



SOFTWARE DEVELOPERS MANUAL for *SigPlus*[®] SHARED C LIBRARY
9/23/05 REVA

FOR TECHNICAL SUPPORT CALL 805 520-8286
OR E-MAIL TO: support@topazsystems.com

The Topaz SigPlus shared C Library is composed essentially of four different classes:

- Signature
- TabletInterface
- LCDInterface
- TabletParameters

The class variables and functions are defined in each of the classes .h files, these are:

- Signature.h
- TabletInterface.h
- LCDInterface.h
- TabletParameters.h

Most of the properties and functions of the SigPlus ActiveX match the class variables and functions of the SigPlus Shared C Library. Therefore, please refer to the SigPlus ActiveX documentation for the majority of the definitions to be found below.

For download: www.topazsystems.com/Software/download/sigplusdocs.zip

For viewing: <http://www.topazsystems.com/Software/sigplushelp/index.htm>

Any definitions **not** found in the ActiveX documentation that are limited to the Shared C Library, are listed and defined below.

Please see SimpleDemo.cpp, SigTabLtTest.cpp and LCD1x5Demo.cpp for real-world implementation of these functions and properties.

INDEX:

Signature Class Functions: p 2-15

TabletInterface Class Functions: p. 16-21

LCDInterface Class Functions: p. 22-25

TabletParameters Class Functions: p. 25

Signature Class Variables: p 26-31

TabletInterface Class Variables: p. 32-34

LCDInterface Class Variables: p. 35

TabletParameters Class Variables: p. 36-38

Signature class

Signature class can be instantiated, allowing the methods in the class to be called.

For example:

```
Signature*          Sig;  
sig = new Signature();
```

CaptureThread()

Function:

Arguments:

Return Value: Void

Remarks:

NewStroke()

Function:

Arguments:

Return Value: Void

Remarks:

AddPointToStroke()

Function:

Arguments: Integers:

XValue

YValue

Return Value: Void

Remarks:

NumberOfStrokes()

Function:

Returns the current number of strokes in the signature. Can be used to verify that the signature is present.

Arguments:

Return Value: Int

Remarks:

NumberOfPointsInStroke()

Function:

Returns the total number of points in the specified stroke.

Arguments: Integers:

Stroke

Return Value: Int

Remarks:



TotalPoints()

Function: Returns the current number of points in the signature. Can be used to verify that the signature is present.

Arguments:

Return Value: Int

Remarks:

GetPointXValue()

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

Arguments: Integers:
Strokeldx
PointIdx

Return Value: Int

Remarks:

GetPointYValue()

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

Arguments: Integers:
Strokeldx
PointIdx

Return Value: Int

Remarks:

IsValidPoint()

Function:

Arguments: Shorts:
StrokeNumber
PointNumber

Return Value: Bool

Remarks:

GetPointForStroke()

Function:

Arguments: Shorts:
StrokeNumber
PointNumber

Return Value: SigPoint

Remarks:

ClearSignature()

Function: Sets the number of signature points to zero, thus clearing out the signature.

Arguments:

Return Value: Void

Remarks:

ClearHotSpotPoints()

Function: Removes all created hotspots.

Arguments:

Return Value: Void

Remarks:

StartCapture()
Function: Must be called prior to capturing pen data.
Arguments: TabletInterface:
*Tablet

Return Value: Bool
Remarks:

StopCapture()
Function: Closes the port and ends pen data capture.
Arguments:

Return Value: Void
Remarks:

ProcessTabletSample()
Function:
Arguments: Integers:
Xpos
YPos
Status

Return Value: Void
Remarks:

IsPenNear()
Function:
Arguments: Integers:
Status

Return Value: Bool
Remarks:

IsPenDown()
Function:
Arguments: Integers:
Status

Return Value: Bool
Remarks:

SamplesAreDifferent()
Function:
Arguments: TabletSamples:
*A
*B

Return Value: Bool
Remarks:

OnPenDown()
Function:
Arguments: TabletSample:
*Sample

Return Value: Bool
Remarks:

OnPenUp ()

Function:
Arguments: TabletSample:
 *Sample

Return Value: Void
Remarks:

OnPenMove ()

Function:
Arguments: TabletSample:
 *Sample

Return Value: Void
Remarks:

InitializeFilter ()

Function:
Arguments: TabletSample:
 *Sample

Return Value: Void
Remarks:

FilterPoint ()

Function:
Arguments: TabletSample:
 *Sample

Return Value: Void
Remarks:

SetSigWindow ()

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: Shorts:
 Coords
 NewXPos
 NewYPos
 NewXSize
 NewYSize

Return Value: Void
Remarks:

ClearSigWindow()

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

Arguments: Short:
Inside

Return Value: Void

Remarks:

ExportSigFile()

Function: Saves signature file to specified path.

Arguments: Char:
*FileName

Return Value: Bool

Remarks:

WriteSigFile()

Function:

Arguments: SigFile:
&DestFile

Return Value: Bool

Remarks:

ExportSigFileComp0()

Function:

Arguments: SigFile:
&DestFile

Return Value: Bool

Remarks:

ExportSigFileComp2()

Function:

Arguments: SigFile:
&DestFile

Return Value: Bool

Remarks:

ImportSigFile()

Function: Opens signature file from specified path.

Arguments: Char:
*FileName

Return Value: Bool

Remarks:

ReadSigFile()

Function:

Arguments: SigFile:
&SourceFile

Return Value: Bool

Remarks:

ImportSigFileComp0()

Function:
Arguments: SigFile:
 &SourceFile

Return Value: Bool
Remarks:

ImportSigFileComp2()

Function:
Arguments: SigFile:
 &SourceFile

Return Value: Bool
Remarks:

LoadSigInfoBinary()

Function:
Arguments: SigFile:
 &SourceFile

Return Value: Void
Remarks:

LoadAnnotationString()

Function:
Arguments: SigFile:
 &SourceFile

Char:
*Buffer

Integer:
Length

Return Value: Void
Remarks:

SaveSigInfoBinary()

Function:
Arguments: SigFile:
 &DestFile

Return Value: Void
Remarks:

FileReadError()

Function:
Arguments: Char:
 *ErrMsg

Return Value: Void
Remarks:

GetTextNextInteger()

Function:
Arguments: SigFile:
 &Source

Return Value: Int
Remarks:

InitSigInfo()

Function:
Arguments:

Return Value: Void
Remarks:

SetEncryptionMode()

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

Arguments: Short:
 Mode

Return Value: Void
Remarks:

EnCryptFile()

Function:
Arguments: SigFile:
 *FromFile
 *ToFile

Return Value: Bool
Remarks:

DeCryptFile()

Function:
Arguments: SigFile:
 *FromFile
 *ToFile

Return Value: Bool
Remarks:

SetKey()

Function: Sets the encryption key for storing the signature data.
Arguments: Unsigned Char:
 *KeyData

Integer:
Length

Return Value: Void
Remarks:

GetKey()

Function:
Arguments: Unsigned Char:
 *KeyData

Integer:
Length

Return Value: Void

Remarks:

GetKeyString()

Function:
Arguments: Char:
 *Result

Integer:
Length

Return Value: Void

Remarks:

SetKeyString ()

Function:
Arguments: Char:
 *KeyString

Return Value: Void

Remarks:

SetAutoKeyData ()

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 BSTR.

Arguments: Unsigned Char:
 *NewValue

Return Value: Void

Remarks:

AutoKeyStart()

Function: Initializes the automatic key generation function.

Arguments:

Return Value: Void

Remarks:

AutoKeyAddData ()

Function:
Arguments: Unsigned Char:
 *Buffer

Integer:
Len

Return Value: Void

Remarks:

AutoKeyFromFile()

Function:

Arguments: Const Char:
 *FileName

Return Value: Void

Remarks:

AutoKeyFinish()

Function:

Completes the auto key generation function. After this call, the key is ready to be used in saving an encrypted file, and can be retrieved using GetKey() functions.

Arguments:

Return Value: Void

Remarks:

GetKeyReceipt()

Function:

Arguments:

Return Value: Long

Remarks:

GetKeyReceiptAscii()

Function:

Arguments: Char:
 *Result

Return Value: Void

Remarks:

GetSigReceipt()

Function:

Arguments:

Return Value: Long

Remarks:

GetSigReceiptAscii()

Function:

Arguments: Char:
 *Result

Return Value: Void

Remarks:

Make40BitKey()

Function:

Arguments: Unsigned Char:
 *Src
 *Dst

Return Value: Void

Remarks:

ChangeKeyAllowed()

Function:

Arguments:

Return Value: Bool

Remarks:

ToAsciiHex()

Function:

Arguments: Integer:
V

Return Value: Char

Remarks:

FromAsciiHex()

Function:

Arguments: Char:
Ch

Return Value: Int

Remarks:

KeyPadAddHotSpot()

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: Shorts:
KeyCode
CoordToUse
XPos
YPos
XSize
YSize

Return Value: Void

Remarks:

KeyPadQueryHotSpot()

Function:

Arguments: Short:
KeyCode

Return Value: Short

Remarks:

KeyPadClearHotSpotList()

Function: Removes all hotspots from the hotspot list.

Arguments:

Return Value: Void

Remarks:

KeyPadConvertToLogical()

Function:

Arguments: Shorts:
LCDPos
LCDSize
LCDStart
LCDStop

Return Value: Short

Remarks:

DrawSignature()

Function: Must be called prior to the WriteImageFile method.

Arguments: Integers:
DrawJustifyX
DrawJustifyY
Width
Height
DrawMode
PenWidth

Return Value: Bool

Remarks:

ScalePoint()

Function:

Arguments: SigWindowTypes:
*Source
*Dest

SigPoint:
&P

Return Value: Void

Remarks:

InitAutoJustify()

Function:

Arguments: SigWindowType:
*Source
*Dest

Return Value: Void

Remarks:

FindMean()

Function:

Arguments: Integers:
*Hist
TotalPoints

Return Value: Int

Remarks:

FindMedian()

Function:

Arguments: Integers:
 *Hist
 TotalPoints

Return Value: Int

Remarks:

FindMode()

Function:

Arguments: Integer:
 *Hist

Return Value: Int

Remarks:

DumpHistogramData()

Function:

Arguments: Integers:
 *Hist
 TotalPoints

Return Value: Void

Remarks:

ScalePenWidth()

Function:

Arguments: SigWindowType:
 *DRect

Integer:

RawWidth

Return Value: Int

Remarks:

DumpImage()

Function:

Arguments: Char:
 *FileName

Return Value: Void

Remarks:

DrawSignatureAntiAlias()

Function:

Arguments: SigWindowTypes:
 *TRect
 *Drect

Integer:

PenWidth

Return Value: Bool

Remarks:



WriteImageFile()

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

Arguments: ImageFileFormats:
Format

Char:
*FileName

Return Value: Bool

Remarks:

TabletInterface class

TabletInterface class can be instantiated, allowing the methods in the class to be called.

For example:

```
TabletInterface* Tablet;  
Tablet = new TabletInterface();
```

OpenTablet()

Function: Checks if the tablet is connected and if so allows it to accept pen data.

Arguments:

Return Value: Bool

Remarks:

CloseTablet()

Function: Stops tablet from accepting pen data.

Arguments:

Return Value: Bool

Remarks:

GetTabletData()

Function:

Arguments: Unsigned Long:
TimeOutInMs

Return Value: Unsigned Long.

Remarks:

GetTabletPointData()

Function:

Arguments: Unsigned Long:
TimeOutInMs

Return Value: Unsigned Long

Remarks:

PutTabletData()

Function:

Arguments: Unsigned Char:
*Buffer

Integer:

Count

Return Value: Bool

Remarks:

TabletDataReady()

Function:

Arguments:

Return Value: Int

Remarks:

TabletPresent()

Function:

Arguments:

Return Value: Bool

Remarks:

ProcessInputData()

Function:

Arguments: ProcessSerialData:
*ProclF

Return Value: Void

Remarks:

SendSerialNPData()

Function:

Arguments: Unsigned Char:
StatusByte

Integers:

XPos

YPos

Return Value: Void

Remarks:

ProcessNonPointData()

Function:

Arguments: Unsigned Char:
StatusByte
*DataBytes

Return Value: Void

Remarks:

WaitForNonPointCmdData()

Function:

Arguments: Unsigned Long:
TimeOut

Integer:

ReturnCount

Return Value: Bool

Remarks:

// TabletCore.cpp

GetTabletIfType()

Function:

Arguments:

Return Value: TabletType.

Remarks:

ScaleCoordData()

Function:

Arguments: Integers:
 *XPos
 *YPos

Return Value: Void

Remarks:

PutInPointBuffer()

Function:

Arguments: Unsigned Long:
 Point

Return Value: Void

Remarks:

GetFromPointBuffer()

Function:

Arguments:

Return Value: Unsigned Long.

Remarks:

PointsInPointBuffer()

Function:

Arguments:

Return Value: Int

Remarks:

// TabletParameters.cpp

InitTabletParams()

Function:

Arguments:

Return Value: Void

Remarks:

GetTabletParameters()

Function: Returns settings from ini to TabletParameters construct.

Arguments: TabletParameters:
*Params

Return Value: Void

Remarks:

SetTabletParameters()

Function: Modify settings in TabletParameters construct.

Arguments: TabletParameters:
*Params

Return Value: Void

Remarks:

LoadDefaultParameters()

Function:

Arguments:

Return Value: Void

Remarks:

LoadIniParameters()

Function:

Arguments:

Return Value: Void

Remarks:

// Serial and HIDSerial support code

IsStartByte()

Function:Arguments: Unsigned Char:
 ByteReturn Value: BoolRemarks:

ConvertCoordData()

Function:Arguments: Unsigned Char:
 Lsb
 MsbReturn Value: IntRemarks:

IsBadPacket()

Function:Arguments: Unsigned Char:
 *BufferReturn Value: BoolRemarks:

IsSerialPenData()

Function:Arguments: Unsigned Char:
 StatusReturn Value: BoolRemarks:

InitProcessInputData()

Function:Arguments:Return Value: VoidRemarks:

CloseProcessInputData()

Function:Arguments:Return Value: VoidRemarks:

ResetSerialInputState()

Function:Arguments:Return Value: VoidRemarks:

LCDInterface class

LCDInterface class can be instantiated, allowing the methods in the class to be called. It takes an instance of TabletInterface argument.

For example:

```
LCDInterface* LCD;
```

```
LCD = new LCDInterface( Tablet );
```

GetLCDCaptureMode()

Function:

Arguments:

Return Value: LCDCaptureModeType.

Remarks:

SetLCDCaptureMode()

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.

Arguments: LCDCaptureModeType:
newMode

Return Value: Void

Remarks:

LCDSetWindow()

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: Shorts:
XPos
YPos
XSize
YSize

Return Value: Bool

Remarks:

LCDWriteString()

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: Shorts:
Dest
Mode
XPos
YPos

CharacterMap:
*Font

Char:
*Str

Return Value: Bool

Remarks:

LCDStringWidth()

Function: CharacterMap() is used to define the font for the string.

Arguments: CharacterMap:
*Font

Char:
*Str

Return Value: Int

Remarks:

LCDStringHeight()

Function:

Arguments: CharacterMap:
*Font

Char:
*Str

Return Value: Int

Remarks:

LCDRefresh()

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: Shorts:

Mode

XPos

YPos

XSize

YSize

Return Value: Bool

Remarks:

LCDSendCmdString()

Function:

Arguments: Char:
*CmdString
*Result

Short:
ReturnCount

Long:
TimeOut

Return Value: Int

Remarks:

LCDSetupTablet()

Function:

Arguments:

Return Value: Void

Remarks:

LCDSendCommand()

Function:

Arguments: LCDWindowedCmdType:
*Cmd

Integer:
DataSize

Return Value: Bool

Remarks:

LCDSendGraphic()

Function: Used to write graphical data to the LCD Display. The data is written at the location specified by the combination of Dest and 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: Shorts:
 Dest
 Mode

LCDGraphicBitmap:
*BitmapData

Return Value: Bool

Remarks:

GetLCDSize()

Function:

Arguments:

Return Value: Long.

Remarks:

GetLCDStringSize()

Function:

Arguments: Char:
 *Str

Return Value: Long.

Remarks:

The TabletParameters class consists of a construct, which obtains tablet X and Y start and stop values, sizing, port enumeration, tablet model among other tablet parameters. Use the GetTabletParameters() and SetTabletParameters() to get these params from the WIN\SigPlus.ini file, and also to change any specific settings you may wish to.

For example:

TabletParameters *Params;*

Listed Class Variables (properties)

Signature Class

General Properties:

CaptureIF;
Variable Type:
CaptureSig*
Remarks:

StrokeList;
Variable Type:
SigStroke*
Remarks:

KillCapture;
Variable Type:
bool
Remarks:

CaptureActive;
Variable Type:
bool
Remarks:

CurrentTablet;
Variable Type:
TabletInterface*
Remarks:

PenState;
Variable Type:
Volatile
Enum
PenStates
Remarks:

SavedSamples;
Variable Type:
TabletSampleList
Remarks:

SamplesToSave;
Variable Type:
short
Remarks:

LastSample;
Variable Type:
TabletSample
Remarks:

SigWindow;

Variable Type:
SigWindowType
Remarks:

MaxPointDelta;
Variable Type:
int
Remarks:

BadPointCounter;
Variable Type:
int
Remarks:

Filter[16];
Variable Type:
TabletSample
Remarks:

NumFilterPoints;
Variable Type:
int
Remarks:

NextPoint;
Variable Type:
int
Remarks:

TimeStamp[512];
Variable Type:
char
Remarks:

DisplayTimeStampPos;
Variable Type:
SigPoint
Remarks:

DisplayTimeStampSize;
Variable Type:
short
Remarks:

DisplayTimeStamp;
Variable Type:
bool
Remarks:

ImageTimeStampPos;
Variable Type:
SigPoint
Remarks:

ImageTimeStampSize;

Variable Type:
short

Remarks:

ImageTimeStamp;

Variable Type:
bool

Remarks:

EncryptionMode;

Variable Type:
EncryptionModes

Remarks:

Three modes:

0 = Default, No Encryption

1 = 40-bit DES Encryption

2 = 128-bit SAFER (lower to 40-bit to be in compliance with
int'l export laws)

Annotate[512];

Variable Type:
char

Remarks:

DisplayAnnotatePos;

Variable Type:
SigPoint

Remarks:

DisplayAnnotateSize;

Variable Type:
short

Remarks:

DisplayAnnotate;

Variable Type:
bool

Remarks:

ImageAnnotatePos;

Variable Type:
SigPoint

Remarks:

ImageAnnotateSize;

Variable Type:
short

Remarks:

ImageAnnotate;

Variable Type:
bool

Remarks:

SaveSigInfo;
Variable Type:
bool
Remarks:

SigKeyData[MaxSigKeySize];
Variable Type:
unsigned char
Remarks:

SigKeyDataLen;
Variable Type:
short
Remarks:

AutoKeyHash[Md5HashSize];
Variable Type:
Unsigned char
Remarks:

AutoKeyContext;
Variable Type:
Xmd5Context
Remarks:

AutoKeyStarted;
Variable Type:
bool
Remarks:

HotSpots;
Variable Type:
HotSpotList
Remarks:

HotSpotPoints;
Variable Type:
SigStroke*
Remarks:

JustifyX;
Variable Type:
short
Remarks:

JustifyY;
Variable Type:
short
Remarks:

JustifyMode;
Variable Type:
JustifyModes
Remarks:

AutoJustifyGain;
Variable Type:
long
Remarks:

AutoJustifyX0;
Variable Type:
long
Remarks:

AutoJustifyY0;
Variable Type:
long
Remarks:

AutoJustifyXOff;
Variable Type:
long
Remarks:

AutoJustifyYOff;
Variable Type:
long
Remarks:

JustifyYExtent;
Variable Type:
double
Remarks:

ImageDrawBuff;
Variable Type:
unsigned char*
Remarks:

ImageDrawBuffXSize;
Variable Type:
int
Remarks:

ImageDrawBuffYSize;
Variable Type:
int
Remarks:

ImageMono;
Variable Type:
bool
Remarks:

AntiAliasLineScale;
Variable Type:
float
Remarks:

AntiAliasSpotSize;



Variable Type:
float
Remarks:

TabletInterface Class

General Properties

TabletParams;

Variable Type:

TabletParameters

Remarks:

ParameterVersion;
PortNumber;
BaudRate;
TabletMode;
TabletMode;
TabletX1;
TabletY1;
TabletX2;
TabletY2;
TabletLogicalXSize;
TabletLogicalYSize;
TabletResolution;
TabletRotation;
UsbMode;
UseMultiUsb;
EnableLogging;
TestIfConnected;
TabletIpAddress;
LCDType;
LCDXSize;
LCDYSize;
LCDXStart;
LCDYStart;
LCDXStop;
LCDYStop;
LCDWriteDelay;
LCDRetryCount;
tabletPortPath[128];

TabletOpen;

Variable Type:

bool

Remarks:

LogFile;

Variable Type:

FILE*

Remarks:

CircBuff;

Variable Type:

CircularBuffer*

Remarks:

PointBuff;

Variable Type:

PointBuffer*

Remarks:

NonPointBuff;
Variable Type:
PointBuffer*
Remarks:

SerialIF;
Variable Type:
SerialIoIF*

Remarks:

HidIF;
Variable Type:
HidIoIF*
Remarks:

UsbIF;
Variable Type:
UsbIoIF*
Remarks:

SockIF;
Variable Type:
SocketIoIF*
Remarks:

ProcIF;
Variable Type:
ProcessSerialData*
Remarks:

SerialNumber;
Variable Type:
unsigned long
Remarks:

ModelNumber;
Variable Type:
unsigned long
Remarks:

Serial and SerialHID thread support stuff

General Properties

Serial InputState;
Variable Type:
volatile
enum
SerialState
Remarks:

SerialStateReset;
Variable Type:
volatile
bool
Remarks:

KillProcessInputData;
Variable Type:
volatile
bool
Remarks:

ProcessInputDataRunning;
Variable Type:
volatile
bool
Remarks:

LCDInterface Class

General Properties:

LCDCaptureModeType ;

Variable Type:

typedefenum

LCDCaptureModes

Remarks:

Tab ;

Variable Type:

TabletInterface*

Remarks:

LCDCaptureMode ;

Variable Type:

LCDCaptureModeType

Remarks:

1 = Default, Auto-erase

2 = No-erase (must be set when performing interactive LCD program)

TabletParameters Class

General Properties

ParameterVersion;

Variable Type:

short

Remarks:

PortNumber;

Variable Type:

short

Remarks:

Sets port for pen data capture (serial, USB, HID USB)

BaudRate;

Variable Type:

long

Remarks:

TabletMode;

Variable Type:

short

Remarks:

TabletX1;

Variable Type:

short

Remarks:

TabletY1;

Variable Type:

short

Remarks:

TabletX2;

Variable Type:

short

Remarks:

TabletY2;

Variable Type:

short

Remarks:

TabletLogicalXSize;

Variable Type:

short

Remarks:

TabletLogicalYSize;

Variable Type:

short

Remarks:

TabletResolution;
Variable Type:
long
Remarks:

TabletRotation;
Variable Type:
int
Remarks:

UsbMode;
Variable Type:
int
Remarks:

UseMultiUsb;
Variable Type:
bool
Remarks:

EnableLogging;
Variable Type:
bool
Remarks:

TestIfConnected;
Variable Type:
bool
Remarks:

TabletIpAddress;
Variable Type:
unsigned long
Remarks:

LCDType;
Variable Type:
short
Remarks:

LCDXSize;
Variable Type:
short
Remarks:

LCDYSize;
Variable Type:
short
Remarks:

LCDXStart;
Variable Type:
short
Remarks:

LCDYStart;
Variable Type:
short
Remarks:

LCDXStop;
Variable Type:
short
Remarks:

LCDYStop;
Variable Type:
short
Remarks:

LCDWriteDelay;
Variable Type:
long
Remarks:

LCDRetryCount;
Variable Type:
unsigned int
Remarks:

tabletPortPath[128];
Variable Type:
char
Remarks:



Statement regarding patents, and warning against modification of Topaz products:

Topaz products are covered under US patents 6,307,955 and 5,120,908 and 5,122,623. Patent work is ongoing. Use of Topaz's products in accordance with our instructions, to the best of our knowledge, does not infringe any patents. However, be aware that there are many patents out there, and modification of our products, or use of our products for other than their intended purpose, could run afoul of these patents. It is the responsibility of Topaz customers to honor relevant patents. U. S. Patent Nos. 5,120,906; 5,195,133; 5,227,590; 5,297,202; 5,322,978; 5,544,255; 5,647,017; 5,818,955; 6,064,751; 6,091,835; 6,381,344; 6,539,363 and others are patents that users of Topaz products should consider if modifications are to be made to our products or if our products are to be used for other than their intended purpose. These patents may be obtained at <http://www.uspto.gov>. Only as examples, without first considering these patents, you should refrain from storing within the signature information (sig file or sig data):

1. A reason for signing or a statement of importance of the signature together with a set of measurements relating to the handwritten signature and either an indication of the signatory or means for comparing said measurements with a set of statistics of a genuine signature to obtain a similarity score.
2. A visual representation of a signature together with a document checksum, hash, or receipt, and the claimed identity of the signatory.
3. A set of measurements relating to biometric information together with a document checksum, hash, or receipt, and the claimed identity of the user.

The foregoing statement is not intended to be an interpretation of the patents or their application to any particular product or implementation.

It may be possible that Federal (FDA) and state digital signature regulations become violated if techniques in 1, 2, and 3 above are employed, so rest assured, Topaz software does not do any of them. Note the information below:

1. The FDA regulations state that "Signed electronic records shall contain information associated with the signing that clearly indicates all of the following ...printed name ... date and time ... The meaning ...". (see FDA guidelines section 11.50) The FDA regulations treat electronic records and electronic signatures as different entities (Section 11.70). Therefore, you should refrain from modifying the Topaz system on your own to store this document-related data in the signature file.

2. State regulations such as CA code of Regulations, Title 2, Division 7, Chapter 10 for example, require that the signature be bound to only the single message that is signed and not to any other message. Therefore you should refrain from modifying the Topaz system on your own to place a checksum in with the signature data and then adding other information in with the signature, and then encrypting them to a secret key. If you were to modify the Topaz system to do this, you would be binding the signature to multiple-messages which is prohibited by the state regulations.

3. State regulations require that the signature data be capable of verification by a forensic document examiner. (CA regulations paragraph 220003(b)(3)(B) for example). Some of the techniques in 1-3 above appear at odds with "lengthy process of handwriting analysis" required by state regulation (see state of CA FAQ section). With the Topaz system, you are assured of the most accurate forensic handwriting analysis results, since the Topaz .sig file is saving all of the original signature data.

In addition, there are potential security and refutability issues if too many things are crammed in with the signature. Leave the document data, reason for signing, receipt, and identity information in the document as SigPlus is designed to work, and as complies with FDA and state regulations. That way, the signature is bound to all of the data, not just to part of the data. It is very easy to create all kinds of standard image files of the signature using SigPlus software. Topaz signature & document security results from our patent pending methods of encrypting the .sig data directly to a document hash (not by putting the hash or receipt into the .sig file), and non-repudiation results from the proper use of (again patent pending) Topaz signature and document receipts. Please contact the factory if you have any additional questions or would like more information about your rights concerning patents as a customer or developer using Topaz products.

Important Notice:

This software or any or all additional documentation, guidelines, or examples do not constitute a warranty about the performance, security, or legal acceptability of SigPlus software or the SigPlus control in any specific use or implementation. To the extent that SigPlus or SigPlus are 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.