

Topaz Systems, Inc.
SigPlus Roadmap

Topaz Systems, Inc.
650 Cochran Street, Unit 6
Simi Valley, CA, 93065
©1995-2005, all rights reserved, US patent 6,307,955, and pending

www.topazsystems.com
tech support: 805 520-8286
support@topazsystems.com

Making Your Way Around SigPlus ActiveX: A Developer's Road Map

This document is a guide to provide developers with a basic understanding of the necessary functions and commands for successful use and integration of the SigPlus ActiveX software tool. In addition, this guide will strive to help developers gain familiarity with the various software documentation, demo's and how-to guides already available online at www.topazsystems.com

Enabling Signature Capture

Before starting the Topaz tablet must be properly connected to the developer's computer. For more information on how to ensure proper physical connection and functionality, click on the troubleshooting guides below: (for information regarding what tablet and connection you have go [here](#) to access online resources)

[LCD Serial Troubleshooting](#)
[Serial Troubleshooting](#)
[USB-HSB Troubleshooting](#)

SigPlus can be easily be instantiated at Design Time by choosing the SigPlus icon from the tool box, then drawing the SigPlus instance on the form in your IDE.

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
```

With the tablet properly physically connected, SigPlus needs to open the computer's serial or USB port in order to receive the signature information recorded from the pen. Setting the tablet state to 'on' (1) allows SigPlus to open the port and "listen" to the tablet for pen data. This is done through the TabletState property. For example:

[SigPlus1.TabletState = 1](#)

Once the computer's port is opened, as soon as the pen hits the tablet pen data will be transferred over the port (click here, for an explanation of the connection settings) and the signature will be

Topaz Systems, Inc.
650 Cochran Street, Unit 6
Simi Valley, CA, 93065
©1995-2005, all rights reserved, US patent 6,307,955, and pending

www.topazsystems.com
tech support: 805 520-8286
support@topazsystems.com

visible in real time as the signing process takes place. As the signature is rendered it is also captured in SigPlus.

How Do I Tell When The Signature is Captured?



There is no sure-fire way programmatically for the developer to deduce whether the author has finished signing programmatically. One possible way might consist of polling and checking for the following:

1. First (within your polling event), check if the author has started signing by querying the NumberOfTabletPoints:

```
If SigPlus1.NumberOfTabletPoints = 0 Then  
    'signature is not yet being captured  
    Exit Sub  
End If
```

A signature is made up of points, captured and accumulated as the pen is drawn across the pad. Zero points means no signature currently exists. Polling to check if the number of points is > 0 will enable the developer to see when the user has started signing by showing when the tablet points are > 0.

2. Tablet points will start increasing with each polling event, indicating the user is still signing:

```
Static intCounter As Integer  
If SigPlus1.NumberOfTabletPoints > 0 Then  
    If SigPlus1.NumberOfTabletPoints > intCounter Then  
        'signature is still being captured, keep polling  
        intCounter = SigPlus1.NumberOfTabletPoints  
    Else  
        'points have stopped accumulating...check again or go on??  
    End If  
End If
```

When the tablet points stop increasing, it is reasonable to assume that the signature has been captured. The problem with relying solely on tablet points is that if the user drops the pen, or for some reason pauses during signing, etc., the tablet points will remain static until the pen is picked up and the signature is completed, which could take seconds or more. After the 'Else' above, you must decide what you want to do as the developer, to comfortably feel that the user is

Topaz Systems, Inc.
SigPlus Roadmap

Topaz Systems, Inc.
650 Cochran Street, Unit 6
Simi Valley, CA, 93065
©1995-2005, all rights reserved, US patent 6,307,955, and pending

www.topazsystems.com
tech support: 805 520-8286
support@topazsystems.com

done signing. That might be poll a few more times to verify that the points do not increase again, for example.

The most reliable means to know when signature capture is completed is to code within a sure-fire event. Some popular events used are as simple as an 'OK' or 'Done' button that the user or operator can click when signature is completed. Depending on the application and tablet model being developed for, these buttons can be programmed to be displayed through the computer the tablet is connected to, or if the tablet has an LCD screen the buttons can be controlled by the signer on the LCD screen itself, by tapping with the pen. *(For more information regarding programming for Topaz LCD equipped signature tablets click [here](#). For information on the easy to use Topaz Active-X Client Demo for programming LCD tablets click [here](#).)*

Once the signature is captured the tablet state should be returned to 0, which will close the port.

Compression and Encryption Issues

The average size of a signature can range from between 6-12 kb (6,000-12,000 characters). After the signature has been captured the developer can set the compression level to greater than 0, (with 0 being default - no compression). As a rule, the compression mode should be set at 0 during signature capture (*click [here](#) for an explanation of the compression mode settings*). The code below illustrates how the compression mode can be set. Each successive compression mode compresses the signature at a 2.5 to 1 ratio.

```
SigPlus1.SigCompressionMode = myCompVal
```

Encryption allows the developer to tie a file, or a string/set of strings, to a signature so that the signature can only be successfully returned if the contents of the file or the strings are returned unchanged. The process is rather like creating a key from the data (the file(s) or string data) which is the only way to unlock the signature. The end result is that the signature is tied to that file(s)/data, providing the signature a context.

IT IS IMPORTANT TO NOTE THAT ENCRYPTION MODE MUST BE 0 DURING SIGNATURE CAPTURE. Signature encryption allows the developer to tie the signature to a set of data, or a file or files. This provides the signature with a context: that the signature belongs with this data or file(s). Therefore, the signature cannot be associated with any other data/file(s) except to those with which it has been encrypted. This is a key component of creating a legally-binding, fully eSign-compliant digital handwritten signature.

Signature encryption is not necessary to capturing and storing the signature, however. If the developer is not interested in eSign law-compliant, legally binding signatures, they certainly do not need to utilize the following encryption methods.

Topaz Systems, Inc.
SigPlus Roadmap

Topaz Systems, Inc.
650 Cochran Street, Unit 6
Simi Valley, CA, 93065
©1995-2005, all rights reserved, US patent 6,307,955, and pending

www.topazsystems.com
tech support: 805 520-8286
support@topazsystems.com

Whether encryption is implemented or not, the signature (when stored in the Topaz SIG file format, or as the ASCII hex string SigString) inherently contains the biometric information useful to a forensic analyst later to determine the signer's identity. Such biometric data as pen stroke order and velocity information, for example, add up to useful data for determining the identity of the signer.

It is the encryption and the biometric data that, together, comprise a complete eSign law-compliant, legally binding digital handwritten signature.

How to Encrypt a Signature

To encrypt/decrypt the signature, use something like the code below. Encryption/Decryption makes use of the AutoKey function comprised of AutoKeyStart, AutoKeyData, and AutoKeyFinish, as well as the EncryptionMode property.

Note that the AutoKeyStart() method is ONLY called if the developer plans on using the AutoKeyData property to pass in strings. If the user passes in a path to a file that SigPlus must go and get, then the developer should NOT call the AutoKeyStart() method.

```
SigPlus1.KeyString = "000000000000000000" //zero out key  
SigPlus1.EncryptionMode = 0 //zero out encryption  
SigPlus1.TabletState = 0  
SigPlus1.AutoKeyStart  
SigPlus1.AutoKeyData = "some sample data" //this could be a path to a file, as well  
SigPlus1.AutoKeyData = "some more sample data"  
SigPlus1.AutoKeyData = "set as much as you like"  
SigPlus1.AutoKeyFinish  
SigPlus1.SigCompressionMode = myCompVal  
SigPlus1.EncryptionMode = myEncVal  
'now you can either extract the signature, or return it...see below
```

The encryption and decryption of signatures are very similar, with the major differences being:

1. If it's a SIG file, whether you use ImportSigFile() or ExportSigFile() at the end of your encryption routine. Import is for bringing back a signature, Export is for getting the signature out, and saved to the location passed in.
2. If it's SigString, then it's which side of the "=" sign the SigString is, for example:
`SigPlus1.SigString = someSigStringValue` : *this puts a signature into SigPlus*
`someSigStringValue = SigPlus1.SigString` : *this copies the signature to the variable...store as you wish*

Topaz Systems, Inc.
SigPlus Roadmap

Topaz Systems, Inc.
650 Cochran Street, Unit 6
Simi Valley, CA, 93065
©1995-2005, all rights reserved, US patent 6,307,955, and pending

www.topazsystems.com
tech support: 805 520-8286
support@topazsystems.com

Essentially, the process for both encryption and decryption can be seen below.

Encrypt

1. Sign
2. AutoKey/Encrypt/Compress
3. Get signature out

Decrypt

1. AutoKey/Encrypt/Compress (same as when encrypted—data or file must match, of course)
2. Get signature out of database / Import from file
3. If the data is the same, and all the modes match, the signature is returned and displayed

To Extract signature from SigPlus after encryption/compression is complete, in either the Topaz .SIG file or as a convenient to use ASCII hex string, use the below calls:

```
'as File  
SigPlus1.ExportSigFile "C:\mypath.sig"
```

```
'as string  
myStrVariable = SigPlus1.SigString
```

To return a saved signature back to SigPlus, use the below code:

```
'as File  
SigPlus1.ImportSigFile "C:\mypath.sig"
```

```
'as string  
SigPlus1.SigString = mySavedSignature
```

LCD Tablet Interactive Functionality

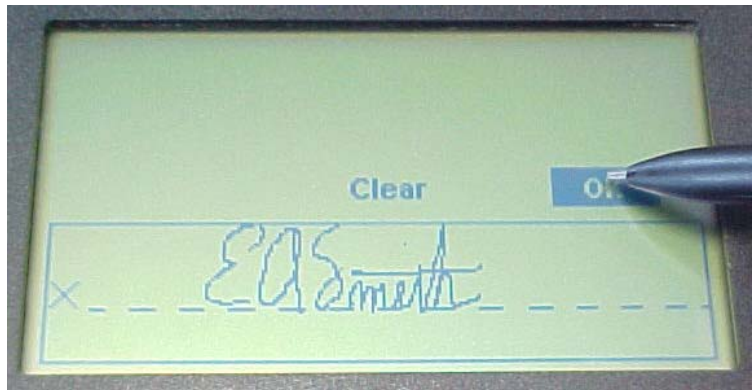
This section of the roadmap deals exclusively with the development of interactive applications for use with the Topaz family of LCD equipped signature capture tablets. For more information regarding these types of tablets click [here](#). For sales and pricing information please contact sales@topazsystems.com. For developer's demos click [here](#), and for further information regarding LCD development click [here](#).

Topaz Systems, Inc.
SigPlus Roadmap

Topaz Systems, Inc.
650 Cochran Street, Unit 6
Simi Valley, CA, 93065
©1995-2005, all rights reserved, US patent 6,307,955, and pending

www.topazsystems.com
tech support: 805 520-8286
support@topazsystems.com

Interactive applications are not a requirement for signature capture, but these customizable programs allow for a more seamless and user-friendly interface for many applications. The primary functionality for programming with the Topaz LCD signature capture tablets has to deal with the use of HotSpots, or areas of the LCD screen which accept pen taps from the user and then run the application as appropriate. Here is a picture of a simple interactive menu displayed on a Topaz SignatureGem LCD4X3 signature capture tablet. The 'Clear' and 'OK' button are both HotSpots. In this case the 'Clear' button is programmed to detect pen taps and then clear the signature accordingly, whereas the 'OK' button is programmed to detect pen taps and load the next menu or screen.



There are numerous methods and properties that are used in the programming of an interactive application, to see all of them please go to the [SigPlus ActiveX Developers Documentation](#) and look under *Methods and Properties for use with LCD Tablets*. Below are a few common calls that can be used with brief descriptions, for a more in depth look at these properties and methods click on the links:

[LCDRefresh](#)- Refreshes the LCD Screen (clears the lcd of ink, or brings up images stored in background memory, or inverts pixels at any specified location/size specified)

[LCDSetWindow](#)- Sets the size/position of the portion of the lcd that you wish to display ink...ink will only appear in the LCD in that section (ie, signature window)

[SetSigWindow](#) – Sets a window so that ink will be collected in SigPlus ONLY when the pen falls within a certain position/size square on the LCD (ie, a signature line)

[LCDWriteBitmap](#)- Used to write a bitmap image to the LCD unit from a Windows handle to a bitmap

[LCDWriteFile](#)- Used to write a bitmap image to the LCD unit from a file path

[LCDWriteString](#)- Used to write a string to the LCD

[LCDSetFont](#)- Sets the font to be used in the LCDWriteString() call

[KeyPadAddHotSpot](#)- Used to add a spot that, when tapped with the pen, is able to be trapped in an event by the developer

[KeyPadClearHotSpot](#)- Clears the control's internal list of HotSpots

[KeyPadQueryHotSpot](#)- Used with AddHotSpot, this is the means by which developers query whether a specific hot spot is tapped

Topaz Systems, Inc.
SigPlus Roadmap

Topaz Systems, Inc.
650 Cochran Street, Unit 6
Simi Valley, CA, 93065
©1995-2005, all rights reserved, US patent 6,307,955, and pending

www.topazsystems.com
tech support: 805 520-8286
support@topazsystems.com

Topaz Systems offers other tools for developers wishing to develop interactive applications called LCD Code Assistants.. These Code Assistants assist in the creation of 'Hotspots' and in placing bitmap images on the LCD screen, by providing a graphical user interface for the placement of these elements...providing the LCD coordinates necessary to size and position items. The assistant provides code for the following SigPlus LCD methods:

[LCDWriteBitmap](#)
[KeyPadAddHotSpot](#)

After visually placing these elements within a rectangular display representing the tablet's LCD, the code is generated to place these elements in the same locations programmatically. Click below to download the code assistants.

[1X5 Code Assistant](#) and [How-To](#)
[4X3 Code Assistant](#) and [How-To](#)

Additionally, there is a SigPlusLCD ActiveX control, that can be used for simple LCD development. Though it has only a limited application and usefulness, it makes rudimentary LCD applications very easy to create. Click on the link below to download SigPlusLCD.ocx (requires SigPlus.ocx to already be installed) and see the developer's documents.

[SigPlusLCD.ocx](#) and [documentation](#)

For optimum speed of operation, and to assure the best and most reliable operation with all versions of windows, it is very important that you set TabletState=1 in your application before calling any LCD functions, writing any text graphics, or signature capture (set TabletState = 1 before calling any methods or set properties that need open communications to exist between the PC and the tablet). To be absolutely sure that there are no unwanted points captured as a part of the signature, you should always call the ClearTablet method just before capturing the signature. For example, see where TabletState=1 is set in the following SigPlus initialization code excerpt:

```
SigPlus1.LCDSetTabletMap 0, 240, 128, 200, 250, 2000, 1300
```

```
    'Sets up SigPlus for LCD 4X3 tablet
```

```
    SigPlus1.TabletXStart = 500
```

```
    SigPlus1.TabletXStop = 2650
```

```
    SigPlus1.TabletYStart = 400
```

```
    SigPlus1.TabletYStop = 2100
```

```
    SigPlus1.TabletLogicalXSize = 2150
```

```
    SigPlus1.TabletLogicalYSize = 1700
```

```
SigPlus1.TabletState = 1 'open com, turn tablet on - OPEN THE PORT AT LEAST HERE
```

Topaz Systems, Inc.
SigPlus Roadmap

Topaz Systems, Inc.
650 Cochran Street, Unit 6
Simi Valley, CA, 93065
©1995-2005, all rights reserved, US patent 6,307,955, and pending

www.topazsystems.com
tech support: 805 520-8286
support@topazsystems.com

SigPlus1.LCDSetWindow 0, 0, 0, 0 *'Prohibit inking on entire LCD*
SigPlus1.LCDRefresh 0, 0, 0, 240, 128 *'Refresh entire tablet*

[SigPlus1.LCDCaptureMode](#) = 2 *'Sets up LCD to retain data rather than 'auto clearing it every few seconds*

SigPlus1.ClearTablet *'Clears the SigPlus object of ink*
SigPlus1.KeyPadAddHotSpot 0, 1, 11, 55, 15, 25 *'set up hot spots SigPlus1.KeyPadAddHotSpot 1, 1, 11, 75, 15, 25*
SigPlus1.KeyPadAddHotSpot 2, 1, 70, 95, 30, 30 SigPlus1.KeyPadAddHotSpot 3, 1, 132, 95, 40, 30
SigPlus1.KeyPadAddHotSpot 4, 1, 17, 53, 60, 15
SigPlus1.KeyPadAddHotSpot 5, 1, 106, 51, 36, 16
SigPlus1.KeyPadAddHotSpot 6, 1, 187, 52, 30, 15
SigPlus1.LCDWriteBitmap 1, 2, 0, 0, 240, 128, Image3.Picture.Handle *'load BMPs 'this BMP is loaded into background*

'The rest of the BMPs are loaded into foreground

SigPlus1.LCDWriteBitmap 0, 2, 10, 0, 100, 10, Prescriptions.Picture.Handle
SigPlus1.LCDWriteBitmap 0, 2, 10, 64, 130, 40, CheckBoxes.Picture.Handle
SigPlus1.LCDWriteBitmap 0, 3, 40, 10, 210, 52, PNumbers.Picture.Handle
SigPlus1.LCDWriteBitmap 0, 2, 70, 110, 100, 10, Next_Clear.Picture.Handle
[SigPlus1.SetEventEnableMask](#) (1)

Essentially, if the port is closed, any methods or properties that must have the port open in order to work will turn the state on (open the port) so the call will work, then (in order to return SigPlus to the state it was in when the method was called) will close the port again. Put together, several calls that do this will result in the port getting opened and closed multiple times in rapid succession, resulting in a non-optimized, slowing (and the possible application failure) of your environment. Such methods include those that write text and or graphics to the LCD, for example, or LCDSetWindow().

Writing Signature Images

SigPlus allows developers to convert the native signature file into an image format if the developer desires. Topaz supports .JPG, .TIF, .BMP, .WMF, and .EMF image file formats. One important factor to be aware of is that once the signature is converted into an image it will no longer contain the biometric information, can no longer be encrypted and tied to a document, and thus, is no longer legally binding.

Topaz Systems, Inc.
SigPlus Roadmap

Topaz Systems, Inc.
650 Cochran Street, Unit 6
Simi Valley, CA, 93065
©1995-2005, all rights reserved, US patent 6,307,955, and pending

www.topazsystems.com
tech support: 805 520-8286
support@topazsystems.com

SigPlus offers many options to handle the converted signature image including ways to alter the size, resolution, image zoom, ink thickness, color, including various types of JPGs, TIFs and BMPs. To see an in-depth explanation of these options click on the property links below.

Below is some sample code to help the developer successfully write signature images through SigPlus. For more information regarding the methods and properties shown, click the links to go to a more in depth explanation.

Code for writing signature images:

[SigPlus1.ImageFileFormat](#) = 0

'0 sets the resulting image's file type to BMP; 0=BMP and 4=JPG 6=TIF

[SigPlus1.ImageXSize](#) = 1500

'Sets resulting image's file width in pixels

[SigPlus1.ImageYSize](#) = 500

'Sets resulting image's file height in pixels

[SigPlus1.ImagePenWidth](#) = 12

'Sets the thickness of the ink (in pixels) for the resulting image

[SigPlus1.JustifyMode](#) = 5

'Blows signature up as large as possible within the bounds of the SigPlus object

[SigPlus1.WriteImageFile](#) "C:\Signature.bmp"

'Method that will write out the image file, using all the parameters set above--pass the complete path, including filename and file extension

Return a Signature Bitmap Byte Array

To return a bitmap byte array, which is an array containing the signature bmp and must be used with ImageFileFormat=0. The benefit of a bitmap array is that you do not have to write the bitmap image to file first:

Dim Size As Long
Dim ByteValue() As Byte

SigPlus1.ImageFileFormat = 0
SigPlus1.ImageXSize = 1500
SigPlus1.ImageYSize = 500

Topaz Systems, Inc.
SigPlus Roadmap

Topaz Systems, Inc.
650 Cochran Street, Unit 6
Simi Valley, CA, 93065

www.topazsystems.com
tech support: 805 520-8286
support@topazsystems.com

©1995-2005, all rights reserved, US patent 6,307,955, and pending

```
SigPlus1.ImagePenWidth = 12  
SigPlus1.JustifyMode = 5  
SigPlus1.BitMapBufferWrite  
Size = SigPlus1.BitMapBufferSize  
ReDim ByteValue(Size)  
ByteValue = SigPlus1.GetBitmapBufferBytes  
SigPlus1.BitMapBufferClose
```

Color Images

To write images in color (this ONLY will function when writing out JPGs):

First, open the WIN\SigPlus.ini file. Find the parameter called EnableColor=0 in the [Tablet] section (below, underlined in red). Change this value to EnableColor=1. Save the file. This sets SigPlus to create color signatures.

```
[Tablet]  
TabletComPort=1  
TabletType=6  
TabletLCDMode=0  
TabletModel=SignatureGemLCD4X3  
EnableColor=0  
Win95USB=0  
ImageScreenResolution=1  
UseMultiUSB=0  
DisableMessages=0
```

Then use the means available through your development environment to set the BackColor and ForeColor of your signature object.

For example, in VB, there is a ForeColor and BackColor property available.

To change the colors, please choose a color from the following list, and replace the decimal value of the ForeColor and BackColor with the new color's decimal value.

Black=0
White=16777215
Light Blue=16776960
Dark Blue=16711680
Red=255
Green=65280
Yellow=65535

Topaz Systems, Inc.
SigPlus Roadmap

Topaz Systems, Inc.
650 Cochran Street, Unit 6
Simi Valley, CA, 93065

www.topazsystems.com
tech support: 805 520-8286
support@topazsystems.com

©1995-2005, all rights reserved, US patent 6,307,955, and pending

Light Gray=14671839

Dark Gray=8355711

Pink=12615935

Purple=16711808

For example:

For dark blue ink on a yellow background, change the parameters in the SigToolImager.ini as follows:

ForeColor=16711680

BackColor=65535

To determine a specific color's decimal value, please first determine the hex value of your color (the RGB value). For example, perfect blue would be 0000FF (where there is 0 red, 0 green, and 255 blue). Next, flip the value backwards (so instead of 0000FF, it becomes FF0000.)

Finally, using the Calculator program provided in Start...Programs...Accessories, click on the View menu. Choose the Scientific calculator. Choose Hex in the radio button selection. Type in your 6-digit hex value, as determined in the above instructions (for our example, type in 0000FF for perfect blue). Finally, choose Dec in the radio button selection, and the Hex value will be converted into the Decimal value necessary for the ForeColor and BackColor parameters in the SigToolImager.ini (as you will see, the Decimal value for perfect blue is 16711680).