

# Tablet USB&RF User Guide Interface

## Specification 1.4

Revised March, 2024

**Target Platform:** Windows

[1]

User Interface	<div>signInitialize</div> <div>Initialize the device application environment.</div>
Declare	int signInitialize ();
Parameters	/
Return Values	If the function succeeds, the return value is ERR_OK. If the function fails, the return value is ERR_DEVICE_OPENFAIL.
Remarks	

[2]

User Interface	<div>signClean</div> <div>Close the device application environment.</div>
Declare	int signClean ();
Parameters	/
Return Values	The return value is ERR_OK.
Remarks	

[3]

User Interface	<div>signGetDeviceStatus</div> <div>Find available tablet devices.</div>
Declare	int signGetDeviceStatus ();
Parameters	/
Return Values	If the function succeeds, the return value is ERR_OK. If the function fails, the return value is ERR_DEVICE_NOTFOUND.
Remarks	

[4]

User Interface	<div>signOpenDevice</div> <div>Open a usable tablet device.</div>
----------------	---



	<pre> unsigned long    pressure;           //pressure  char    vendor[32];           //verdor name  char    product[32];           //product name  unsigned long    version;           //driver version  char    serialnum[32];           //serial number  } TABLET_DEVICEINFO, *PTABLET_DEVICEINFO; </pre>

[7]

<b>User Interface</b>	<p><b>signRegisterDataCallBack</b></p> <p>Register a pen data callback function.</p>
<b>Declare</b>	int signRegisterDataCallBack (PACKDATAPROC lpPackDataProc);
<b>Parameters</b>	<p>lpPackDataProc</p> <p>[in]Pointer to the callback function.</p> <p>The <b>PACKDATAPROC</b> type defines a pointer to this callback function.</p>
<b>Return Values</b>	<p>If the function succeeds, the return value is <b>ERR_OK</b>.</p> <p>If the function fails, the return value is <b>ERR_INVALIDPARAM</b>.</p>
<b>Remarks</b>	<pre> PACKDATAPROC typedef int (CALLBACK * PACKDATAPROC) (PDATAPACKET pktObj);  //    PDATAPACKET    structure typedef struct tagDATAPACKET {     EventType        eventtype;           //event type 4     unsigned short    physical_key;        //physical key    2      unsigned short    virtual_key;        //virtual key2      KeyStatus        keystatus;           //key status 4      PenStatus        penstatus;           //pen status 4      unsigned short    x;                   //x    2      unsigned short    y;                   //y    2      unsigned short    pressure;           //pressure    2      short    wheel_direction;           //wheel 2 </pre>

	<pre>unsigned short  button;          //pen button      2  char tiltX;          //tilt X  char tiltY;          //tilt Y  }DATAPACKET, *PDATAPACKET;  enum EventType {     EventType_Pen = 1,     EventType_Key = 2,     EventType_Eraser = 3,     EventType_Wheel = 4,     EventType_ALL = 0xfe };  enum PenStatus {     PenStatus_Hover,     PenStatus_Down,     PenStatus_Move, PenStatus_Up,     PenStatus_Leave };  enum KeyStatus {     KeyStatus_Up,     KeyStatus_Down };</pre>
	<p>When the eventtype is EventType_Key, if physical_key is greater than 0, gets the physical key mask status bool</p> <pre>Pkey_01 = physical_key&amp;(0x1&lt;&lt;0); bool Pkey_02 = physical_key&amp;(0x1&lt;&lt;1); bool Pkey_03 = physical_key&amp;(0x1&lt;&lt;2); bool Pkey_04 = physical_key&amp;(0x1&lt;&lt;3); ...</pre> <p>if virtual _key is greater than 0, get the key number int</p> <pre>Vkey = virtual_key;</pre>

[8] [9]

<b>User Interface</b>	<p><b>signRegisterDevNotifyCallBack</b></p> <p>Register a status callback function.</p>
<b>Declare</b>	<pre>int signRegisterDevNotifyCallBack (DEVNOTIFYPROC lpDevNotifyProc);</pre>
<b>Parameters</b>	<p>lpDevNotifyProc</p> <p>[in] Pointer to the callback function.</p> <p>The <b>DEVNOTIFYPROC</b> type defines a pointer to this callback function.</p>
<b>Return Values</b>	<p>If the function succeeds, the return value is <b>ERR_OK</b>.</p> <p>If the function fails, the return value is <b>ERR_INVALIDPARAM</b>.</p>
<b>Remarks</b>	<p><b>DEVNOTIFYPROC</b></p> <pre>typedef          int  (CALLBACK * DEVNOTIFYPROC) (PENVPACKET pktObj);  typedef struct tagSTATUSPACKET {     INT          penAlive;     INT          penBattery;     INT          status; //0 DISCONNECTED 1 CONNECTED 2 SLEEP 3 AWAKE 4 BATTERY }STATUSPACKET,* PSTATUSPACKET;</pre>

[10]

<b>User Interface</b>	<p><b>signUnregisterDevNotifyCallBack</b></p> <p>Unregister the status callback function.</p>
<b>Declare</b>	<pre>void signUnregisterDevNotifyCallBack (long handler);</pre>
<b>Parameters</b>	<p>handler</p> <p>[in] Handle returned by the bleRegisterDataCallBack callback function.</p>
<b>User Interface</b>	<p><b>signUnregisterDataCallBack</b></p> <p>Unregister the pen data callback function.</p>
<b>Declare</b>	<pre>void signUnregisterDataCallBack (long handler);</pre>
<b>Parameters</b>	<p>handler</p> <p>[in] Handle returned by the bleRegisterDataCallBack callback function.</p>
<b>Return Values</b>	<p>/</p>
<b>Remarks</b>	

Return Values	/
Remarks	

[11]

User Interface	<p><code>signRegisterTouchCallBack</code></p> <p>Register a touch data callback function.</p>
Declare	<code>void signRegisterTouchCallBack (long handler);</code>
Parameters	<p><code>lpDevNotifyProc</code></p> <p>[in] Pointer to the callback function.</p> <p>The <code>TOUCHPROC</code> type defines a pointer to this callback function.</p>
Return Values	<p>If the function succeeds, the return value is <code>ERR_OK</code>.</p> <p>If the function fails, the return value is <code>ERR_INVALIDPARAM</code>.</p>
Remarks	<p><code>TOUCHPROC</code></p> <pre>typedef          int (_stdcall * TOUCHPROC) (TOUCHDATA td);  enum TouchStatus {     TouchStatus_Up,     TouchStatus_Down,     TouchStatus_Move };  typedef struct tagTOUCHDATA {     TouchStatus    status[10];     unsigned int   x[10];     unsigned int   y[10]; } TOUCHDATA, *PTOUCHDATA;</pre>

[12]

User Interface	<p><code>signUnregisterTouchCallBack</code></p> <p>Unregister the touch data callback function.</p>
Declare	<code>void signUnregisterTouchCallBack (long handler);</code>
Parameters	<p><code>handler</code></p> <p>[in] Handle returned by the <code>signRegisterTouchCallBack</code> callback function.</p>
Return Values	/
Remarks	

[13]

<b>User Interface</b>	<p><code>signChangeDeviceMode</code></p> <p>The function changes the running mode of the device.</p>
<b>Declare</b>	<code>int signChangeDeviceMode(int mode);</code>
<b>Parameters</b>	<p>mode</p> <p>[in] running mode</p>
<b>Return Values</b>	<p>If the function succeeds, the return value is <code>ERR_OK</code>.</p> <p>If the function fails, the return value is <code>ERR_ ERR_NOSUPPORTED</code> or <code>ERR_DEVICE_OPENFAIL</code>.</p>
<b>Remarks</b>	<pre>//run mode enum DeviceRunMode {     DeviceRunMode_Mouse = 1,        //system mouse     DeviceRunMode_Pen = 2,          //pen data     DeviceRunMode_MousePen = 3,     //system mouse and pen data     DeviceRunMode_StdPen = 4        //standard pen };</pre>

[14]

<b>User Interface</b>	<p><code>signGetScreenRect</code></p> <p>The function retrieves the dimensions of the bounding rectangle of the signaturescreen.</p>
<b>Declare</b>	<code>int signGetScreenRect(RECT* lpRect);</code>
<b>Parameters</b>	<p>lpRect</p> <p>[in] Pointer to a RECT structure that receives the screen coordinates.</p>
<b>Return Values</b>	<p>If the function succeeds, the return value is <code>ERR_OK</code>.</p> <p>If the function fails, the return value is <code>ERR_ ERR_NOSUPPORTED</code>.</p>
<b>Remarks</b>	

[15]

<b>User Interface</b>	<p><code>signMouseControl</code></p> <p>The function enables or disables the virtual mouse.</p>
<b>Declare</b>	<code>bool signMouseControl(bool bControlled);</code>
<b>Parameters</b>	<p>bControlled</p> <p>[in] If this parameter is true, the mouse is enabled. If the parameter is false, the mouse is disabled.</p>
<b>Return Values</b>	Returns the current mouse available status.

Remarks	
---------	--

[16]

User Interface	<b>signSetExtendDisplay</b>  The function changes the mouse to extend mode.
Declare	void signSetExtendDisplay(bool bExtendDisplay);
Parameters	bExtendDisplay [in] If this parameter is true, the mouse show on the extend screen. If the parameter is false, the mouse show on the primary screen.
Return Values	/
Remarks	

[17]

User Interface	<b>signRotateMode</b>  The function changes the running rotation angle of the device for pen.
Declare	int signRotateMode(int mode);
Parameters	mode [in] rotation smode
Return Values	If the function succeeds, the return value is <b>ERR_OK</b> . If the function fails, the return value is <b>ERR_NOSUPPORTED</b>
Remarks	Set the screen rotation angle. The default angle is 0 degrees. [parameter] angle: the values can be specified as 0, 90, 180, 270. Mode=0; //0 degrees Mode=1; //90 degrees Mode=2; //180 degrees Mode=3; //270 degrees

Constant Definition		
<b>ERR_OK</b>	0	Is ok.
<b>ERR_DEVICE_NOTFOUND</b>	-1	No available devices were found.
<b>ERR_DEVICE_OPENFAIL</b>	-2	The function fails.
<b>ERR_DEVICE_NOTCONNECTED</b>	-3	Not connected.
<b>ERR_INVALIDPARAM</b>	-101	Invalid parameter.
<b>ERR_NOSUPPORTED</b>	-102	This operation is not supported.



### Support Device List

Type	Pen	Physics Key	Virtual key	Display	Touch
CS01	yes	no	no	no	no
CS03	yes	no	no	no	no
EX07	yes	yes	no	no	no
EX08	yes	yes	no	no	no
UG05	yes	no	yes	yes	no
TabletA5	yes	no	yes	no	no
ET0A4KW	yes	yes	yes	no	no
101NF	yes	no	no	yes	no
101TF	yes	no	no	yes	yes
UD13	yes	yes	no	yes	no