FAQ Guide

Software Development

Table of Contents

Table of Contents

Software Development FAQ

For all Properties, the function names are documented in the form of a Get/Set pair for (Visual C++). For Visual Basic, VBA, Powerbuilder, etc. users, just use the property name itself, without the Get/Set prefix.

The full name is used when the object is referenced from a Visual C++ application. There are syntax differences between the Visual C++ syntax noted herein and VB or VBA syntax, most notably involving the use of the parenthesis in the argument.

To create a single instance of the control called SigPlusControl_1:

Set SigPlusControl_1 = CreateObject("SIGPLUS.SigPlusCtrl.1")

The control names for using CreateObject Script are:

"SIGPLUS.SigPlusCtrl.1"
"SIGSIGN.SigSignCtrl.1"

The following is how to write a BMP to the SignatureGem® LCD 4x3 signature pad using a Windows API bitmap handle call within PowerBuilder.

1. Declare the following external function to the Windows API:

   ```
   FUNCTION ULONG LoadImageA (ULONG a_hInst, REF STRING a_lpszName, &
   ULONG a_uType, LONG a_cxDesired, LONG a_cyDesired, ULONG a_fuLoad) &
   LIBRARY "USER32.DLL"
   ```

2. The following code gets the bitmap handle using the above windows API function and writes the BMP to the pad.

   ```
   string ls_filename
   ulong hbitmap
   ls_filename = "c\sigplus.bmp"
   hbitmap = LoadImageA(0, ls_filename, 0, 0, 0, 16)
   ole_SigPlus.object.LCDWriteBitmap(1, 2, 0, 0, 240, 128, hbitmap)
   ```

General Software Development FAQ

How do I bind a signature to a document?

A signature is bound to a document or data by using common cryptography and secure hashing techniques. If the document data is changed, the binding prevents the electronic signature from being viewed or printed. Binding only applies to .sig-format data and not to image or bitmap data. If you need to use image files for reports, viewing, or printing, they can be generated from the control using the WriteImageFile whenever the signature is rendered in the control.

To bind an electronic signature to a document when the signature is stored separately from the document:

1. Capture the signature with EncryptionMode set to 0.
2. Use AutokeyData, with or without first using AutoKeyStart (see AutoKeyStart in the SigPlus Developers Guide for details). When done passing in data, use AutoKeyFinish.
3. Set the EncryptionMode > 0.
4. Use GetSigString to return the signature as an ASCII-hex string (alternately, ExportSigFile writes the signature as a file to a location you specify).

NOTE: When using EncryptionMode = 2 especially, the sequence of steps is important.

To read in and verify the binding of the signature to the document file, the steps are similar:

1. Set EncryptionMode to 0 and clear the control (ClearTablet).
2. Use AutokeyData, with or without first using AutoKeyStart (see AutoKeyStart in the SigPlus Developers Guide for details). When done passing in data, use AutoKeyFinish.
3. Set the EncryptionMode > 0.
4. Use SetSigString to return the ASCII-hex string to the new instance of SigPlus (alternately, ImportSigFile imports the signature from the file location you specified).
5. To verify the binding view the signature and check automatically to determine the number of strokes, or simply inspect the return value that is generated when using the ImportSigFile or SetSigData methods. If returned successfully, the signature will render automatically.

In Visual Basic, to Autokey to a path and file name, use the following technique:

```
Dim text1 As String
text1 = "c:\windows\file.ext"; note that this should contain your path and file name, the quotes are needed.
SigPlus1.AutoKeyData = text1
SigPlus1.AutoKeyFinish
SigPlus1.EncryptionMode = 2
```

In Visual Basic, to Autokey to string literals, use the following technique:

Dim text1 As String
text1 = "c:\windows\file.ext"; note that this should contain your path and file name, the quotes are needed.
SigPlus1.AutoKeyStart
SigPlus1.AutoKeyData = "this is my sample data"
SigPlus1.AutoKeyData = "add as much as desired"
SigPlus1.AutoKeyFinish
SigPlus1.EncryptionMode = 2

To bind an electronic signature to a document when the signature is stored within the document (within each instance of the control placed in the document, note the following example written in VBA for Word:

Private Sub Done_Click(); This is the signature encryption step, binding to the document. The signature would have already been captured into the control by setting EncryptionMode = 0 and then TabletState = 1.
SigPlus1.TabletState = 0; Always set TabletState = 0 when done with electronic signature capture.

Dim text3 As String
text3 = ActiveDocument.Content; This is a Word VBA command.
SigPlus1.AutoKeyStart
SigPlus1.AutoKeyData = text3
SigPlus1.AutoKeyFinish
SigPlus1.EncryptionMode = 1; or Encryption Mode = 2;
SigPlus1.ExportSigFile (""); ("") is used to store the electronic signature within the control
End Sub

Private Sub Document_Open(); This is a Word event that occurs on document opening
SigPlus1.ClearTablet ; Clears the control prior to keying and signature import.
SigPlus1.EncryptionMode = 0 ; Clears EncryptionMode
Dim text3 As String
text3 = ActiveDocument.Content ; uses the content of the document to generate sig encryption key.
SigPlus1.AutoKeyStart ; here the Autokey mode is used to bind to text data
SigPlus1.AutoKeyData = text3 ; rather than a file
SigPlus1.AutoKeyFinish SigPlus1.JustifyMode = 5 ; this does the auto-zooming to fill white space.
SigPlus1.DisplayPenWidth = 12 ; this sets the display and printing pen width to suit your application
SigPlus1.EncryptionMode = 1 ; or EncryptionMode = 2;
SigPlus1.ImportSigFile ("") ; ("") is used to retrieve the electronic signature within the control.
End Sub

What is the difference between a .sig format electronic signature and a bitmap?

Topaz® .sig format data is a vector file containing the original signature data from the pad at high speed. This means that the .sig data represents the entire signature and the exact sequence of all of the points, loops, strokes, and timing data.

When do I set TabletState to true and false?

The TabletState is what turns on and off the capture of signatures via the pen and pad. Use TabletState = 1 to capture signatures, and use TabletState = 0, the default, to turn capture off.

If you are using the JustifyMode = 5 (AutoJustify) to zoom the signature to fill the control and eliminate white space from the control, the TabletState must be set to zero. The AutoJustify action will only occur when TabletState = 0.

Be sure to set TabletState = 0 when changing tablet type. Also be sure to set TabletState = 0 before exiting the application.

How do I get rid of the white space around a signature?

Be sure to set JustifyMode in your application code. The signature will zoom with the correct aspect ratio into any size of control, from anywhere on the pad, if TabletState = 0 and if the justify mode is being used.

0 Normal, no justification.
1 Auto-Justify signature to upper left corner of signature box.
2 Auto-Justify signature to upper right corner of signature box.
3 Auto-Justify signature to lower right corner of signature box.
4 Auto-Justify signature to lower-left corner of signature box.
5 Auto-Justify signature to center of signature box.

How do I save an electronic signature in a Memo field in MS Access?

To save a Topaz signature in a Memo field of a database, extract the signature using the SigString property of SigPlus.

To put the signature into the control:
SigCtl.SigString = SigData

To get the electronic signature from the control:
SigData = SigCtl.SigString

Where is the forensic and biometric information?

All of the data needed for forensic and biometric verification of an electronic signature is stored in the Topaz format signature. The Topaz format signature can either be a .sig file or the ASCII hex-converted SigString. You cannot view this forensic and biometric information. The objective of the Topaz system is to only allow forensic or biometric characteristics to be available to a handwriting expert and forensic document examiner.

How do I automatically detect if the pad is plugged in?

Use the TabletConnectQuery() function. A pad plug-in detection scheme is shown below and assumes that the COM port was set in the Sigplus.ini file during install.

```
Dim TabletConnect as Boolean
TabletConnect = SigPlus1.TabletConnectQuery()
If TabletConnect = True Then
MsgBox "Tablet connected"
Else
MsgBox "No Tablet Detected"
End If
```

Can I sign files that are not traditional documents?

Yes, a simple custom application can be made to sign all kinds of files, drawings, notes, or documents for approval and archival purposes. To do so, the electronic file to be signed is represented by the file.ext in the example below:

```
Dim text1 As String
text1 = "c:\windows\file.ext" 'note that this should contain your path and file name, the quotes are needed.
'SigPlus1.AutoKeyStart 'MAKE SURE THIS IS COMMENTED OUT
SigPlus1.AutoKeyData = text1
SigPlus1.EncryptionMode = 2
```

Can I have multiple signatures in the same document?

Yes, you can add multiple signatures to the same document. There are several ways to do so. You can use SigPlus controls as placeholders and import the signatures from an external forms program or a database. Or you can store each signature within the control instance itself, such as we do in the supplied Topaz MS Word Plug-In. In this case, each signature is stored within the instance of the control and the signature travels along in the control and embedded in the document. Also, the Topaz Acrobat Plug-In will publish multiple signatures into truly portable .pdf documents.

What options do I have for Linux and Unix environments?

Topaz offers a C Library and also a SigPlus Java API you can use in Linux or Unix.

SigPlus ActiveX Software Development FAQ

When and how should I create an image file?

Image files of an electronic signature are useful in many applications for printing, viewing or merging into reports. They allow a significant level of versatility with a wide range of application programs. For example, in an environment where you cannot instantiate a SigPlus control, an image of the signature might serve as a good means for rendering the signature. The method for writing an image file is:

SigPlus1.WriteImageFile c:\test.bmp

See the property "ImageFileFormat" to set an image type other than bitmap. Other supported image formats include JPG, TIF, WMF, and EMF The full file name must be provided, including the standard file extension.

How do I set the ink thickness for display, printing, and images?

The ink thickness for direct display and printing of the signature from the control is set by the DisplayPenWidth property, called from your code. The width specified is in pixels:

SigPlus1.DisplayPenWidth = 12

The ink thickness for creating image files and bitmaps of the signature from the control is set by the ImagePenWidth property. The width is specified in pixels. Default is 1. This property does not affect the pen width shown in the control signature window. You will notice a natural interaction in perceived pen thickness depending upon the x and y resolution selected for the image.

SigPlus1.ImagePenWidth = 4

How do I set the resolution of the signature file or an image file?

The resolution of a signature file or image file is normally set by default upon software installation to match the resolution of the pad being used. To change the resolution of an image file, use the ImageXSize and ImageYSize properties to directly set the resolution of the image file. We recommend that the aspect ratio of the image file match the aspect ratio of the control. Normally, with the SignatureGem pad settings, you would get a bitmap resolution of 2000 x 600, which are the LogicalX and Y values. To reduce the resolution by a factor of 2, you would use:

SigPlus1.ImageXSize = 1000
SigPlus1.ImageYSize = 300

ImageSize defaults to the pad’s LogicalXSize and LogicalYSize values.

How do I automatically detect if the pad is plugged in?

Use the TabletConnectQuery() function. This is the preferred method in SigPlus 4.29 and above. A pad plug-in detection scheme is shown below and assumes that the COM port was set in the Sigplus.ini file during install.

```
Dim TabletConnect as Boolean
TabletConnect = SigPlus1.TabletConnectQuery()
If TabletConnect = True Then
MsgBox "Tablet connected"
Else
MsgBox "No Tablet Detected"
End If
```

How do I save and display the time and date stamp and annotation?

To set the time and date and annotation stamps to display and to be sure to save the time and date and annotation data within the .sig date, you must set the following properties:

```
SigPlus1.DisplayAnnotate = True
SigPlus1.DisplayTimeStamp = True
SigPlus1.SaveSigInfo = True
SigPlus1.AutoTimeStamp
SigPlus1.Annotation="My Annotation String"
```

How do I set the size of the time and date stamp and annotation?

To set the time and date and annotation stamp sizes use the properties shown below. The size of the text is specified proportional to the LogicalYSize settings of that particular control instance. If the LogicalYSize is 600 for example, the size of the text in the example below would be 1/3 of the control height (200/600).

```
SigPlus1.DisplayAnnotateSize = 200
SigPlus1.DisplayTimeStampSize = 200
```

How can I clear and reset SigPlus to be ready to capture and bind a new signature?

In order to successfully clear and reset SigPlus so it is ready to capture and bind a new signature, when it has already been used to capture and bind a signature using the AutoKey methods, and setting the EncryptionMode property > 0, use the following code:

```
SigPlus1.ClearTablet
SigPlus1.KeyString = "0000000000000000"
SigPlus1.SigCompressionMode = 0
SigPlus1.EncryptionMode = 0
SigPlus.TabletState = 1
```

SigPlus is now clear, reset, and ready to capture and bind a new signature.

When I instantiate SigPlus at runtime, do I need to do anything in particular?

Anytime SigPlus is created dynamically, rather than at Design Time, it is important that you call the InitSigPlus() method. Additionally, setting the TabletInvisible property to True is important if SigPlus is not going to be visible on any form. For example (VB sample code):

```
Dim SigPlus1 As Object
Set SigPlus1 = CreateObject("SIGPLUS.SigPlusCtrl.1")
SigPlus1.InitSigPlus
SigPlus1.TabletInvisible = True
```

SigWeb Software Development FAQ

What is the SigWebTablet.js file, and do I need to use it?

The SigWebTablet.js file contains function calls from the SigWeb service and your web page's call to SigWeb. It is not included in the SigWeb installer, but is installed on the webpage by the developer. The SigWebTablet.js file should be placed into the same location on your web server as the page that hosts SigWeb. Alternately, you may put the SigWebTablet.js file anywhere on your web server that you prefer, as long as you reference this location in the <script> tag's "src" parameter.

You will need to add a <script> tag to your web page to reference the SigWebTablet.js file. Here is an example: <script type=text/javascript" src="SigWebTablet.js"></script>

Download the latest version of the SigWebTablet.js file; Topaz recommends that you do not alter the SigWebTablet.js file.

Is it important to deactivate signature capture before the web page closes?

It is critical that you deactivate the connection to the signature pad every time before you close the page. The appropriate call to do this is: SetTabletState(0, tmr);

Note that the second argument in the SetTabletState() call above is the target of the call to SetTabletState() when used to open it originally. For example:

```
var tmr;
tmr = SetTabletState(1, ctx, 50) || tmr;
```

assuming that 'ctx' is the context for your web page's canvas.

When I sign in Chrome using SigWeb, I do not see the signature as I am signing. However, I do see the signature in other browsers.

As you sign, SigWeb draws the signature into the canvas signing area in real time. Unlike other browsers, Chrome saves temporary files associated with this process on your computer. Chrome deletes these temporary files when the browser closes normally. In rare cases, when Chrome does not close normally, the files may not be deleted properly. Delete the temporary files left by Chrome from the following folder: Windows\Users\%UserName%\AppData\Local\Temp. After the files have been deleted, update your web server to use the latest version of SigWebTablet.js to mitigate this.

I updated Chrome, and SigWeb no longer captures signatures.

Chrome Version 57 or higher should use SigWeb Version 1.5 or higher. To set-up, follow these steps:

1. Log in as Administrator to your computer.
2. In your list of Services (Start - Control Panel - Administrative Tools - Services), locate the "Topaz SigWeb Tablet Service", and stop it.
3. Under Programs and Features (Start - Control Panel - Programs and Features), right-click on "SigWeb" and choose "Uninstall". Allow the uninstall to complete.
4. If you also locate "Topaz SigWeb Tablet" in the list under Programs and Features (you may or may not find it listed), please uninstall it as well.
5. Once done, look on the back of your signature pad to find your model number. Then, follow the SigWeb Installation Steps (this is critical to successful installation).

When I sign using SigWeb in IE11, I don't see the signature in the signature box, but after signing and clicking "Done", the data is provided. How can IE11 be set to show the signature when signing?

In Internet Explorer 11, navigate to "Tools"-"Internet Options"-"General"-"Browser History"-"Settings". Change this to read "Every time I visit the web page".

SigPlus Java Software Development FAQ

Where should I place the SigUsb.dll when using SigPlus Java with HSB pads in Windows?

For fastest results when capturing signatures, Topaz recommends SigUsb.dll be placed in your machine's SYS directory (either System32 or SysWOW64, depending on your OS). The DLL will be found by Windows much more quickly here than if it resides in another folder. Alternately, you can place the SigUsb.dll in any location specified in the PATH variable.

How do I run the Topaz Java Bean headless under Linux?

You'll need to set the java.awt.headless property to true (allowing use of the awt library), using the line of code below (under main is the suggested location):

System.setProperty("java.awt.headless","true");

Additionally, you will need to have XWindows installed, however it does not have to be running.

SigPlusNET Software Development FAQ

Do I need separate versions of SigPlusNET for 32-bit and 64-bit Windows?

The current version of SigPlusNET is 'AnyCPU' compatible, meaning that it will run appropriately in either 32-bit or 64-bit mode depending upon your .NET project's platform.

LCD Software Development FAQ

When discussing x, y positions and sizes below, it is important to keep in mind the LCD specs of each Topaz® pad (when using coordinates in your method calls, be sure to keep your positions and sizes within these boundaries):

LCD 1x5 Pads: 240px x 64px (x, y origin: 0, 0 when using LCD coordinates)
LCD 4x3 Pads: 240px x 128px (x, y origin: 0, 0 when using LCD coordinates)
LCD 4x5 Pads: 320px x 240px (x, y origin: 0, 0 when using LCD coordinates)
Color 5.7 Pads: 640px x 480px (x, y origin: 0,0 when using LCD coordinates)
Color 4.3 Pads: 480px x 272px (x, y origin: 0,0 when using LCD coordinates)

How do I write images to the LCD?

ActiveX Example: Using either the LCDWriteBitmap() method or the LCDWriteFile() method. Both methods take the following 6 arguments: Dest, Mode, XPosition, YPosition, XSize, YSize

Dest: 0=foreground, 1=background
Mode: 0=Clear, 1=Complement, 2=Write Opaque, 3=Write Transparent (mode 2 is most-often used)
XPosition: The x coord location of the upper-left corner of the image
YPosition: The y coord location of the upper-left corner of the image
XSize: The width of the image to display in px
YSize: The height of the image to display in px

The final argument is the difference between the two methods. LCDWriteBitmap() takes a Windows handle to the image (HBITMAP). For example, this can be obtained in Visual Basic by adding an Image object to the application, setting the Picture property to the image in question, and then using th Handle property of that object to assign to this argument. LCDWriteFile() instead takes the path to the file in question. For example, "C:\somefolder\myimage.bmp". In either case, the image must be a b&w 1-bit bmp file.

How do I manage hotspots correctly?

ActiveX Example: The KeyPadAddHotspot() method adds hotspots to the lcd tablet: hotspots are used to determine if the user has tapped within a specific area of the LCD, so the appropriate action can be taken in code. For example, if you place a "clear button" image on the LCD, you'll probably want to place a hotspot under this button so you'll know if the user has tapped it with the pen, and then take the appropriate action in code. KeyPadAddHotspot() takes the following 6 arguments: KeyCode, Coords, XPos, YPos, XSize, YSize

Keycode: This is essentially a hotspot numbering system, controlled by the developer Coords: Coordinate system used for this hotspot: 0 = Logical tablet coordinates, 1 = LCD Coordinates (Topaz always recommends using 1 = LCD Coordinates).

XPosition: The x coord location of the upper-left corner of the hotspot
YPosition: The y coord location of the upper-left corner of the hotspot
XSize: The width of the hotspot in px
YSize: The height of the hotspot in px

For example: KeyPadAddHotspot(0, 1, 20, 30, 40, 40)
This is hotspot #0, using LCD coordinates. This hotspot's upper left corner is at 20, 30. It is 40 px wide and 40 px tall.

Hotspots have several characteristics worth mentioning that developers should note:

1. The KeyPadAddHotspot() method will require slight variations in px coord location (arguments 3 and 4, varying for about 1 px to 5 px, the closer you get to the bottom of the lcd) from its counterpart LCDWriteBitmap(), LCDWriteFile(), or LCDWriteString() method calls. In other words, whereas LCDWriteBitmap(), LCDWriteFile(), or LCDWriteString() will line up precisely as expected, hotspots will start shifting slightly from 1 to 5 px as you move down the LCD.

   For example, if you called:

   LCDWriteBitmap(0, 2, 20, 90, 50, 30)

   your call to KeyPadAddHotSpot might be something like:

   KeyPadAddHotspot(1, 1, 18, 87, 52, 33)

   For best results, Topaz recommends the following in terms of adding hotspots:

   a. Make the hotspot larger than the image/text representing it. This eliminates “hunting and tapping” on the user's part.
   b. Make all hotspots no smaller than 10 px in both height and width. c. Leave at least 10 px of space between hotspots.

2. Do not add all the hotspots you will be using in your application at once if your users will be moving from one "screen" to another. In other words, if you have, for example, a page that the user first makes choices on, then they click something like "OK" (or equivalent) to move to the next "screen" where they might sign, you will need to create the hotspots for each page separately. You do this by:

   a. Creating the hotspots for screen 1 ONLY. When you move to the next screen, do item "b."
   b. Calling KeyPadClearHotSpotList() to clear the hotspots out, then doing item "c."
   c. Creating the hotspots for the next screen, and so on.

   For example, let's assume I plan to display 2 screens: the first uses 3 hotspots, and the second uses 2 hotspots.

   I would create the first three hotspots something like the following:
   KeyPadAddHotspot(0, 1, 0, 10, 40, 20)
   KeyPadAddHotspot(1, 1, 0, 50, 40, 20)
   KeyPadAddHotspot(2, 1, 0, 90, 40, 20)

When it's time to move to the next screen, first call:
KeyPadClearHotSpotList()

Then, create screen 2 hotspots, something like:
KeyPadAddHotspot(3, 1, 20, 10, 20, 10)
KeyPadAddHotspot(4, 1, 80, 10, 20, 10)

The issue involved is possible overlapping hotspots, which will cause issues to arise.

How do I tell when a user has tapped a hotspot?

ActiveX Example: Let's assume you have hotspots created as outlined above. You can choose to manage them either by polling at a given rate of time (perhaps 200-300 milliseconds or so) or you can use the SigPlus-provided pen events (Pen_Up and Pen_Down).

Within the polling or pen event routine, you call a method KeyPadQueryHotSpot(), which takes the following argument:

KeyCode: This is the number assignment of the hotspot, as assigned in the Keycode argument of the KeyPadAddHotspot() method: Call this method within an if statement, checking whether the value returned from KeyPadQueryHotSpot() > 0.

For example:

```
'assume within polling routine or pen event
If SigPlus1.KeyPadQueryHotSpot(0) > 0 Then
'user has tapped hotspot #0
SigPlus1.ClearTablet
End If
```

or

```
//assume within polling routine or pen event
if(sigObj.keyPadQueryHotSpot(0) != 0)
{
//user has tapped hotspot #0
sigObj.clearTablet();
sigObj.ClearSigWindow(1);
}
```

It is very important that as soon as you fall within a KeyPadQueryHotSpot() routine you clear out the hotspot buffer (as shown in the code above), or your hotspot will trip every time your polling routine runs, or a pen event occurs. You can clear the hotspot buffer by calling ClearTablet() and/or ClearSigWindow(1).

Be sure that in your polling routine or pen event routine, you call ClearTablet() and/or ClearSigWindow(1), in case you do not trip any hotspots. If your code makes it through the routine without getting into one of the KeyPadQueryHotspot() routines, the hotspot buffer will still need to be cleared.

How do I write text to the LCD?

ActiveX Example: Using the LCDWriteString() method. This method takes 8 arguments: Dest, Mode, XPos, YPos, XSize, YSize, Format, HexString

Dest: 0=foreground, 1=background
Mode: 0=Clear, 1=Complement, 2=Write Opaque, 3=Write Transparent (mode 2 is most-often used)
XPosition: The x coord location of the upper-left corner of the image
YPosition: The y coord location of the upper-left corner of the image
XSize: The width of the image to display in px
YSize: The height of the image to display in px
Format: NOT CURRENTLY IMPLEMENTED - PASS A 0
HexString: String value to display

If 0s are passed for the size arguments, then the default size will be used. For example:

SigPlus1.LCDWriteString(0, 2, 0, 0, 0, 0, 0, "This is a test string")

You can also control the font more completely using the LCDSetFont() method. The arguments for this method are all defined in the LOGFONT data structure (see CreateFont function of Windows API) in Windows for logical fonts. They are:

Height, Width, Weight, Italic, Underline, PitchAndFamily, FaceName

The FaceName argument is used to specify a particular font. Again, see the CreateFont function of Windows API for details on the other args. Leaving the args at 0 will result in the default for each value to be used. An example is below:

SigPlus1.LCDSetFont(0, 0, 0, 0, 0, 0, "Courier New")

How can I manage displaying variable text of unknown length without line breaks on a Topaz LCD pad?

There are two methods that will allow for sizing. One will return the current LCD size, and the other (based on the text you pass in, and also on your current LCDSetFont() settings) provides the size in px you will need for that text.

GetLCDSize()
Function: Returns the XSize and YSize (in LCD coords) of the pad currently specified in the SigPlus.ini.
Return Value: Unsigned long - (YSize upper 16 bits, XSize lower 16 bits)

GetLCDTextSize(str StringToBeDisplayedOnLCD)
Function: Returns the XSize and YSize (in LCD coords) necessary to display on the LCD the string argument, based on the font parameters (see LCDSetFont() method for details). Use to determine the LCD size necessary for your display Text.
Argument: String Text to be displayed on the LCD
Return Value: Unsigned long

How can I set a section of the LCD so that the signature will only be accepted in that section?

ActiveX Example: There are two methods that work together for this functionality (although you can certainly use them separately should your application have specific requirements):

SetSigWindow() and LCDSetWindow()

SetSigWindow() defines a specific area on the pad that will accept pen data as "signature" data (collected in the signature object). Outside of this window, pen data will not be added to the signature. This is useful especially when hotspots are used; the area where the hotspots reside can be left out of the signature window, so that hotspot taps do not accumulate in the signature. It is important to indicate to the user, of course, the specific area set aside as the "signature" area (either with text or an image) when using the SetSigWindow() method. This method takes 5 arguments - Coords, XPos, YPos, XSize, YSize:

Coords: Coordinate system used for this hotspot:
0 = Logical tablet coordinates, 1 = LCD Coordinates (Topaz always recommends using 1 = LCD Coordinates).
XPosition: The x coord location of the upper-left corner of the signature window
YPosition: The y coord location of the upper-left corner of the signature window
XSize: The width of the signature window in px
YSize: The height of the signature window in px

For example: SigPlus1.SetSigWindow(1, 3, 30, 122, 30)

LCDSetWindow() defines a specific area on the LCD itself where "electronic ink" is displayed. It is important to note that this method has nothing to do with the captured signature...only the signature as it is displayed on the LCD. Typically, when you call the SetSigWindow() method to create a window which limits the area in which signature data is captured, you want to also limit the LCD itself to display the signature representation in the same manner. This method takes 4 arguments - XStart, YStart, XSize, YSize:

XPosition: The x coord location of the upper-left corner of the signature window
YPosition: The y coord location of the upper-left corner of the signature window
XSize: The width of the signature window in px
YSize: The height of the signature window in px

For example: SigPlus1.LcdSetWindow(0, 0, 60, 32)

In other words, if you call SetSigWindow() without calling LCDSetWindow(), you will limit the signature in the SigPlus object to a certain area, but the LCD will display all of the ink drawn by the pen, even if it exceeds the window set by SetSigWindow(). Likewise, if you call LCDSetWindow() without calling SetSigWindow(), all of what is drawn by the pen will be captured in the SigPlus object, but the LCD will be limited to displaying "electronic ink" in the specified area only.

To inhibit ink on LCD screen completely, use the following code:
SigPlus1.LCDSetWindow(0, 0, 1, 1)

To inhibit ink in the SigPlus object completely, use the following code:
SigPlus1.SetSigWindow(1, 0, 0, 1, 1) (be sure to leave at least one pixel in the Xsize and Ysize arguments)

If you are calling the LCDSetWindow() or SetSigWindow() and clearing and adding new hotspots in the same routine, be sure to first clear the hotspot list using SigPlus1.KeyPadClearHotSpotList(), then calling LCDSetWindow() or SetSigWindow(), then adding your new hotspots using the KeyPadAddHotSpot() method.

The idea is to set the windows while the hotspots are clear.

How many characters can I fit per line and how many lines can I fit when writing text to the LCD?

Assuming you are using a fixed-width font (with a 5 px spread), you can expect the following (you may need to extrapolate further data based on these stats--and remember these are rough, rule-of-thumb values):

SignatureGem LCD 1x5 (TL462) and SigLite LCD 1x5 (TL460):
At font size 8, the total number of possible lines is: 12
Number of characters per line: 39

At font size 10, the total number of possible lines is: 10
Number of characters per line: 34

At font size 12, the total number of possible lines is: 9
Number of characters per line: 26

SignatureGem LCD 4x3 (TL755) and SigLite LCD 4x3 (TLBK750):
At font size 8, the total number of possible lines is: 24
Number of characters per line: 39

At font size 10, the total number of possible lines is: 20
Number of characters per line: 34

At font size 12, the total number of possible lines is: 18
Number of characters per line: 26

SignatureGem LCD 4x5 (TLBK766):
At font size 8, the total number of possible lines is: 45
Number of characters per line: 52

At font size 10, the total number of possible lines is: 37
Number of characters per line: 45

At font size 12, the total number of possible lines is: 33
Number of characters per line: 34

NOTE: All characters after the cutoff will be lost.

How do I use the pen events: Pen_Up and Pen_Down? Can I poll instead?

ActiveX Example: You will need an event handler for Pen_Up, Pen_Down or for both, for example:

SigPlus1_PenUp()

or

SigPlus1_PenDown()

Place the code you wish to run in the event of a pen event in the appropriate event handler. The methods used by the pad which will need to be called in the event handler is:

KeyPadQueryHotSpot()

which is used to check if a hotspot has been tapped.

It is very important to note that there is a method call necessary for the pen events to function: SetEventEnableMask(). The SetEventEnableMask() method takes a single arg - EventMask:

EventMask: an integer value used to set the mask for either 1=Pen Down, 2=Pen Up, 3=Both Pen Down and Pen Up

For each time the pen_up or pen_down event fires, the SetEventEnableMask() method must be called. Therefore, at the end of each event, you'll need to call the SetEventEnableMask() method to trigger the next pen event. This method is used to offer further control to the developer. A pen event will only fire once and will not fire again until the SetEventEnableMask() method is called. Otherwise, it could fire repeatedly should the user keep the pen pressed to the pad. Instead of using the pen events, you may choose to poll. This methodology entails the use of a timer of some sort that checks at a certain rate for pen activity (calling KeyPadQueryHotSpot(), etc.) Typically, the rate is somewhere between 100-300 ms. This methodology does not require the use of the pen event handlers, nor a call to SetEventEnableMask().

What is the difference between tablet logical coordinates and LCD coordinates? Which should I use when passing the 'Coords' argument in a method call requiring this argument?

Tablet Logical coordinates are the original coordinate system designed for Topaz pads. They are based on a 0, 0 position which exists up and left from where the active area of the pad begins. So, the point at which the active area of a particular pad (where the pad will start capture pen data) never begins at 0, 0 in tablet logical coordinates, and can vary to different degrees. For example, the active coordinate origin for a SignatureGem LCD 1x5 is 400, 350. For the SignatureGem LCD 4x3, it is 500, 400. Other pads will vary.

LCD coordinates were added to SigPlus® for ease of use in terms of calls to methods where an x and y position are required, for instance KeyPadAddHotspot(). The LCD coordinate system makes the upper-left corner of the LCD itself the 0, 0 position, and the bottom-right corner of the LCD itself the extent of the LCD, which is:

For SignatureGem LCD 1x5 and SigLite LCD 1x5: 240, 64
For SignatureGem LCD 4x3 and SigLite LCD 4x3: 240, 128
For SignatureGem LCD 4x5: 320, 240

In terms of tablet logical coordinates, the top-left corner of the LCD is a counterintuitive value, far from 0, 0 since 0, 0 in tablet coordinates is a position not only far beyond the upper-left corner of the LCD, it is beyond the active area. This is the reason Topaz highly recommends using LCD coordinates over tablet logical coordinates.

How do I clear the background memory of the LCD pad?

The easiest way is to call the LCDWriteFile() method, passing an initial argument of 1 (to indicate write to the background memory), then followed by a 0 (indicating complete refresh). This will erase the background memory of the LCD. For example:

Clear background memory of LCD 1x5:
SigPlus1.LCDWriteFile(1, 0, 0, 0, 240, 64, 0, "")

Clear background memory of LCD 4x3:
SigPlus1.LCDWriteFile(1, 0, 0, 0, 240, 128, 0, "")

Clear background memory of LCD 4x5:
SigPlus1.LCDWriteFile(1, 0, 0, 0, 320, 240, 0, "")

Clear background memory of SigLite Color 4.3:
SigPlus1.LCDWriteFile(1, 0, 0, 0, 480, 272, 0, "")

Clear background memory of SigGem Color 5.7:
SigPlus1.LCDWriteFile(1, 0, 0, 0, 640, 480, 0, "")

To clear only a portion of the LCD's background memory, use arguments 3, 4, 5, and 6 (XPos, YPos, XSize and YSize, respectively). For example:

SigPlus1.LCDWriteFile(1, 0, 160, 120, 160, 120, 0, "")

clears only the bottom-right 1/4 of the LCD's background memory.