bottom coordinates of pRect, effectively shifting pRect by pPoint.

Errors:

This function may generate an assert if either pointer argument in invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegRectAddVal, PegRectShift

Notes:

2.22 PegRectAddVal

Synopsis:

```
    void PegRectAddVal(PegRect *pRect, PEGINT iVal);
```

Arguments:

pRect

Pointer to a PegRect structure.

iVal

Value to add to pRect.

Returns:

None

Description:

This function inflates pRect by sVal. If sVal is negative, then pRect is deflated.

The left and top members of pRect subtract sVal, while the right and bottom members of pRect add sVal.

Errors:

This function may generate an assert if either pointer argument in invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegRectAddPoint, PegRectShift

Notes:

2.23 PegRectAnd

Synopsis:

```
    void PegRectAnd(PegRect *pTarget, PegRect *pSource);
```

Arguments:

pTarget
    Pointer to a PegRect structure.
pSource
    Pointer to a PegRect structure.

Returns:

None

Description:

This function evaluates the matching coordinates in pTarget and pSource and stores the coordinates to the largest overlapping rectangle described by pTarget and pSource in the coordinates of pTarget.

Errors:

This function may generate an assert if either pointer is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegRectOr

Notes:

2.24 PegRectContainsCoords

Synopsis:

```
    PEGBOOL PegRectContainsCoords(PegRect *pRect, PEGINT x, PEGINT
      y);
```

Arguments:

pRect
    Pointer to a PegRect structure.

x
    The coordinate along the x axis.

y
    The coordinate along the y axis.

Returns:

If the x and y coordinates lie inside the boundaries of pRect, then TRUE, otherwise, FALSE.

Description:

This function tests if x and y are contained within the rectangle described by the data members of pRect.

Errors:

This function may generate an assert if pRect is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegRectContainsPoint, PegRectContainsRect

Notes:

2.25 PegRectContainsPoint

Synopsis:

```
    PEGBOOL PegRectContainsPoint(PegRect *pRect, PegPoint *pPoint);
```

Arguments:

pRect

Pointer to a PegRect structure.

pPoint

Pointer to a PegPoint structure.

Returns:

If the coordinates contained in pPoint are contained in pRect, then TRUE, otherwise, FALSE.

Description:

This function tests if the x and y coordinates of pPoint lie within the rectangle described by pRect.

Errors:

This function may generate an assert if either pRect or pPoint are invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegRectContainsCoords, PegRectContainsRect

Notes:

2.26 PegRectContainsRect

Synopsis:

```
    PEGBOOL PegRectContainsRect(PegRect *pRect, PegRect *pTest);
```

Arguments:

pRect

Pointer to a PegRect structure.

pTest

Pointer to a PegRect structure.

Returns:

Returns TRUE if pRect contains pTest, otherwise, returns FALSE.

Description:

This function tests if the rectangle described by pTest lies completely within the rectangle described by pRect.

Errors:

This function may generate an assert if either pointer is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegRectContainsCoord, PegRectContainsPoint

Notes:

2.27 PegRectCopy

Synopsis:

```
    void PegRectCopy(PegRect *pTarget, PegRect *pSource);
```

Arguments:

pTarget
	Pointer to a PegRect structure.
pSource
	Pointer to a PegRect structure.

Returns:

None

Description:

This function copies the rectangle described by pSource into pTarget.

Errors:

This function may generate an assert if either pointer is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

None

Notes:

2.28 PegRectEqual

Synopsis:

```
    PEGBOOL PegRectEqual(PegRect *pLeft, PegRect *pRight);
```

Arguments:

pLeft
    Pointer to a PegRect structure.
pRight
    Pointer to a PegRect structure.

Returns:

If the rectangle described by pLeft is equal to the rectangle described by pRight, the TRUE, otherwise, FALSE.

Description:

This function evaluates the coordinates of both rectangles. If they are exactly equal, meaning that the rectangles are the same size and in the same position, the function returns TRUE. Otherwise, it returns FALSE.

Errors:

This function may generate an assert if either pointer is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegRectNotEqual

Notes:

2.29 PegRectHeight

Synopsis:

```
PEGSHORT PegRectHeight(PegRect *pRect);
```

Arguments:

pRect

    Pointer to a PegRect structure.

Returns:

The height of the rectangle described by pRect.

Description:

This function evaluates the top and bottom members of pRect and returns the distance between the members plus 1.

If the bottom member is less than the top member, this function returns a negative value.

Errors:

This function may generate an assert if pRect is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegRectWidth

Notes:

2.30 PegRectMoveTo

Synopsis:

```
void PegRectMoveTo(PegRect *pRect, PEGINT x, PEGINT y);
```

Arguments:

pRect
: Pointer to a PegRect structure.

x
: Coordinate on which to place the left member of the rectangle.

y
: Coordinate on which to place the top member of the rectangle.

Returns:

None

Description:

This function moves pRect so that it's left and top members are at the coordinates described by x and y, respectively. The right and bottom members of pRect are then shifted the same distance to preserve the width and height of the rectangle.

Errors:

This function may generate an assert if pRect is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegRectShift

Notes:

2.31 PegRectNotEqual

Synopsis:

```
    PEGBOOL PegRectNotEqual(PegRect *pLeft, PegRect *pRight);
```

Arguments:

pLeft

    Pointer to a PegRect structure.

pRight

    Pointer to a PegRect structure.

Returns:

If any coordinate of pLeft is different that it's matching coordinate in pRight, TRUE, otherwise, FALSE.

Description:

This function tests the coordinates of both rectangles. If the rectangles are exactly the same, their size and position are identical, then this function returns FALSE. If any coordinate in one differs from the matching coordinate in the other, this function returns TRUE.

Errors:

This function may generate an assert if either pointer is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegRectEqual

Notes:

2.32 PegRectOr

Synopsis:

```
    void PegRectOr(PegRect *pTarget, PegRect *pSource);
```

Arguments:

pTarget
  Pointer to a PegRect structure.
pSource
  Pointer to a PegRect structure.

Returns:

None

Description:

This function evaluates the corresponding coordinates in pTarget and pSource and stores the coordinates that defines the smallest rectangle that encompasses both pTarget and pSource in pTarget.

Errors:

This function may generate an assert if either pointer is invalid and the C/ PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegRectAnd

Notes:

2.33 PegRectOverlaps

Synopsis:

```
    PEGBOOL PegRectOverlaps(PegRect *pRect, PegRect *pTest);
```

Arguments:

pRect
    Pointer to a PegRect structure.
pTest
    Pointer to a PegRect structure.

Returns:

TRUE if pTest overlaps pRect, otherwise, FALSE.

Description:

This function tests pTest for overlapping pRect.

Errors:

This function may generate an assert if either pointer is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegRectContainsRect

Notes:

2.34 PegRectSet

Synopsis:

```
    void PegRectSet(PegRect *pRect, PEGINT iLeft, PEGINT iTop,
      PEGINT iRight, PEGINT iBottom);
```

Arguments:

pRect
: Pointer to a PegRect object.

iLeft
: Value describing the left coordinate of the rectangle.

iTop
: Value describing the top coordinate of the rectangle.

iRight
: Value describing the right coordinate of the rectangle.

iBottom
: Value describing the bottom coordinate of the rectangle.

Returns:

None

Description:

This function allows the application to the set coordinates describing a rectangle in pRect.

Errors:

This function may generate an assert if pRect is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegRectCopy

Notes:

2.35 PegRectShift

Synopsis:

```
    void PegRectShift(PegRect *pRect, PEGINT xshift, PEGINT
      yshift);
```

Arguments:

pRect
  Pointer to a PegRect structure.
xshift
  The amount to shift the rectangle along the x axis.
yshift
  The amount to shift the rectangle along the y axis.

Returns:

None

Description:

This function shifts the rectangle described in pRect the amount along the x and y axis's of xshift and yshift, respectively.

This function does not modify the size of the rectangle, only the position.

Errors:

This function may generate an assert if pRect is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegRectMoveTo

Notes:

2.36 PegRectWidth

Synopsis:

```
    PEGSHORT PegRectWidth(PegRect *pRect);
```

Arguments:

pRect

  Pointer to a PegRect structure.

Returns:

The height of the rectangle described by pRect.

Description:

This function evaluates the left and top members of pRect and returns the distance between the members plus 1.

If the right member is less than the left member, this function returns a negative value.

Errors:

This function may generate an assert if pRect is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegRectHeight

Notes:

2.37 PegScrollInfoSet

Synopsis:

```
    void PegScrollInfoSet(PegScrollInfo *pScrollInfo, PEGINT iMin,
      PEGINT iMax, PEGINT iCurrent, PEGINT iStep, PEGINT
      iVisible);
```

Arguments:

pScrollInfo
: Pointer to a PegScrollInfo structure.

iMin
: Minimum value of the scrolling range.

iMax
: Maximum value of the scrolling range.

iCurrent
: Current value in the scrolling range.

iStep
: Value to step for a up or down operation.

iVisible
: Value that describes the amount of the range that is visible at one time.

Returns:

None

Description:

This function sets the data members of pScrollInfo.

Errors:

This function may generate an assert if pScrollInfo is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

None

Notes:

The define PEG_SCROLLBAR_SUPPORT must be turned on in order for this function to be available.
See the section on scrolling in the “C/PEG Programmers Manual”.

CHAPTER 3

C/PEG GRAPHICS FUNCTIONS

3.1 Overview

The following functions deal with graphics primitive support. Any of these functions may be called from within the drawing context of any PegThing or derived object. This allows the application designer a great deal of freedom in defining the visual output of a custom object.

It is important to note that an object must be visible and must have drawable status in order to actually do any drawing to the screen.

The first status, visibility, is quite obvious. If the object is not visible, it should not be allowed to draw. That's self evident.

The drawable status is a little trickier. An object may be visible but may not be able to draw. This notion is enforced to insure that objects do not draw into each other's screen space. While all objects are aware of their size and position on the screen, and also have a clipping rectangle, they are not always aware of when they are partially covered by another object that is not a sibling.

See the “C/PEG Programmers Manual” for a detailed discussion on drawable status in PegThing derived objects.

3.2 PegDrawArc

Synopsis:

```
void PegDrawArc(void *pThing, PEGINT ixCenter, PEGINT iyCenter,
     PEGINT iRadius, PEGINT iStartAngle, PEGINT iEndAngle,
     PegBrush *pBrush);
```

Arguments

pThing
: Pointer to a PegThing object which is used for the drawing context.

ixCenter, iyCenter
: Coordinates describing the center point of the arc.

iRadius
: Defines the radius of the arc.

iStartAngle
: Defines the start angle where drawing will begin. Valid values are from 1 to 359.

iEndAngle
: Defines the end angle where drawing will cease. Valid values are from 1 to 359.

pBrush
: Pointer to a PegBrush object.

Returns:

None

Description:

This function draws an arc, or, if optionally filled, a pie slice.

The pBrush object controls the visual appearance of the arc. If the ubFlags member is set to CF_FILL, then the arc is filled with the Background color. In this case, if the Foreground and Background colors are different and the iWidth member is greater than 0, then the arc is outlined with the Foreground color.

If the ubFlags member is CF_NONE and iWidth is greater than 0, then the arc is outlined with the Foreground color and is not filled.

Errors:

This function may generate an assert if any of the pointer parameters are invalid and the C/PEG library was compiled with the PEG_USE_ASSERT directive defined.

Related Functions:

PegDrawCirlce, PegDrawArc

Notes:

This function is only available if the C/PEG library was compiled with the PEG_ARC_GRAPHICS directive defined.

3.3 PegDrawBegin

Synopsis:

```
    void PegDrawBegin(void *pThing);
```

Arguments:

pThing

Pointer to the PegThing object that is beginning the draw operation.

Returns:

None

Description:

This function is called prior to any draw functions called by the object when the object is drawing. For all drawing operations, a pointer to a PegThing object is used for the context of the operation. This allows the draw operation to properly clip to the clip region of the object.

Errors:

This function may generate an assert if the PegThing object is invalid and the C/PEG library was compiled with the PEG_USE_ASSERT directive defined.

Related Functions:

PegDrawBeginBitmap, PegDrawEnd, PegRectRegionInvalidate

Notes:

The PegScreen object tracks the invalid region of the screen, and only drawing that takes place within this region actually updates the visible screen. If drawing is done outside of the draw function of the object, then the object must first invalidate the portion of the screen into which it is expecting to draw. This can be accomplished by calling PegRectRegionInvalidate before calling PegDrawBegin. Any call to PegRectRegionInvalidate made after PegDrawBegin will be ignored by PegScreen.

PegDrawBegin and PegDrawEnd must be called in balanced pairs for the visible screen to update properly. It is an error to call PegDrawBegin without calling PegDrawEnd and will result in undefined behavior by the display.

3.4 PegDrawBeginBitmap

Synopsis:

```
void PegDrawBeginBitmap(void *pThing, PegBitmap *pBitmap);
```

Arguments:

pThing
: Pointer to the object which is beginning the drawing operation.

pBitmap
: Pointer to a valid PegBitmap which all subsequent drawing operations will target.

Returns:

None

Description:

This function shifts drawing operations away from the screen and to the bitmap pointed to by pBitmap. This allows the application to draw into an off-screen memory region without disturbing the contents of the visible display.

Errors:

None

Related Functions:

PegDrawBegin, PegDrawEndBitmap.

Notes:

No drawing will take place on the screen until PegDrawEndBitmap is called once for each call to PegDrawBeginBitmap. Once PegDrawEndBitmap has been called sufficient number of times, the scan line pointers in the screen object will reset and drawing to the screen will resume.

3.5 PegDrawBitmap

Synopsis:

```
void PegDrawBitmap(void *pThing, PegPoint *pWhere, PegBrush
    *pBrush, PEGBOOL bOnTop);
```

Arguments:

pThing
: A pointer to a PegThing object that is used for the drawing context.

pWhere
: A pointer to a PegPoint structure that defines the coordinates for the placement of the top/left corner of the bitmap.

pBrush
: A pointer to a PegBrush structure that contains a pointer to the bitmap to draw.

bOnTop
: Determines if the bitmap will be drawn on top of all other visible objects.

Returns:

None

Description:

The is the principle function used to draw bitmaps to the screen. The draw operation begins drawing the bitmap at the coordinates defined in pWhere.

Errors:

This function may generate an assert if any of the pointer parameters are invalid and the C/PEG library was compiled with the PEG_USER_ASSERT directive defined.

Related Functions:

PegDrawBitmapFill, PegBrushFlagsSet

Notes:

It is not an error for pWhere to specify coordinates that are outside of the object's clipping region or the visible screen.

The bOnTop parameter should normally be set to FALSE. Setting this argument to TRUE will cause the bitmap to be drawn over top of any existing, viewable objects on the screen. This is usually not the desired effect.

The flags set in the brush determine how the bitmap is to be drawn. If the flags are set to CF_NONE, the function simply draws the bitmap on the screen at the given point pWhere. If the CF_CLIP flag is set, the drawing operation will clip the bitmap drawing to the rectangle contained in the brush. If the CF_CENTER flag is set, the bitmap will be drawn in the center of the brush's clip rectangle. The CF_CENTER flag implies the CF_CLIP flag. Finally, if CF_TILE is set, the bitmap is tiled over the rectangle in the brush's clip rectangle. Again, the CF_TILE flag implies the CF_CLIP flag.

3.6 PegDrawBitmapFill

Synopsis:

```
void PegDrawBitmapFill(void *pThing, PegBrush *pBrush);
```

Arguments:

pThing
: Pointer to a PegThing object used for the drawing context.

pBrush
: Pointer to a PegBrush structure that contains the bitmap and rectangle to fill.

Returns:

None

Description:

This function fills a rectangle region with a bitmap. If necessary, the bitmap will be tiled to cover the entire rectangular region.

Errors:

This function may generate an assert if any of the parameters are invalid and the C/PEG library was compiled with the PEG_USER_ASSERT directive defined.

Related Functions:

PegDrawBitmap

Notes:

It is not an error for the coordinates defined in pRect are outside of the object's clipping are or the visible screen.

The fill algorithm always begins by drawing the bitmap in the top/left point defined by the rectangle. If this point does not line up with the object's clipping region, the top/left of the bitmap will not line up with the object's clip region.

There is no carry over from fill rows when tiling the bitmap. In other words, the algorithm always begins with the start of the bitmap at the left edge of the rectangle and fills the region until the left edge of the bitmap passes beyond the right edge of the rectangle.

It is not necessary for the width of the rectangle to be a multiple of the width of the bitmap. The fill algorithm clips the final bitmap on a row to the right edge of the rectangle, even if that width is not the full width of the bitmap.

Calling this function is identical to calling PegDrawBitmap with the CF_TILE and CF_CLIP flags set in the brush.

3.7 PegDrawCircle

Synopsis:

```
    void PegDrawCircle(void *pThing, PEGINT ixCenter, PEGINT
        iyCenter, PEGINT iRadius, PegBrush *pBrush);
```

Arguments:

pThing
: Pointer to a PegThing object used for the drawing context.

ixCenter, iyCenter
: Coordinates for the center of the circle.

iRadius
: The radius of the circle

pBrush
: Pointer to a PegBrush object that controls the visual appearance of the circle.

Returns:

None

Description:

This function draws a circle described by the radius at the given coordinates.

If the ubFlags member of the brush is set to CF_FILL, the circle is filled with the Background color. In this case, if the Foreground and Background colors are different and the iWidth member is greater than 1, then the circle is also outlined in the Foreground color.

If ubFlags is set to CF_NONE and iWidth is greater than 1, then the circle is drawn as an outline in the Foreground color.

Errors:

This function may generate an assert if pThing or pBrush are invalid and the C/PEG library was compiled with the PEG_USE_ASSERT directive defined.

Related Functions:

PegDrawEllipse

Notes:

This function is only available if the C/PEG library was compiled with the PEG_FULL_GRAPHICS directive defined.

It is not an error if the bounding rectangle of the circle lies outside of the clipping region of the calling object or valid screen coordinates.

3.8 PegDrawEllipse

Synopsis:

```
void PegDrawEllipse(void *pThing, PegRect *pRect, PegBrush
     *pBrush);
```

Arguments:

pThing
: Pointer to a PegThing object which defines the drawing context.

pRect
: Pointer to a PegRect structure that describes the boding rectangle of the ellipse.

pBrush
: Pointer to a PegBrush object which controls the visual appearance of the ellipse.

Returns:

None

Description:

This function draws an ellipse described by pRect. It is not an error for pRect to fall outside of the caller's clipping region or the valid screen.

If the ubFlags member of the brush is set to CF_FILL, the ellipse is filled with the Background color. In this case, if the Foreground and Background colors are different and the iWidth member is greater than 1, then the ellipse is also outlined in the Foreground color.

If ubFlags is set to CF_NONE and iWidth is greater than 1, then the ellipse is drawn as an outline in the Foreground color.

Errors:

This function may generate an assert if pThing or pBrush are invalid and the C/PEG library was compiled with the PEG_USE_ASSERT directive defined.

Related Functions:

PegDrawCircle

Notes:

This function is only available if the C/PEG library was compiled with the PEG_FULL_GRAPHICS directive defined.

3.9 PegDrawEnd

Synopsis:

```
    void PegDrawEnd(void *pThing);
```

Arguments:

pThing
Pointer to a PegThing object that is completed with drawing

Returns:

None

Description:

This function is called when an object has completed drawing. It is the complement to PegDrawBegin.

Errors:

This function may generate an assert if the PegThing parameter is invalid and the C/PEG library was compiled with the PEG_USER_ASSERT directive defined.

Related Functions:

PegDrawBegin, PegEndDrawBitmap

Notes:

If this function is called without a prior balanced PegDrawBegin function call made, it is ignored.

3.10 PegDrawEndBitmap

Synopsis:

```
    void PegDrawEndBitmap(void *pThing, PegBitmap *pBitmap);
```

Arguments:

pThing
: Pointer to the object doing the drawing.

pBitmap
: Pointer to a valid PegBitmap object where drawing operations are taking place.

Returns:

None

Description:

This function ends drawing into the PegBitmap pointed to by pBitmap.

Errors:

None

Related Functions:

PegDrawBeginBitmap, PegDrawEnd

Notes:

This function must be called once for every call to PegDrawBeginBitmap. Once this function has been called a sufficient number of times, the screen object’s internal scan pointers revert to pointing to the screen. Therefore, any subsequent drawing will take place in the screen.

3.11 PegDrawLine

Synopsis:

```
    void PegDrawLine(void *pThing, PEGINT ixStart, PEGINT iyStart,
         PEGINT ixEnd, PEGINT iyEnd, PegBrush *pBrush);
```

Arguments:

pThing

Pointer to a PegThing object that provides the drawing context.

ixStart, iyStart, ixEnd, iyEnd

Starting and ending x and y coordinates that defines the position of the line. It is not mandatory for these coordinates to lie within the visible screen or the caller's clipping region.

pBrush

Pointer to a PegBrush object that is used to define the color, style and width of the line being drawn.

Returns:

None

Description:

This is the standard way for objects to draw a line within the C/PEG framework. The function determines how to handle the line drawing based on the context that is provided by the PegThing object. The coordinates can be located outside of the clipping region of the object, or even off of the visible screen. Either case will be properly clipped to prevent drawing into the region described by other objects or outside of the valid frame buffer.

The PegBrush argument defines how the line will be represented. This object defines the width and color of the line and may optionally define a pattern to be used to draw the line as well as an ancillary clipping rectangle with which to constrain drawing.

If the coordinates fall outside of the invalid region maintained by the PegScreen object, or the object is not drawable, the command will be ignored and no drawing will take place.

Errors:

If the coordinates fall outside of the invalid region maintained by the PegScreen object, the call will silently return without drawing.

The function may generate an assert if either the PegThing or PegBrush pointer are invalid and the C/PEG library was compiled with the PEG_USE_ASSERT directive defined.

Notes:

The line primitive is optimized to draw horizontal and vertical lines differently than angled lines.

3.12 PegDrawPolyLine

Synopsis:

```
void PegDrawPolyLine(void *pThing, PegPoint *pPoints, PEGUINT
    uiNumPoints, PegBrush *pBrush);
```

Arguments:

pThing
: Pointer to a PegThing object that is used as the drawing context.

pPoints
: Pointer to an array of PegPoint structures that define the coordinates used for drawing. The first point is used as the starting x and y, the second point is used as the ending coordinates of the first line and the starting coordinates of the second line and so forth.

uiNumPoints
: The number of PegPoint structures contained in the array pointed to by pPoints.

pBrush
: Pointer to a PegBrush object that described the style used to draw the lines.

Returns:

None

Description:

This function draws a series of lines based on the coordinates provided in the PegPoint array pointer parameter. Internally, this function calls PegDrawLine to do the actual drawing, therefore, the PegBrush parameters works the same way as in the PegDrawLine call.

Errors:

It is an error if the uiNumPoints is greater than the number of points passed in pPoints. This would cause the function to run past the end of the point array and cause memory problems.

If uiNumPoints is one, the function will silently return.

The function may generate an assert if the PegThing, PegPoint array or PegBrush pointers are invalid and the C/PEG library was compiled with the PEG_USE_ASSERT directive defined.

Related Functions:

PegDrawLine, PegDrawPolygon

Notes:

As with the PegDrawLine function, it is not an error if the coordinates fall outside of the clipping region defined by the calling object or outside of the visible screen.

3.13 PegDrawPolygon

Synopsis:

```
    void PegDrawPolygon(void *pThing, PegPoint *pPoints, PEGINT
        iNumPoints, PegBrush *pBrush);
```

Arguments:

pThing
: Pointer to a PegThing object that is used for the drawing context.

pPoints
: Pointer to an array of PegPoints that define the screen coordinates used to defined the outline of the polygon.

iNumPoints
: The number of PegPoints included in the array pointed to by pPoints.

pBrush
: Pointer to a PegBrush object that defines how the polygon is drawn.

Returns:

None

Description:

This function draws a polygon as defined by the coordinates passed in pPoints. The polygon may be convex or concave.

The pBrush object controls the visual appearance of the polygon. The following table illustrates the how the visual appearance of the polygon is determined by the pBrush parameter.

ubFlags	**iWidth**	**pBitmap**	**Visual Appearance**
CF_NONE	>= 1	NULL	Outlined polygon
CF_FILL	0	NULL	Filled polygon, no border
CF_FILL	>= 1	NULL	Filled polygon with border if Foreground and Background colors are different.
CF_PATTERN	0	Valid	Bitmap filled polygon, no border
CF_PATTERN	>= 1	Valid	Bitmap filled polygon with border if Foreground and Background colors are different.

Errors:

This function may generate an assert if any of the pointer parameters are invalid and the C/PEG library was compiled with the PEG_USE_ASSERT directive defined.

Related Functions:

PegDrawPolyLine

Notes:

If the polygon formed by pPoints is not closed, the function will create a new PegPoint array with the contents of pPoints plus the addition of a closing point that equals the coordinates in the first point. To avoid this overhead, the application code should ensure that the polygon is closed, meaning that the first point and the last point in the array have the same coordinates.

It is not an error for any polygon coordinate to be outside of the calling object's clipping region or beyond the screen coordinates.

This function is only available if PEG_FULL_GRAPHICS was defined for the C/PEG library build.

3.14 PegDrawRectangle

Synopsis:

```
void PegDrawRectangle(void *pThing, PegRect *pRect, PegBrush
     *pBrush);
```

Arguments:

pThing
: Pointer to a PegThing object that is used for the drawing context.

pRect
: Pointer to a PegRect structure that defines the coordinates for the rectangle to be drawn.

pBrush
: Pointer to a PegBrush object that describes the style to be used to draw the rectangle.

Returns:

None

Description:

This is the basic function for painting a rectangular region of the screen.

The PegBrush controls how the rectangle is drawn. The Foreground member of the brush defines the frame color. The Background member defines the background, or fill, color. The iWidth member determines the width of the frame and may be set to zero. If the ubFlags member of the PegBrush contains the constant CF_FILL, then the rectangle will be filled with the background color, otherwise, only the frame will be drawn.

Errors:

If the iWidth member of the PegBrush object is set to zero and the ubFlags member of the PegBrush is set to CF_NONE, then no drawing will occur.

This function may generate an assert if any of the parameters are invalid and the C/PEG library was compiled with the PEG_USER_ASSERT directive defined.

Related Functions:

PegDrawRectangleXOR, PegDrawLine

Notes:

It is not an error if any of the coordinates for the rectangle fall outside of the caller's clip region or the visible screen. Likewise, in these cases, the frame may not show if the distance the rectangle is out of these boundaries is greater than the width of the frame.

3.15 PegDrawRectangleXOR

Synopsis:

```
    void PegDrawRectangleXOR(void *pThing, PegRect *pRect);
```

Arguments:

pThing

Pointer to a PegThing object that is used for the drawing context.

pRect

Pointer to a PegRect structure that defines the coordinates for the rectangle.

Returns:

None

Description:

This function is used to draw a framed XOR rectangle bounded by pRect.

Errors:

This function may generate an assert if either parameter is invalid and the C/PEG library was compiled with the PEG_USER_ASSERT directive defined.

Related Functions:

PegDrawRectangle

Notes:

3.16 PegDrawText

Synopsis:

```
    void PegDrawText(void *pThing, PegPoint *pPoint, PEGCHAR
         *pText, PegBrush *pBrush, PegFont *pFont, PEGINT iCount);
```

Arguments:

pThing
: Pointer to a PegThing object that is used for the drawing context.

pPoint
: Pointer to a PegPoint structure that defines the top and left coordinates where drawing will begin.

pText
: PEGCHAR pointer that is a NULL terminated string to draw.

pBrush
: Pointer to a PegBrush object that determines the colors used to draw the text.

pFont
: Pointer to a PegFont structure that defines the glyphs to use for the text drawing.

iCount
: The number of characters to draw. Passing the constant PEG_TEXT_ALL draws all of the text.

Returns:

None

Description:

This function is the basic way to draw text on the screen. This function may be called by any object that wishes to display text.

The iCount parameter allows the application to determine how many characters in the string passed in pText are actually drawn. If the number of characters is unknown, and the string is NULL terminated, the caller may pass PEG_TEXT_ALL in this parameter and the function will draw the entire string. This saves the caller from unnecessarily calling a string library function just to discover the length of the string.

Errors:

This function may generate an assert if any of the pointer parameters are invalid and the C/PEG library was compiled with the PEG_USE_ASSERT directive defined.

Passing an iCount of zero, while not an error, will not produce any output.

Related Functions:

PegTextHeight, PegTextWidth

Notes:

Every PegThing derived objects that display text use this function to draw text on the screen.

The ubFlags member in the pBrush parameter determine if the background rectangle that contains the text drawing will be filled or not. To fill the background, set ubFlags to CF_FILL, otherwise, set it to CF_NONE. If the background is not to be filled, it is wise for the application code to invoke drawing of the object through it's parent, which will paint over the object and allow it to draw the text on a clean slate. A common error is to set the Foreground and Background colors the same when using CF_FILL. This, obviously, results in a rectangle drawn in the background color, with invisible text.

3.17 PegPaletteGet

Synopsis:

```
    PEGCOLORVAL *PegPaletteGet(PEGULONG *pSize);
```

Arguments:

pSize
: Pointer to a PEGULONG that receives the size of the palette.

Returns:

A pointer to the start of the palette data.

Description:

This function returns a pointer to the currently installed palette. The application may call this function to save off the palette before installing a custom palette using PegPaletteSet.

Errors:

None

Related Functions:

PegPaletteSet, PegPaletteReset.

Notes:

The parameter pSize is set to an integer value that describes the size of the palette. When running on color depths of 8 bit per pixel and lower, this number represents the total size, in bytes, of the palette. Each color entry in this type of palette is comprised of 3 bytes of data, one each for the red, blue and green values that make up the color. Therefore, pSize will be 3 times the number of entries in the palette.

When running at 16 bits per pixel, each color value in the palette is made up of a 16 bit unsigned short value which represents the red, blue and green components in the packed pixel format used by the platform (5:5:5 mode or 5:6:5 mode). Therefore, pSize will equal the number of entries in the palette.

3.18 PegPaletteSet

Synopsis:

```
    void PegPaletteSet(PEGINT iFirst, PEGINT iNum, PEGUBYTE *pPal);
```

Arguments:

iFirst
    Denotes the first index into the palette.
iNum
    How many palette indices to update.
pPal
    Pointer to new palette data.

Returns:

None

Description:

This function updates the system palette. It begins with palette index iFirst and updates iNum number of entries in the palette.

pPal points to a buffer of RGB triplet data. The data should always be in the format of one byte for each color triplet (1 byte for red, 1 byte for green and 1 byte for blue). Therefore, one palette entry would comprise three bytes of incoming RGB data. This is true regardless of the color depth of the target. If the incoming data is not arranged in this manner, the results will be unpredictable.

For color depths of 8 bit per pixel and lower, the palette entries are written to the video hardware as well as stored locally. For a color depth of 16 bits per pixel, the palette data is not written to the video hardware and only stored locally. The palette in 16 bit per pixel mode is used only for mapping 8 bit per pixel bitmap data to a corresponding 16 bit color value.

Errors:

None

Related Functions:

PegPaletteGet, PegPaletteReset.

Notes:

It is permissible for the incoming palette data to not be persistent. The screen object maintains a local copy of the palette data, and, therefore, does not keep a pointer to the incoming palette data.

When running in 16 bit per pixel mode, this palette is only used for drawing 8 bit per pixel bitmaps, and does not effect 16 bit per pixel bitmap drawing.

3.19 PegPaletteReset

Synopsis:

```
    void PegPaletteRest(void);
```

Arguments:

None

Returns:

Nothing.

Description:

When the video hardware is initially configured, a default palette is installed. During the course of running, the application may choose to install a custom palette to display a particular bitmap or to perform color rotation. At any time, the display may be restored to it’s original palette by calling this function.

Errors:

None

Related Functions:

PegPaletteSet, PegPaletteGet.

Notes:

3.20 PegPixelGet

Synopsis:

```
PEGCOLORVAL PegPixelGet(void *pThing, PEGINT x, PEGINT y);
```

Arguments:

pThing
    Pointer to a PegThing or derived object.

x
    Horizontal coordinate

y
    Vertical coordinate

Returns:

The color value of the pixel at coordinate x,y. If the values of x or y lie outside of the screen rectangle, then a value of 0 is returned.

Description:

This is a simple function to safely retrieve the color data for a given pixel at coordinate x,y.

Errors:

None

Related Functions:

PegPixelSet

Notes:

3.21 PegPixelSet

Synopsis:

```
    void PegPixelSet(void *pThing, PEGINT x, PEGINT y, PEGCOLORVAL
        c);
```

Arguments:

pThing
    Pointer to a PegThing or derived object.

x
    Horizontal coordinate

y
    Vertical coordinate

c
    The color value to write

Returns:

Nothing

Description:

This function sets a pixel at the x,y coordinate to the value of c. The pixel will not be written if any of the following are true: x or y fall outside of the area of the screen owned by pThing; pThing is not drawable; there is no invalid region of the screen.

Errors:

None

Related Functions:

PegPixelGet, PegDrawLine, PegDrawRectangle

Notes:

This function is convenient to use when only setting a few pixels to a certain color. Since the drawing has to be clipped and verified for each call, it is not advised to call this function to draw lines or to fill areas of the screen. It is much more efficient to call PegDrawLine or PegDrawRectangle for large operations since all of the clipping and verification is done only once for the entire operation.

3.22 PegPointerPosGet

Synopsis:

```
    void PegPointerPosGet(PegPoint *pPoint);
```

Arguments:

pPoint

Pointer to a PegPoint structure to receive the coordinates.

Returns:

None

Description:

This function returns the current x and y coordinates of the mouse pointer in pPoint.

Errors:

None

Related Functions:

PegPointerPosSet

Notes:

3.23 PegPointerPosSet

Synopsis:

```
void PegPointerPosSet(PegPoint *pPoint);
```

Arguments:

pPoint
: Pointer to a PegPoint structure.

Returns:

None

Description:

This function will programatically ‘warp’ the mouse pointer to the position specified in pPoint.

Errors:

None

Related Functions:

PegPointerPosGet

Notes:

This function locks the screen resource since it has to modify the frame buffer. Be careful when calling this function from an ancillary thread or task.

3.24 PegPointerShow

Synopsis:

```
    void PegPointerShow(PEGBOOL bShow);
```

Arguments:

bShow

Set to TRUE to show the pointer, FALSE to hide the pointer.

Returns:

None

Description:

This function shows or hides the mouse pointer.

Errors:

None

Related Functions:

PegPointerTypeAdd, PegPointerTypeGet, PegPointerTypeSet

Notes:

3.25 PegPointerTypeAdd

Synopsis:

```
    PEGUBYTE PegPointerTypeAdd(PegBitmap *pBitmap, PEGUINT
        uiXOffset, PEGUINT uiYOffset);
```

Arguments:

pBitmap
: PegBitmap pointer which describes the image that will be used for the pointer.

uiXOffset
: Horizontal offset of the hot spot.

uiYOffset
: Vertical offset of the hot spot.

Returns:

A value between PEG_SYS_POINTER_TYPES and PEG_NUM_POINTER_TYPES. If the pointer can not be added, then 0 is returned.

Description:

This function allows the application to add a new mouse cursor bitmap to the array maintained by the library. By default, 5 new pointer types may be added to the array by the application. This number is controlled by the PEG_USER_POINTER_TYPES directive. The application is free to define this themselves to a different number, but, bear in mind that each entry in the pointer array occupies up to 8 bytes of data space on the final target.

A value other than 0 returned by this call may then be used to cause the system to switch mouse pointer bitmaps by calling PegPointerTypeSet.

The x and y offsets are used to determine the hotspot of the mouse. In other words, these numbers figure in to the actual x and y values reported by C/PEG to the application in button click mouse messages. The common 'Arrow' bitmap that is most often used has a hot spot of 0, 0; meaning that the hot spot is the top, left corner of the bitmap. Contrast this with a 'Cross Hair' mouse pointer which would have the offsets set to put the hotspot directly in the middle of the bitmap.

Errors:

None

Related Functions:

PegPointerTypeGet, PegPointerTypeSet.

Notes:

3.26 PegPointerTypeGet

Synopsis:

```
PEGUBYTE PegPointerTypeGet(void);
```

Arguments:

None.

Returns:

The index of the currently selected pointer type.

Description:

This function allows the application to discover which pointer type is currently active.

Errors:

None

Related Functions:

PegPointerTypeSet

Notes:

3.27 PegPointerTypeSet

Synopsis:

```
void PegPointerTypeSet(PEGUBYTE ubType);
```

Arguments:

ubType
Index of the pointer to set.

Returns:

None

Description:

This function sets the bitmap used to display the mouse cursor. Acceptable values for ubType are any of the system types listed in the table below, or an index value returned from a successful call to PegPointerTypeAdd.

The following table lists the system pointer index defines.

Pointer Index	Pointer Type
PPT_NORMAL	Arrow
PPT_VSIZE	Double arrow oriented vertically
PPT_HSIZE	Double arrow oriented horizontally
PPT_NWSE_SIZE	Double arrow angled toward the top/ left
PPT_NESW_SIZE	Double arrow angled toward the top/ right
PPT_IBEAM	I-Beam used for edit fields
PPT_HAND	Hand selection

Errors:

None

Related Functions:

PegPointerTypeAdd, PegPointerTypeGet

Notes:

3.28 PegRectRegionCapture

Synopsis:

```
    void PegRectRegionCapture(PegCapture *pCapture, PegRect
         *pRect);
```

Arguments:

pCapture
: Pointer to a PegCapture structure.

pRect
: Pointer to a PegRect structure.

Returns:

None

Description:

This function captures an area of the screen described by pRect and saves the data to the PegCapture structure pointed to by pCapture. The PegBitmap structure embedded in pCapture can then be used to draw to another portion of the screen, or as the target for off screen bitmap drawing.

Errors:

None

Related Functions:

PegRectRegionRestore

Notes:

This function allocates enough memory to hold the captured area of the screen. The memory is pointed to by the PegBitmap structure embedded in pCapture. It is the responsibility of the caller to free this memory.

3.29 PegRectRegionMove

Synopsis:

```
    void PegRectRegionMove(void *pThing, PegRect *pRect, PegRect
         *pClip, PEGINT xshift, PEGINT yshift);
```

Arguments:

pThing
    Pointer to a PegThing object used as the drawing context.
pRect
    Pointer to a PegRect structure which describes the source area of the screen to move.
pClip
    Pointer to a PegRect structure which describes the clip area for the target area.
xshift
    The amount to shift the source area along the horizontal axis.
yshift
    The amount to shift the source area along the vertical axis.

Returns:

None

Description:

This function does the work of PegRectRegionCapture and PegRectRegionRestore all in one function call.

Errors:

None

Related Functions:

PegRectRegionCapture, PegRectRegionRestore

Notes:

This function allocates and frees the necessary memory to capture and restore the bitmap so the caller is not responsible for this. The destination rectangle will always be clipped to both the caller's pThing and the pClip rectangle. The pClip rectangle may be used if the caller wishes to clip the target rectangle to an area other than the clip region owned by pThing. pClip can not be NULL.

3.30 PegRectRegionRestore

Synopsis:

```
    void PegRectRegionRestore(void *pThing, PegCapture *pCapture,
         PEGBOOL bOnTop);
```

Arguments:

pThing
    Pointer to a PegThing object used as the drawing context.

pCapture
    Pointer to a PegCapture structure.

bOnTop
    Boolean to control how the capture bitmap is restored.

Returns:

None

Description:

This function restores an area of the screen that was previously captured by the PegRectRegionCapture function. The pCapture holds the bitmap to restore.

Errors:

None

Related Functions:

PegRectRegionCapture, PegDrawBitmap

Notes:

This function does not free the memory associated with the bitmap in pCapture. It is the responsibility of the caller to free this memory.

CHAPTER 4

C/PEG LIBRARY FUNCTIONS

4.1 Overview

This collection of functions do not necessarily deal with a particular object or structure type.

Some of the functions are informational in that they return data about the system on which the application is running, PegColorDepthGet would be an example of this type of function.

Other functions do relate to PegThing objects, but are generic enough that they are listed here. The PegStyleAdd and PegStyleHas functions are examples of this.

There is also a group of functions that deal with the mouse pointer. Which object owns the pointer, where the pointer is located on the screen as well as capturing and releasing the pointer.

Finally there are functions that return pointers to the PegPresentation, PegScreen and the main task's PegMessageQueue objects that may be used in application code.

4.2 PegCenterOf

Synopsis:

```
    void PegCenterOf(void *pThing, PegPoint *pPoint);
```

Arguments:

pThing

Pointer to a PegThing or derived object.

pPoint

Pointer to a PegPoint structure.

Returns:

Returns the center coordinates for pThing in pPoint.

Description:

This function calculates the center of the rectangular region occupied by pThing and returns the coordinates in pPoint.

The object pointed to by pThing does not have to be visible on the screen.

Errors:

This function may generate an assert if either pointer is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegCenterOver

Notes:

4.3 PegCenterOver

Synopsis:

```
    void PegCenterOver(void *pThing, void *pCenter);
```

Arguments:

pThing
    Pointer to a PegThing or derived object.
pCenter
    Pointer to a PegThing or derived object.

Returns:

None

Description:

This function centers pCenter over pThing. pCenter does not have to be a child of pThing. Any PegThing derived object may be centered over any other PegThing derived object.

Errors:

This function may generate an assert if either pointer is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegCenterOf

Notes:

4.4 PegChildrenNotify

Synopsis:

```
void PegChildrenNotify(void *pParent, const PegMessage *pMesg);
```

Arguments:

pParent
    Pointer to a PegThing or derived object.
pMesg
    Pointer to a PegMessage structure that contains the message data.

Returns:

None

Description:

This function is an easy way for a PegThing or derived object to pass a message to all of it's children.

Errors:

This function may generate an assert if pParent is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegNotify, PegThingNotify

Notes:

4.5 PegChildrenShift

Synopsis:

```
void PegChildrenShift(void *pParent, PEGINT ixShift, PEGINT
  iyShift, PEGBOOL bRedraw);
```

Arguments:

pParent
: PegThing pointer who is the parent of the child objects to shift.

ixShift
: The amount to shift along the horizontal axis.

iyShift
: The amount to shift along the vertical axis.

bRedraw
: Boolean to determine whether to immediately redraw or not.

Returns:

None

Description:

This function is the preferred method for scrolling the children of pParent.

The function first invalidates the area occupied by pParent and repositions all of pParent’s children. If bRedraw is FALSE, then the function returns. If not, drawing is accomplished one of two ways. If PEG_FAST_BLIT in defined, the function PegRectRegionMove is used to move the portions of pParent’s client region which will still be visible after applying the horizontal and vertical shift. It then only redraws the portion of the client area which has been newly exposed. If PEG_FAST_BLIT it not turned on, then pParent is entirely redrawn.

It is often much faster, even without hardware acceleration to use PegRectRegionMove to draw the portions of the screen that remain visible and have just been moved. With hardware acceleration, it is often very fast and a trivial system load.

Errors:

None

Related Functions:

PegRectRegionMove, PegRectRegionInvalidate, PegResize

Notes:

See the scroll example in your C/PEG distribution for a demonstration on how to use this function in quickly scrolling objects.

4.6 PegClientInit

Synopsis:

```
    void PegClientInit(void *pThing);
```

Arguments:

pThing

Pointer to a PegThing or derived object.

Returns:

None

Description:

This function initializes the Client and Clip rectangles of pThing based on the object's frame style.

This function should be called on an object after the style and Real members have been set.

This function should not be used on an object while it is visible.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

None

Notes:

This function is usually called during the creation process of an object.

4.7 PegColorDepthGet

Synopsis:

```
PEGUBYTE PegColorDepthGet(void);
```

Arguments:

None

Returns:

The color depth at which the system is currently running.
Valid returns values are:

- 1
- 2
- 4
- 8
- 16

Description:

This function returns the color depth, or bits per pixel, of the system.

Errors:

None

Related Functions:

PegNumColorsGet

Notes:

4.8 PegColorGet

Synopsis:

```
PEGCOLORVAL PegColorGet(void *pThing, PEGUBYTE ubIndex);
```

Arguments:

pThing

    Pointer to a PegThing or derived object.

ubIndex

    Color index to return.

Returns:

The color value associated with the index described by ubIndex. If ubIndex is less than 0 or greater than PEG_THING_COLOR_LIST_SIZE, 0 is returned.

Description:

This function retrieves the color value at color index ubIndex of pThing.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegColorSet

Notes:

On most platforms, the error return value of 0 is often a legitimate color value, usually mapped to BLACK.

4.9 PegColorSet

Synopsis:

```
    void PegColorSet(void *pThing, PEGUBYTE ubIndex, const
      PEGCOLORVAL c);
```

Arguments:

pThing

Pointer to a PegThing or derived object.

ubIndex

Color index in which to store the color value.

c

Color value to store.

Returns:

None

Description:

This function is used to set a particular color index in pThing with PEGCOLORVAL c.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegColorGet

Notes:

If ubIndex is less than 0 or greater than PEG_THING_COLOR_LIST_SIZE, the function silently does nothing and returns.

4.10 PegCurrentThingGet

Synopsis:

```
    PegThing *PegCurrentThingGet(void);
```

Arguments:

None

Returns:

Returns a pointer to the object that currently has input focus. It is possible for this to be NULL if no object that can accept input focus currently has focus.

Description:

This function returns a pointer to the current object that has input focus. The object that has input focus is the current object on the screen to which all key messages are routed.

Errors:

None

Related Functions:

PegFocusTreeMove

Notes:

The PegPresentation object keeps track of which object on the screen currently has input focus. If there are no objects on PegPresentation that are on the current object tree that can accept input, then this function will return NULL.

4.11 PegDefaultFontGet

Synopsis:

```
    PegFont *PegDefaultFontGet(PEGUBYTE ubIndex);
```

Arguments:

ubIndex

Index of the font type.

Returns:

Upon success, a pointer to a PegFont structure that coincides to the index type. If the type is out of bounds, the default font pointer is returned.

Description:

This function is the compliment to PegDefaultFontSet.

Errors:

If ubIndex is out of array bounds for the system font table, the default font is returned.

Related Functions:

PegDefaultFontSet

Notes:

See the Notes section in PegDefaultFontSet

4.12 PegDefaultFontSet

Synopsis:

```
void PegDefaultFontSet(PEGUBYTE ubIndex, PegFont *pFont);
```

Arguments:

ubIndex
: Zero based index value to offset into the C/PEG font table.

pFont
: Pointer to a PegFont structure.

Returns:

None

Description:

This function set the C/PEG system's font table index to point to pFont.

Errors:

ubIndex must be less than PEG_NUM_FONTS.

This function may generate an assert if pFont is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegDefaultFontGet

Notes:

For every object in C/PEG that derives from PegTextThing, there is a corresponding entry to that object in a small table that is used to assign the default fonts to an object of a given type. This is a simple way to enforce a standard look across the application and individual objects.

This also provides an easy way for the application designer to set the default font used by a particular object type, so every object of that type that is created after the default font has been modified, will automatically begin using the new font.

The following table lists the built in font types and the object type to which they correspond.

Font Index Identifier	Referring Object Type
PEG_DEFAULT_FONT	Any text object that does not have a specific index for that object type.
PEG_TITLE_FONT	PegTitle
PEG_TEXT_BUTTON_FONT	PegTextButton, PegMLTextButton, PegDecoratedButton
PEG_RADIO_BUTTON_FONT	PegRadioButton
PEG_CHECKBOX_FONT	PegCheckBox
PEG_PROMPT_FONT	PegPrompt, PegMLPrompt, PegVertPrompt
PEG_EDIT_FIELD_FONT	PegEditField
PEG_GROUP_FONT	PegGroup
PEG_ICON_FONT	PegIcon
PEG_PANEL_FONT	PegPanel, PegStatusPanel
PEG_MESSAGE_PANEL_FONT	PegMessagePanel

C/PEG provides a mechanism for the application designer to add new fonts to the system font table by defining PEG_SPARE_FONT_INDEXES to a number greater than 0, which is the default. This define can be found in the cpeg/include/pfonts.h file.

The PEG_NUM_FONTS define is the number of system fonts plus PEG_SPARE_FONT_INDEXES. PEG_NUM_FONTS is used to allocate the storage for the system font table.

Once the number of fonts has been modified, the application designer can then define new font indices for the new object types. These defines should begin at PEG_NUMBER_OF_DEFAULT_FONTS and enumerate up from there.

4.13 PegDrawChildren

Synopsis:

```
    void PegDrawChildren(void *pThing);
```

Arguments:

pThing
Pointer to a PegThing or derived object.

Returns:

None

Description:

This function iterates through each child of pThing and calls the child's Draw function. Most every object's Draw function calls this directly or indirectly by deferring the call to it's immediate base object, which, in turn, ends up at PegThingDraw which calls this function.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegThingDraw

Notes:

This function is called by PegThingDraw. All C/PEG objects expect that they may have children, so every object calls this function in some way.

4.14 PegFind

Synopsis:

```
    PegThing *PegFind(void *pStart, PEGUSHORT usId, PEGBOOL
      bRecurse);
```

Arguments:

pStart
: Pointer to a PegThing or derived object where the find operation begins.

usId
: ID of the object to find.

bRecurse
: Optionally recurs the search through the children of pStart.

Returns:

A pointer to the first object whose ID matches usId, or NULL if no children matched.

Description:

This function searches through the children of pStart and compares the child's ID with usId. The first match that is found completes the search and the pointer is returned.
If bRecurse is TRUE, the find operation first exhausts the branch below each child before moving on to the next child.

Errors:

This function may generate and assert if pStart is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegChildFirst, PegChildNext, PegParent

Notes:

Although a globally unique ID for every object in use by an application is sometimes impossible, if the application relies heavily on this function for finding child objects, it would be in the best interest of the application developer to make every effort to do so.

4.15 PegFocusTreeMove

Synopsis:

```
    void PegFocusTreeMove(void *pThing);
```

Arguments:

pThing
    Pointer to a PegThing or derived object.

Returns:

None

Description:

This function may be used by the application to manually move focus to a particular object on the screen. This will also give pThing input focus, if so appropriate.

Errors:

This function may generate and assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegCurrentThingGet

Notes:

If the object that had focus before this call and pThing are on different branches of the object tree belonging to PegPresentation, this function removes focus from every object in the old branch and ensures that focus is properly moved to the new branch.

4.16 PegFuncPtrSet

Synopsis:

```
PEGBOOL PegFuncPtrSet(void *pThing, PEGINT iFuncID, ...);
```

Arguments:

pThing

Pointer to a PegThing derived object.

iFuncID

Index ID of the function pointer in the object to set.

Returns:

If iFuncID is a valid index for a function pointer in the pThing object, then TRUE is returned, otherwise, FALSE is returned.

Description:

This function sets a specific function pointer in pThing to point to pFunc. iFuncID controls which function pointer in pThing is set to pFunc.

Errors:

If iFuncID does not refer to a function pointer index supported by the object, the function fails and returns.

Related Functions:

None

Notes:

In every PegThing derived object, there are a number of function pointers handle calling the proper function for the type of object. PegTextThing adds two more function pointers and PegPanel adds one more function pointer.

A list of the function pointer index names and the corresponding function pointer are listed in the table below.

Function Index	Supported Object	Function
PFP_NOTIFY	PegThing	Notify
PFP_DRAW	PegThing	Draw
PFP_ADD	PegThing	Add
PFP_ADDTOEND	PegThing	AddToEnd
PFP_REMOVE	PegThing	Remove

Function Index	Supported Object	Function
PFP_DESTROY	PegThing	Destroy
PFP_RESIZE	PegThing	Resize
PFP_DRAWBORDER	PegThing	DrawBorder
PFP_ERASEFOCUS	PegThing	EraseFocus
PFP_DRAWFOCUS	PegThing	DrawFocus
PFP_PARENTSHIFT	PegThing	ParentShift
PFP_FONTSET	PegTextThing	FontSet
PFP_TEXTSET	PegTextThing	TextSet
PFP_EXECUTE	PegPanel	Execute

For any given object that inherits from PegThing, it is only valid to call this function using function pointer indices that are listed here as supported by PegThing. Likewise for PegTextThing. And, since PegPanel derives from PegTextThing, any of these function indices are acceptable.

It is strongly suggested that the application code use this function to set function pointers in custom objects. The name of a function pointer in a structure may change during C/PEG development, and any application code that directly touched that function pointer directly would certainly break.

This function provides a thin layer of encapsulation of the function pointers in various structures, and the application code should take advantage of that.

4.17 PegLineClipToRect

Synopsis

```
    PEGBOOL*PegLineClipToRect(PEGINT *px1, PEGINT *py1, PEGINT
      *px2, PEGINT *py2, PegRect *pRect
```

Arguments

px1,py1, px2, py2

Pointers to integers describing the two points that make up the line segment.

pRect

Point to a PegRect which describes the rectangle to use for clipping the line segment.

Returns

Returns TRUE if the line was successsfully clipped, or if the original line segment was already encompassed by the area described by the rectangle. If the line segments falls outside of the rectangle, then FALSE is returned.

If the line can be clipped, then px1, py1, px2, and py2 will be modified and contain the new end points to the line.

Description

This function is used throughout the library to clip line segments for drawing. This allows for fast line drawing since the end points are clipped before the actual drawing begins. This significantly speeds up angled line drawing algorithms. It is also possible to call this functrion from user code in order to clip lines if necessary.

Errors

This function may generate an assert if pRect is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions

PegLtoU

Notes

It is important to remember px1, py1, px2, and py2 will be modified by this function if the line was successfully clipped.

4.18 PegLtoA

Synopsis:

```
char *PegLtoA(long lval, char *s, int rad);
```

Arguments:

lval

Integer value to convert to a NULL terminated string.

s

Pointer to a char array to hold the converted value of lval.

rad

Numerical radix in which lval is to be represented.

Returns:

A pointer to the converted string.

Description:

Converts a long integer value to a null-terminated string using the specified radix and stores the result in the given buffer. If radix is 10 and value is negative the string is preceded by the minus sign (-). With any other radix, value is always considered unsigned. The buffer should be large enough to contain any possible value: (sizeof(long)*8+1) for radix=2.

Internally, the C/PEG library always calls PegLtoA when converting a integral value to a string. If PEG_LTOA is not defined, then PegLtoA is mapped to _ltoa, which is often provided by the compiler's library.

Errors:

It is important that the caller allocates a buffer large enough to hold the string conversion of lval, plus the NULL terminator, otherwise a buffer overrun may occur.

This function may generate an assert if s is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegLtoU

Notes:

This function is not part of the ANSI C specification, but it is supported by a number of compilers. For compilers which do not support this function, the

C/PEG version of the function may be included in the library by defining PEG_LTOA in the pconfig.h file.

The C/PEG version of this function has two forms. One that deals with the ordinary ANSI character set, and a second version that deals properly with strings that are in wide byte format for when the library is compiled for Unicode support.

The PegProgressBar and PegSpinButton objects use this function to convert integer data to a string.

4.19 PegMessageQueueGetByTask

Synopsis:

```
PegMessageQueue *PegMessageQueueGetByTask(PEG_TASK_TYPE task);
```

Arguments:

task

A task identifier.

Returns:

If the task, or thread, denoted by the task argument currently has a unique PegMessageQueue object associated with it, this function returns a pointer to the PegMessageQueue in use by the task. Otherwise, it returns NULL.

Description:

This function queries the tasks currently executing under the watch of PegPresentation. Typically, this list of tasks of which PegPresentation is aware are separate tasks that are currently executing a modal PegPanel or derivative. If a task is found that matches the passed in task, then the PegMessageQueue associated with that task is returned.

Errors:

None

Related Functions:

PegMessageQueueGetByThing

Notes:

This function is not normally called from application code. Typically, any task's message queue may be reached by a PegMessage by simply pushing the message into the default system message queue. It is really transparent to the application that a particular PegPanel object may have a message queue separate from the system message queue. PegPresentation handles routing messages to tasks on it's own.

This function is only available if the C/PEG library was built with the PEG_MULTITHREAD directive defined.

4.20 PegMessageQueueGetByThing

Synopsis:

```
PegMessageQueue *PegMessageQueueGetByThing(void *pThing);
```

Arguments:

pThing

Pointer to a PegThing or derivative object.

Returns:

If pThing is associated with a unique PegMessageQueue, then a pointer to the message queue is returned, otherwise, NULL is returned.

Description:

This function queries the tasks currently executing under the watch of PegPresentation. Typically, this list of tasks of which PegPresentation is aware are separate tasks that are currently executing a modal PegPanel or derivative. If an object is found that matches pThing, then the PegMessageQueue associated with that object is returned.

Errors:

None

Related Functions:

PegMessageQueueGetByThing

Notes:

This function is not normally called from application code. Typically, any task's message queue may be reached by a PegMessage by simply pushing the message into the default system message queue. It is really transparent to the application that a particular PegPanel object may have a message queue separate from the system message queue. PegPresentation handles routing messages to tasks on it's own.

This function is only available if the C/PEG library was built with the PEG_MULTITHREAD directive defined.

4.21 PegMessageQueuePtr

Synopsis:

```
    PegMessageQueue *PegMessageQueuePtr(void);
```

Arguments:

None

Returns:

Returns a pointer to the C/PEG system message queue.

Description:

This function is used in application code to obtain a pointer to the default C/PEG system message queue. The pointer returned from this call may be used by application calls to specify a PegMessageQueue in which to push messages.

Errors:

This function may generate an assert if the system message queue is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegMessageQueuePush, PegPresentationPtr, PegMessageQueueGetByTask, PegMessageQueueGetByThing

Notes:

This function will always return the default system message queue. In stand alone operation, this will always be the only message queue used by the system. In multi-threaded mode, there may be more than one message queue in use at a time. If the application must obtain a pointer to a message queue in use by a modally executing PegPanel, then the application should not use this function, but instead use PegMessageQueueGetByTask or PegMessageQueueGetByThing, both of which are only available if PEG_MULTITHREAD was defined for the C/PEG library build.

4.22 PegMoveToFront

Synopsis:

```
void PegMoveToFront(void *pParent, void *pMove, PEGBOOL
  bRedraw);
```

Arguments:

pParent
: Pointer to a PegThing or derived object.

pMove
: Pointer to a PegThing or derived object.

bRedraw
: Optionally redraw pParent after the move operation has completed.

Returns:

None

Description:

This function moves pMove to the front of the child list owned by pParent. The object pointed to by pMove will only be moved as far up the child list as permitted by it's status. In other words, if pMove does not have PSF_ALWAYS_ON_TOP status, and other children of pParent do, then pMove will be placed directly after the last child object that does have PSF_ALWAYS_ON_TOP status.

Errors:

This function may generate an assert if either pointer is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegAdd, PegAddToEnd

Notes:

This function works very similar to calling PegAdd for an object that is already a child of the parent. But, this function does not move the tab order around as PegAdd does.

4.23 PegNumColorsGet

Synopsis:

```
    PEGULONG PegNumColorsGet(void);
```

Arguments:

None

Returns:

Returns the number of colors at which the system is currently running.

Valid return values are:

- 1
- 2
- 4
- 16
- 256
- 65535

Description:

This function returns the number of colors currently used by the system. The return value from this will often match the constant PEG_NUM_COLORS, but, on some systems, the color depth is configurable at run time, so these values may differ. It is best to use the return value from this function for determining the number of colors supported by the system.

Errors:

None

Related Functions:

PegColorDepthGet

Notes:

4.24 PegPointerCapture

Synopsis:

```
void PegPointerCapture(void *pThing);
```

Arguments:

pThing

Pointer to a PegThing or derived object.

Returns:

None

Description:

This function captures all pointer messages and directs them to pThing. Calls to this function may be nested by objects that belong to the same task.

If the pointer is captured by an object of a task, then an object in a separate task calls this function to capture the pointer, the second task will block until the first task has released the pointer.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegPointerRelease

Notes:

It is important to remember in application code to balance calls to PegPointerCapture with calls to PegPointerRelease. Multi-threaded applications will not work properly if these calls are out of balance.

4.25 PegPointerIsCaptured

Synopsis:

```
    PEGBOOL PegPointerIsCaptured(void);
```

Arguments:

None

Returns:

Returns TRUE if the pointer is currently captured, otherwise returns FALSE.

Description:

This function allows the application to query if any object currently owns the pointer.

Errors:

None

Related Functions:

PegPointerCapture, PegPointerRelease, PegPointerOwnerGet

Notes:

4.26 PegPointerLastOverGet

Synopsis:

```
    PegThing *PegPointerLastOverGet(void);
```

Arguments:

None

Returns:

Returns a pointer to a PegThing or derived object. It is not an error for this function to return NULL.

Description:

This function returns a pointer to an object that occupies the screen region over which the mouse pointer is positioned.

Errors:

None

Related Functions:

PegPointerOwnerGet

Notes:

4.27 PegPointerOwnerGet

Synopsis:

```
    PegThing *PegPointerOwnerGet(void);
```

Arguments:

None

Returns:

Returns a pointer to the object that currently has the mouse pointer captured. If no object currently has the pointer captured, NULL is returned.

Description:

This function allows the application to discover which object currently has the mouse pointer captured.

Errors:

None

Related Functions:

PegPointerCapture, PegPointerRelease

Notes:

4.28 PegPointerRelease

Synopsis:

```
    void PegPointerRelease(void *pThing);
```

Arguments:

pThing

Pointer to a PegThing or derived object that is releasing the mouse pointer.

Returns:

None

Description:

This function releases the mouse pointer from pThing. This is the complement of PegPointerCapture and must be called exactly once for each call of the object to PegPointerCapture.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegPointerCapture

Notes:

C/PEG allows pointer captures to be nested for objects that are executing within the same task. If an object from a separate task attempts to capture the pointer, the task will block until the pointer is completely released by all of the objects in the task which currently owns the pointer.

4.29 PegPresentationPtr

Synopsis:

```
PegPresentation *PegPresentationPtr(void);
```

Arguments:

None

Returns:

Returns a pointer to the PegPresentation object.

Description:

This function is used by application code to obtain a pointer to the PegPresentation object.

Errors:

None

Related Functions:

PegPresentationPtrSet, PegMessageQueuePtr

Notes:

This function is used often by the application to add top level objects to the PegPresentation object. The following code snippet is a short example of how to use this function:

```
PegAdd(PegPresentationPtr(), MyObjectCreate(), TRUE);
```

If the application needs to continually make use of this pointer over the span of a function, it is best to call this function once and keep a local pointer to the PegPresentation object to alleviate the overhead of the function call.

The caveat to this is that the call itself is protected by serialization objects, so no two tasks can access the PegPresentation pointer simultaneously. But, there is no protection against one task replacing the PegPresentation object, thus making a pointer that another task may have invalid. This is a matter for the application developer. It is not recommended for the PegPresentation object to be replaced during normal operation of a system. It is recommended for the PegPresentation object to be replaced at the start of the application before any ancillary tasks are created or children added to the PegPresentation object.

4.30 PegPresentationPtrSet

Synopsis:

```
    PegPresentation *PegPresentationPtrSet(PegPresentation
      *pPresent);
```

Arguments:

pPresent

Pointer to a PegPresentation or derivative object.

Returns:

Returns a pointer to the PegPresentation object in use at the time this function was called.

Description:

There are situations where replacing the default PegPresentation object may be an appropriate course of action for a system. This function does allow the application developer to replace the PegPresentation object that was installed at the time the library initialized the basic objects of the C/PEG library at run time.

Errors:

This function may generate an assert if pPresent is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegPresentationPtr, PegAppInitialize

Notes:

If an application designer decides that the application cannot function properly with the default PegPresentation object, and commits to creating a custom PegPresentation object for the application, then it is best to replace the PegPresentation object as the first order of business in the PegAppInitialize function, before any objects are added to the current PegPresentation object.

It is also important to note that this function does not destroy the existing PegPresentation object, it only modifies the internal pointer that points to the PegPresentation object in use. It is the responsibility of the caller to free the instance of the PegPresentation object that is returned from this call when it is appropriate to do so.

4.31 PegRectRegionInvalidate

Synopsis:

```
    void PegRectRegionInvalidate(PegRect *pRect);
```

Arguments:

pRect

Pointer to a PegRect structure that defines the region to be invalidated.

Returns:

None

Description:

This function call informs the PegScreen that an area of the screen is invalid and is in need of updating.

Much of this work is done within the library itself in response to system events such as new objects being added to visible objects or moving an object around. In these cases, C/PEG calls into application code to update the objects that are within the invalid rectangle.

When an object's color or text are updated, the rectangular invalid region of the screen is updated to include the rectangle occupied by the modified object. In these instances, C/PEG does not force an automatic update of the modified objects. This is so many objects may be updated at once, then all visibly updated together, saving unnecessary re-draws.

The PegScreen maintains this invalid region on behalf of the entire system. Therefore, this call is thread protected. This protection means that the thread calling this function should be ready to draw, because once this function is called by the thread, that thread owns the screen resource and no other threads are allowed to draw. The screen resource is not released from the thread until the PegDrawBegin/PegDrawEnd stack unwinds. What this means is that ancillary drawing, meaning any drawing that may take place that is not a direct result of C/PEG calling into the application to draw, must invalidate then draw as soon as possible to ensure that the application does not reach a deadlock condition.

A common error in the application code is to change an object's text and not tell the object to redraw itself. The reason why this is an error is because the call to set an object's color, PegColorSet, calls

PegRectRegionInvalidate on behalf of the object that is being modified. In turn, what this does, is lock the screen resource. Even though, to the caller, this is not readily apparent.

Errors:

This function may generate an assert if the PegScreen object is invalid and the C/PEG library was compiled with the PEG_USER_ASSERT directive defined.

Related Functions:

PegDrawBegin, PegDrawEnd

Notes:

4.32 PegScratchPadGet

Synopsis:

```
PEGCHAR *PegScratchPadGet(void);
```

Arguments:

None

Returns:

The contents of the system scratch pad.

Description:

This function returns a pointer to a NULL terminated string that is maintained by C/PEG as a scratch pad. If the scratch pad is empty, this function returns NULL.

Errors:

None

Related Functions:

PegScratchPadSet

Notes:

C/PEG maintains a scratch pad that may contain a single, NULL terminated string, or may be NULL. This scratch pad is completely independent of the operating system. Most real time operating systems do not provide scratch pad, or clipboard, functionality.

The PegEditField object is able to copy the marked contents of it's string to the scratch pad and also provides functionality to paste the contents of the scratch pad at the current location of the cursor.

Any C/PEG object may use the scratch pad as they see fit, but it is important to remember that by setting the scratch pad, the caller will destroy any previously held string.

4.33 PegScratchPadSet

Synopsis:

```
    void PegScratchPadSet(PEGCHAR *pText);
```

Arguments:

pText

    Pointer to a NULL terminated string.

Returns:

None

Description:

This function copies the NULL terminated string pointed to by pText to the C/PEG scratch pad.

Errors:

None

Related Functions:

PegScratchPadGet

Notes:

See the Notes section of PegScratchPadGet.

4.34 PegSignalCheckSend

Synopsis:

```
PEGBOOL PegSignalCheckSend(void *pThing, PEGUSHORT signal);
```

Arguments:

pThing
    Pointer to a PegThing or derived object.
signal
    The signal to check.

Returns:

TRUE if pThing meets the criteria for sending a signal of signal, otherwise, FALSE.

Description:

This function evaluates whether or not pThing is able to send a signal of type signal. The criteria is the object must support that signal, must have a parent, must have an ID, and must be visible.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegSignalSend

Notes:

This function is called throughout the C/PEG library to check that pThing is able to send the signal. This saves a great deal of overhead in the long run. It is recommended that custom application objects also call this function when in a situation where the object may send a signal to it's parent.

4.35 PegSignalSend

Synopsis:

```
    void PegSignalSend(void *pThing, PEGUSHORT signal);
```

Arguments:

pThing	Pointer to a PegThing or derived object.
signal	Signal to send.

Returns:

None

Description:

This function sends a signal from pThing to pThing's parent object. The signal is in the form of a PegMessage structure that is pushed into the PegMessageQueue.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegSignalCheckSend, PegMessageQueuePush

Notes:

This function does not check for the validity of pThing wishing to send signal. Please use PegSignalCheckSend before calling PegSignalSend to ensure that pThing is able to send the signal.

4.36 PegSinCosLookup

Synopsis:

```
void PegSinCosLookup(PEGINT iAngle, PEGINT *pSin, PEGINT
  *pCos);
```

Arguments:

iAngle
: The angle to lookup.

pSin
: Pointer to a PEGINT data type that will hold the return sine of iAngle.

pCos
: Pointer to a PEGINT data type that will hold the return cosine of iAngle.

Returns:
: The sine and cosine of iAngle in pSin and pCos, respectively.

Description:

This function allows a set of geometric functions to exist in the C/PEG library without the need for floating point mathematics. This function is also available for application use, if so needed.

Errors:

None

Related Functions:

None

Notes:

C/PEG maintains a table of values that represent sine/cosine values for one quadrant of the Cartesian coordinate system. By interpolating iAngle, this single table is able to cover all four quadrants of the coordinate system. These values are stored as the actual sine/cosine of the angle multiplied by 1024, thus allowing them to work in integer mathematics. What this means to the application developer is that any operations done with these return values must then be divided by 1024 (or shifted right 10 places) to arrive at the actual value.

For an example, the Peg2DPolygon object uses this function to rotate the points of it's polygon for a given angle, theta. The following is a sample of code that rotates these points:

```
PegSinCosLoopkup(theta, &sin, &cos);

for (i = 0; i < pdp->uiNumPoints; i++)
{
    ixo = pdp->pPoints[i].x - ixc;
    iyo = pdp->pPoints[i].y - iyc;

    pdp->pRotPoints[i].x = (((ixo * cos) >> 10) -
        ((iyo * sin) >> 10) * -1) + ixc;
    pdp->pRotPoints[i].y = (((iyo * cos) >> 10) +
        ((iyo * sin) >> 10) * -1) + iyc;
}
```

You will note how the return values are treated.

C/PEG also uses this function when drawing circles and arcs.

The directives PEG_FULL_GRAPHICS and/or PEG_ARC_GRAPHICS must be defined in pconfig.h for this function to included in the C/PEG library build.

4.37 PegStatusInit

Synopsis:

```
void PegStatusInit(void *pThing);
```

Arguments:

pThing

Pointer to a PegThing or derived object.

Returns:

None

Description:

This function is eventually called by most objects during creation. This unitizes the status of the object to a common default setting.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegThingInit

Notes:

While not imperative, it is good practice for custom objects to call this function at some point during object creation.

4.38 PegStringIDSet

Synopsis:

```
    void PegStringIDSet(void *pTextThing, PEGSTRINGID id);
```

Arguments:

pTextThing
    Pointer to a PegTextThing or derived object.
id
    A string ID to assign to the object.

Returns:

None

Description:

This function sets the string ID associated with pTextThing to the value passed in id. This function then looks up the string in the string tables and assigns that string to pTextThing by calling PegTextSet, which also implies that the region of the screen occupied by pTextThing is then invalidated. Therefore, pTextThing should be redrawn after this call.

Errors:

This function may generate an assert if pTextThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegStringIDGet

Notes:

This function is only available if PEG_STRING_TABLES and PEGSTRING_IS_ID are both defined.

4.39 PegStyleAdd

Synopsis:

```
    void PegStyleAdd(void *pThing, PEGUSHORT usStyle);
```

Arguments:

pThing
    Pointer to a PegThing or derived object.
usStyle
    The style flags to add to pThing.

Returns:

None

Description:

This macro adds usStyle to the style mask of pThing.

Errors:

None

Related Macros:

PegStyleHas, PegStyleRemove

Notes:

Changing the style of an object will not automatically cause it to redraw. To see the effect of changing the object's style, first add the style, then invalidate it's screen area (PegRectRegionInvalidate), then call the object's Draw function (PegDraw).

4.40 PegStyleHas

Synopsis:

```
    PEGBOOL PegStyleHas(void *pThing, PEGUSHORT usStyle);
```

Arguments:

pThing
    Pointer to a PegThing or derived object.
usStyle
    Style flag to check.

Returns:

TRUE if the style mask in pThing contains the usStyle style, otherwise, FALSE.

Description:

Checks for usStyle in pThing's style mask.

Errors:

None

Related Macros:

PegStyleAdd, PegStyleRemove

Notes:

4.41 PegStyleRemove

Synopsis:

```
    void PegStyleRemove(void *pThing, PEGUSHORT usStyle);
```

Arguments:

pThing
    Pointer to a PegThing or derived object.
usStyle
    Style flag to remove.

Returns:

None

Description:

This function removes usStyle from pThing's style mask.

Errors:

None

Related Macros:

PegStyleAdd, PegStyleHas

Notes:

Changing the style of an object will not automatically cause it to redraw. To see the effect of changing the object's style, first remove the style, then invalidate it's screen area (PegRectRegionInvalidate), then call the object's Draw function (PegDraw).

4.42 PegTabOrderSet

Synopsis:

```
    void PegTabOrderSet(void *pThing, PEGUSHORT *pIds, PEGINT
      iNumIds);
```

Arguments:

pThing
: Pointer to a PegThing or derived object.

pIds
: Pointer to an array of ID values.

iNumIds
: The number of ID’s in pIds

Returns:

None

Description:

This function forces an application defined tab order to the children of pThing.

pIds points to an array of ID’s. The order of ID’s in the array determines the new tab order for the children of pThing. If the ID of any child of pThing is not contained in pIds, it will be left off of the tab list.

Errors:

None

Related Functions:

PegTabOrderSetDefault

Notes:

This function is only available if PEG_TAB_KEY_SUPPORT is enabled.

4.43 PegTabOrderSetDefault

Synopsis:

```
    void PegTabOrderSetDefault(void *pThing);
```

Arguments:

pThing

    Pointer to a PegThing object.

Returns:

None

Description:

This function sets the tab order for the children of pThing. It uses a nearest neighbor algorithm to determine the links between the child objects. Children that do not accept focus (do not have the PSF_ACCEPTS_FOCUS style turned on) are excluded from consideration and are not included in the tab list.

Errors:

None

Related Functions:

PegTabOrderSet

Notes:

This function is only available if PEG_TAB_KEY_SUPPORT is enabled.

4.44 PegTextCopyModeOn

Synopsis:

```
    void PegTextCopyModeOn(void *pTextThing);
```

Arguments:

pTextThing

    Pointer to a PegTextThing object.

Returns:

None

Description:

A PegTextThing or derived object may be optionally instructed to copy it's assigned text into a private buffer by passing the TT_COPY flag as a style in the object's Create function. If this is not done in the Create function, this function allows this style to be turned on after object creation.

Errors:

This function may generate an assert if pTextThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegTextThingCreate, PegTextThingTextSet

Notes:

Once this style is turned on, it can not be turned off for that particular object instance.

If the object will be handling text that was allocated on the stack frame, then it is a good idea to turn this on, simply because once the original text goes out of scope, the pointer that the object would have to it will be garbage and the next time the object draws, the results are undefined.

4.45 PegTextMaxLenSet

Synopsis:

```
void PegTextMaxLenSet(void *pTextThing, PEGINT iMaxLen);
```

Arguments:

pTextThing
    Pointer to a PegTextThing or derived object.
iMaxLen
    The maximum length of an acceptable string, in characters.

Returns:

None

Description:

This function sets the maximum allowable string that may be assigned to pTextThing. The maximum length of a string is used if the object's TT_COPY style has been turned on, otherwise, it is ignored.

Errors:

This function may generate an assert if pTextThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegTextCopyModeOn, PegTextSet

Notes:

This determines the maximum size of the internal buffer maintained by the object. If PEG_TEXT_MAX_IGNORE is passed in iMaxLen, then the object will not check the size of the assigned string before allocating a buffer to hold the string.

Every PegTextThing object has a defined allowable character length limit. These defines control the default values for the maximum allowable string length for each type of PegTextThing or derived object.

If the object does not have the TT_COPY style enabled, the internal maximum length buffer length member is ignored. The assumption is the string is maintained by the application code.

If iMaxLen is less than the length of the current string internalized by the object, the current string is truncated to iMaxLen – 1 and NULL terminated.

No object should be allowed to accept a string greater than the maximum value for a system native signed integer (PEGINT), regardless of the TT_COPY style setting.

4.46 PegTextHeight

Synopsis:

```
    PEGINT PegTextHeight(const PEGCHAR *pText, PegFont *pFont);
```

Arguments:

pText

Pointer to a NULL terminated character string.

pFont

Pointer to a PegFont structure that is used for determining the height of the text.

Returns:

Upon success, returns the height of the text using the given font. If the font is a multi-paged font, the return value will be the greatest character height of the pages in the font.

If pFont is NULL, the function returns zero.

Description:

This function measures the height of text using pFont. The height includes the ascenders and descenders of the glyphs, so is, therefore, the overall character height of the font.

If pText is NULL, the function simply returns the ubHeight value of the pFont structure.

Errors:

None

Related Functions:

PegDrawText, PegTextWidth

Notes:

The return value is more a measure of the overall character height of the font than a measurement of the height of the text.

4.47 PegTextWidth

Synopsis:

```
    PEGINT PegTextWidth(const PEGCHAR *pText, PegFont *pFont,
      PEGINT iLen);
```

Arguments:

pText
: Pointer to NULL terminated string.

pFont
: Pointer to a PegFont structure.

iLen
: The number of characters in pText to use for the return sum. By passing PEG_TEXT_ALL in this parameter, the function will evaluate the entire string pointed to by pText.

Returns:

Upon success, returns the length of the string pointed to by pText, or a substring of pText is iLen is less than the length of the string and not PEG_TEXT_ALL, using the widths in the characters contained in pFont. If either pText or pFont are NULL, then zero is returned.

Description:

This functions sums the width of a given string pointed to by pText using the character width table in the font pointed to by pFont.

Errors:

If pText or pFont is NULL, the function returns a value of zero.

Related Functions:

PegTextHeight, PegDrawText

Notes:

This function uses the offset table contained in the font to determine the aggregate width of the string. Therefore, this call may produce different results for the same string using different fonts.

Pass PEG_TEXT_ALL in iLen to evaluate the entire string. This saves a string library call before hand to determine the length of an unknown string.

4.48 PegVersionMajor

Synopsis:

```
    PEGINT PegVersionMajor(void);
```

Arguments:

None

Returns:

Returns the major build number of the C/PEG library in use.

Description:

This function returns the major build number of the C/PEG library in use.

Errors:

None

Related Functions:

PegVersionRelease, PegVersionMajor, PegVersionString

Notes:

Please see the Notes section of PegVersionString.

4.49 PegVersionMinor

Synopsis:

```
    PEGINT PegVersionMinor(void);
```

Arguments:

None

Returns:

Returns the minor version of the C/PEG library.

Description:

This function returns the minor version of the currently in use C/PEG library.

Errors:

None

Related Functions:

PegVersionRelease, PegVersionMajor, PegVersionString

Notes:

Please see the Notes section of PegVersionString.

4.50 PegVersionRelease

Synopsis:

```
    PEGINT PegVersionRelease(void);
```

Arguments:

None

Returns:

The release number of the C/PEG library.

Description:

This function returns the value defined in PEG_VERSION_RELEASE. This is the release number of the C/PEG library.

Errors:

None

Related Functions:

PegVersionString, PegVersionMajor, PegVersionMinor

Notes:

Please see the Notes section in PegVersionString.

4.51 PegVersionString

Synopsis:

```
    const PEGCHAR* PegVersionString(void);
```

Arguments:

None

Returns:

Returns a NULL terminated string describing the major, minor and release version of the C/PEG library.

Description:

This function returns a NULL terminated string that follows this basic format:

“C/PEG major.minor.release”

Errors:

None

Related Functions:

PegVersionRelease, PegVersionMajor, PegVersionMinor

Notes:

A particular release of the C/PEG library is governed by major, minor and release numbers. The numbering scheme is very simple and intended to provide C/PEG customers with a precise library version when corresponding with Swell Software's technical support department. These numbers are also useful when upgrading to a newer release of the library.

Release notes for a particular C/PEG release are included in the C/PEG distribution.

The defines for this string can be found in the cpeg/include/version.h file.

The major release number is reserved for defining major releases of the C/PEG library.

The minor release number is reserved for defining minor changes to the library. Even numbered releases denote a release that is available to C/PEG users, while odd numbered releases denote in house releases.

For instance, if the current minor version number releases to customers were 2, then the in house release at Swell Software would be 3. When version 3 were deemed satisfactory to release to customers, the release version would be moved to version 4 and released to customers. At that point, the in house version would be moved to version 5, and so on.

The release number signifies the release number of the current major and minor numbers. This number may be even or odd. Not all release numbers are released to customers.

4.52 PegWallpaperGet

Synopsis:

```
    PegBitmap *PegWallpaperGet(void *pPanel);
```

Arguments:

pPanel

Pointer to a PegPanel object.

Returns:

If pPanel has a wallpaper bitmap, then a pointer to the bitmap is returned. Otherwise, the return value is NULL.

Description:

This function returns a pointer to pPanel's wallpaper bitmap, if it has one.

Errors:

This function may generate an assert if pPanel is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegPanelDraw, PegPanelWallpaperSet

Notes:

Any PegPanel or derived object may have a background bitmap that is tiled to fill it's client area.

4.53 PegWallpaperSet

Synopsis:

```
    void PegWallpaperSet(void *pPanel, PegBitmap *pBitmap, PEGBOOL
      bTile);
```

Arguments:

pPanel
: Pointer to a PegPanel object.

pBitmap
: Pointer to a PegBitmap structure.

bTile
: Optionally tile the bitmap in the client area rectangle of pPanel. If set to FALSE, the bitmap will draw itself centered in the client region of pPanel.

Returns:

None

Description:

This function sets the background wallpaper for pPanel. If bTile is set to TRUE, the bitmap is tiled over the entire client area of pPanel beginning in the top, left corner of the client area. If bTile is FALSE, the bitmap will only be drawn once, centered in the client area.

Errors:

This function may generate an assert if pPanel is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegWallpaperGet, PegPanelDraw

Notes:

4.54 PegXResGet

Synopsis:

```
    PEGSHORT PegXResGet(void);
```

Arguments:

None

Returns:

Returns the horizontal resolution of the display in pixels.

Description:

Returns the horizontal resolution of the display expressed in pixels. This will usually match the constant PEG_VIRTUAL_XSIZE, but, on some systems the display size may be adjusted at run time, so it is best not to rely on PEG_VIRTUAL_XSIZE and instead query the PegScreen object for the current horizontal resolution.

Errors:

None

Related Functions:

PegYResGet

Notes:

4.55 PegYResGet

Synopsis:

```
    PEGSHORT PegYResGet(void);
```

Arguments:

None

Returns:

Returns the vertical resolution of the display in pixels.

Description:

Returns the vertical resolution of the display expressed in pixels. This will usually match the constant PEG_VIRTUAL_YSIZE, but, on some systems the display size may be adjusted at run time, so it is best not to rely on PEG_VIRTUAL_YSIZE and instead query the PegScreen object for the current vertical resolution.

Errors:

None

Related Functions:

PegXResGet

Notes:

CHAPTER 5

C/PEG MESSAGE QUEUE FUNCTIONS

5.1 Overview

The PegMessageQueue functions have been given their own section simply because they are unique unto themselves.

The functions that deal directly with the message queues are implementation dependent. C/PEG provides default implementations for these message queue functions that are suitable for running in stand alone or single threaded mode. But, if the application will be using multiple threads or tasks to work with the GUI, then new implementations of these functions must replace the default C/PEG functions.

For most supported operating systems, these functions are properly implemented. For instance, when running C/PEG on ThreadX, C/PEG uses services provided by ThreadX for creating message queues and passing messages between objects and between tasks. The default implementation of the message queue functions are not used.

If you are porting C/PEG to an unsupported operating system, please contact Swell Software for assistance on getting the message queues up and running.

5.2 PegMessageQueueCreate

Synopsis:

```
    PegMessageQueue *PegMessageQueueCreate(void);
```

Arguments:

None

Returns:

Returns a pointer to a newly allocated PegMessageQueue object.

Description:

This function creates a new PegMessageQueue object and returns a pointer to it.

Errors:

This function may generate an assert if the necessary memory for the object can not be allocated from the system and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegMessageQueueDestroy

Notes:

It is not recommended for application level code to call this function.

Part of the C/PEG startup procedure is to allocate one PegMessageQueue that is used for all system message passing. If the library was configured to run with a multi-threaded operating system, then the C/PEG library may allocate new message queues when necessary to properly handle message passing during multi-threaded operation.

The default implementation of this function allocates a new PegMessageQueue object as well as a pool of free PegMessage structures that is used as the messaging mechanism for the system. This implementation is not thread safe and should not be used with a multi-threading operating system.

It is highly recommended that the default implementation be replaced by a version that is coupled with the operating system and used facilities provided by the operating system to allocate message pools as well as blocking techniques that may be used when the queue is empty.

Please see the section on PegMessageQueue's in the “C/PEG Programmers Manual” for a more detailed look on how to integrate the message queue to an operating system.

5.3 PegMessageQueueDestroy

Synopsis:

```
void PegMessageQueueDestroy(PegMessageQueue *pQueue);
```

Arguments:

pQueue

Pointer to a PegMessageQueue object.

Returns:

None

Description:

This is called by the C/PEG library to close and release the resources owned by pQueue.

Errors:

This function may generate an assert if pQueue is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegMessageQueueCreate

Notes:

As with PegMessageQueueCreate, this function should not be called by application code. The C/PEG library keeps track of the system as well as any ancillary message queues in use by the application.

5.4 PegMessageQueueEnqueue

Synopsis:

```
void PegMessageQueueEnqueue(PegMessageQueue *pQueue, PegMessage
  *pMesg);
```

Arguments:

pQueue
: Pointer to a PegMessageQueue object.

pMesg
: Pointer to a PegMessage structure.

Returns:

None

Description:

This function provides an alternate to pushing a message into pQueue.

Errors:

This function may generate an assert if pQueue is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegMessageQueuePush

Notes:

It is not recommended to call this function from use application code. Instead, use the PegMessageQueuePush function.

This function is only available if C/PEG was compiled with the PEG_MULTITHREAD symbol defined.

The integration package of C/PEG to the operating system implements this function.

5.5 PegMessageQueueFold

Synopsis:

```
void PegMessageQueueFold(PegMessageQueue *pQueue, PegMessage
  *pMesg);
```

Arguments:

pQueue
Pointer to a PegMessageQueue object.
pMesg
Pointer to a PegMessage structure.

Returns:

None

Description:

This function folds a message pointed to by pMesg into the queue pointed to by pQueue. This folding process aids in eliminating duplicate message types in the queue.

Errors:

This function may generate an assert if pQueue is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegMessageQueuePush

Notes:

This function should not be called directly by application code. The function exists to provide a portable way for device driver authors to push device messages into the C/PEG system queue without overloading the queue with device messages.

For example, if the a driver for a mouse device sends messages into the C/PEG system queue every time the mouse is moved and the driver reads the hardware, this could potentially generate enough messages to quickly fill the C/PEG system queue and create a bottle neck in the system. But, if the mouse movement messages were folded into the queue instead of pushed, the queue examines the messages already in the queue. If an existing message has the same type as the incoming message, the data from the incoming message is folded into the data of the existing message and the incoming message is then discarded.

Obviously this could incur some overhead, depending on the implementation of the message queue within the operating system, and, as such, is not ideal for general purpose application use.

5.6 PegMessageQueuePop

Synopsis:

```
void PegMessageQueuePop(PegMessageQueue *pQueue, PegMessage
  *pMesg);
```

Arguments:

pQueue
Pointer to a PegMessageQueue object.
pMesg
Pointer to a PegMessage structure.

Returns:

The contents of the next message in pQueue are copied into pMesg.

Description:

This function is used by the C/PEG library to pop messages from the message queues.

Errors:

This function may generate an assert if pQueue is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegMessageQueuePush

Notes:

It is not recommended for application code to directly call this function. The C/PEG library handles popping messages from the system queues and routing the messages to the proper objects in the proper tasks.

5.7 PegMessageQueuePurge

Synopsis:

```
    void PegMessageQueuePurge(PegMessageQueue *pQueue, void
      *pThing);
```

Arguments:

pQueue
    Pointer to a PegMessageQueue object.
pThing
    Pointer to a PegThing or derivative object.

Returns:

None

Description:

This function purges all messages from pQueue whose target is pThing. This is usually called from the destroy function for the object. This insures that the message queue will not attempt to send a message to an object that has been destroyed.

Errors:

This function may generate an assert if pQueue is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegMessageQueuePop

Notes:

This function should not normally be called from application code.

If the application designer creates a custom object and overrides the destroy function of the base object, it is imperative for the destroy function in the custom object to call the destroy function of the base object.

Eventually, every C/PEG object's destroy function calls the PegThingDestroy function that properly purges the object from the system message queue.

5.8 PegMessageQueuePush

Synopsis:

```
void PegMessageQueuePush(PegMessageQueue *pQueue, PegMessage
  *pMesg);
```

Arguments:

pQueue
: Pointer to a PegMessageQueue object.

pMesg
: Pointer to a PegMessage structure.

Returns:

None

Description:

This function pushes the PegMessage pointed to by pMesg into the PegMessageQueue pointed to by pQueue.

Errors:

This function may generate an assert if pQueue is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegMessageQueuePop, PegMessageQueueEnqueue, PegMessageQueueFold

Notes:

This is the only PegMessageQueue function that should be called from application code. This is the standard way for any task to push a message into any queue. To push a message into the C/PEG system queue, call the PegMessageQueuePtr to attain a pointer to the queue, then use this pointer as the first argument in this function.

In a multi-threaded environment, it is perfectly legal for any thread to push a message into the C/PEG system queue. This is often the most efficient way for inter-task communication when relating messages to the C/PEG system and objects.

5.9 PegTimerKill

Synopsis:

```
    void PegTimerKill(void *pThing, PEGUSHORT usId);
```

Arguments:

pThing
: Pointer to a PegThing or derived object.

usId
: ID of the timer to kill.

Returns:

None

Description:

This function stops and destroys the timer of usId that is owned by pThing.

Errors:

This function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegTimerStart, PegTimerTick

Notes:

This function is used by the application code to stop and destroy a timer started by calling PegTimerStart.

If usId is 0, then all of the timers for the object pointed to by pThing, as well as any timers owned by children of pThing, are halted and destroyed.

5.10 PegTimerStart

Synopsis:

```
    void PegTimerStart(void *pThing, PEGUSHORT usId, PEGLONG
      lExpire, PEGLONG lReset);
```

Arguments:

pThing
: Pointer to a PegThing or derivative that is to be notified when the timer expires.

usId
: ID of the timer. This does not have to be a application wide unique ID number. It only needs to be unique to pThing.

lExpire
: The number of timer ticks that will pass before the timer will expire.

lReset
: The number of timer ticks the timer will be reset to after lExpire has expired.

Returns:

None

Description:

This function starts a timer in with an ID of usId. When the timer reaches lExpire number of timer ticks, the object pointed to by pThing will be sent a PegMessage with a type of PM_TIMER. The ID of the timer is in the sData member of the message, to allow the object to distinguish between multiple timers that it may have started.

After the first expiration of lExpire number of timer ticks, lReset is examined. If lReset is zero, then the timer is killed. Otherwise, subsequent expirations of the timer will occur at lReset intervals until the application manually kills the timer by calling PegTimerKill.

Errors:

This function may generate an assert if pQueue is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegTimerKill, PegTimerTick

Notes:

The ID number of the timer is only important to the object that is installing the timer. If the object only starts up one timer, an ID of 1 is often used. Any PegThing or derived object may start any number of timers.
The following code snippet is a common use of timers. In this example, the object will start two timers, to further illustrate how the object may need to distinguish between which timer has expired.

```
PEGINT MyObjectNotify(void *pThing, const PegMessage *pMesg)
{
      switch(pMesg->usType)
      {
      case PM_SHOW:
          PegTimerStart(pThing, 1, 10, 30);
          PegTimerStart(pThing, 2, 5, 5);
          PegThingNotify(pThing, pMesg);
          break;

      case PM_HIDE:
          PegTimerKill(pThing, 1);
          PegTimerKill(pThing, 2);
          PegThingNotify(pThing, pMesg);
          break;

      case PM_TIMER:
          if (pMesg->sData == 1)
          {
          /* first timer expired */
          }
          else
          {
          /* second timer expired */
          }
          break;

      default:
```

```
                return(PegThingNotify(pThing, pMesg);
            }


            return(0);
        }
```

This example starts the timers then the object receives a PM_SHOW message. This is a typical course of action when the object wishes to have a timer expiring for the duration of it's stay on the visible screen. Timers can be started in response to any sort of event or state that the object may receive or be in, but it is always a good idea to kill the timers when the object is hidden. Thus, when the object receives a PM_HIDE message, it kills it's timers.

When the object receives a PM_TIMER message, it inspects the sData member of the message to the ID of the timer that expired. Notice that the first timer has an initial expiration of 10 timer ticks, and a reset of 30 timer ticks, while the second timer has a expiration and reset of 5 timer ticks.

It is important to note that a timer tick is system dependent. It is usually a coarse grained timer that generates roughly 20 timer ticks per second. But, that can not always be assumed to be true. The number of timer ticks per second is dependent on the operating system, the hardware on which the application is running and the means by which timer ticks are generated in the C/PEG library.

If the application needs a fine grained, deterministic, timer, then it is best to use operating system features to generate reliable timers.

5.11 PegTimerTick

Synopsis:

```
    void PegTimerTick(void);
```

Arguments:

None

Returns:

None

Description:

This function causes the default PegMessageQueue to evaluate it's timer list. If the timer list contains any timers that reach expiration, then the timer's corresponding object is sent a PM_TIMER message.

Errors:

This function may generate an assert if pQueue is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegTimerStart, PegTimerKill

Notes:

This function should not be called by application code. This function should only be called by the device responsible for generating timer ticks into the C/PEG system message queue.

For multi-threading environments, this is often a separate thread designed for this purpose. On some integrations, the timer tick facility is programmed directly into hardware and vectored to on a hardware interrupt.

When running stand alone, it is acceptable to generate these timer ticks by polling a timer function during PegIdleFunction.

CHAPTER 6

C/PEG MACROS

6.1 Overview

Many of the common “functions” used in application code are actually macros. This saves space, is more efficient at run time, and generally makes application code much easier to read.

For instance, the PegDraw macro can be used on any object that inherits from PegThing, and turns this bit of code:

```
    pThing->funcDraw(pThing);
```

into this code:

```
    PegDraw(pThing);
```

For all practical purposes, both calls are absolutely equivalent. The advantage of the second form is that the application code is insulating itself from the inner working of the PegThing object. So, if the function pointer named funcDraw should ever change in the PegThing object, the application code would not break if using the second form.

Every function pointer in the PegThing, PegTextThing and PegPanel objects may be called using a macro that has the same form as the function name. It is highly recommended that the application code use these macros to implement the application.

The “C/PEG Programmers Manual” has a more detailed look at how these macros are implemented.

6.2 PEG_ALLOC

Synopsis:

```
    void *PEG_ALLOC(size)
```

Arguments:

size

Integer value representing the size of the needed memory block.

Returns:

Upon success, returns a pointer to the newly allocated block of memory. Upon failure, NULL is most often returned.

Return values may be dependent on the implementation and specific to the operating system upon which the application is running.

Description:

The default implementation of this function (the implementation that is used when running a C/PEG application without an operating system, or on a development platform operating system such as Windows or Unix/Linux), the macro expands to call malloc.

With the exception of stack, or automatic, variables, this macro is used for all memory allocation within the C/PEG library.

Errors:

Errors are dependent on the implementation.

Related Functions:

PEG_FREE, malloc

Notes:

For some real time operating systems, the default implementation of calling malloc may be a problem. Often times a compiler version of malloc is incompatible with the memory allocation algorithms of a particular RTOS. It is for this reason that this macro may need to be adapted to use RTOS provided functionality for memory block allocation.

6.3 PEG_ASSERT

Synopsis:

```
    #define PEG_ASSERT(_exp) assert(_exp)
    #define PEG_ASSERT(_exp) ((void)0)
```

Arguments:

_exp

The expression to be evaluated

Returns:

None

Description:

The directive PEG_USE_ASSERT must be turned on in the pconfig.h file for the macro to evaluate to the first form listed above. If PEG_USE_ASSERT is not defined, then the second form is used.

This macro is used throughout the C/PEG library to test incoming pointers and return values from PEG_ALLOC.

Errors:

It is an error to define PEG_USE_ASSERT and attempt to compile the library with a compiler that does not support the assert function call.

Related Macros:

PEG_ALLOC

Notes:

6.4 PEG_CHECK_VALID_DRAW

Synopsis:

```
PEGBOOL PEG_CHECK_VALID_DRAW(PegThing *pThing);
```

Arguments:

pThing

Pointer to a PegThing or derived object.

Returns:

TRUE if pThing is allowed to draw, otherwise, FALSE.

Description:

This macro checks for PSF_VISIBLE and PSF_DRAWABLE status in pThing. If pThing has both, then this macro returns TRUE.

Errors:

None

Related Macros:

PegStatusIs

Notes:

This macro should be used in application code before any drawing calls. This will insure that the object that is wishing to draw may do so before making any calls into the graphics subsystem. And while the graphics subsystem catches calls from objects that are not allowed to draw, it is more efficient for the application to make this call once before any draw operations begin.

The following is a small code example using this macro:

```
void MyObjectDraw(void *pThing)
{
    if (!PEG_CHECK_VALID_DRAW(pThing))
    {
        return;
    }

    /* continue with the drawing */
```

```
        PegDrawBegin(pThing);

        /* do some drawing for the object */

        /* end drawing */
        PegDrawEnd(pThing);
    }
```

Since PEG_CHECK_VALID_DRAW is a macro that contains two other macros, there is no function call overhead, and is therefore very cheap to call.

It is important to note an object that attempts to draw when it is not visible or when it does not have drawable status is caught by the PegScreen object and any calls made by the object are ignored. This insures that one object's drawing does not corrupt another object that may happen to be on top of the first object. But, with a little bit of foresight, the application developer can save the overhead of calling the PegScreen object in the first place, and determine for themselves if the object should be trying to draw or not. In the long run, this could potentially save a great number of wasted system cycles.

6.5 PEG_FREE

Synopsis:

```
void PEG_FREE(void *pMem)
```

Arguments:

pMem

Pointer to a block of memory to return to the system.

Returns:

None

Description:

The default implementation of this function (the implementation that is used when running a C/PEG application without an operating system, or on a development platform operating system such as Windows or Unix/Linux), the macro expands to call free.

This macro is used throughout the C/PEG library to return memory allocated with PEG_ALLOC back to the system.

Errors:

Errors are dependent on the implementation.

Related Functions:

PEG_ALLOC, free

Notes:

For real time operating systems that do not use malloc for memory allocation, this function would also need to be modified to use the operating system functionality for releasing a block of memory back to the operating system.

6.6 PEG_SIGNAL

Synopsis:

```
(PEGUSHORT)PEG_SIGNAL(PEGUSHORT usId, PEGINT iSignal);
```

Arguments:

usId

ID of the object that is sending or has sent the signal.

iSignal

Contstant of the signal type.

Returns:

A PEGUSHORT that is the merging of usId with iSignal.

Description:

This macro is used to encode and decode unique message types that represent a particular object sending a particular message. This type of message is referred to as a signal, in that it is sent by object's based on user interaction.

Errors:

None

Related Functions:

PegNotify

Notes:

This macro is integral in the messaging mechanism supported by C/PEG in relating events from a child object to it's parent object. To allow a parent object to receive particular messages from a child object, this macro combines the ID of the object with the type of signal that the object wishes to send to create a unique message type that is then sent to the parent object.

In turn, the parent object then uses this macro to decode the message type to determine the ID of the object that sent the message as well as the signal being sent by the child object.

The table below lists the signal types supported by C/PEG. Not all object types send each type of signal.

Signal Type	Supported Objects	Sent When
PSF_SIZED	PegThing	object is moved or resized
PSF_FOCUS_RECEIVED	PegThing	object receives input focus
PSF_FOCUS_LOST	PegThing	object loses input focus
PSF_KEY_RECEIVED	PegThing	object receives an unsupported input key
PSF_CLICKED	PegThing	default left click notification
PSF_RIGHTCLICK	PegThing	right click notification
PSF_TEXT_SELECT	PegEditField	all or a portion of the text is selected
PSF_TEXT_EDIT	PegEditField	each time the text is modified
PSF_TEXT_EDIT_DONE	PegEditField	text modification is complete
PSF_CHECK_ON	PegCheckBox	object is checked
PSF_CHECK_OFF	PegCheckBox	object is unchecked
PSF_DOT_ON	PegRadioButton	object is selected
PSF_DOT_OFF	PegRadioButton	object is unselected
PSF_LIST_SELECT	PegList	list child object is selected
PSF_SCROLL_CHANGE	PegVertScroll, PegHorzScroll	scroll position is changed
PSF_SLIDER_CHANGE	PegSlider	slider thumb position is changed
PSF_SPIN_MORE	PegSpinButton	the up or right button is clicked
PSF_SPIN_LESS	PegSpinButton	the down or left button is clicked

The following code snippet is an example of a PegTextButton object sending a signal to it's parent object which is a PegPanel derivative.

```
1   enum ChildIds
2   {
3       IDB_BUTTON_ONE = 1,
4       IDB_CHECKBOX_ONE
5   };
6
7   PEGINT MyPanelNotify(void *pThing, const PegMessage
    *pMesg)
8   {
9       switch(pMesg->usType)
10      {
11      case PEG_SIGNAL(IDB_BUTTON_ONE,PSF_CLICKED):
12           /* process the button's clicked signal */
13           break;
14
15      case PEG_SIGNAL(IDB_CHECKBOX_ONE, PSF_CHECK_ON):
16           /* process the checkbox's check on signal */
17           break;
18
19      default:
20           return(PegPanelNotify(pThing, pMesg));
21      }
22
23      return(0);
24  }
```

The lines 1 through 5 simply enumerate the button ID's used in this example. It is important that ID's always start with a value greater than zero. The child ID's do not need to be system unique. It is perfectly acceptable for different parent objects to have children that have similar IDs. Likewise, it is important for all children of a given parent object to have unique IDs, otherwise, the signal messages received by a parent would not be unique for every child.

Lines 7 through 24 show how a parent object catches a signal from a child object, based on the ID and the signal being sent. Note we are still interrogating the usType member of the message for the signal that was sent. This is important from a system perspective since the intervals of C/PEG handle system messages differently than object signals, and this provides an easy way for the system to determine the type of message, regardless of how the message was generated.

6.7 PegAdd

Synopsis:

```
void PegAdd(void *pThing, void *pAdd, PEGBOOL bRedraw);
```

Arguments:

pThing
Pointer to a PegThing or derived object.
pAdd
Pointer to a PegThing or derived object.
bRedraw
Optionally instructs pThing to redraw once pAdd has been added.

Returns:

None

Description:

This macro calls the add function of pThing through pThing's add function pointer.

Errors:

The add function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegThingAdd

Notes:

This macro encapsulates the object and hides the details of calling the object's add function. This macro should always be used by application code to call the object's add function.

6.8 PegAddToEnd

Synopsis:

```
    void PegAddToEnd(void *pThing, void *pAdd, PEGBOOL bRedraw);
```

Arguments:

pThing
    Pointer to a PegThing or derived object.
pAdd
    Pointer to a PegThing or derived object.
bRedraw
    Optionally instructs pThing to redraw once pAdd has been added.

Returns:

None

Description:

This macro calls the add to end function of pThing through pThing's add to end function pointer.

Errors:

The add to end function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegThingAddToEnd

Notes:

This macro encapsulates the object and hides the details of calling the object's add to end function. This macro should always be used by application code to call the object's add to end function.

6.9 PegBrushBitmapSet

Synopsis:

```
    void PegBrushBitmapSet(PegBrush *pBrush, PegBitmap *pBitmap);
```

Arguments:

pBrush
    Pointer to a PegBrush object.
pBitmap
    Pointer to a PegBitmap structure.

Returns:

None

Description:

This macro set the bitmap that is used for drawing with pBrush.

Errors:

None

Related Macros:

PegBrushColorsSet, PegBrushFlagsSet, PegBrushPatternSet, PegBrushSet

Notes:

The caller should also set the pBrush flags to define how the bitmap will be used for drawing.

6.10 PegBrushClipRectSet

Synopsis:

```
void PegBrushClipRectSet(PegBrush *pBrush, PegRect *pRect);
```

Arguments:

pBrush
: Pointer to a PegBrush object.

pRect
: Pointer to a PegRect structure.

Returns:

None

Description:

This macro copies the data member of pRect into the internal clipping rectangle of pBrush. If pRect is NULL, the internal clip rectangle members are set to -1.

Errors:

None

Related Macros:

PegBrushSet, PegBrushColorsSet, PegBrushFlagsSet, PegBrushPatternSet, PegBrushBitmapSet

Notes:

It is not an error to pass NULL in pRect.

The flags of pBrush should be updated to use the rectangle when needed.

6.11 PegBrushColorsSet

Synopsis:

```
    void PegBrushColorsSet(PegBrush *pBrush, PEGCOLORVAL fore,
      PEGCOLORVAL back);
```

Arguments:

pBrush
    Pointer to a PegBrush object.
fore
    Color value to use for drawing foreground colors.
back
    Color value to use for drawing background colors.

Returns:

None

Description:

This macro sets the Foreground and Background members of pBrush.

Errors:

None

Related Macros:

PegBrushSet, PegBrushFlagsSet, PegBrushPatternSet,
PegBrushBitmapSet, PegBrushClipRect Set

Notes:

6.12 PegBrushFlagsSet

Synopsis:

```
void PegBrushFlagsSet(PegBrush *pBrush, PEGUBYTE ubFlags);
```

Arguments:

pBrush
    Pointer to a PegBrush object.
ubFlags
    Modifier flags.

Returns:

None

Description:

This None sets the flags member of pBrush with ubFlags.

Errors:

None

Notes:

The following table lists valid values for ubFlags. Some of these flags may be combined.

Flag Name	Description
CF_NONE	No modifier
CF_FILL	Draw operations will fill with the current background color.
CF_XOR	Draw operations will be performed using an XOR algorithm and ignore the foreground and background colors.
CF_ANTIALIAS	Draw operations will perform anti-aliased.
CF_PATTERN	Draw operations will use the internal pattern to draw lines and fills. Also used to bitmap flood fill a polygon.
CF_TILE	Used to tile a bitmap over the clip rectangle.
CF_STRETCH	Used to fill a rectangle with a bitmap.

Flag Name	Description
CF_CENTER	Used to center a bitmap in the brush's clip rectangle.
CF_CLIP	Determines if the drawing operation should optionally clip to the brush's internal rectangle.

6.13 PegBrushPatternSet

Synopsis:

```
    void PegBrushPatternSet(PegBrush *pBrush, PEGULONG ulPattern);
```

Arguments:

pBrush
    Pointer to a PegBrush object.
ulPattern
    Pattern data.

Returns:

None

Description:

This macro sets the internal pattern member of pBrush to ulPattern. This pattern may be used in some drawing operations to draw pattern lines and fills.

ulPattern is a PEGULONG, which, on most systems is 32 bits wide. Each bit in the pattern represents a corresponding pixel on the screen as on or off. Therefore, the pattern is 32 pixels wide. A bit that is turned on tells the drawing operations that a pixel should be set in the foreground color of the brush.

Errors:

None

Related Functions:

PegBrushSet, PegBrushColorsSet, PegBrushFlagsSet, PegBrushPatternSet, PegBrushBitmapSet, PegBrushClipRectSet

Notes:

The default value for ulPattern in any PegBrush that was created using PegBrushCreate or PegBrushCreateDefault is PEG_DEF_FILL_PATTERN, which is currently defined as 0xccccccccUL. This pattern is 2 bits on followed by 2 bits off repeated 8 times.

6.14 PegDestroy

Synopsis:

```
    void PegDestroy(void *pThing);
```

Arguments:

pThing
        Pointer to a PegThing or derived object.

Returns:

None

Description:

This macro calls the destroy function of pThing through pThing's destroy function pointer.

Errors:

The destroy function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegThingDestroy

Notes:

This macro encapsulates the object and hides the details of calling the object's destroy function. This macro should always be used by application code to call the object's destroy function.

6.15 PegDraw

Synopsis:

```
    void PegDraw(void *pThing);
```

Arguments:

pThing

    Pointer to a PegThing or derived object.

Returns:

None

Description:

This macro calls the draw function of pThing through pThing's draw function pointer.

Errors:

The draw function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegThingDraw

Notes:

This macro encapsulates the object and hides the details of calling the object's draw function. This macro should always be used by application code to call the object's draw function.

6.16 PegDrawBorder

Synopsis:

```
void PegDrawBorder(void *pThing, PEGCOLORVAL c);
```

Arguments:

pThing

Pointer to a PegThing or derived object.

c

Color value to use for filling the background.

Returns:

None

Description:

This macro calls the draw border function of pThing through pThing's draw border function pointer.

Errors:

The draw border function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegThingDrawBorder

Notes:

This macro encapsulates the object and hides the details of calling the object's draw border function. This macro should always be used by application code to call the object's draw border function.

6.17 PegDrawFocus

Synopsis:

```
    void PegDrawFocus(void *pThing, PEGBOOL bRedraw);
```

Arguments:

pThing
Pointer to a PegThing or derived object.
bRedraw
Optionally redraws the entire object to draw the focus indicator.

Returns:

None

Description:

This macro calls the draw focus function of pThing through pThing's draw focus function pointer.

Errors:

The draw focus function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegThingDrawFocus

Notes:

This macro encapsulates the object and hides the details of calling the object's draw focus function. This macro should always be used by application code to call the object's draw focus function.

6.18 PegEraseFocus

Synopsis:

```
    void PegEraseFocus(void *pThing);
```

Arguments:

pThing
Pointer to a PegThing or derived object.

Returns:

None

Description:

This macro calls the erase focus function of pThing through pThing's erase focus function pointer.

Errors:

The erase focus function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegThingEraseFocus

Notes:

This macro encapsulates the object and hides the details of calling the object's erase focus function. This macro should always be used by application code to call the object's erase focus function.

6.19 PegExecute

Synopsis:

```
    PEGINT PegExecute(void *pThing, PEGUSHORT usMode);
```

Arguments:

pThing

Pointer to a PegPanel or derived object.

usMode

The mode at which to execute.

Returns:

Various depending on how the object was dismissed by the user.

Description:

This macro calls the execute function of pThing through pThing's execute function pointer.

Errors:

The execute function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegPanelExecute

Notes:

This macro encapsulates the object and hides the details of calling the object's execute function. This macro should always be used by application code to call the object's execute focus function.

6.20 PegFirst

Synopsis:

```
    PegThing *PegFirst(void *pThing);
```

Arguments:

pThing

  Pointer to a PegThing or derived object.

Returns:

Returns a pointer to pThing's first child. If pThing has no children, this returns NULL.

Description:

This macro returns a pointer to pThing's first child.

Errors:

None

Related Macros:

PegNext, PegParent

Notes:

If the macros PegAdd and PegRemove are always used to add and remove children from an object, then pThing's first child pointer should always be valid. In this case, a NULL return is valid. But, if the application code does not adhere to this practice, pThing's first child pointer may become corrupt, and the return value from this macro may point to garbage.

6.21 PegFontGet

Synopsis:

```
    PegFont *PegFontGet(void *pTextThing);
```

Arguments:

pTextThing
    Pointer to a PegTextThing or derived object.

Returns:

A pointer to the font currently in use by pTextThing.

Description:

This function returns a pointer to the font being used by pTextThing.

Errors:

This macro may generate a run-time error if pTextThing is not a PegTextThing or derivative.

Related Macros:

PegFontSet

Notes:

6.22 PegFontSet

Synopsis:

```
    void PegFontSet(void *pThing, PegFont *pFont);
```

Arguments:

pThing
: Pointer to a PegTextThing or derived object.

pFont
: Pointer to a PegFont structure.

Returns:

None

Description:

This macro calls the font set function of pThing through pThing's font set function pointer.

Errors:

The font set function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegTextThingFontSet

Notes:

This macro encapsulates the object and hides the details of calling the object's font set function. This macro should always be used by application code to call the object's font set function.

6.23 PegIDGet

Synopsis:

```
    PEGUSHORT PegIDGet(void *pThing);
```

Arguments:

pThing
    Pointer to a PegThing or derived object.

Returns:

The ID of the object.

Description:

This function returns the user assigned ID of the object pThing.

Errors:

None

Related Macros:

PegIDSet

Notes:

6.24 PegIDSet

Synopsis:

```
    void PegIDSet(void *pThing, PEGUSHORT usID);
```

Arguments:

pThing
  Pointer to a PegThing or derived object.

usID
  The ID to assign to pThing.

Returns:

None

Description:

This macro sets the ID of pThing to usID.

Errors:

None

Related Macros:

PegIDGet

Notes:

6.25 PegMessageTypeGet

Synopsis:

```
    PEGUSHORT PegMessageTypeGet(PegMessage *pMesg);
```

Arguments:

pMesg
    Pointer to a PegMessage structure.

Returns:

The type of the message expressed as an unsigned short integer.

Description:

This macro hides the implementation of the type member of a PegMessage structure. It is preferable to use this construct rather than addressing the PegMessage's data member directly from application code.

Errors:

None

Related Macros:

None

Notes:

None

6.26 PegNext

Synopsis:

```
    PegThing *PegNext(void *pThing);
```

Arguments:

pThing

Pointer to a PegThing or derived object.

Returns:

A pointer to pThing's first sibling. If pThing has no siblings, this returns NULL.

Description:

This macro returns the first sibling of pThing.

Errors:

None

Related Macros:

PegFirst, PegParent

Notes:

Please refer to the Notes section of the PegFirst macro.

6.27 PegNotify

Synopsis:

```
    PEGINT PegNotify(void *pThing, const PegMessage *pMesg);
```

Arguments:

pThing

Pointer to a PegThing or derived object.

pMesg

Pointer to a PegMessage structure that contains the message data.

Returns:

Various based on the type of message.

Description:

This macro calls the notify function of pThing through the notify function pointer in pThing.

Errors:

The notify function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegThingNotify

Notes:

This macro encapsulates the object and hides the details of calling the object's notify function. This macro should always be used by application code to call the object's notify function.

6.28 PegParent

Synopsis:

```
    PegThing *PegParent(void *pThing);
```

Arguments:

pThing

Pointer to a PegThing or derived object.

Returns:

Pointer to pThing's parent.

Description:

This macro returns a pointer to pThing's parent object.

Errors:

None

Related Macros:

PegFirst, PegNext

Notes:

Please see the Notes section of the PegFirst macro.

6.29 PegParentShift

Synopsis:

```
    void PegParentShift(void *pThing, PEGSHORT xshift, PEGSHIFT
      yshift);
```

Arguments:

pThing
: Pointer to a PegThing or derived object.

xshift
: Amount to shift along the x axis.

yshift
: Amount to shift along the y axis.

Returns:

None

Description:

This macro calls the parent function of pThing through the parent function pointer in pThing.

Errors:

The parent shift function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegThingParentShift

Notes:

This macro encapsulates the object and hides the details of calling the object's parent shift function. This macro should always be used by application code to call the object's parent shift function.

6.30 PegRemove

Synopsis:

```
void *PegRemove(void *pThing, void *pAdd, PEGBOOL bRedraw);
```

Arguments:

pThing
: Pointer to a PegThing or derived object.

pAdd
: Pointer to a PegThing or derived object.

bRedraw
: Optionally instructs pThing to redraw once pAdd has been removed.

Returns:

Pointer to the object that was removed. If pAdd is not a child object of pThing, then NULL is returned.

Description:

This macro calls the remove function of pThing through pThing's remove function pointer.

Errors:

The remove function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegThingRemove

Notes:

This macro encapsulates the object and hides the details of calling the object's remove function. This macro should always be used by application code to call the object's remove function.

6.31 PegResize

Synopsis:

```
    void PegResize(void *pThing, PegRect *pRect);
```

Arguments:

pThing
    Pointer to a PegThing or derived object.

pRect
    Pointer to a PegRect structure that defines the size and position to be given to pThing.

Returns:

None

Description:

This macro calls the resize function of pThing through pThing's resize function pointer.

Errors:

The resize function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegThingResize

Notes:

This macro encapsulates the object and hides the details of calling the object's resize function. This macro should always be used by application code to call the object's resize function.

6.32 PegSignalAdd

Synopsis:

```
    void PegSignalAdd(void *pThing, PEGUSHORT signal);
```

Arguments:

pThing
: Pointer to a PegThing or derived object.

signal
: A signal value.

Returns:

None

Description:

This function adds the signal value to the signal mask of pThing.

Errors:

None

Related Macros:

PegSignalSet, PegSignalRemove, PegSignalHas

Notes:

This macro may not be used to add several signals at once.

6.33 PegSignalHas

Synopsis:

```
    PEGBOOL PegSignalHas(void *pThing, PEGUSHORT signal);
```

Arguments:

pThing

Pointer to a PegThing or derived object.

signal

Signal value to check.

Returns:

TRUE if signal was in pThing's signal mask, otherwise, FALSE.

Description:

This macro checks for the existence of signal in pThing's signal mask.

Errors:

None

Related Macros:

PegSignalSet, PegSignalAdd, PegSignalRemove

Notes:

The signal argument may be logically OR'd together with multiple signals to test for the existence of any signal.

6.34 PegSignalRemove

Synopsis:

```
    void PegSignalRemove(void *pThing, PEGUSHORT signal);
```

Arguments:

pThing
    Pointer to a PegThing or derived object.

signal
    The signal value to remove.

Returns:

None

Description:

This function removes signal from pThing's signal mask.

Errors:

None

Related Macros:

PegSignalAdd, PegSignalHas, PegSignalSet

Notes:

This macro is able to remove only one signal at a time.

6.35 PegSignalSet

Synopsis:

```
    void PegSignalSet(void *pThing, PEGUSHORT signal);
```

Arguments:

pThing
: Pointer to a PegThing or derived object.

signal
: The signal to set in pThing.

Returns:

None

Description:

This macro sets the signal mask of pThing to signal, removing any signals that may have already been set.

Errors:

None

Related Macros:

PegSignalAdd, PegSignalRemove

Notes:

This macro will remove any previously set signals in the object pointed to by pThing. If the intention is to only add a signal to an existing signal mask, then use the PegSignalAdd macro instead.

Signals may not be combined when using this macro.

6.36 PegStatusIs

Synopsis:

```
PEGBOOL PegStatusIs(void *pThing, PEGUSHORT status);
```

Arguments:

pThing
        Pointer to a PegThing or derived object.
status
        The status to check.

Returns:

TRUE if pThing has status, otherwise, FALSE.

Description:

This macro checks the existence of status in pThing's status mask. If status is found, the return value is TRUE.

Errors:

None

Related Macros:

None

Notes:

It is not recommended that application code ever modify the status of an object at runtime. Doing so may cause unpredictable behavior on the part of the GUI objects.

6.37 PegStringIDGet

Synopsis:

```
    PEGSTRINGID PegStringIDGet(PegTextThing *pTextThing);
```

Arguments:

pTextThing

Pointer to a PegTextThing or derived object.

Returns:

The String ID associated with pTextThing.

Description:

This macro returns the string ID associated with this object. The string ID may be the constant PEG_NULL_STRING, which is not an error. Normal string ID's that correspond to a user defined string table begin at 0 and move in a positive direction. String ID's that correspond to the C/PEG system string table, begin at -2 and move in a negative direction.

Errors:

None

Related Macros:

None

Notes:

This macro is only available if PEG_STRING_TABLES and PEGSTRING_IS_ID are both defined.

6.38 PegTextGet

Synopsis:

```
PEGCHAR *PegTextGet(void *pTextThing);
```

Arguments:

pTextThing
    Pointer to a PegTextThing or derived object.

Returns:

A pointer to the string maintained by pTextThing.

Description:

This macro returns a pointer to the string maintained by pTextThing. It is not an error for this function to return NULL if the object does not have a string assigned to it.

Errors:

This macro may generate a run-time error if pTextThing is not a PegTextThing or derivative.

Related Macros:

PegTextSet

Notes:

6.39 PegTextSet

Synopsis:

```
void PegTextSet(void *pThing, const PEGCHAR *pText);
```

Arguments:

pThing
: Pointer to a PegTextThing or derived object.

pText
: Pointer to a NULL terminated string.

Returns:

None

Description:

This macro calls the text set function of pThing through pThing's text set function pointer.

Errors:

The text set function may generate an assert if pThing is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegTextThingTextSet

Notes:

This macro encapsulates the object and hides the details of calling the object's text set function. This macro should always be used by application code to call the object's text set function.

6.40 PegTypeGet

Synopsis:

```
    PEGUBYTE PegTypeGet(void *pThing);
```

Arguments:

pThing

    Pointer to a PegThing or derived object.

Returns:

The object type of pThing.

Description:

This function returns the object type of pThing.

Errors:

None

Related Macros:

PegTypeSet

Notes:

The base type is PEG_TYPE_THING. Every PegThing derived object is of at least this type.

See the “C/PEG Programmers Manual” for a description of types used in C/PEG.

6.41 PegTypeSet

Synopsis:

```
    void PegTypeSet(void *pThing, PEGUBYTE ubType);
```

Arguments:

pThing

    Pointer to a PegThing or derived object.

ubType

    The type to set pThing.

Returns:

None

Description:

This macro sets the type of pThing to ubType.

Errors:

None

Related Macros:

PegTypeGet

Notes:

See the “C/PEG Programmers Manual” for a description of types in C/PEG.

CHAPTER 7

C/PEG STRING LIBRARY

7.1 Overview

This section contains functions that C/PEG implements as a subset of the ANSI C string library.

Internally, C/PEG calls the C/PEG string library versions of common ANSI C functions such as strcpy and strlen. How these C/PEG function calls are mapped is configured in the configuration file pconfig.h. If the application developer wishes to use the standard ANSI C functions as provided by the compiler vendor, then the symbol PEG_STRLIB should not be defined.

Although string handling functions are defined in the ANSI standard, not all string libraries implemented by compiler vendors are created equal. In an effort to give the application developer consistency and portability, the C/PEG string library implements a wide range of functions for handling 8 bit ASCII and 16 bit Unicode character strings. The table below shows the ANSI C function call and the equivalent C/PEG function call.

ANSI C Function	C/PEG Function
strcat	PegStrCat
strncat	PegStrnCat
strcpy	PegStrCpy
strncpy	PegStrnCpy
strcmp	PegStrCmp
strncmp	PegStrnCmp
strlen	PegStrLen
atol	PegAtoL
atoi	PegAtoI

Table 2- C/PEG String Library Functions

The parameters to the C/PEG versions are identical to the ANSI C versions of the functions.

The first question that usually comes to mind is “Where is sprintf?”. The short answer is that sprintf is usually a problem waiting to happen. Most implementations of this function are not re-entrant, and are, therefore, not thread safe. Some complier vendors do make a thread safe version of this function, although they may very well call it something else. The goal of the C/PEG string library is to provide a consistent and portable string library, and sprintf does not fit in with this goal. Internally, the C/PEG library does not use sprintf. When compound formatting is necessary, as is the case for a PegProgressBar, the listed functions above are used to build the proper output string.

Part of the portability issue with using strings lies in the fact that some applications work with 8 bit ASCII character sets, while others work with 16 bit Unicode character sets. The C/PEG library supports either way. In some circumstances, it may even support both. To the application developer, the differences should be negligible, as outlined in the following table.

C/PEG Function maps to	**ANSI C (!PEG_STRLIB)&& (!PEG_UNICODE)**	**C/PEG ASCII PEG_STRLIB && (!PEG_UNICODE)**	**C/PEG Unicode PEG_STRLIB && PEG_UNICODE**
PegStrCat	strcat	PegStrCatA	PegStrCatU
PegStrnCat	strncat	PegStrnCatA	PegStrnCatU
PegStrCpy	strcpy	PegStrCpyA	PegStrCpyU
PegStrnCpy	strncpy	PegStrnCpyA	PegStrnCpyU
PegStrCmp	strcmp	PegStrCmpA	PegStrCmpU
PegStrnCmp	strncmp	PegStrnCmpA	PegStrnCmpU
PegStrLen	strlen	PegStrLenA	PegStrLenU
PegAtoL	atol	PegAtoLA	PegAtoLU
PegAtoI	atoi	PegAtoIA	PegAtoIU

Table 3- C/PEG String Function Mappings

Following the above table, one can see that using the PegStrCat function call actually maps the call to one of three possible choices, depending on how the C/PEG library build was configured.

For example, if PEG_STRLIB and PEG_UNICODE are not defined, then PegStrCat maps to the ANSI C strcat function. But, if PEG_STRLIB is defined, then the call will map to PegStrCatA (with the A at the end of the function name denoting the ASCII version of the function) which expects to work with 8 bit ASCII character strings. Finally, if PEG_UNICODE is defined, at which point, PEG_STRLIB will be automatically defined by the library, then PegStrCat will map to PegStrCatU (with the U at the end of the function name denoting the Unicode version of the function) which expects to work with 16 bit Unicode character strings.

It is important to note that if a particular version of the function must be used in a given situation, it is possible to call the function directly without using the common C/PEG mapping. For instance, if an application uses Unicode, but the underlying file system uses ASCII for file names, and the application developer wishes to concatenate a postfix to a file name, then it is perfectly acceptable to call PegStrCatA to do so. Internally, C/PEG always uses the general form of the function call which is always mapped to the native character width in which the application is running.

In general, it is usually always a good idea to use the C/PEG string library in the application code. Even if the application developer does not wish to use the C/PEG versions of these functions, it is a good idea to use the C/PEG form of calling the functions. This aids in portability issues as the application matures. It is very possible that an application may eventually end up with the need for Unicode support somewhere down the road, and using the C/PEG form of the functions insulates the application developer from the need to go back and fix all the instances where they may be calling the ANSI version of these functions.

With regard to wide character encoding function available with some compilers on some platforms (GCC on Unix is a good example), again, not all support is created equal. On most systems, a wchar_t is a 32 bit data type, often defined as an int. In C/PEG, a wide character is typically defined as 16 bits, which, of course, raises problems. The good news is, C/PEG offers the application developer a way to define the width of the wide character data type in C/PEG by defining PEGWIDECHAR in pconfig.h. If this symbol is not defined, then C/PEG defines it as an unsigned short int, which corresponds to the usual width of Unicode characters. In turn, the PEGCHAR data type, when PEG_UNICODE is enabled, is defined as a PEGWIDECHAR. This allows the application developer to configure the width of the wide data type.

C/PEG never uses the system wide char (often prefixed with wcs instead of str) since this would make the C/PEG string library non-portable.

CHAPTER 8

C/PEG STRING TABLE FUNCTIONS

8.1 Overview

This section contains functions that C/PEG uses in support of string tables. String tables in C/PEG are an integral part of application development for those types of applications that must support multi-lingual text display, either in 8 bit ASCII format or 16 bit Unicode format.

The C/PEG library provides a set of functions to work with tables of strings that are associated with a language. The strings themselves come from PEG WindowBuilder and are generated using the String Table Editor tool. Please see the PEG WindowBuilder documentation for more coverage of that topic.

8.2 PegLanguageGet

Synopsis:

```
    PEGINT PegLanguageGet(void);
```

Arguments:

None

Returns:

An integral value that denotes the language currently in use.

Description:

This function allows the application to discover the language currently in use. The integral value returned by this function will be one of the language names defined in the PEG WindowBuilder project which generated the string table in use by the application.

Errors:

None

Related Functions:

PegLanguageSet

Notes:

This function is only available if PEG_STRING_TABLES is defined.

8.3 PegLanguageMaxSet

Synopsis:

```
    void PegLanguageMaxSet(PEGINT iMax);
```

Arguments:

iMax

Integral value that sets the maximum number of languages.

Returns:

None

Description:

This function sets the maximum number of languages supported by the application. It is important this number not be greater than the number of languages in the language tables outputted by the String Table Editor in PEG WindowBuilder.

Errors:

None

Related Functions:

PegLanguageSet, PegLanguageGet

Notes:

This function is only available if PEG_STRING_TABLES is defined.

8.4 PegLanguageSet

Synopsis:

```
    void PegLanguageSet(PEGINT iNum);
```

Arguments:

iNum

Integral value that sets the current language.

Returns:

None

Description:

This function sets the current language that the system will be running in. If the application is running in string ID mode, then all objects that are descendents of the PegPresentation object are informed that the language has changed.

Errors:

None

Related Functions:

PegLanguageGet, PegLanguageMaxSet

Notes:

This function is only available if PEG_STRING_TABLES is defined.

8.5 PegLanguageStringLookup

Synopsis:

```
const PEGCHAR *PegLanguageStringLookup(PEGSTRINGID id);
const PEGCHAR *PLSL(PEGSTRINGID id);
```

Arguments:

id

Integral value of the string ID to lookup.

Returns:

A const PEGCHAR pointer to the string.

Description:

This function returns the string associated with id for the currently selected language. The macro form of this function, PLSL, may be used as shorthand for the function call.

Errors:

If id is PEG_NULL_STRING, the return value will be NULL.

Related Functions:

PegLanguageGet, PegLanguageMaxSet, PegLanguageSet, PegStringTableAssign

Notes:

This function is only available if PEG_STRING_TABLES is defined.

8.6 PegStringTableAssign

Synopsis:

```
void PegStringTableAssign(const PEGHCAR ***pTable, PEGINT
  iNumLanguages);
```

Arguments:

pTable

Pointer to the new language table.

iNumLanguages

The number of languages in pTable.

Returns:

None

Description:

By default, the user string table points to the string table generated by PEG WindowBuilder with the name of wbStringTable. At run time, this table may be substituted for an alternative string table.

Errors:

None

Related Functions:

PegLanguageGet, PegLanguageMaxSet, PegLanguageSet

Notes:

This function is only available if PEG_STRING_TABLES is defined.

CHAPTER 9

C/PEG MEMORY MANAGEMENT FUNCTIONS

9.1 Overview

This section deals with a group of functions that relate to memory management.

In the embedded systems world, when dealing with compact micro-kernel operating systems and stripped down compiler libraries, it is often the case where the memory management requirements of the system out pace the services provided. A common example of this is dealing with a multi-tasking operating system coupled with a compiler which does not provide task aware system libraries. In this case, it would be very dangerous to allocate any memory on the heap from more than one task. There would never be any guarantees of the memory's integrity.

Another scenario that is serviced by these functions is one of managing excess video memory. On high end hardware, it is typical for the video controller to include more memory to than what is necessary to support both the visible frame buffer and back buffer. The excess memory can then be utilized to store bitmaps in use by the application. This usually provides quite a boost in speed for graphically intensive applications.

For the most part, these functions are invisible to the user. C/PEG works with these when instructed to do so (when PEG_HEAP_MANAGER and/or PEG_VID_MEM_MANAGER is defined). The application is also free to use these functions to manager a block of memory if necessary. Usually, the block is set aside on the system by constructing a linker file to ignore certain portions of system memory. The address is then given to the PegHeapManagerInitialize function for the heap manager to maintain. The application would then use PegByteAlloc and PegByteFree to allocate and free sections of memory. Internally, when necessary, the macros PEG_ALLOC and PEG_FREE are mapped to PegByteAlloc and PegByteFree.

The remaining functions are available only if PEG_HEAP_DEBUG is defined. These functions allow the application to discover usage and test the integrity of the heap. Usually, these are not necessary in a production system.

9.2 PegHeapManagerInitialize

Synopsis:

```
    void PegHeapManagerInitialize(PegHeapManager *phm, void *pmem,
      PEGULONG size);
```

Arguments:

phm

Pointer to a PegHeapManager object.

pmem

Pointer to the base address of available memory.

size

The size of the memory to use at pmem.

Returns:

None

Description:

This function initializes the PegHeapManager pointed to by phm. The memory pointed to by pmem should not be part of any section of the application. It is also important that size not overrun a application section or the end of memory.

Errors:

This function may generate an assert if phm is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegByteAlloc, PegByteFree

Notes:

This is the function to call after a PegHeapManager object instance has been created to properly initialize the instance. This sets the internal memory pointers and memory size as well as initializes the free and used lists of phm. After calling this function, it is possible to use phm for allocating and freeing memory in the block pointed to by pmem.

9.3 PegByteAlloc

Synopsis:

```
void PegByteAlloc(PegHeapManager *phm, void **pmem, PEGULONG
  size);
```

Arguments:

phm
: Pointer to a PegHeapManager object.

pmem
: Pointer to a void pointer which will receive the memory address of the allocated memory block.

size
: Size of the memory block to allocate.

Returns:

Returns the address of the allocated memory in pmem. If it is not able to allocate a block of size, pmem will contain NULL.

Description:

This function replaces the ANSI C malloc function to allocate memory managed by the PegHeapManager pointed to by phm. If the size requested is larger than any available free blocks of memory, then NULL is returned; therefore, it is important for the caller to check the return value before making assumptions regarding the memory.

Errors:

This function may generate an assert if phm or pmem is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegByteFree, pegHeapTrap

Notes:

It is very important to remember that if the memory pool managed by phm is in use by more than one task, the caller must use system serialization functions to protect calls when allocating or freeing memory from the pool. There is no internal protection mechanism for either allocating or freeing memory.

The size which is asked for may not always match the size of the block allocated. The function rounds size so that it always allocates a block of memory which is divisible by 4 to make memory access efficient.

This function also uses an efficient algorithm to split large blocks into smaller pieces when small amounts of memory are requested. The threshold for splitting blocks depends on the size being requested as well as the define PEG_HEAP_BLOCK_SPLIT_THRESHOLD. By default, this is set to 256 bytes, but can be adjusted up or down depending on the requirements of the system.

When a block is requested, the requested size is combined with the threshold. If any free block is larger than this size, it is split, with a portion being moved to the used blocks and the remainder being returned to the free pool.

If PEG_HEAP_TRAP_ENABLE is defined and the function is not able to allocate the necessary memory, it will call the function pegHeapTrap. pegHeapTrap is a sink hole for trapping program execution when Bad Things happen in the heap management. This allows the application developer the opportunity to stop the debugger and easily trace the stack back to the point in the application where memory had been exhausted.

9.4 PegByteFree

Synopsis:

```
    void PegByteFree(PegHeapManager *phm, void *pmem);
```

Arguments:

phm

    Pointer to a PegHeapManager object.

pmem

    Pointer to the base address of the memory to be freed.

Returns:

None

Description:

This function returns the memory pointed to by pmem to the free pool managed by phm.

Errors:

This function may generate an assert if phm is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegByteAlloc, pegHeapTrap

Notes:

It is very important to remember that if the memory pool managed by phm is in use by more than one task, the caller must use system serialization functions to protect calls when allocating or freeing memory from the pool. There is no internal protection mechanism for either allocating or freeing memory.

Only call this function to release memory which was acquired by calling PegByteAlloc.

For every block of memory assigned by phm, the header portion contains a magic number which the heap manager uses to determine if the block of memory is valid. If the memory was over written by the application, then the magic number would be corrupt. If the heap manager detects a corrupted magic number and PEG_HEAP_TRAP_ENABLE is defined, then pegHeapTrap is called and the execution goes into an infinite loop.

If the block being released is verified as being valid, then it is put back into the free pool. When being put back into the pool, the function attempts to combine the newly freed block with a previous block. If this is unsuccessful, it then puts the free block at the head of the free block list and combines it with the previous first block, if there was one. This is an efficient way to help maintain integrity and avoid problems arising from the defragmentation of the memory pool.

9.5 PegHeapListVerify

Synopsis:

```
    PEGBOOL PegHeapListVerify(PegHeapManager *phm);
```

Arguments:

phm

Pointer to a PegHeapManager object.

Returns:

TRUE if the list is successfully verified, otherwise, FALSE.

Description:

This function walks the linked list of memory blocks managed by phm and verifies the list pointer’s integrity.

Errors:

This function may generate an assert if phm is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegHeapSizeVerify

Notes:

This function is only available when PEG_HEAP_DEBUG is enabled and is meant to be used to verify the integrity of the heap during debug sessions.

9.6 PegHeapNumFreeBlocks

Synopsis:

```
    PEGINT PegHeapNumFreeBlocks(PegHeapManager *phm);
```

Arguments:

phm

Pointer to a PegHeapManager object.

Returns:

The number of free blocks in the pool maintained by phm.

Description:

This function counts the number of free blocks in the pool maintained by phm.

Errors:

This function may generate an assert if phm is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegHeapNumUsedBlocks

Notes:

This function is only available when PEG_HEAP_DEBUG is enabled and is meant to be used to verify the integrity of the heap during debug sessions.

9.7 PegHeapNumUsedBlocks

Synopsis:

```
    PEGINT PegHeapNumUsedBlocks(PegHeapManager *phm);
```

Arguments:

phm

Pointer to a PegHeapManager object.

Returns:

The number of used blocks in the memory pool maintained by phm.

Description:

This function walks the list of used blocks in the memory pool and returns the number of blocks.

Errors:

This function may generate an assert if phm is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegHeapNumFreeBlocks

Notes:

This function is only available when PEG_HEAP_DEBUG is enabled and is meant to be used to verify the integrity of the heap during debug sessions.

9.8 PegHeapSizeFreeMem

Synopsis:

```
    PEGULONG PegHeapSizeFreeMem(PegHeapManager *phm);
```

Arguments:

phm

Pointer to a PegHeapManager object.

Returns:

The total size of free memory in the pool maintained by phm.

Description:

This function walks the list of free memory and adds up the size of each free block. This sum is for informational purposes and does not necessarily imply that all of this memory is available for use by the application.

Errors:

This function may generate an assert if phm is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegHeapSizeUsedMem, PegHeapSizeVerify

Notes:

This function is only available when PEG_HEAP_DEBUG is enabled and is meant to be used to verify the integrity of the heap during debug sessions.

The returned sum does not include the size of the header for each block.

9.9 PegHeapSizeUsedMem

Synopsis:

```
PEGULONG PegHeapSizeUsedMem(PegHeapManager *phm);
```

Arguments:

phm

Pointer to a PegHeapManager object.

Returns:

The total size of used memory in the pool maintained by phm.

Description:

This function walks the list of used memory and adds up the size of each used block.

Errors:

This function may generate an assert if phm is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegHeapSizeFreeMem, PegHeapSizeVerify

Notes:

This function is only available when PEG_HEAP_DEBUG is enabled and is meant to be used to verify the integrity of the heap during debug sessions.

The returned sum does not include the size of the header for each block.

9.10 PegHeapSizeVerify

Synopsis:

```
PEGBOOL PegHeapSizeVerify(PegHeapManager *phm);
```

Arguments:

phm

Pointer to a PegHeapManager object.

Returns:

Returns TRUE if the sum of all blocks and headers equals the size of the memory pool originally assigned to phm. Otherwise, returns FALSE.

Description:

This function sums the headers and blocks in both the free and used lists and checks this value against the original size value it was assigned. This is a very good integrity check for the memory pool.

Errors:

This function may generate an assert if phm is invalid and the C/PEG library was built with the PEG_USE_ASSERT directive defined.

Related Functions:

PegHeapNumFreeBlocks, PegHeapNumUsedBlock, PegHeapSizeFreeMem, PegHeapSizeUsedMem

Notes:

This function is only available when PEG_HEAP_DEBUG is enabled and is meant to be used to verify the integrity of the heap during debug sessions.

9.11 pegHeapTrap

Synopsis:

```
    void pegHeapTrap(void);
```

Arguments:

None

Returns:

None

Description:

This function may be called when there is a problem either allocating or freeing memory.

Errors:

None

Related Functions:

PegByteAlloc, PegByteFree

Notes:

Even though this is a static function, and not directly available from application code, it has been included here since the application developer should have knowledge of this function if they are using the heap management facilities in their application.

CHAPTER 10

C/PEG INTEGRATION FUNCTIONS

10.1 Overview

This section contains functions that the application must implement that are called from within the C/PEG library.

10.2 PegAppInitialize

Synopsis:

```
    void PegAppInitialize(PegPresentation *pPresent);
```

Arguments:

pPresent
    Pointer to the PegPresentation object.

Returns:

None

Description:

On most platforms that C/PEG supports, this function is the entry point into user application code.

Usually, C/PEG implements the 'main' function and is allowed to set up the screen, message queue and presentation objects before calling into user code. Therefore, the application can view this function as its 'main' function when running with C/PEG.

A typical use for this function would be for the application to put at least a C/PEG object on the PegPresentation object.

Errors:

It is important that the application returns from this function without executing any PegPanel or derived objects. It is fine to add the objects to the PegPresentation object, but, by executing a PegPanel object within this context blocks the C/PEG message loop from starting, and will thus freeze the application.

Related Functions:

main

Notes:

On some platforms, this function may be implemented differently depending on how the operating system defines application startup.

10.3 PegIdleFunction

Synopsis:

```
    void PegIdleFunction(void);
```

Arguments:

None

Returns:

None

Description:

This function is called by the PegMessagQueue object when the main task's message queue is empty.

This function is only used on systems that are running in stand alone or single threaded mode.

Errors:

None

Related Functions:

PegMessageQueuePop

Notes:

This function must be filled in by the system designer when integrating C/PEG into the target operating system, if any, and hardware.

Please see the “C/PEG Programmers Manual” for a discussion on how to implement this function in the application.

10.4 PegKeyInputInit

Synopsis:

```
void PegKeyInputInit(void *pData);
```

Arguments:

pData

    void pointer to user defined data.

Returns:

None

Description:

This function is responsible for initializing any key input devices on the system.

Errors:

None

Related Functions:

PegKeyInputShutdown

Notes:

This function is not part of the core C/PEG library. It is used by the integration to initialize any key input devices on the system when PEG_KEYBOARD_SUPPORT is turned on.

The pData parameter is implementation specific and may be NULL. It is often used to pass a PegMessageQueue pointer to the key input driver as a destination for key input messages.

10.5 PegKeyInputShutdown

Synopsis:

```
void PegKeyInputShutdown(void *pData);
```

Arguments:

pData
    void pointer to user defined data.

Returns:

None

Description:

This function releases any key input devices on the system when the system is shutting down.

Errors:

None

Related Functions:

PegKeyInputInit

Notes:

This function is not part of the core C/PEG library. It is called by the integration when the system is shutting down.

The pData parameter is driver specific and may be NULL.

10.6 PegPointerInputInit

Synopsis:

```
    void PegPointerInputInit(void *pData);
```

Arguments:

pData
    void pointer to user data.

Returns:

None

Description:

This function initializes the pointer device on the system.

Errors:

None

Related Functions:

PegPointerInputShutdown

Notes:

This function is not part of the C/PEG core library. It is used to initialize the system pointer device when either PEG_MOUSE_SUPPORT or PEG_TOUCH_SUPPORT is turned on.

This function is called by the initialization routine in the platform integration startup.

The pData parameter is implementation specific and may be NULL. It is often used to pass a PegMessageQueue pointer to the pointer input driver as a destination for pointer input messages.

10.7 PegPointerInputShutdown

Synopsis:

```
    void PegPointerInputShutdown(void *pData);
```

Arguments:

pData

void pointer to user defined data.

Returns:

None

Description:

This function shuts downs and releases the system pointer device.

Errors:

None

Related Functions:

PegPointerInputInit

Notes:

This function is not part of the core C/PEG library. It is called by the integration when the system is shutting down.