USER MANUAL

uSign™
API Reference Guide
V1.01

81084503-001-A
Dec 28, 2007

Revision History

Revision	Date	Description of Changes	By
A	12/28/2007	Initial Release	GS

This document contains proprietary information of ID TECH. Its receipt or possession does not convey any rights to reproduce or disclose its contents or to manufacture, use or sell anything it may describe. Reproduction, disclosure, or use without specific written authorization from ID TECH is strictly forbidden.

Software License Agreement

CAREFULLY READ ALL THE TERMS, CONDITIONS AND RESTRICTIONS OF THIS LICENSE AGREEMENT BEFORE USING OR INSTALLING THE SOFTWARE. YOUR USE OR INSTALLATION OF THE SOFTWARE PRESUMES YOUR AGREEMENT WITH AND ACCEPTANCE OF THE TERMS, CONDITIONS, AND RESTRICTIONS CONTAINED IN THIS AGREEMENT. IF YOU DO NOT AGREE WITH THESE TERMS, CONDITIONS, AND RESTRICTIONS, PROMPTLY RETURN THE SOFTWARE AND RELATED DOCUMENTATION TO – ID TECH Support, 10721 Walker Street, Cypress, CA 90630.

TERMS, CONDITIONS AND RESTRICTIONS

ID TECH, Incorporated (the "Licensor") owns and has the right to distribute the described software and documentation, collectively referred to as the "Software".

LICENSE: Licensor grants you (the "Licensee") the right to use the Software in conjunction with ID TECH products.

LICENSEE MAY NOT COPY, MODIFY OR TRANSFER THE SOFTWARE IN WHOLE OR IN PART EXCEPT AS EXPRESSLY PROVIDED IN THIS AGREEMENT. Licensee may not decompile, disassemble, or in any other manner attempt to reverse engineer the Software. Licensee shall not tamper with, bypass, or alter any security features of the software or attempt to do so.

TRANSFER: Licensee may not transfer the Software or license to the Software to another party without prior written authorization of the Licensor. If Licensee transfers the Software without authorization, all rights granted under this Agreement are automatically terminated.

COPYRIGHT: The Software is copyrighted. Licensee may not copy the Software except to archive the Software or to load the Software for execution purposes. All other copies of the Software are in violation of this Agreement.

TERM: This Agreement is in effect as long as Licensee continues the use of the Software. The Licensor also reserves the right to terminate this Agreement if Licensee fails to comply with any of the terms, conditions, or restrictions contained herein. Should Licensor terminate this Agreement due to Licensee's failure to comply, Licensee agrees to return the Software to Licensor. Receipt of returned Software by the Licensor shall mark the termination.

LIMITED WARRANTY: Licensor warrants to the Licensee that the disk(s) or other media on which the Software is recorded to be free from defects in material or workmanship under normal use. THE SOFTWARE IS PROVIDED AS IS WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Because of the diversity of conditions and PC hardware under which the Software may be used, Licensor does not warrant that the Software will meet Licensee specifications or that the operation of the Software will be uninterrupted or free of errors.

IN NO EVENT WILL LICENSOR BE LIABLE FOR ANY DAMAGES, INCLUDING ANY LOST PROFITS, LOST SAVINGS OR OTHER INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE SOFTWARE. Licensee's sole remedy in the event of a defect in material or workmanship is expressly limited to replacement of the Software disk(s) if applicable.

GOVERNING LAW: If any provision of this Agreement is found to be unlawful, void or unenforceable, that provision shall be removed from consideration under this Agreement and will not affect the enforceability of any of the remaining provisions. This Agreement shall be governed by the laws of the State of California and shall insure to the benefit of International Technologies & Systems Corporation (d/b/a ID TECH), its successors, or assigns.

ACKNOWLEDGMENT: LICENSEE ACKNOWLEDGES THAT HE HAS READ THIS AGREEMENT, UNDERSTANDS ALL OF ITS TERMS, CONDITIONS, AND RESTRICTIONS AND AGREES TO BE BOUND BY THEM. LICENSEE ALSO AGREES THAT THIS AGREEMENT SUPERSEDES ANY AND ALL, VERBAL AND WRITTEN, COMMUNICATIONS BETWEEN LICENSOR AND LICENSEE OR THEIR ASSIGNS RELATING TO THE SUBJECT MATTER OF THIS AGREEMENT. QUESTIONS REGARDING THIS AGREEMENT SHOULD BE ADDRESSED IN WRITING TO ID TECH, INCORPORATED, ATTENTION: CUSTOMER SUPPORT, AT THE ABOVE ADDRESS OR E-MAILED TO: support@idtechproducts.com

Information Provided

The information contained herein is provided to the user as a convenience. While every effort has been made to ensure accuracy, ID TECH is not responsible for damages that might occur because of errors or omissions, including any loss of profit or other commercial damage, nor for any infringements or patents or other rights of third parties that may result from its use. The specifications described herein were current at the time of publication, but are subject to change at any time without prior notice.

ID TECH is a registered trademark of International Technologies & Systems Corporation. uSign and Value through Innovation are trademarks of International Technologies & Systems Corporation.

CONTENTS

1.0 Introduction

The ID TECH's new product, uSign™, is an electronic signature capture device with a LCD and pressure sensitive screen. The uSign provides a real-time stylus trace when a user authors their signature. The DLL is into the Host application processor memory and allows access to all the functions needed by an application operating with the uSign. This document provides the information for using the API. A separate document is available for uSign operation and installation.

2.0 The DLL

The {name}.DLL file is fully contained, which means the Host machine does not need any other support files for its function. The design will allow operation on Terminals and PC platforms. The DLL zip folder is available free, on line at www.idtechproducts.com. Sample dll calls are provided to show how to use the API commands in single thread method.

3.0 API Functions

The API Functions are listed below. Each function call provides the operation and passes a parameter or parameters as listed. The returned result indicates the function result. 1 indicates a function success; other value indicates the function operation was not successful. See Appendix A for more information of return value.

3.1 uSign_ GetSDKVersion

Function: uSign_GetSDKVersion
Description: Get DLL version number.
Prototype: BYTE uSign_GetSDKVersion(char *sVersion, int sLength, int *rLength)
Parameters: sVersion The data of DLL version number
sLength The length of sVersion
rLength The length of get DLL version number
Return: Appendix A
Example: BYTE res = uSign_GetSDKVersion(Version,128, &length);

3.2 uSign_OpenPort

Function: uSign_OpenPort
Description: Open port for RS232 communication.
Prototype: BYTE uSign_OpenPort(int Comport, long Baud, char Parity, int Stopbit)
Parameters: Baud baud rate
Parity parity bit
Stopbit stop bit
ComPort Port for RS232 communication
Return: Appendix A
Example: BYTE res = uSign_OpenPort(1,9600, ‘E’, 1)

3.3 uSign_ClosePort

Function: uSign_ ClosePort
Description: Close port for RS232 communication and USB HID interface
Prototype: bool uSign_ClosePort()
Parameters: None
Return: Appendix A
Example: uSign_ClosePort()

3.4 uSign_SetBaud

Function: uSign_SetBaud
Description: Set up baud rate for RS232 communication.
Prototype: BYTE uSign_SetBaud(long Baud)
Parameters: Baud (1200, 2400, 4800, 9600, 14400, 19200, 38400, 57600 or 115200)
Return: Appendix A
Example: uSign_SetBaud(9600)

3.5 uSign_OpenUSBHID

Function: uSign_OpenUSBHID
Description: Open the USB HID interface
Prototype: BYTE uSign_OpenUSBHID( )
Parameters: None
Return: Appendix A
Example: uSign_OpenUSBHID( )

3.6 uSign_SetParity

Function: uSign_SetParity
Description: Set up parity for RS232 communication
Prototype: BYTE uSign_SetParity(char Parity)
Parameters: Parity N (None), O (Odd) , E (Even), M (Mark) or S(Space)
Return: Appendix A
Example: uSign_SetParity('E')

3.7 uSign_SetStopBit

Function: uSign_SetStopBit
Description: Set up number of stop bits for RS232 communication
Prototype: BYTE uSign_SetStopBit(int Stopbit)
Parameters: Stopbit 1 or 2
Return: Appendix A
Example uSign_SetStopBit(1)

3.8 uSign_GetFirmware

Function: uSign_GetFirmware
Description: Get firmware version
Prototype: uSign_GetFirmware(char *sFirmware, int sLength, int *rLength)

Parameters	sFirmware	The data of firmware version
	sLength	The length of sFirmware
	rLength	The length of getting formware version

Return: Appendix A
Example BYTE res = uSign_GetFirmware(Version, 128,&length)

3.9 uSign_GetSerialNum

Function: uSign_GetSerialNum
Description: Get Serial number
Prototype: BYTE uSign_GetSerialNum(char *sSerialNum,int sLength, int *rLength);
Parameters:

sSerialNum	The data of Serial number
sLength	The length of sSerialNum
rLength	The length of get Serial number

Return: Appendix A
Example: BYTE res = uSign_GetSerialNum(Version,128, &length)

3.10 uSign_GetSignFormat

Function: uSign_GetSignFormat
Description: Get Signature data
Prototype: BYTE uSign_GetSignFormat(BYTE s_Type, char *Sign_Data, int sLength, int *rLength)
Parameters:

s_Type	0x01	SIG format signature data
	0x02	CMP format signature data
	0x04	RAW format signature data
	0x10	BMP format signature data
Sign_Data	Get the signature data	
sLength	The length of Sign_Data	
rLength	The length of getting the signature data	

Return: Appendix A
Example: BYTE res = uSign_GetSignFormat(0x01, Sign_Data, 8192, &length)

3.11 uSign_SetSignFormat

Function: uSign_SetSignFormat
Description: Send signature to uSign
Prototype: BYTE uSign_SetSignFormat(BYTE s_Type, char *Sign_Data, int sLength)
Parameters:

s_Type	0x01/0x81	SIG format signature data,0x81 means to clear old signature data,while 0x01 not.
	0x02/0x82	CMP format signature data,0x82 means to clear old signature data,while 0x02 not.
	0x04/0x84	RAW format signature data,0x84 means to clear old signature data,while 0x04 not.
	0x10/0x90	BMP format signature data,0x90 means to clear old signature data, while 0x10 not.
Sign_Data	the signature data for sending to uSign	
sLength	The length of signature data	

Return: Appendix A
Example: BYTE res = uSign_SetSignFormat(0x01, Sign_Data, 128)

3.12 uSign_GetClipArea

Function:	uSign_GetClipArea	
Description:	Get current clip area the clip area range is (0,0) – (191,63)	
Prototype:	BYTE uSign_GetClipArea(int *left, int *top, int *right, int *bottom)	
Parameters	left	The left of clip area
	top	The top of clip area
	right	The right of clip area
	bottom	The bottom of clip area
Return:	Appendix A	
Example	uSign_GetClipArea(10,10,150,50)	

3.13 uSign_SetClipArea

Function:	uSign_SetClipArea	
Description:	Set new clip area the clip area range is (0,0) – (191,63)	
Prototype:	BYTE uSign_SetClipArea(int left, int top, int right, int bottom)	
Parameters:	left	The left of new clip area
	top	The top of new clip area
	right	The right of new clip area
	bottom	The bottom of new clip area
Return:	Appendix A	
Example:	uSign_SetClipArea(10,10,150,50)	

3.14 uSign_ClearData

Function:	uSign_ClearData
Description:	Clear buffered data and display
Prototype:	BYTE uSign_ClearData()
Parameters:	None
Return	Appendix A
Example:	uSign_ClearData()

3.15 uSign_StartBufCapture

Function:	uSign_StartBufCapture
Description:	Start capture without data out during script
Prototype:	BYTE uSign_StartBufCapture()
Parameters:	None
Return:	Appendix A
Example:	uSign_StartBufCapture()

3.16 uSign_StopBufCapture

Function:	uSign_StopBufCapture
Description:	Stop buffered capture without clearing buffered data
Prototype:	BYTE uSign_StopBufCapture(long *pCount)
Parameters:	pCount Buffered points count
Return:	Appendix A
Example:	uSign_StopBufCapture(&pCount)

3.17 uSign_GetSampleRate

Function:	uSign_GetSampleRate
Description:	Get current sample rate
Prototype:	BYTE uSign_GetSampleRate(int *rate)
Parameters:	rate Sample rate
Return:	Appendix A
Example:	uSign_GetSampleRate(&rate)

3.18 uSign_SetSampleRate

Function:	uSign_SetSampleRate
Description:	Get new sample rate
Prototype:	BYTE uSign_SetSampleRate(int rate)
Parameters:	rate Sample rate (0x09 to 0x22)
Return:	Appendix A
Example:	uSign_SetSampleRate(0x20)

3.19 uSign_LEDControl

Function:	uSign_LEDControl
Description:	Control the red led and green led
Prototype:	BYTE uSign_LEDControl(BYTE l_Mask, BYTE c_Mask)
Parameters:	l_Mask Defined as bxxxxxxGR,where 1 means select
	c_Mask Defined as bxxxxxxGR,where 1 means ON and 0 mean OFF
Return:	Appendix A
Example:	uSign_LEDContro(1,1)

3.20 uSign_TurnRedled

Function:	uSign_TurnRedled		
Description:	Turn red LED on or off		
Prototype:	BYTE uSign_TurnRedled(bool f_Led)		
Parameters:	f_Led	true	Turn red LED on
		false	Turn red LED off
Return:	Appendix A		
Example:	uSign_TurnRedled(true)		

3.21 uSign_GetScriptCount

Function:	uSign_GetScriptCount	
Description:	Get script points count	
Prototype:	BYTE uSign_GetScriptCount(long * pCount)	
Parameters:	pCount	The script points count
Return:	Appendix A	
Example:	uSign_GetScriptCount(&pCount)	

3.22 uSign_StopCapturing

Function:	uSign_StopCapturing
Description:	Stop capturing sign
Prototype:	BYTE uSign_StopCapturing()
Parameters:	None
Return:	Appendix A
Example:	uSign_StopCapturing()

3.23 uSign_StartCapturing

Function:	uSign_StartCapturing		
Description:	Start capturing sign using non-buffered mode		
Prototype:	BYTE uSign_StartCapturing(BYTE s_Type)		
Parameters:	s_Type	0x00	Start capturing use FBP format 0
		0x01	Start capturing use FBP format 1
Return:	Success no return value and other return value see Appendix A		
Example:	uSign_StartCapturing(BYTE s_Type)		

3.24 uSign_AddPointHandle

Function:	uSign_AddPointHandle
Description:	Register a call-back function for sending the sign data
Prototype:	BYTE uSign_AddPointHandle(PKEY_FUNC func,LPVOID pParam)
Parameters:	func The name of call-back function pParam The currently pointer
Return:	Appendix A
Example:	uSign_AddPointHandle(key_handle,this)

3.25 uSign_ClearScreen

Function:	uSign_ClearScreen
Description:	Clear display
Prototype:	BYTE uSign_ClearScreen()
Parameters:	None
Return	Appendix A
Example:	uSign_ClearScreen()

3.26 uSign_ClearCMPData

Function:	uSign_ClearCMPData
Description:	Reset CMP format mode data
Prototype:	BYTE uSign_ClearCMPData()
Parameters:	None
Return:	Appendix A
Example:	uSign_ClearCMPData()

3.27 uSign_SimulatePW120

Function:	uSign_SimulatePW120
Description:	Simulate PW120 command
Prototype:	BYTE uSign_SimulatePW120()
Parameters:	None
Return:	Appendix A
Example:	uSign_SimulatePW120()

3.28 uSign_ResetPointsCounter

Function:	uSign_ResetPointsCounter
Description:	Reset the points counter in using
Prototype:	BYTE uSign_ResetPointsCounter()
Parameters:	None
Return:	Appendix A
Example:	uSign_ResetPointsCounter()

3.29 uSign_InitDevice

Function: uSign_InitDevice
Description: Initialize uSign
Prototype: BYTE uSign_InitDevice ()
Parameters: None
Return: Appendix A
Example: uSign_InitDevice ()

3.30 uSign_GetPW120Version

Function: uSign_GetPW120Version
Description: Get PW120 version
Prototype: BYTE uSign_GetPW120Version(char *sVersion, int sLength, int *rLength)
Parameters: sVersion PW120 version
sLength The length of sVersion
rLength The length of PW120 version
Return: Appendix A
Example: uSign_GetPW120Version(sVersion,128,&length)

3.31 uSign_TestPW120Device

Function: uSign_TestPW120Device
Description: Test PW120 device.
Prototype: BYTE uSign_TestPW120Device()
Parameters: None
Return: Appendix A
Example: uSign_TestPW120Device()

3.32 uSign_SetCTSControl

Function: uSign_SetCTSControl
Description: Enable or disable CTS control for RS232 interface.
Prototype: BYTE uSign_SetCTSControl(bool f_CTS)
Parameters: f_CTS true Enabe
false Disable
Return: Appendix A
Example: uSign_SetCTSControl(true)

3.33 uSign_SetInterval

Function: uSign_SetInterval
Description: Set interval between two consecutive points.if exceeded,the display will disappear
Prototype: BYTE uSign_SetInterval(BYTE Time)
Parameters: Time The interval between two consecutive points
Return: Appendix A
Example: uSign_SetInterval(10)

3.34 uSign_CalibrateDevice

Function: uSign_CalibrateDevice
Description: Calibrate uSign
Prototype: BYTE uSign_CalibrateDevice()
Parameters: None
Return: Appendix A
Example: uSign_CalibrateDevice()

3.35 uSign_SetOffset

Function: uSign_SetOffset
Description: Set the offset position
Prototype: BYTE uSign_SetOffset(int x_Off, int y_Off)
Parameters: x_Off X axial
y_Off Y axial
Return: Appendix A
Example: uSign_SetOffset(10,10)

3.36 uSign_EnableOffset

Function: uSign_EnableOffset
Description: Enable of disable offset
Prototype: BYTE uSign_EnableOffset(bool f_Off)
Parameters: f_Off true Enable
false Disable
Return: Appendix A
Example: uSign_EnableOffset(true)

3.37 uSign_DrawLine

Function:	uSign_DrawLine	
Description:	Draw line defined by two points	
Prototype:	BYTE uSign_DrawLine(int x1, int y1, int x2, int y2)	
Parameters:	x1	The x axial of start point
	y1	The y axial of start point
	x2	The x axial of end point
	y2	The y axial of end point
Return:	Appendix A	
Example:	uSign_DrawLine(10,10,30,30)	

3.38 uSign_DrawRectangle

Function:	uSign_DrawRectangle		
Description:	Draw hollow or solid rectangle		
Prototype:	BYTE uSign_DrawRectangle(bool d_Type,int x1, int y1, int width, int height)		
Parameters:	d_Type	true	Draw solid rectangle
		false	Draw hollow rectangle
	x1	The x axial of start point	
	y1	The y axial of start point	
	width	The width of rectangle	
	height	The height of rectangle	
Return:	Appendix A		
Example:	uSign_DrawRectangle(false,10,10,40,40)		

3.39 uSign_DrawText

Function:	uSign_DrawText	
Description:	Write characters in uSign	
Prototype:	uSign_DrawText(int x1, int y1,int sLength, char *strData)	
Parameters:	x1	The x axial of start point
	y1	The y axial of start point
	sLength	The length of strData
	strData	Write characters
Return:	Appendix A	
Example:	uSign_DrawText(10, 10,24, strData)	

4.0 Return Values

Return Value	Description
0	FAIL
1	SUCCESS
50	NOT_SUPPORTED
99	PARAMETER_ERR
101	BUFFER_LACK
200	PORT_ OPENED
201	PORT_CLOSED

5.0 API Example

Target Device:
uSign
Describe:
Support uSign
Platform:
Microsoft Windows XP, Windows 2000, Vista
DLL Usage (Microsoft Visual C++ 6.0)
Add uSignKit.lib to Project->Settings->Link->Object/library modules and include the head file uSignKit.h , then call the DLL function directly.

Example for DLL call:

```
//include head file
#include "uSignKit.h"
//add Lib(uSignKit.lib):Add uSignKit.lib to Project->Settings->Link->Object/library
```

//Call DLL functions using single-thread method:

```
//uSign_GetSDKVersion
    char Version[128];
    int length = 0;
    BYTE res = uSign_GetFirmware(Version, 128,&length);
//uSign_OpenPort
       BYTE res = uSign_OpenPort(1,9600, ‘N’, 1);
//uSign_ClosePort
       bool Flag = uSign_ClosePort();
//uSign_SetBaud
       BYTE res = uSign_SetBaud(9600);
//uSign_OpenUSBHID
       int res = uSign_OpenUSBHID();
//uSign_SetParity
       BYTE res = uSign_SetParity('N');
//uSign_SetStopBit
       BYTE res = uSign_SetStopBit(1);
//uSign_GetFirmware
       char Version[128];
       int length = 0;
       BYTE res = uSign_GetFirmware(Version, 128,&length);
//uSign_GetSerialNum
       char SerialNum[128];
       int length = 0;
       BYTE res = uSign_GetSerialNum(SerialNum,128, &length);
//uSign_GetSignFormat
       char Sign_Data[8192];
       int length = 0;
       BYTE res = uSign_GetSignFormat(0x01, Sign_Data, 8192, &length);
```

// Call DLL functions using multi-threads methods:

```
// uSign_StartCapturing
     void __stdcall key_handle (int *buf,int rev,LPVOID pParam)
   {
          CUSignTestDlg* pthis = (CUSignTestDlg*)pParam;
          pthis->p_Count = 0;
          memset(pthis->p_Array,0,50);
          pthis->SendMessage(WM_SWITCH_UPDATE, 0, 0);
          for(int i = 0; i < rev;)
          {
               pthis->p_Array[pthis->p_Count].x = *buf++;
               pthis->p_Array[pthis->p_Count].y = *buf++;
               i++;
               i++;
               pthis->p_Count++;
          }
          pthis->ClearRect(false);
          pthis->SendMessage(WM_SWITCH_UPDATE, 0, 0);
   }
   static UINT ThreadProc( LPVOID pParam )
   {
          CUSignTestDlg* pthis = (CUSignTestDlg*)pParam;
          int res = uSign_StartCapturing(0);
          TRACE("Start capturing result:%d\n", res);
          return 0;
   }

  void CUSignTestDlg::OnStartcapturing()
  {
     // TODO: Add your control notification handler code here
     uSign_AddPointHandle(key_handle,this);
     AfxBeginThread(ThreadProc, this);
  }
```