Software Developer Guide

SigPlus ActiveX Control

Version 4.4.0.24

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Recent Version Changes

- ### Current Version: 4.4.0.24

- ### Version 4.4.0.22
  - Resolved uncommon rendering dependency.

- ### Version 4.4.0.19
  - Minor software update.

- ### Version 4.4.0.16
  - Fixed TabletModelNumber() issue pertaining to calling the function multiple times with a tablet disconnect between calls. Added support for newer signature pads to support change from 'full-speed' USB to 'high-speed' USB.

- ### Version 4.4.06
  - Changed default SocketTimeout to 15 seconds (using SigSock). Changed GetTabletModelNumber to default to 0. Added Enable/Disable of ClearType. Updated SigPlus logging for increased information. Added SigPlusRoot.ini support.

- ### Version 4.3.0
  - Optimized LCD pad interactivity. Optimized SetSigWindow() performance. Optimized TabletSerialNumber() and TabletModelNumber(). Updated SetBackgroundHandle() to do FitToWindow. Added new MultiUsb functionality: GetNumberOfAttachedTablets(), SetMultiUseEnable(), GetTabletSerialNumber(), SetSerialNumberToUse(). Created a new SigPlus 64-bit version. Added SetTabletPortPath() function. Added CCITT Group 4 fax tiff image support to ImageFileFormat property.

- ### Version 3.95
  - Optimized LCD pad interactivity. Improved logging. Improved performance in thin client environments. Revamped TabletConnectQuery() function for optimal signature hardware detection. State logic optimization in communication between hardware and SigPlus, resulting in improved LCD performance.

- ### Version 3.89
  - Added timeout for SigSock support. If SigSock client is not located within 5 seconds, timeout is implemented. Added support for ZLib compression on specific LCD pads. Optimized scaling for TabletPC usage. Optimized signature buffer for heavy-load CPUs. Optimized SigPlus. Added SetLogEnable and SetLogFilePath for customizing SigPlus.log file save location. Added support for SigSock socket interface. Optimized ImageFileFormat=7 for writing WMF files. Added support for SigGemColor 5.7 signature pad. Added SetLCDPixelDepth() and LCDCompressMode() for SigGem Color 5.7.

General Release Notes

I. Setting the COM Port

- Be sure that tablet state is off when selecting the COM port. The COM port must be selected first, and then tablet state turned on. Also, be sure to set tablet state off before exiting the application. Alternately, set TabletType=6 instead.

II. Very Important Note for Non-C++ Users

- For all Properties, the function names are in the form of a Get/Set pair. The full name is used when the object is referenced from a Visual C++ application. For Visual Basic, Powerbuilder, and other users, just use the property name itself, **without the Get/Set prefix (NOTE: Method calls in VB still require the Get prefix. See the specific "Remark" section for each method for details).**

- Note that in version 2.30, the "add 32 TabletMode" was added to provide lower-resolution signatures sometimes needed by VB. There are other 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.

III. Using the SigPlus.ini Feature

- The "SigPlus.exe" program installs the "OCX" control in the WIN\SigPlus directory, together with the "SigPlus.ini" file in the WIN directory. After you have installed the .ini file, you may edit it or delete it easily. This .ini is not functional unless it has been placed in the WIN directory. If the .ini file is not present, then the properties of the control in your application are unaffected and the application runs on its own. When the .ini file is present in the WIN directory, the values present are used to override the properties set in the control proper in the application. The .ini file allows a common application to use different tablet types and interfaces.

- For example, if one machine uses the signature pad on COM1, another on COM2, a third on the USB port with a Topaz HSB tablet, the SigPlus.ini files for these three cases would be:

Computer 1 (COM1)	Computer 2 (COM 2)	Computer 3 (HSB)
[Tablet]	[Tablet]	[Tablet]
TabletComPort=1	TabletComPort=2	TabletComPort=1
TabletType=0	TabletType=0	TabletType=6
TabletLCDMode=0	TabletLCDMode=0	TabletLCDMode=0
etc.	etc.	etc.

- **It is very important to note that the .ini settings can only be used to override the OCX default property settings. If coding or script is used to override a property setting in your application, the .ini settings will not override the scripted setting.**

- TabletType settings: Default is 0
  - 0 Serial mode. When tablet is activated it will accept input from the selected com port
  - 6 HSB mode (USB tablet using Windows HID driver, models ending in –HSB-R). Please note that the TabletComPort value is not relevant when TabletType is set to 6.

- The TabletComPort and TabletLCDMode configuration lines are identical to the properties with the same names as described in this document. The TabletType configuration line uses that same settings as the lower order settings of the TabletMode property as described later in this document.

- Version 3.01 and above of the SigPlus control allows the background and ink color of the control to be set under program or property control. To enable these properties, set the line EnableColor = 1 in the [Tablet] portion of the SigPlus.ini file at the top.

- Rendering color images to either the TLBK-57GC or the T-LBK43LC requires passing an argument of int value = 8 to the SetLCDPixelDepth() function. For black and white images, an argument of int value = 0 can be passed in. The default value for SetLCDPixelDepth() is 0.

- To create BMP, JPG or TIF image files at tablet resolution, please set ImageScreenResolution=0.

- To connect multiple HSB tablets to a USB hub, and function independently, please set the UseMultiUSB=1. For a typical, single HSB tablet setup, keep UseMultiUSB=0

- To suppress error messages generated by SigPlus, please set DisableMessages=1.

- Topaz HSB tablets use the built-in HID USB driver rather than a separate Topaz USB driver. No driver loading is necessary since it is already automatically installed. Requires the SigPlus.ini parameter TabletType=6.

IV. Note Regarding Tablet Settings

- Signatures are stored relative to the LogicalX and LogicalY sizes, so it is important to be consistent in all instances of the control. If you change the LogicalX and LogicalY settings, you will need to convert any .sig files saved with the previous LogicalX and LogicalY settings. (LogicalX and LogicalY determined by Start values minus Stop values for that tablet).

V. **SigPlus.ini File Parameters for LCD Tablets**

➢ LCD Type	Type of LCD display
➢ LCD XSize	X Size of LCD display, in pixels
➢ LCDYSize	Y Size of LCD display, in pixels
➢ LCDXStart	X Pos in logical tablet coordinates of LCD
➢ LCDYStart	Y Pos in logical tablet coordinates of LCD
➢ LCDXStop	X Pos in logical tablet coordinates of LCD
➢ LCDYStop	Y Pos in logical tablet coordinates of LCD

- **Wrappers:**
  - ➢ For Visual C++ programmers, the SigPlus wrappers are provided in the installation of the software. By default, the SigPlus.cpp and SigPlus.h files are provided in WIN\sigplus\wrappers.

VI. **Important Notice**

- These guidelines or any or all additional documentation or examples do not constitute a warranty about the performance, security, or legal acceptability of SigPlus software in any specific use or implementation. To the extent that SigPlus is used to achieve regulatory or other specific objectives within an industry, you must consult competent experts or regulatory officials together with your own plan to achieve your desired business objectives using the Topaz tools.

Events for SigPlus OCX Control

SigPlus_PenDown

This event will fire when the pen has made contact with the tablet.

SigPlus_PenUp

This event will fire when the pen has been lifted from the tablet.

- *Note: Events in SigPlus must be used in conjunction with the* ***SetEventEnableMask(int EventMask)*** *method of SigPlus. Please refer to this method in the "Methods Section" further in this documentation.*

Methods and Properties for Use with LCD Tablets

I. General Methods:

ClearSigWindow(int Location)

Function: Clears pen data either inside or outside the SigWindow (see SetSigWindow).

Arguments: Integer:

Location	Specifies clearing inside or outside the SigWindow
If Location = 0	Then signature data is cleared (inside SigWindow)
If Location = 1	Then points are cleared from the hot spot buffer

Return Value: Void

LCDSetTabletMap(LCDType, LCDXSize, LCDYSize, LCDXStart, LCDXStop, LCDXStop, LCDYStop)

Function: Used to override the default values for the LCD parameters at run time.

Arguments: Integers:

LCDType	Specifies LCD type and format, 0 for 240x128, 1 for 128x64
LCDXSize	X Size of LCD display, in pixels
LCDYSize	Y Size of LCD display, in pixels
LCDXStart	X Pos in logical tablet coordinates of LCD
LCDYStart	Y Pos in logical tablet coordinates of LCD
LCDXStop	X Pos in logical tablet coordinates of LCD
LCDYStop	Y Pos in logical tablet coordinates of LCD

Return Value: Void

Remarks: This method is set at load time by parameters in the SigPlus.ini file, or can be run in the program to override the .ini file settings. Note that LCDType = 1 is not supported in this release.

SetSigWindow(Coords, XPos, YPos, XSize, YSize)

Function: This function sets a window in the logical tablet space that restricts the operation of some functions to the specified window. The functions behave as follows:

JustifyMode will only operate on points inside of this window.
ExportSigFile and WriteImageFile will only operate on points inside the window.
SigString only operates on points inside of the window.
ClearTablet will only clear in the window.

Arguments: Integers:

Coords	Coordinate system used for this hot spot 0 = Logical tablet coordinates, 1 = LCD Coordinates
XPos	Location in logical tablet coordinates (upper left - 0,0)
YPos	Same
XSize	XSize in logical tablet pixels
YSize	YSize in logical tablet pixels

Return Value: Void

Remarks: This behavior is enabled by setting the start and stop values to non-zero. The window defaults to (0,0,0,0). The window can be enabled at one spot, re-enabled at another, etc., without disabling in between, and then disabled when the various parts of the tablet data have been separated and stored. To determine the logical values in the control for the installed tablet, see the TabletLogicalXSize and TabletLogicalYSize properties.

II. Graphics Support: Graphics Methods:

LCDSetFont(Height As Long, Width As Long, Weight As Long, Italic As Integer, Underline As Integer, PitchAndFamily As Integer, FaceName As String)

Function: Sets the size, type, and properties of font used when calling the LCDWriteString method. The arguments are all defined in the LOGFONT data structure (see CreateFont function of Windows API) in Windows for logical fonts.

Arguments:

Height	Height of font in pixels
Width	Width of font in pixels (If 0, the font mapper uses a default width that matches the height.)

Weight: Font weight as a number between 0 and 900. 0=default, 400=normal, and 700=bold.

Italic: If this value is non-zero, the text is italicized.

Underline: If this value is non-zero, the text is underlined.

PitchAndFamily: Specifies the pitch (fixed or variable width) and font family used if the font you request is unavailable. If you specify a font that's likely to be, then this argument can be left as 0.

FaceName: Font’s name—for example, "Times New Roman", "Courier New", “Arial”

LCDSetWindow(XStart, YStart, XSize, YSize)

Function: This function sets the tablet so that the LCD display will be showing ink only in a restricted area when data is input with a pen in LCDCaptureMode = 2 and = 3 inking modes. Returning the tablet to default state (such as using LCDCaptureMode = 1) will reset these values.

Arguments: Integers:

XPos	Location in LCD coordinates (upper left - 0,0)
YPos	Same
XSize	XSize in LCD pixels
YSize	YSize in LCD pixels

Return Value: True if checksum received and verified. False if no or incorrect checksum received from tablet.

Remarks: Do not send a command with XSize or YSize = 0.

GetLCDSize()

Function: Returns the XSize and YSize (in LCD coords) of the tablet currently specified in the SigPlus.ini.

Return Value:

Unsigned long	(YSize – upper 16 bits, XSize – lower 16 bits)

GetLCDTextSize(String 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	(YSize – upper 16 bits, XSize – lower 16 bits)

Remarks: Be sure to call the LCDSetFont() method prior to calling GetLCDTextSize(). LCDTextSize() bases its return value on the sizes set using LCDSetFont().

LCDRefresh(Mode, XPos, YPos, XSize, YSize)

Function: The tablet is sent a refresh command with 4 possible modes.

Mode 0 - Clear: The Display is cleared at the specified location.
Mode 1 - Complement: The Display is complemented at the specified location.
Mode 2 - WriteOpaque: The contents of the background memory in the tablet are transferred to the LCD display, overwriting the contents of the LCD display.
Mode 3 - WriteTransparent: The contents of the background memory in the tablet are transferred to the LCD display, and combined with the contents of the LCD display.

Arguments: Integers:

Mode	0, 1, 2, 3 as defined above
XPos	Location in LCD coordinates (upper left - 0,0)
YPos	Same
XSize	XSize in LCD pixels
YSize	YSize in LCD pixels

Return Value: True if checksum received and verified. False if no or incorrect checksum received from tablet.

Remarks: NOTE: that this function only can occur on horizontal 8 LCD-pixel boundaries on the LCD tablet unit.

LCDWriteBitmap(Dest, Mode, XPos, YPos, XSize, YSize, Bitmap)

Function: Used to write windows bitmap data to the LCD Display. The data is written at the location specified by the combination of Dest, XPos, and YPos. The Mode determines how the data is written.

Mode 0 - Clear: The Display is cleared at the specified location.
Mode 1 - Complement: The Display is complemented at the specified location.
Mode 2 - WriteOpaque: The contents of the background memory in the tablet are transferred to the LCD display, overwriting the contents of the LCD display.
Mode 3 - WriteTransparent: The contents of the background memory in the tablet are combined with and transferred to the visible LCD memory.

Arguments: Integers:

Dest:	0 = Foreground, 1 = Background memory in tablet
Mode	0, 1, 2, 3 as defined above
XPos	Location in LCD coords to draw at
YPos	Same
XSize	Width in LCD pixels
YSize	Height in LCD pixels
Bitmap	Windows handle (HBITMAP) to a bitmap to be displayed. See Object. Handle property in VB Docs

Return Value: True if checksum received and verified. False if no or incorrect checksum received from tablet.

LCDWriteFile(Dest, Mode, XPos, YPos, XSize, YSize, Format, FilePath&Name)

Function: Used to write the image data to the LCD Display. The data is written at the location specified by the combination of Dest, XPos, and YPos. The Mode determines how the data is written.

Mode 0 - Clear: The Display is cleared at the specified location.
Mode 1 - Complement: The Display is complemented at the specified location.
Mode 2 - WriteOpaque: The contents of the background memory in the tablet are transferred to the LCD display, overwriting the contents of the LCD display.
Mode 3 - WriteTransparent: The contents of the background memory in the tablet are combined with and transferred to the visible LCD memory

Arguments: Integers:

Dest:	0 = Foreground, 1 = Background memory in tablet
Mode	0, 1, 2, 3 as defined above
XPos	Location in LCD coords to draw at
YPos	Same
XSize	Width in LCD pixels
YSize	Height in LCD pixels
Format	Image file format, see WriteImageFile
FileName	Path and name of BMP image file to load as string

Return Value: None

Remarks: None

LCDWriteString(Dest, Mode, XPos, YPos, XSize, YSize, Format, String)

Function: Used to write the image data to the LCD Display. The data is written at the location specified by the combination of Dest, XPos, and YPos. The Mode determines how the data is written. See remarks below on the format argument.

Mode 0 - Clear: The Display is cleared at the specified location.
Mode 1 - Complement: The Display is complemented at the specified location.
Mode 2 - WriteOpaque: The contents of the background memory in the tablet are transferred to the LCD display, overwriting the contents of the LCD display.
Mode 3 - WriteTransparent: The contents of the background memory in the tablet are combined with and transferred to the visible LCD memory.

Arguments: Integers:

Dest:	0 = Foreground, 1 = Background memory in tablet
Mode	0, 1, 2, 3 as defined above
XPos	Location in LCD coords to draw at
YPos	Same
XSize	Width in LCD pixels
YSize	Height in LCD pixels
Format	Not currently implemented, pass a 0
String:	Text to display on the LCD

Return Value: True if checksum received and verified. False if no checksum or checksum incorrect.

Remarks: Currently, the Format argument of this method is non-functional. Please pass a 0 for this argument.

SetLCDPixelDepth(Depth)

Function: Color image use only with T-LBK57GC and T-LBK43LC devices.
Used to specify color or black and white images when passing an image to paint to the LCD using the LCDWriteFile() or LCDWriteBitmap() functions. Call SetLCDPixelDepth() appropriately prior to painting to the LCD.

Depth = 0 - Tells SigPlus to expect a black and white image for painting.
Depth = 8 - Tells SigPlus to expect a color image for painting.

Arguments: Integers:

Depth:	0 = Black and White, 8 = Color

Return Value: True if checksum received and verified. False if no checksum or checksum incorrect.

SetLCDCompression(LCDCompMode, (LCDZCompMode), LCDCompFast, LCDCompSlow)

Function: Used to specify whether to include data compression when writing text/graphics to a “SE” signature pad, the T-LBK57GC pad, or the T-LBK43LC pad. Also, used to specifiy the level of compression to use (max = 127 for both fast and slow compression).
Arguments: Integers:

LCDCompMode:	0 = Off, 1 = On
LCDZCompMode:	0 = Off, 1 = On
LCDCompFast:	(max = 127)
LCDCompMode:	(max = 127)

Return Value: True if checksum received and verified. False if no checksum or checksum incorrect.

Remarks: To maximize the compression, simply use 127, 127 as the final 2 arguments in the call to SetLCDCompression(). Note that the WIN\SigPlus.ini file, when properly set for the T-LBK57GC pad or T-LBK43LC pad, will already by default include these values, so this function do not necessarily have to be invoked. It would only be pertinent to invoke given the absence of the WIN\SigPlus.ini file, or if set incorrectly.

III. Graphics Properties:

LCDCaptureMode

Function: This property sets the current LCD Mode for the tablet, the tablet is put into the mode as well.

Mode 0 – No LCD Tablet. No LCD commands are sent to the tablet
Mode 1 - Capture Default. CTRL-D is sent to the tablet, which clears the tablet and sets capture mode to be active with Autoerase in the tablet.
Mode 2 - Capture Ink CTRL-T is sent to the tablet, putting the tablet in persistent ink capture mode where the tablet does not automatically clear the display.
Mode 3 - Capture Ink Inverted: CTRL-I is sent to the tablet, where signature ink is displayed inverted against a suitable dark background set using the Graphic functions. Autoerase in the tablet is disabled.

Remarks: If TabletState is TRUE, the mode command is sent to the tablet immediately. If Tabletstate = false, then this inking mode command is sent when the tablet state is next set to true. When LCDWrite functions, or LCDRefresh are called, and tablet state is TRUE, the mode will automatically be set after completing the Write or Refresh function.

IV. Keypad Support: Keypad Methods

KeyPadAddHotSpot(KeyCode, Coords, XPos, YPos, XSize, YSize)

Function: Defines in software the location of a tablet hotspot in logical tablet coordinates. The coordinates of the HotSpot are defined in logical tablet coordinates with (0,0) being the upper left-most pixel. The ini file parameters are used to map the points to logical coordinates if LCD coordinates are used.

Arguments: Integers

KeyCode	Integer value defining the HotSpot
Coords	Coordinate system used for this hot spot 0 = Logical tablet coordinates 1 = LCD Coordinates
XPos	Location (upper left - 0,0)
YPos	Same
XSize	XSize in pixels
YSize	YSize in pixels

Return Value: Void

Remarks: The KeyPadAddHotspot() method will require slight variations in px coord location (arguments 3 and 4, varying for about 1 px to 5 px) from its counterpart LCDWriteBitmap(), LCDWriteFile(), or LCDWriteString() method call. For best results, Topaz recommends the following in terms of adding hot spots:

1. Make the hotspot larger than the image/text representing it…this eliminates “hunting and tapping” on the part of the user.
2. Making all hotspots no smaller than 10 px in both the Y and X axis.

KeyPadQueryHotSpot(KeyCode)

Function: This method queries the data points currently in the control against the logical tablet coordinates mapped by KeyCode. This method returns a true if the control contains data that is within the definition of the KeyCode area on the tablet.

Arguments: Integer

Return Value:

Integer	The number of points within the KeyCode definition

Remarks: None

KeyPadClearHotSpotList()

Function: This method clears the controls internal list of hotspots, created using KeyPadAddHotSpot.

Arguments: None

Return Value: None

Remarks: None

Methods for SigPlus OCX Control

AutoKeyFinish()

Function: Completes the auto key generation function. After this call, the key is ready to be used in saving a bound file, and can be retrieved using GetKey().

Argument: None

Return Value: Void

Remarks: Please see SetAutoKeyData property in the SigPlus Property list, to be used in conjunction with this method.

AutoKeyStart()

Function: Initializes the automatic key generation function.

Argument: None

Return Value: Void

Remarks: Please see SetAutoKeyData property in the SigPlus Property list, to be used in conjunction with this method. The automatic key generation function will derive a key from the data fed to it via the AutoKeyData property. AutoKeyStart is called to initialize the operation, then AutoKeyData is called repeatedly to input more data to the key generation function. When all of the data has been added, then AutoKeyFinish must be called to complete key generation. This feature can be used to produce a key that is uniquely derived from the input data, and can be used to verify that the data has not changed.

AutoTimeStamp()

Function: Sets the Time and Date stamp to the current time and date derived from the clock function of the computer.

Argument: None

Return Value: None

Remarks: None

BitMapBufferWrite()

Function: Creates a bmp file in memory.

Argument: None

Return Value: None

Remarks: Four methods are used to execute the creation of a buffered BMP: BitMapBufferWrite(),BitMapBufferSize(),BitMapBufferByte(Idx), BitMapBufferClose().

BitMapBufferSize()

Function: Returns size of the bmp buffer.

Argument: None

Return Value: Long

Remarks: Use to determine the size of the byte array necessary.

BitMapBufferByte (Idx)

Function: Returns the buffer byte at Idx.

Argument: None

Return Value: Short

Remarks: Use to return the byte at particular location.

BitMapBufferClose()

Function: Frees the buffer.

Argument: None

Return Value: Short

ClearTablet()

Function: Causes the control to clear the current signature, and begin a new one. The display is cleared as well as the signature.

Argument: None

Return Value: Short

Remarks: None

DisableMessageBoxes(Int)

Function: Disables all error messages returned by SigPlus when an error occurs.

Argument: Int value: 1=Disable, 0=Enable

DisplayRefreshInterval(Int Milliseconds)

Function: Set, in milliseconds, the rate at which SigPlus is refreshed during signature capture. Default = 30 ms

Argument: Int milliseconds

Return Value: None

Remarks: None

ExportSigFile(Filename)

Function: The control will write out a signature file in the Topaz image-free raw tablet data vector file format (.sig extension).

Argument: FileName is a string, containing the path and filename to write to.

Return Value:

BOOL	TRUE if successful, FALSE if not successful

Remarks: The full Filename must be provided, including the .sig extension. New in version 2.23 is the (“”) argument feature. This allows the control to export the signature to be saved in the control itself. It can be retrieved into the control window using the ImportSigFile (“”) command. If a blank signature is exported into the control using (“”), the signature-storage contents of the control are erased.

GetBitMapBufferBytes()

Function: Returns all the buffer bytes

Argument: None

Return Value: Byte Array

Remarks: Use to return all the bitmap bytes at particular location.

GetKey(String KeyData)

Function: Returns the current binding key, as a string. The returned string must be allocated by the caller

Argument:

String KeyData, Pointer to a string containing the raw key data, the number of key data bytes used depends on the algorithm used. See Encryption Mode for more details.

Return Value: Void

Remarks: This mode is not active.

GetKeyReceipt()

Function: Returns a 32 bit value that is uniquely derived from the key, it can be used to verify that a document has not been modified if the Auto key feature was used to generate the key.

Argument: None

Return Value: Long 32 bit binary receipt.

Remarks: IN VISUAL BASIC, MUST BE CALLED WITH "GET" PREFIX.

GetKeyReceiptAscii()

Function: Returns the key receipt as a 8 character Ascii hex string

Argument: None

Return Value:

String	The key receipt

Remarks: IN VISUAL BASIC, MUST BE CALLED WITH "GET" PREFIX. See GetKeyReceipt().

GetNumberOfStrokes()

Function: Returns the total number of strokes in the current signature. Can be used to detect if a signature is present, or not.

Argument: None

Return Value:

Short	Decimal value of number of strokes in the signature

Remarks: IN VISUAL BASIC, MUST BE CALLED WITH "GET" PREFIX. A signature may have as few as one stroke. However, the term "stroke" can be interchanged with the term "segment". Since the Topaz tablet is collecting raw tablet data, it cannot determine and does not assume that the data is a signature. The term "stroke" is used to describe the number of segments in the raw data.

GetNumPointsForStroke(StrokeNumber)

Function: Returns the total number of points in the specified.

Argument:

Short	StrokeNumber is the number of the stroke to inquire about. Ranges from 0 to NumberOfStrokes - 1

Return Value:

Short	Decimal value of number of points in the stroke

Remarks: IN VISUAL BASIC, MUST BE CALLED WITH "GET" PREFIX.

GetOposPointArray(String PointArray)

Function: Returns the Opos representation of the signature in a string. Upon return PointArray will point to a string containing the data in Opos format.

Argument:

String	PointArray, Pointer to a string containing the signature data

Return Value:

Long	The total number of points in the array

Remarks: IN VISUAL BASIC, MUST BE CALLED WITH "GET" PREFIX. The string returned must be de-allocated by the caller. Disabled if encryption mode is >/= to 1.

GetOposTotalPoints()

Function: Returns the total number of points in the Opos. PointArray representation of the signature.

Argument: None

Return Value:

Long	The number of points in the array

Remarks: IN VISUAL BASIC, MUST BE CALLED WITH "GET" PREFIX. The number includes the pen lift points at the end of each stroke.

GetNumberOfAttachedTablets()

Function: Returns the total number of Topaz signature pads attached to the target computer.

Argument: None

Return Value:

Int	The number of Topaz pads attached

GetPointXValue(Short StrokeIdx, Short PointIdx)

Function: Returns the X coordinate value for the specified point. The value is in LogicalTablet Coordinates.

Argument:

Short	StrokeIdx, the index of the stroke for the point desired
Short	PointIdx, the index of the point in the stroke

Return Value:

Short	Decimal value of the x coordinate for the point

Remarks: IN VISUAL BASIC, MUST BE CALLED WITH "GET" PREFIX. The Stroke and Point Index must be valid, or 0 is returned. Disabled if encryption mode >/= 2.

GetPointYValue(Short StrokeIdx, Short PointIdx)

Function: Returns the Y coordinate value for the specified point. The value is in LogicalTablet Coordinates.

Argument:

Short	StrokeIdx, the index of the stroke for the point desired
Short	PointIdx, the index of the point in the stroke

Return Value:

Short	decimal value of the y coordinate for the point.

Remarks: IN VISUAL BASIC, MUST BE CALLED WITH "GET" PREFIX. The Stroke and Point Index must be valid, or 0 is returned. Disabled if encryption mode >/= 2.

GetSigData (String SigData)

Function: Returns the sig file representation of the signature in the form of a string, that is allocated, filled and returned in *SigData.

Argument:

String	SigData, Pointer to a string containing the signature data

Return Value: None

Remarks: IN VISUAL BASIC, MUST BE CALLED WITH "GET" PREFIX. The string returned must be de-allocated by the caller. When TabletMode = 1024 (400H), the data format is Memo Field, ASCII, and unicode compatible.

GetSigPlusVersionString()

Function: Returns the installed version of SigPlus.

Argument: None

Return Value: String version of SigPlus (ie, "3.46")

Remarks: IN VISUAL BASIC, MUST BE CALLED WITH "GET" PREFIX. Can only be used with SigPlus 3.46 or above.

GetSigReceipt()

Function: Returns a 32 bit receipt similar to the key receipt. The receipt is formed, by using the auto key generation algorithm on the signature file data. The result can be used to verify that the signature has not been modified.

Argument: None

Return Value:

Long	32 bit binary receipt

Remarks: IN VISUAL BASIC, MUST BE CALLED WITH "GET" PREFIX.

GetSigReceiptAscii(Short StrokeIdx, Short PointIdx)

Function: Same as GetKeyReceiptAscii, but for Sig receipt.

Argument: None

Return Value:

String	The sig receipt

Remarks: IN VISUAL BASIC, MUST BE CALLED WITH "GET" PREFIX. See GetSigReceipt.

GetTabletSerialNumber(Int Index)

Function: Returns the virtual serial number of a signature pad attached based on an index value (total index value obtained using the GetNumberOfAttachedTablets() function). Used when multiple signature pads are to be managed.

Argument: None

Remarks: WILL ONLY FUNCTION WITH TOPAZ PADS THAT CONTAIN THE SIGSERIAL CHIP.

HighlightPoint()

Function: Draws a highlighted circle about the specified point in the currently highlighted stroke.

Argument: StrokeNumber is a short, the number of the point to highlight about. Ranges from 0 to NumPointsForStroke - 1

Return Value: None

Remarks: None

HighlightStroke(StrokeNumber)

Function: Draws the specified stroke highlighted.

Argument: StrokeNumber is a short, the number of the stroke to highlight. Ranges from 0 to NumberOfStrokes - 1

Return Value: None

Remarks: None

ImportSigFile(Filename)

Function: The control will Clear the current signature, read in a signature file in the Topaz vector file format, and display it.

Argument: FileName is a string, containing the path and filename to read from.

Return Value:

BOOL	TRUE if successful, FALSE if not successful

Remarks: New in version 2.23 is the (“”) argument feature. This allows the control to import the signature into the control window from the control itself if a signature has been stored in the control using the ExportSigFile (“”) command. Note that importing signature files into the control is limited to the Topaz vector format files. Image files such as bitmap cannot be imported into SigPlusTM. The full Filename must be provided including the .sig extension. **SIGPLUS OCX V 1.13 AND LATER USES AND CREATES A MODIFIED .SIG FILE COMPARED TO VERSIONS 1.11 AND EARLIER. TO READ-IN PRIOR .SIG FILES, INVERT THE TABLET LOGICAL Y SETTING.**

InitSigPlus()

Function: Used to properly instantiate SigPlus. Must call when SigPlus is instantiated dynamically. No need to call when using a design-time instance of SigPlus.

Remarks: None

LoadTrackGemSig(LPCTSTR FileName, Short Index)

Function: Loads the TrackGem signature corresponding to Index from the TrackGem file FileName.

Argument:

LPCTSTR	FileName, the TrackGem file to extract the signature from
Short	Index, The index of the signature to load from FileName

Return Value:

BOOL	Returns TRUE if the signature was loaded, FALSE if not

Remarks: See TrackGem documentation for details of the file format**.** New in version 2.14.

NumberOfTabletPoints()

Function: Returns the total number of points in the current signature. Can be used to detect if a signature is present, or not.

Argument: None

Return Value:

Short	Decimal value of number of points in the signature

Remarks: A signature should consist of at least 200 points to be considered valid, as this represents approximately 1 second of active signature time. If TabletMode Add 16 is set, then this only reports the number of tablet points that are within the Xstart/Stop and Ystart/Stop boundaries.

SetDisplayAnnotateData(X, Y, Size)

Function: Sets display screen info for Annotate string.

Argument:

Short	X, Location to display Annotate string at in signature display window. Uses logical tablet coordinates, relative to the control rectangle.
Short	Y Location to display Annotate string at in signature display window. Uses logical tablet coordinates, relative to the control rectangle.
Short	Size, size to display Annotate string, in logical tablet coordinate height.

Return Value: None

Remarks: The default is the lower right corner, at ~8% of the screen high. Note that in version 2.25, the Annotation string can have multiple lines. You may wish to position the annotation string at the top of the control rectangle instead of at the bottom. To do this, set DisplayAnnotateData to the same height as the chosen text height. For example, for the default text height of 8% (120 logical tablet coordinates), setting the Y location to 120 will position the annotation string to start in the upper right corner of the control.

SetDisplayTimeStampData(X, Y, Size)

Function: Sets display screen info for Time and Date stamp.

Argument:

Short	X, Location to display TimeStamp string at in signature display window. Uses logical tablet coordinates, relative to the control rectangle.
Short	Y Location to display TimeStamp string at in signature display window. Uses logical tablet coordinates, relative to the control rectangle.
Short	Size, size to display TimeStamp string, in logical tablet coordinates high.

Return Value: None

Remarks: The default is the lower left corner, at ~8% of the screen height.

SetEventEnableMask(Int EventMask)

Function: Enables Pen Up and Pen Down events in SigPlus.

Argument:

Short	1 = Pen Down
	2 = Pen Up
	3 = Pen Down and Pen Up

Return Value: None

Remarks: **IN VISUAL BASIC, USE "SET" PREFIX.** Once the pen event has fired, the SetEnableEventMask method must be called again before the event will once again fire.

SetImageAnnotateData(X, Y, Size)

Function: Sets image file info for Annotate string.

Argument:

Short	X, Location to draw Annotate string at in signature image file. Coordinates range from 0 to TabletLogicalXSize - 1.
Short	Y, Location to draw Annotate string at in signature image file. Coordinates range from 0 to TabletLogicalYSize - 1.
Short	Size, size to draw Annotate string, in pixels high.

Return Value: None

Remarks: The default is the lower left corner, at ~8% of the image high.

SetImageTimeStampData(X, Y, Size)

Function: Sets image file info for Time and Date stamp.

Argument:

Short	X, Location to draw TimeStamp string at in signature image file. Coordinates range from 0 to ImageXSize - 1.
Short	Y, Location to draw TimeStamp string at in signature image file. Coordinates range from 0 to ImageYSize - 1.
Short	Size, size to draw TimeStamp string, in pixels high.

Return Value: None

Remarks: The default is the lower left corner, at ~8% of the image high.

SetLogEnable(Boolean State)

Function: Sets SigPlus up to log into the root folder by default (SigPlus#.log, where # is the current com port assignment). Logging keeps track of SigPlus usage, and can be used for troubleshooting, depending upon the issues at hand.

Return Value: None

SetLogFilePath(String FilePath)

Function: Requires logging be activated (see SetLogEnable) Specifies a location and name for the SigPlus log file to be produced Default is %ROOT%\SigPlus#.log, where # is the current com port assignment).

Return Value: None

SetKey(String KeyData)

Function: Sets the binding key for storing the signature data

Argument:

String	KeyData, Pointer to a string containing the raw key data, the number of key data bytes used depends on the algorithm used. See Encryption Mode for more details.

Return Value: Void

Remarks: None

SetSerialNumberToUse(Int SerialNumber)

Function: Assigns a specific signature pad to an instance of SigPlus when multiple pads are plugged into a single target computer. Argument passed in is the virtual serial number of one of these pads, obtained using GetTabletSerialNumber().

Argument:

Int	The virtual serial number of one of these pads, obtained using GetTabletSerialNumber().

Return Value: Void

Remarks: WILL ONLY FUNCTION WITH TOPAZ PADS THAT CONTAIN THE SIGSERIAL CHIP.

SetSigData(String SigData)

Function: Loads the signature as if it were a sig file, from the data in the string.

Argument:

String	SigData, Pointer to a string containing the signature data

Return Value: None

Remarks: When TabletMode = 1024 (400H), the data format is Memo Field, ASCII, and unicode compatible.

SetBackground(Filename, Mode)

Function: Sets a BMP image as the background to a SigPlus control.

Argument 1:
FileName is a string, containing the BMP path and filename to write
Argument 2:
Int value, not implemented at this time, so pass a 0

Remarks: You will need to set TabletState=1 prior to calling this method. In addition, set the DisplayWindowRes=True prior to calling this method in Visual Basic. BMP will display at true size.

SetBackgroundHandle(Windows Image Handle, Mode)

Function: Sets a BMP image as the background to a SigPlus control.

Argument 1:
Windows Handle to an image, containing the BMP path and filename to write.
Argument 2:
0 - Default, write at size
1 - FitToWindow

Remarks: You will need to set TabletState=1 prior to calling this method. In addition, set the DisplayWindowRes=True prior to calling this method in Visual Basic. BMP will display at true size.

SetEnableColor(Int)

Function: Sets SigPlus up for color Signatures: Background and Foreground Color.

Argument:
0 = off, 1 = on, Default = 0

SetMultiUseEnable(BOOL Enable)

Function: Sets SigPlus up to allow for the use of multiple signature pads with one SigPlus instance.

Argument: 0 = off, 1 = on, Default = 0

SetTabletPortPath(String Clientname)

Function: Overrides the SigPlus.ini's TabletPortPath property when using SigSock.

TabletConnectQuery()

Function: Checks whether a Topaz signature pad is plugged in.

Return Value: BOOL

ImageScreenResolution()

Function: Sets SigPlus up to write images at TabletResolution (settable by setting the TabletResolution property in SigPlus), or at ScreenResolution (the current resolution of the target machine's screen). NOTE: If TabletResolution is to be used, and you change it, make sure it is set to the default value for actual signature capture, or capture results will vary.

Return Value:
Int 0= TabletResolution, 1=ScreenResolution

WriteImageFile(Filename)

Function: The control will write out a signature file in the current Image file format. The default is .BMP.

Argument:

FileName is a string, containing the path and filename to write to

Return Value:

BOOL	TRUE if successful, FALSE if not successful

Remarks: See the property “SetImageFileFormat” to set an image type other than BMP. Other supported image formats include JPG, TIF, WMF, EMF. The full Filename must be provided, including the standard file extension.

SetDynamicPressure(Factor, Final Factor, Offset)

Function: Visually renders velocity information by increasing or decreasing the ink thickness of the signature based on the velocity. Slower movement renders thicker ink, and faster pen movement renders thinner ink.

Argument:

Factor: Determine the input scaling
Final Factor: Determines the output gain
Offset: Determine input scaling

Remarks: Input scaling determines over what portion of the raw pen velocity curve the scaling occurs. It is suggested to contact Topaz support should you be interested in using this feature in your software.

SetAntiAliasParameters(Enable, Linescale, Spotsize)

Function: Adds antialias information to the signature to produce a smoothing effect.

Argument:

Enable: 0 = Off, 1 = On
Linescale: The depth of spots from the line
Spotsize: The size of the spots drawn around the line edge

Properties for SigPlus OCX Control

I. General Properties

SetAsciiDataMode()

Function: This is the Memo Field mode, where the GetSigData and SetSigData formats are unicode compliant for applications using a memo field to store the signature cannot accept binary data. This Memo Field mode is active for all .sig file formats, including normal, cryptographically bound, and compressed. If you save the .sig data in this mode, you must also retrieve the data in the same mode.

Argument:

BOOL	Value for AsciiDataMode
TRUE	AsciiDataMode mode = active
FALSE	AsciiDataMode mode = inactive

Return Value: None

Remarks: Note: This is the preferred way of setting TabletMode = add 1024.

GetAsciiDataMode()

Function: Gets state of AsciiDataMode.

Argument: None

Return Value:

BOOL	Value for AsciiDataMode

Remarks: Note: This is the preferred way of setting TabletMode = add 1024.

SetAnnotate(String)

Function: Sets Annotate string for signature.

Argument:

String	String to use for Annotate string, an ASCII new line character (LF) will break the text into multiple lines. Not that the annotate string is right justified in the OCX control window. Limited to 512 characters.

Return Value: None

Remarks: When enabled, care must be taken as to what is stored in Annotation field. For example, the annotation string should not be used to store a set of measurements relating to biometric information together with a document checksum, hash, or receipt, and the claimed identity of the user. It should also not be used to store a reason for signing or a statement of importance of the signature in relation to a document. One reason to avoid this is that a conflict may occur between the statement and the contents of the document. The annotation can be used to save tablet setup parameters like TabletType, signature compression setting, and the logical X and Y settings. Saving the configuration this way is especially useful when different tablets are used in different applications. During signing, the annotation string can be used to display auxiliary information in the vicinity of the signature on the screen, or can be used to display and store the name of the signature box or signer for reference purposes.

GetAnnotate()

Function: Gets Annotate string stamp for signature display.

Argument: None

Return Value:

String	Annotation

Remarks: When enabled, care must be taken as to what is stored in Annotation field. For example, the annotation string should not be used to store a set of measurements relating to biometric information together with a document checksum, hash, or receipt, and the claimed identity of the user. It should also not be used to store a reason for signing or a statement of importance of the signature in relation to a document. One reason to avoid this is that a conflict may occur between the statement and the contents of the document. The annotation can be used to save tablet setup parameters like TabletType, signature compression setting, and the logical X and Y settings. Saving the configuration this way is especially useful when different tablets are used in different applications. During signing, the annotation string can be used to display auxiliary information in the vicinity of the signature on the screen, or can be used to display and store the name of the signature box or signer for reference purposes.

SetAutoKeyData(String Buffer)

Function: Adds data to the auto key generation function. If called with file name (and path) when AutoKeyStart has not been initialized, this command will generate AutoKey data from a file rather than adding data via string.

Argument:

String	Buffer, Pointer to a string containing the data, to be added to the key generation.

Return Value: Void

Remarks: **Do not confuse this property with the function AutoKeyAddData(), a function for use specifically with C++.** Used with AutoKeyStart and AutoKeyStop methods, but called as a property.

SetCompressionMode()

Function: Sets the current compression mode for .sig files

Argument:

Short	Integer value as follows:
0	No compression (default)
1	Lossless compression with compacted data format.
2-8	Compression ratio of signature stored in .sig file “2” = 1KB typ. “4” = 500 byte typ. “8” = 250 byte typ

Return Value: None

Remarks: When loading a .sig file, the compression mode must be set to the same value that was used when the .sig file was created. Do not use compression with MicroGem™ tablet data collection.

GetCompressionMode()

Function: Returns the current setting of the compression mode for .sig files.

Argument: None

Return Value:

Short	Integer value of .sign fil compression mode as shown above.

Remarks: None

SetEncryptionMode(Short Mode)

Function: Sets Encryption mode. This function is used to set the encryption mode used for importing and exporting sig files.

Argument:

Short Mode	
0	Clear text mode
1	40-bit DES. If a longer key is set, only 40 bits used
2	Higher security encryption mode

Return Value: None

Remarks: If Encryption mode >/= 2, the key can only be set if there are no points in the signature (ClearTablet). When the signature has points while in encryption mode 2, the encryption mode cannot be changed to another mode unless the points are cleared.

GetEncryptionMode()

Function: Gets the current value for the encryption mode.

Argument: None

Return Value:

Short	Current encryption mode.

Remarks: See SetEncryptionMode.

SetJustifyMode(Short Mode)

Function: Sets Justification mode.

Argument:

Short Mode

0	Normal, no justification
1	Justify and zoom signature (upper-left corner)
2	Justify and zoom signature (upper-right corner)
3	Justify and zoom signature (lower-right corner)
4	Justify and zoom signature (lower-left corner)
5	Justify and zoom signature (center of control)
Add 16	Justify based on median value of signature data
Add 32	Justify based on mean value of signature data
Add 48	Justify based on mode value of signature data (These modes can recreate a synthetic baseline To Autojustify the median value of the signature data to the center of the OCX box, you would set JustifyMode to 16 + 5 = 21.)
6	Justify Upper Left Scale
7	Justify Upper Left Scale
8	Justify Upper Left Scale
9	Justify Upper Left Scale
10	Scale without justify of signature.

Return Value: None

Remarks: When using JustifyMode 1-5 and Add modes, a border area around the signature limits can be set in logical tablet coordinates using the JustifyX and JustifyYvalues. In modes 6-9, JustifyX = width of the control in inches x 1000. If JustifyX = 0 when used with modes 6,7,8,9,10 and Acrobat, automatic 1:1 scaling will be set for the transfer of the signature into Acrobat.

GetJustifyMode()

Function: Gets Justification Mode.

Argument: None

Return Value:

Short	Current Justification Mode

Remarks: See SetJustifyMode().

SetJustifyX(Short XPos)

Function: Sets Justification point X coordinate.

Argument:

Short XPos	X coord in Logical Tablet coordinates When JustifyMode = 1-5, sets autojustify X border. In modes 6-10, JustifyX = width of the control in inches x 1000

Return Value: None

Remarks: Implemented for Autojustify mode only.

GetJustifyX()

Function: Gets Justification point X coordinate.

Argument: None

Return Value:

Short	X coord in Logical Tablet coordinates

Remarks: Implemented for Autojustify mode only.

SetJustifyY(Short YPos)

Function: Sets Justification point Y coordinate.

Argument:

Short YPos	Y coord in Logical Tablet coordinates When JustifyMode = 1-5, sets autojustify Y border. In modes 6-10, JustifyY is not used.

Return Value: None

Remarks: Implemented for Autojustify mode only.

GetJustifyY()

Function: Gets Justification point Y coordinate.

Argument: None

Return Value:

Short	Y coord in Logical Tablet coordinates

Remarks: Implemented for Autojustify mode only.

GetKeyString(String SigData)

Function: Provides from the control in Ascii Data (VB script) compatible format, the binding key.

Argument: None

Return Value:

String	SigData, Pointer to a string containing the signature data

Remarks: Can be used to move keys from one instance of the control to another.

SetOpaqueMode(Mode)

Function: Controls whether the signature is displayed on an opaque background, or transparently.

Argument:

BOOL	
TRUE	Use opaque background
FALSE	Use transparent background.
Defaults	To True

Return Value: None

Remarks: Transparent background is only supported in Visual C++ applications. In VB and VBA applications windows causes the transparent background of the control to shine to the desktop.

GetOpaqueMode()

Function: Gets the value of Opaque Mode.

Argument: None

Return Value:

BOOL	
TRUE	Use opaque background
FALSE	Use transparent background.

Remarks: None

SetSaveSigInfo(SaveSigInfo)

Function: Enables/Disables the saving of TimeStamp and Annotate data in the sig file.

Argument:

BOOL	Value for SaveSigInfo
TRUE	The SigInfo will be saved (default)
FALSE	The SigInfo will not be saved

Return Value: None

Remarks: By default, the SaveSigInfo property is disabled. When enabled, care must be taken as to what is stored in Annotation field. For example, the annotation string should not be used to store a set of measurements relating to biometric information together with a document checksum, hash, or receipt, and the claimed identity of the user. It should also not be used to store a reason for signing or a statement of importance of the signature in relation to a document. One reason to avoid this is that a conflict may occur between the statement and the contents of the document. The annotation can be used to save tablet setup parameters like TabletType, signature compression setting, and the logical X and Y settings. Saving the configuration this way is especially useful when different tablets are used in different applications. During signing, the annotation string can be used to display auxiliary information in the vicinity of the signature on the screen, or can be used to display and store the name of the signature box or signer for reference purposes.

SetSigString(String SigData)

Function: Loads sig data into the control in Ascii Data (VB script) compatible format. Data is in the form of an Ascii string.

Argument:

String	SigData, Pointer to a string containing the signature data.

Return Value: None

Remarks: Used to retrieve or place sig data in the control as a property rather than as a method. The data format is Memo Field, ASCII, and unicode compatible.

GetSigString()

Function: Gets sig data from the control in Ascii Data (VB script) compatible format. Data is in the form of an Ascii string.

Argument: None

Return Value:

String	Signature data in hexadecimal string format.

Remarks: Used to retrieve or place sig data in the control as a property rather than as a method. The data format is Memo Field, ASCII, and unicode compatible.

SetTimeStamp(TimeStamp)

Function: Sets Time and Date stamp for signature.

Argument:

String	String to use for TimeStamp string, an ASCII new line character (LF) will break the text into multiple lines. Not that the TimeStamp string is left justified in the OCX control window. Limited to 512 characters.

Return Value: None

Remarks: None

GetTimeStamp()

Function: Gets Time and Date for signature display.

Argument: None

Return Value:

String	Time stamp

Remarks: None

SetZoomPower(Short Power)

Function: Sets Zoom power.

Argument:

Short Power	The magnification power to use, 1 = normal

Return Value: None

Remarks: Not implemented in this version.

GetZoomPower()

Function: Gets Zoom power.

Argument: None

Return Value:

Short	Current Zoom power.

Remarks: See SetZoomPower()**.** Not implemented in this version.

SetZoomX(Short XPos)

Function: Sets Zoom point X coordinate.

Argument:

Short XPos	X coord in Logical Tablet coordinates

Return Value: None

Remarks: Not implemented in this version.

GetZoomX()

Function: Gets Zoom point X coordinate.

Argument: None

Return Value:

Short	X coord in Logical Tablet coordinates

Remarks: Not implemented in this version.

SetZoomY(Short YPos)

Function: Sets Zoom point X coordinate.

Argument:

Short YPos	Y coord in Logical Tablet coordinates

Return Value: None

Remarks: Not implemented in this version.

GetZoomY()

Function: Gets Zoom point Y coordinate.

Argument: None

Return Value:

Short	Y coord in Logical Tablet coordinates

Remarks: Not implemented in this version.

II. Tablet Properties

SetTabletBaudRate (Rate)

Function: Sets Baud rate of tablet.

Argument:

Long	19200 for SignatureGem™ and MicroGem™ tablets 9600 for ClipGem™ tablets. 19200 = default

Return Value: None

Remarks: None

GetTabletBaudRate()

Function: Gets Baud rate of tablet.

Argument: None

Return Value: Long, 19200 or 9600

SetTabletClipping()

Function: Sets mode where signature points are not reported if the points are drawn outside the XStart/Stop, and YStart/Stop window. When active, pen down writing outside the Start/Stop window will return zero points as the number of tablet points.

Argument:

BOOL	Value for TabletClipping
TRUE	TabletClipping mode = active
FALSE	TabletClipping mode = inactive

Return Value: None

Remarks: Note: This is the preferred way of setting TabletMode = add 16.

GetTabletClipping()

Function: Gets state of TabletClipping.

Argument: None

Return Value:

BOOL	Value for TabletClipping

Remarks: Note: This is the preferred way of setting TabletMode = add 16.

SetTabletCOMPort(PortNumber)

Function: Sets the COM port to use.

Argument:

Short	PortNumber is an integer value from 1 to 99.

Return Value: None

Remarks: The SigPlus™ OCX control does not lock up a port as is the case with mousetype drivers. The port is only used when tablet is selected on. Take care to only set COM port when tablet state is OFF.

GetTabletComPort()

Function: Gets the COM port in use.

Argument: None

Return Value:

Short	PortNumber is an integer value

Remarks: None

SetTabletComTest()

Note: This function is deprecated in the SigPlus.ocx file.
Function: Sets hardware connection mode test. When this mode is active, if Topaz tablet is plugged into the appropriate port, TabletState can be set to 1. If tablet is not plugged into the port, TabletState cannot be set to 1.

Argument:

BOOL	Value for TabletComTest
TRUE	TabletComTest mode = active
FALSE	TabletComTest mode = inactive

Return Value: None

Remarks: Note: This is the preferred way of calling SetTabletComTest.
Sample code:

```
m_SigPlus.SetTabletComTest(true);
m_SigPlus.SetTabletState(1);
if(m_SigPlus.GetTabletState() == 1)
{
    //located signature pad
    m_SigPlus.SetTabletComTest(false); //turn off test mode
}
```

GetTabletComTest()

Note: This function is deprecated in the SigPlus.ocx file.
Function: Gets state of TabletComTest.

Argument: None

Return Value:

BOOL	Value for TabletComTest

Remarks: Note: This is the preferred way of determining the current value of the TabletComTest.

SetTabletFilterPoints(Count)

Function: Sets the number of samples to Filter in order to optimize the smoothness of the captured signature.

Argument:

Short	Integer value of filter points. Default = 4.

Return Value: None

Remarks: This property is normally set to 4 (default) for the SignatureGem™ products.

GetTabletFilterPoints()

Function: Gets the number of samples to Filter that is set.

Argument: None

Return Value:

Short	Integer value of filter points

Remarks: See SetTabletFilterPoints above.

SetTabletInvisible()

Function: Sets mode where control becomes invisible and does not draw visibly.

Argument:

BOOL	Value for TabletInvisible
TRUE	TabletInvisible mode = active
FALSE	TabletInvisible mode = inactive

Return Value: None

Remarks: Note: This is the preferred way of setting the control instance to become invisible.

GetTabletInvisible()

Function: Gets the state of TabletInvisible.

Argument: None

Return Value:

BOOL	Value for TabletInvisible

Remarks: Note: This is the preferred way of setting TabletMode = add 64.

SetTabletLCDMode(Mode)

Function: No need to set this unless you are not using the SigPlus.ini file.

Argument:

BOOL
FALSE indicates other Topaz tablet.
Defaults to FALSE

Return Value: None

Remarks: None

GetTabletLCDMode()

Function: Gets LCD Tablet Mode

Argument: None

Return Value:

BOOL
FALSE indicates other Topaz tablet.

Remarks: See SetTabletLCDMode property above.

SetTabletLogicalXSize(Xsize)

Function: Sets the Range of horizontal values to be used in representing signatures. This has no relation to the displayed, image file, or tablet sizes.

Argument:

Short	Integer value of Tablet logical size. Default is 2150.

Return Value: None

Remarks: This is the X-range used for the Topaz vector format, and the internally used format. In most applications, this should be set to match the active tablet X resolution.

GetTabletLogicalXSize()

Function: Returns the Tablet Logical X Size.

Argument: None

Return Value:

Short	Integer value of tablet logical X size

Remarks: None

SetTabletLogicalYSize(Ysize)

Function: Sets the Range of vertical values to be used in representing signatures. This has no relation to the displayed, image file, or tablet sizes.

Argument:

Short	Integer value of Tablet logical size. Default is 1400.

Return Value: None

Remarks: This is the Y-range used for the Topaz vector format, and the internally used format. In most applications, this should be set to match the active tablet Y resolution.

GetTabletLogicalYSize()

Function: Returns the Tablet Logical Y Size.

Argument: None

Return Value:

Short	Integer value of tablet logical Y size

Remarks: None

SetTabletMode(Mode)

Function: Determines if the tablet will accept data from a com port, or from the WinTab driver if WinTab support is desired.

Argument:

Short Mode

0.	Normal mode. When tablet is activated it will accept input from the selected COM port. Default is 0.
1	WinTab mode, when the tablet is activated, it will accept data from the Topaz WinTab driver only.
2	USB mode, when the tablet is activated, it will accept data from the Topaz USB driver.
3	TracGemPOST signature format, Transparent Mode (CTRL T).
4	Older-model SigLite touch tablet format (obsolete mode).

Preferred method of setting the modes above is with TabletType property.

For Reference only:

Add 32	Rendering of signatures is lower resolution for better Visual Basic compatibility. **Preferred method of setting this mode is with DisplayWindowRes property.**
Add 64	Control becomes invisible and does not draw visibly (0X40) **Preferred method of setting this mode is with TabletInvisible property.**
Add 128	Hardware check mode. When this mode is active, Normal mode is also set. If Topaz tablet is plugged into selected COM port, TabletState can be set to 1. If tablet is not plugged into serial port, TabletState cannot be set to 1. **Preferred method of setting this mode is with TabletComTest property.**
Add 256	Allows for signature rotation in the control after capture. Does not save the .sig file rotated. Display rotation only. **Preferred method of setting this mode is with DisplayRotate property.**
Add 768	Allows for signature rotation in the control after capture. Does save the .sig file rotated in this mode. **Preferred method of setting this model is with DisplayRotateSave property.**
Add 1024	(400H). This is the Memo Field mode, where the GetSigData and SetSigData formats are unicode compliant for applications where the field use to store the signature cannot accept binary data. This Memo Field mode is active for all .sig file formats, including normal, cryptographically bound, and compressed. If you save the .sig data in this mode, you must also retrieve the data in the same mode. **Preferred method of setting this mode is with AsciiDataMode property.**

Add modes can be used in any additive combination. For example, to implement Add 32 in WinTab mode, the argument is set to 1 + 32 = 33.

Return Value: None

Remarks: If WinTab support is not available, it will not do anything when in the active state. Conversely, if WinTab is present and the mode is not correct, the tablet will also do nothing when active, because the WinTab driver takes exclusive possession of the COM port.

GetTabletMode()

Function: Returns the current tablet mode of the control.

Argument: None

Return Value: Short

Remarks: See SetTabletMode().

SetTabletRotation(Short TabletRotation)

Function: Sets the orientation for display of tablet data. The data in the sig file is stored in the native tablet orientation.

Argument:

Short	Tablet rotation in degrees, allowed values are: 0 90 180 270

Return Value: None

Remarks: Implemented new in version 2.22.

GetTabletRotation()

Function: Gets current tablet orientation.

Argument: None

Return Value:

Short	Current tablet rotation in degrees

Remarks: See SetTabletRotation().

SetTabletState(State)

Function: Set the capture state of the control. When the control is active, pen data is captured and added to the current signature.

Argument: Short State

1	Active Attaches the control to the COM port and starts accepting data.
0	Inactive. Detaches from the port and stops gathering data. Default state is 0.

Return Value: None. Tablet State is retrieved by using GetTabletState. This can be used for tablet connect or driver detect logic. See TabletMode for further information.

Remarks: Only one tablet control can be active on a given serial port at any point in time.

GetTabletState()

Function: Returns the current capture state of the control.

Argument: None

Return Value:

Short	1 if accepting data, 0 if not

Remarks: None

SetTabletTimingAdvance(Count)

Function: Sets the number of timing ticks set into software to match the tablet being used. This is an internal Topaz function.

Argument:

Short	Integer value of timing ticks. Default = 4.

Return Value: None

Remarks: This property is normally set to 4 (default) for the SignatureGem™ products and is set to 2 for the MicroGem™ or ClipGem™ products.

GetTabletTimingAdvance()

Function: Gets the decimal number of timing ticks that are used. (Internal Topaz function).

Argument: None

Return Value:

Short	Integer value of timing ticks

Remarks: See SetTimingAdvance above.

SetTabletType(Mode)

Function: Determines if the tablet will accept data from a com port, WinTab driver, USB driver or other method of data input.

Argument: Short Mode

0	Normal mode. When tablet is activated it will accept input from the selected com port. Default is 0.
6	-HSB tablet (USB mode for tablets using HID driver).

Return Value: None

Remarks: If WinTab support is not available, it will not do anything when in the active state. Conversely, if WinTab is present and the mode is not correct, the tablet will also do nothing when active, because the WinTab driver takes exclusive possession of the Com Port. This is the preferred way of setting TabletMode for tablet input.

GetTabletType()

Function: Gets TabletType value.

Argument: None.

Return Value:

Short	Integer value of TabletType

Remarks: Note: This is the preferred way of setting TabletMode for tablet input.

SetTabletXStart(XStart)

Function: Sets the X position in tablet coordinates, of the upper left hand corner of the control signature box.

Argument:

Short	The integer value, in tablet coordinates representing the left-hand edge of the signature box: See table of recommended values on Page 2 of this document. Default = 500

Return Value: None

Remarks: To position the starting point of the signature box in tablet space to a particular spot on the tablet, simply calculate the tablet coordinate of where you wish the left-hand edge of the signature box to start. For example, for the box to be active starting 1 inch in from the left edge on a SignatureGem™ 4X5 tablet, set XStart to 500 + 410 = 910.

GetTabletXStart(XStart)

Function: Gets the X position of the left-hand edge of the active area of the signature box, in tablet coordinates.

Argument: None

Return Value:

Short	Integer value of tablet X start

Remarks: See SetTabletXStart above.

SetTabletXStop(XStop)

Function: Sets the X position in tablet coordinates, of the lower right hand corner of the control signature box.

Argument:

Short	The integer value, in tablet coordinates representing the right-hand edge of the signature box: See table of recommended values on Page 2 of this document. Default = 2650.

Return Value: None

Remarks: To position the ending point of the signature box in tablet space to a particular spot on the tablet, simply calculate the tablet coordinate of where you wish the right-hand edge of the signature box to stop. For example, for the box to stop being active at 4 inches from the left edge on a SignatureGem® 4X5 tablet, set XStart to 500 + (410x4) = 2140.

GetTabletXStop(XStop)

Function: Gets the X position of the right-hand edge of the active area of the signature box, in tablet coordinates.

Argument: None

Return Value:

Short	Integer value of tablet X stop

Remarks: See SetTabletXStart above.

SetTabletYStart(YStart)

Function: Sets the Y position in tablet coordinates, of the upper-edge of the control signature box.

Argument:

Short	The integer value, in tablet coordinates representing the upper edge of the signature box: See table of recommended values on Page 2 of this document. Default = 350.

Return Value: None

Remarks: To position the starting point of the signature box in tablet space to a particular spot on the tablet, simply calculate the tablet coordinate of where you wish the upper left corner of the signature box to start. For example, for the box to be active starting 1 inch below the top usable edge (bottom of the paperguide) on a SignatureGem™ 4X5 tablet, set YStart to 575 + 410 = 985.

GetTabletYStart(YStart)

Function: Gets the Y position of the left-hand edge of the active area of the signature box, in tablet coordinates.

Argument: None

Return Value:

Short	Integer value of tablet Y start

Remarks: See SetTabletXStart above.

SetTabletYStop(YStop)

Function: Sets the Y position in tablet coordinates, of the lower right hand corner of the control signature box.

Argument:

Short	The integer value, in tablet coordinates representing the lower edge of the signature box: See table of recommended values on Page 2 of this document. Default = 2100.

Return Value: None

Remarks: To position the ending point of the signature box in tablet space to a particular spot on the tablet, simply calculate the tablet coordinate of where you wish the right-hand edge of the signature box to stop. For example, for the box to stop being active 3 inches below the top usable edge (bottom of the paper guide on a SignatureGem™ 4X5 tablet, set XStart to 575+(410x2) =1395.

GetTabletYStop(YStop)

Function: Gets the Y position of the bottom edge of the active area of the signature box, in tablet coordinates.

Argument: None

Return Value:

Short	Integer value of tablet Y stop.

Remarks: See SetTabletXStart above.

III. Display Properties

SetDisplayAnnotate(DisplayAnnotate)

Function: Enables/disables the display on the Annotate string in the window.

Argument:

BOOL	Value for DisplayAnnotate
TRUE	The Annotate string will be displayed
FALSE	The Annotate string will not be displayed

Return Value: None

Remarks: When enabled, care must be taken as to what is stored in Annotation field. For example, the annotation string should not be used to store a set of measurements relating to biometric information together with a document checksum, hash, or receipt, and the claimed identity of the user. It should also not be used to store a reason for signing or a statement of importance of the signature in relation to a document. One reason to avoid this is that a conflict may occur between the statement and the contents of the document. The annotation can be used to save tablet setup parameters like TabletType, signature compression setting, and the logical X and Y settings. Saving the configuration this way is especially useful when different tablets are used in different applications. During signing, the annotation string can be used to display auxiliary information in the vicinity of the signature on the screen, or can be used to display and store the name of the signature box or signer for reference purposes.

GetDisplayAnnotate()

Function: Returns the value of DisplayAnnotate.

Argument: None

Return Value:

BOOL	Value of DisplayAnnotate

Remarks: See SetDisplayAnnotate.

SetDisplayAnnotatePosX(Short XPos)

Function: Sets Display Annotate X location.

Argument:

Short	XPos X coord

Return Value: None

Remarks: See SetDisplayAnnotateData().

GetDisplayAnnotatePosX()

Function: Gets Display Annotate X location.

Argument: None

Return Value:

Short	X coord

Remarks: See SetDisplayAnnotateData().

SetDisplayAnnotatePosY(Short YPos)

Function: Sets Display Annotate Y location.

Argument:

Short	YPos Y coord.

Return Value: None

Remarks: See SetDisplayAnnotateData()

GetDisplayAnnotatePosY()

Function: Gets Display Annotate Y location.

Argument: None

Return Value:

Short	Y coord

Remarks: See SetDisplayAnnotateData().

SetDisplayAnnotateSize(Short Size)

Function: Sets Display Annotate size.

Argument:

Short Size	Font size

Return Value: None

Remarks: See SetDisplayAnnotateData().

GetDisplayAnnotateSize()

Function: Gets Display Annotate Size.

Argument: None

Return Value:

Short	Font size

Remarks: See SetDisplayAnnotateData().

SetDisplayPenWidth(Int Width)

Function: Sets pen width for the displayed signature, in pixels.

Argument:

Short	Integer value for display pen width

Return Value: None

Remarks: None

GetDisplayPenWidth()

Function: Gets pen width for the displayed signature, in pixels.

Argument: None

Return Value:

Short	Integer value for display pen width

Remarks: None

SetDisplayRotate()

Function: Sets mode allowing signature rotation in the control after capture. Does not save the .sig file rotated. Display rotation only. The rotation value is set by TabletRotation.

Argument:

BOOL	Value for DisplayRotate
TRUE	DisplayRotate mode = active
FALSE	DisplayRotate mode = inactive

Return Value: None

Remarks: Note: This is the preferred way of setting TabletMode = add 256**.** Can be used to rotate signatures after capture, if the signature was accidentally taken in a rotation orientation during signature capture. Normally, to change the tablet orientation during capture, the TabletRotation property is used.

GetDisplayRotate()

Function: Gets state of DisplayPenWidth.

Argument: None

Return Value:

BOOL	Value for DisplayRotate

Remarks: Can be used to rotate signatures after capture, if the signature was accidentally taken in a rotation orientation during signature capture. Normally, so change the tablet orientation during capture, the TabletRotation property is used.

SetDisplayRotateSave()

Function: Sets mode allowing signature rotation and the save of signature in rotated format after capture. Does not save the .sig file rotated. Display rotation only. The rotation value is set by TabletRotation.

Argument:

BOOL	Value for DisplayRotateSave
TRUE	DisplayRotateSave mode = active
FALSE	DisplayRotateSave mode = inactive

Return Value: None

Remarks: Note: This is the preferred way of setting TabletMode = add 768**.** Can be used to rotate and then save in rotated format, signatures after capture, if the signature was accidentally taken in a rotated orientation during signature capture. Normally, to change the tablet orientation during capture, only the TabletRotation property is used.

GetDisplayRotateSave()

Function: Gets state of DisplayRotateSave.

Argument: None

Return Value:

BOOL	Value for DisplayRotateSave

Remarks: Can be used to rotate and then save in rotated format, signatures after capture, if the signature was accidentally taken in a rotated orientation during signature capture. Normally, so change the tablet orientation during capture, only the TabletRotation property is used.

SetDisplayTimeStamp(DisplayTimeStamp)

Function: Enables/Disables the display on the Time and Date stamp in the window.

Argument:

BOOL	Value for DisplayTimeStamp
TRUE	The TimeStamp string will be displayed
FALSE	The TimeStamp will not be displayed

Return Value: None

Remarks: None

GetDisplayTimeStamp()

Function: Returns the value of DisplayTimeStamp.

Argument: None

Return Value:

BOOL	Value of DisplayTimeStamp

Remarks: See SetDisplayTimeStamp.

SetDisplayTimeStampPosX(Short XPos)

Function: Sets Display TimeStamp X location.

Argument:

Short XPos	X coord

Return Value: None

Remarks: See SetDisplayTimeStampData().

GetDisplayTimeStampPosX()

Function: Gets Display TimeStamp X location.

Argument: None

Return Value:

Short	X coord

Remarks: See SetDisplayTimeStampData().

SetDisplayTimeStampPosY(Short YPos)

Function: Sets Display TimeStamp Y location.

Argument:

Short YPos	Y coord

Return Value: None

Remarks: See SetDisplayTimeStampData().

GetDisplayTimeStampPosY()

Function: Gets Display TimeStamp Y location.

Argument: None

Return Value:

Short	Y coord

Remarks: See SetDisplayTimeStampData().

SetDisplayTimeStampSize(Short Size)

Function: Sets Display TimeStamp size.

Argument:

Short Size	Font size

Return Value: None

Remarks: See SetDisplayTimeStampData().

GetDisplayTimeStampSize()

Function: Gets Display TimeStamp Size.

Argument: None

Return Value:

Short	Font size

Remarks: See SetDisplayTimeStampData().

SetDisplayWindowRes()

Function: Sets mode which renders signatures in lower (screen) resolution for compatibility in printing directly from VB. **MUST BE USED WHEN PRINTING DIRECTLY FROM A VISUAL BASIC FORM.**

Argument:

BOOL	Value for DisplayWindowRes
TRUE	DisplayWindowRes mode = active
FALSE	DisplayWindowRes mode = inactive

Return Value: None

Remarks: Note: This is the preferred way of setting TabletMode = add 32.

GetDisplayWindowRes()

Function: Gets state of DisplayWindowRes.

Argument: None

Return Value:

BOOL	Value for DisplayWindowRes

Remarks: Note: This is the preferred way of setting TabletMode = add 32.

TabletModelNumber()

Function: Returns the tablet’s internal SigModel number (Must be used in conjunction with specialized Topaz tablets that support this functionality).

Return Value: None

Remarks: None

TabletSerialNumber()

Function: Returns the tablet’s internal SigSerial number (Must be used in conjunction with specialized Topaz tablets that support this functionality).

Return Value: None

Remarks: None

IV. Image Properties

SetImageAnnotate(ImageAnnotate)

Function: Enables/Disables the display on the Annotate string in the image file

Argument:

BOOL	Value for ImageAnnotate
TRUE	The ImageAnnotate string will be drawn
FALSE	The ImageAnnotate will not be drawn

Return Value: None

Remarks: None

GetImageAnnotate()

Function: Returns the value of ImageAnnotate.

Argument: None.

Return Value:

BOOL	Value of ImageAnnotate

Remarks: See SetImageAnnotate.

SetImageAnnotatePosX(Short XPos)

Function: Sets Image Annotate X location.

Argument:

Short XPos	X coord

Return Value: None

Remarks: See SetImageAnnotateData()

GetImageAnnotatePosX()

Function: Gets Image Annotate X location.

Argument: None

Return Value:

Short	X coord

Remarks: See SetImageAnnotateData()

SetImageAnnotatePosY(Short YPos)

Function: Sets Image Annotate Y location.

Argument:

Short YPos	Y coord

Return Value: None

Remarks: See SetImageAnnotateData().

GetImageAnnotatePosY()

Function: Gets Image Annotate Y location.

Argument: None

Return Value:

Short	Y coord

Remarks: See SetImageAnnotateData().

SetImageAnnotateSize(Short Size)

Function: Sets Image Annotate size.

Argument:

Short Size	Font size

Return Value: None

Remarks: See SetImageAnnotateData().

GetImageAnnotateSize()

Function: Gets Image Annotate Size.

Argument: None

Return Value:

Short	Font size

Remarks: See SetImageAnnotateData().

SetImageFileFormat(Format)

Function: Sets the current format to use for Image files. The default is .BMP. The choices are:

Argument:

Short	Integer value as follows:
0	Compressed BMP (default) must have .bmp ext
1	Uncompressed BMP must have .bmp ext
2	Mono. BMP must have .bmp ext
3	JPG Q=20 must have .jpg ext
4	JPG Q=100 must have .jpg ext
5	Uncompressed TIF must have .tif ext
6	Compressed TIF must have .tif ext
7	WMF (windows metafile) must have .wmf ext
8	EMF (enhanced metafile) must have .emf ext
9	TIF (1-bit) must have .tif ext
10	TIF (1-bit inverted) must have .tif ext
11	TIF (Group 4) must have .tif ext

Return Value: None

Remarks: The file extension is not assumed in the WriteImageFile function. Any extension can be specified, but it should match the specified file format. Note that writing an image file is completely different from a .sig file. An image file is just a standard image format of what is seen in the control and cannot be cryptographically bound or decrypted by the SigPlus control. The .sig file format does not store an image, but rather uses a unique method of preserving the original signature data from the tablet. To create a metafile with a transparent background use a trio of instructions to set TabletOpaque = False, then WriteImageFile, the TabletOpaque = True. For all other image files, TabletOpaque must be true when the image file is written.

GetImageFileFormat()

Function: Returns the current setting of the image file format.

Argument: None

Return Value:

Short	Integer value of image file type as shown above

Remarks: None

SetImagePenWidth(Width)

Function: Sets the current pen width to use for signature in Image files.

Argument:

Short	The decimal integer representing the number of pixels of width to make the pen width for image files. Default is 1.

Return Value: None

Remarks: This command 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.

GetImagePenWidth()

Function: Gets the current pen width used for signature in Image files.

Argument: None

Return Value:

Short	Integer value of image pen width

Remarks: None

SetImageTimeStamp(ImageTimeStamp)

Function: Enables/Disables the display on the Time and Date stamp in the image file.

Argument:

BOOL	Value for ImageTimeStamp
TRUE	The TimeStamp string will be drawn
FALSE	The TimeStamp will not be drawn

Return Value: None

Remarks: None

GetImageTimeStamp()

Function: Returns the value of ImageTimeStamp.

Argument: None

Return Value:

BOOL	Value of ImageTimeStamp

Remarks: See SetImageTimeStamp.

SetImageTimeStampPosX(Short XPos)

Function: Sets Image TimeStamp X location.

Argument:

Short XPos	X coord

Return Value: None

Remarks: See SetImageTimeStampData().

GetImageTimeStampPosX()

Function: Gets Image TimeStamp X location.

Argument: None

Return Value:

Short	X coord

Remarks: See SetImageTimeStampData().

SetImageTimeStampPosY(Short YPos)

Function: Sets Image TimeStamp Y location.

Argument:

Short YPos	Y coord

Return Value: None

Remarks: See SetImageTimeStampData().

GetImageTimeStampPosY()

Function: Gets Image TimeStamp Y location.

Argument: None

Return Value:

Short	Y coord

Remarks: See SetImageTimeStampData().

SetImageTimeStampSize(Short Size)

Function: Sets Image TimeStamp size.

Argument:

Short Size	Font size

Return Value: None

Remarks: See SetImageTimeStampData().

GetImageTimeStampSize()

Function: Gets Image TimeStamp Size.

Argument: None

Return Value:

Short	Font size

Remarks: See SetImageTimeStampData().

SetImageXSize(X Size)

Function: Sets the current Image file width in pixels.

Argument:

Short	Integer number of pixels desired in the image file. Defaults to 0, which links image size to equal LogicalXSize value.

Return Value: None

Remarks: Maximum useful resolution is based on 410 points per inch of tablet active area. Minimum useful resolution is based upon trade-off of desired image coarseness versus resulting image file size.

GetImageXSize

Function: Gets the current Image file width in pixels.

Argument: None

Return Value:

Short	Integer value of image file X size. If zero, then value is as specified by the LogicalXSize.

Remarks: The currently set image file width is a decimal integer in pixels.

SetImageYSize(Y Size)

Function: Sets the current Image file height in pixels.

Argument:

Short	Integer number of pixels desired in the image file. Defaults to 0, which links image size to equal LogicalYSize value.

Return Value: None

Remarks: Maximum useful resolution is based on 410 points per inch of tablet active area. Minimum useful resolution is based upon trade-off of desired image coarseness versus resulting image file size.

GetImageYSize()

Function: Gets the current Image file height in pixels.

Argument: None

Return Value:

Short	Integer value of image file Y size. If zero, then value is as specified by the LogicalYSize.

Remarks: The currently set image file width is a decimal integer in pixels.