80088506-001-A

User Manual

SmartPIN B100 API

Revision History

Version	Date	Description	By
A	11/07/2013	Initial Release	Candy H

Target Device:

SmartPINB100 and SecureHead

Describe:

Support SmartPINB100 and SecureHead device. For the paring functions, the two devices must be connected at the same time.

Platform:

Microsoft Windows XP, Windows 2000, Vista, Windows 7

DLL Usage (Microsoft Visual C++ 6.0)

Add PINB100_SecureHead_SDK.lib to Project->Settings->Link->Object/library modules and include the head file PINB100_SecureHead_SDK.h, then call the DLL function directly.

Command Summary

All commands supported are listed below.

Com_OpenDevice
Com_Close
Com_PairingKey
PIN_GetEncryptedPIN
PIN_Cancel
PIN_ControlAudio
PIN_GenerateTone
PIN_GetModelNumber
PIN_SetPINLen
PIN_GetPINLen
PIN_Reset
PIN_GetFirmware
PIN_GetSerialNumber
PIN_SetSerialNumber
PIN_GetFuncKey
PIN_BaudRate
PIN_Parity
PIN_StopBits
MSR_EnableSecureHead
MSR_DisableSecureHead
MSR_DefaultSetting
MSR_SetDecodeMethod
MSR_SetBaudrate
MSR_ReviewSetting
MSR_ReviewFirmware
MSR_ReviewSerialNumber
MSR_SetTerminatorSetting
MSR_SetPostambleSetting
MSR_SetPreambleSetting

MSR_SetPrefix
MSR_SetSuffix
MSR_SetTrackSelection
MSR_SetTrackSeparator
MSR_SetSentinel

Function description

SmartPIN B100 function API

Function:	Com_OpenDevice	
Description:	Open SecureHead(RS232) and SmartPIN B100(HID or RS232) devices.	
Format:	int Com_OpenDevice(char* pszSecureHead, char* pszPIN)	
Parameter:	pszSecureHead	The communicate parameter for SecureHead RS232 interface. The format is "COM%d/baud=%d/parity=%c/stop=%d/data=%d" Null: If there is no SeucreHead
	pszPIN	The communicate parameter for SmartPIN B100. Null:HID interface; Non-Null: RS232 interface.
Return:	Appendix A	
Example	char*SecureHead_Por = "COM1/baud=38400/parity=N/stop=1/data=8"; char *PIN_Port = ""; int res = Com_OpenDevice(SecureHead_Port,PIN_Port);	
Function:	Com_Close	
Description:	Close SecureHead and SmartPIN B100 devices.	
Format:	int Com_Close ()	
Parameter:	None	
Return:	Appendix A	
Example	Com_Close()	
Function:	Com_PairingKey	
Description:	Generate the Pairing Key.SecureHead and SmartPIN B100 devices must be connected.	
Format:	int Com_PairingKey()	
Parameter:	None	
Return:	Appendix A	
Example:	Com_PairingKey()	
Function:	PIN_GetEncryptedPIN	
Description:	Get Encrypted PIN.	

Format:	int PIN_GetEncryptedPIN(BYTE m_kType, char *m_tPAN, char *EncryptedPIN, int *Length)	
Parameter:	m_kType	The key type; SecureHead and SmartPIN B100 devices must be connected when m_kType > 3. 1: Get Encrypted PIN with DUKPT Key under Triple DES or Single DES mode using Plaintext PAN(Primary Account Number). 2: Get Encrypted PIN with DUKPT Key under AES mode using Plaintext PAN. 3: Get Encrypted PIN with MKSK using Plaintext PAN 4: Get Encrypted PIN with DUKPT Key under Triple DES or Single DES mode using Encrypted PAN. 5: Get Encrypted PIN with DUKPT Key under AES mode using Encrypted PAN. 6: Get Encrypted PIN with MKSK using Encrypted PAN.
	m_tPAN	16 bytes PAN(this will be used when m_kType == 1,2 or 3.
	EncryptedPIN	The Encrypted PIN block; m_kType == 1: 20 Ascii code KSN + 16 Ascii code Encrypted PIN block. m_kType == 2: 20 Ascii code KSN + 32 Ascii code Encrypted PIN block. m_kType == 3: 16 Ascii code Encrypted PIN block. m_kType == 4: 20 Ascii code KSN + 16 Ascii code Encrypted PIN block. m_kType == 5: 20 Ascii code KSN + 32 Ascii code Encrypted PIN block. m_kType == 6: 16 Ascii code Encrypted PIN block.
	Length	The length of EncryptedPIN string.
Return:	Appendix A	
Example:	PIN_GetEncryptedPIN(0x03,"1234567890123456",EncryptedPIN, &length); Please see demo code for more information.	
Function:	PIN_Cancel	
Description:	Cancel to input.Encrypted PIN and Function Key	
Format:	int PIN_Cancel()	
Parameter:	None	
Return:	Appendix A	
Example:	PIN_Cancel()	
Function:	PIN_ControlAudio	
Description:	Open/close beeper.	
Format:	int PIN_ControlAudio(bool m_bAudio)	

Parameter:	m_bAudio	True:open beeper False:close beeper
Return:	Appendix A	
Example	PIN_ControlAudio(true);	
Function:	PIN_GenerateTone	
Description:	Generate beep	
Format:	int PIN_GenerateTone(int m_iFrequency, int m_iDuration)	
Parameter	m_iFrequency	Frequence need be more than 1000Hz and less than 20000Hz.
	m_iDuration	Duration need be more than 16ms and less than 65535ms
Return:	Appendix A	
Example	PIN_GenerateTone (2000, 300)	
Function:	PIN_GetModelNumber	
Description:	Review device model number	
Format:	int PIN_GetModelNumber(char *m_sModelNumber, int *Length)	
Parameter:	m_sModelNumber	The model number string
	Length	The length of model number string
Return:	Appendix A	
Example:	PIN_GetModelNumber(ModelNumber, &length)	
Function:	PIN_SetPINLen	
Description:	Set PIN length	
Format:	int PIN_SetPINLen(BYTE m_bMinLen, BYTE m_bMaxLen)	
Parameter:	m_bMinLen	The minimum length 4~12 m_bMinLen need be same or less than m_bMaxLen.
	m_bMaxLen	The maximum length 4~12
Return	Appendix A	
Example:	PIN_SetPINLen(4,12);	
Function:	PIN_GetPINLen	
Description:	Review the minimum and minimum value for PIN.	
Format:	int PIN_GetPINLen(BYTE *m_bMinLen, BYTE *m_bMaxLen)	
Parameter:	m_bMinLen	The minimum length
	m_bMaxLen	The maximum length
Return:	Appendix A	
Example:	PIN_GetPINLen(&m_bMinLen, &m_bMaxLen)	
Function:	PIN_Reset	
Description:	Reset SmartPIN B100 device.	
Format:	int PIN_Reset()	
Parameter:	None	

Return:	Appendix A	
Example:	PIN_Reset()	
Function:	PIN_GetFirmware	
Description:	Get the firmware version.	
Format:	int PIN_GetFirmware(char *m_sFirmware, int *Length)	
Parameter:	m_sFirmware	The firmware version string.
	Length	The length of the firmware version string
Return:	Appendix A	
Example:	PIN_GetFirmware(m_sFirmware, &Length)	
Function:	PIN_GetSerialNumber	
Description:	Get Serial Number	
Format:	int PIN_GetSerialNumber(char *m_sSerialNum, int *Length)	
Parameter:	m_sSerialNum	The Serial Number string
	Length	The length of Serial Number string
Return:	Appendix A	
Example:	PIN_GetSerialNumber(m_sSerialNum, &Length)	
Function:	PIN_SetSerialNumber	
Description:	Set Serial Number to device	
Format:	int PIN_SetSerialNumber(char *m_sSerialNum)	
Parameter:	m_sSerialNum	The Serial Number string. This length must be 10.
Return:	Appendix A	
Example:	PIN_SetSerialNumber(m_sSerialNum)	
Function:	PIN_GetFuncKey	
Description:	Get function key. PIN_Cancel can cancel this API.	
Format:	int PIN_GetFuncKey(char *FuncKey, int *Length)	
Parameter:	FuncKey	The function key string.
	Length	The length of function key string.
Return:	Appendix A	
Example:	PIN_GetFuncKey(FuncKey, &Length)	
Function:	PIN_BaudRate	
Description:	Set or review the baud rate.	

Format:	int PIN_BaudRate(bool m_bHandle, long *baud);	
Parameter:	m_bHandle	True:set the baud rate to device. False:review the baud rate from device.
	baud	The baud rate. 2400 ,4800 ,9600,19200,38400 ,115200
Return:	Appendix A	
Example:	PIN_BaudRate(true, 38400);	
Function:	PIN_Parity	
Description:	Set or review Parity.	
Format:	int PIN_Parity(bool m_bHandle, char *parity)	
Parameter:	m_bHandle	True:set the Parity to device. False:review the Parity from device.
	parity	‘N’:None ‘O’:Odd ‘E’:Even
Return:	Appendix A	
Example:	PIN_Parity(ture,’E’)	
Function:	PIN_StopBits	
Description:	Set or review the stop bits.	
Format:	int PIN_StopBits(bool m_bHandle, int *stop)	
Parameter	m_bHandle	True:set the stop bits to device. False:Review the stop bits from device.
	stop	The stop bits,1 or 2.
Return:	Appendix A	
Example:	PIN_StopBits(true,1);	

SecureHead function API

Function:	MSR_EnableSecureHead	
Description:	Enable MSR Reading and register a call-back function,the function will be called when receiving SecureHead data.	
Format:	int MSR_EnableSecureHead(PMSR_FUNC func,LPVOID pParam)	
Parameter:	func	The name of call-back function. the format of PMSR_FUNC is typedef void (WINAPI *PMSR_FUNC)(unsigned char*,int, LPVOID);
	pParam	The currently pointer.
Return:	Appendix A	
Example:	MSR_EnableSecureHead(SecureHead_handle,pParam);	
Function:	MSR_DisableSecureHead	

Description:	Disable MSR Reading.	
Format:	int MSR_DisableSecureHead()	
Parameter:	None	
Return:	Appendix A	
Example:	MSR_DisableSecureHead()	
Function:	MSR_DefaultSetting	
Description:	Reset all setting to default.	
Format:	int MSR_DefaultSetting()	
Parameter:	None	
Return:	Appendix A	
Example:	MSR_DefaultSetting()	
Function:	MSR_SetDecodeMethod	
Description:	Set decode method,it supports five kinds of decode.	
Format:	int MSR_SetDecodeMethod(BYTE m_bDecode)	
Parameter:	m_bDecode	0x30: Raw Data Decoding in Both Directions, send out in ID TECH mode. 0x31: Decoding in Both Directions**.** If the encryption feature is enabled, the key management method used is DUKPT. 0x32: Moving stripe along head in direction of encoding. If the encryption feature is enabled, the key management method used is DUKPT. 0x33: Moving stripe along head against direction of encoding. If the encryption feature is enabled, the key management method used is DUKPT. Ox34: Raw Data Decoding in Both Directions, send out in other mode. If the encryption feature is enabled, the key management method used is fixed key.
Return:	Appendix A	
Example:	MSR_SetDecodeMethod(0x31);	
Function:	MSR_SetBaudrate	
Description:	Set the baud rate to SecureHead device.	
Format:	int MSR_SetBaudrate(long baud)	
Parameter:	baud	The baud rate. 4800,9600,19200,38400,57600 and 115200
Return:	Appendix A	
Example:	MSR_SetBaudrate (38400)	

Function:	MSR_ReviewSetting	
Description:	Review all setting from device.	
Format:	int MSR_ReviewSetting(BYTE *m_bSet, int *Length)	
Parameter:	m_bSet	The setting buffer. The format is <FuncSETBLOCK1>…<FuncSETBLOCKn> <FuncSETBLOCK> has the following format: <FuncID><Len><FuncData> Where: <FuncID> is one byte identifying the setting(s) for the function. <Len> is a one byte length count for the following function-setting block <FuncData> <FuncData> is the current setting for this function.
	Length	The length of setting.
Return:	Appendix A	
Example:	MSR_ReviewSetting(m_bSet, &Length)	
Function:	MSR_ReviewFirmware	
Description:	Review the firmware version from device.	
Format:	int MSR_ReviewFirmware(char *m_sFirm, int *Length)	
Parameter:	m_sFirm	The firmware version string.
	Length	The length of firmware version string.
Return:	Appendix A	
Example:	MSR_ReviewFirmware(m_sFirm, &Length)	
Function:	MSR_ReviewSerialNumber	
Description:	Review Serial Number from device.	
Format:	int MSR_ReviewSerialNumber(char *m_sSerial, int *Length)	
Parameter:	m_sSerial	The Serial Number sting.
	Length	The length of Serial Number sting
Return:	Appendix A	
Example:	MSR_ReviewSerialNumber(m_sSerial, &Length)	
Function:	MSR_SetTerminatorSetting	
Description:	Set terminator setting to device. Terminator characters are used to end a string of data in some applications.	
Format:	int MSR_SetTerminatorSetting(BYTE m_bTerm)	
Parameter	m_bTerm	Any one character, 00h is none; default is **CR** (0Dh)

Return:	Appendix A	
Example:	MSR_SetTerminatorSetting(0x0D);	
Function:	MSR_SetPostambleSetting	
Description:	Set postamble characters to device. The characters can be added to the end of the data string, after any terminator characters.	
Format:	int MSR_SetPostambleSetting(char *m_sPost)	
Parameter	m_sPost	The postamble string,the maximum length is 0x0F.
Return:	Appendix A	
Example:	MSR_SetPostambleSetting(m_sPost)	
Function:	MSR_SetPreambleSetting	
Description:	Set preamble characters to device.the characters can be added to the beginning of a string of data. These can be special characters for identifying a specific reading station, to format a message header expected by the receiving host, or any other character string. Up to fifteen ASCII characters can be defined.	
Format:	int WINAPI MSR_SetPreambleSetting(char *m_sPream)	
Parameter	m_sPream	The Preamble string,the maximum length is 0x0F.
Return:	Appendix A	
Example:	MSR_SetPreambleSetting(m_sPream)	
Function:	MSR_SetPrefix	
Description:	Set Prefix characters to device.the characters can be added to the beginning of a track data. These can be special characters to identify the specific track to the receiving host, or any other character string. Up to six ASCII characters can be defined.	
Format:	int MSR_SetPrefix(BYTE m_bTrack, char *m_sPrefix)	
Parameter	m_bTrack	0x01 for track 1; 0x02 for track 2 and 0x03 for track 3
	m_sPrefix	The prefix string,the maximum length is 6.
Return:	Appendix A	
Example:	MSR_SetPrefix(0x01, m_sPrefix)	
Function:	MSR_SetSuffix	
Description:	Set Suffix characters to device.the characters can be added to the end of track data. These can be special characters to identify the specific track to the receiving host, or any other character string. Up to six ASCII characters can be defined.	

Format:	int MSR_SetSuffix(BYTE m_bTrack, char *m_sSuffix)	
Parameter	m_bTrack	0x01 for track 1; 0x02 for track 2 and 0x03 for track 3
	m_sSuffix	The suffix string,the maximum length is 6.
Return:	Appendix A	
Example:	MSR_SetSuffix (0x01, m_sSuffix)	
Function:	MSR_SetTrackSelection	
Description:	Set the track selection	
Format:	int MSR_SetTrackSelection(BYTE m_bTrack)	
Parameter	m_bTrack	**0x00: Any Track** 0x01: Require Track 1 Only 0x02: Require Track 2 Only 0x03: Require Track 1 & Track 2 0x04: Require Track 3 Only 0x05: Require Track 1 & Track 3 0x06: Require Track 2 & Track 3 0x07: Require All Three Tracks 0x08: Any Track 1 & 2 0x09:Any Track 2 & 3
Return:	Appendix A	
Example:	MSR_SetTrackSelection(0x00)	
Function:	MSR_SetTrackSeparator	
Description:	Set track separator character, the character to be used to separate data decoded by a multiple-track reader.	
Format:	int MSR_SetTrackSeparator(BYTE m_bSeparator)	
Parameter	m_bSeparator	One ASCII Character. The default value is **CR,** 0h means no track separator.
Return:	Appendix A	
Example:	MSR_SetTrackSeparator(0);	
Function:	MSR_SetSentinel	
Description:	Set Start/End Sentinel and Track 2 Account Number Only.	
Format:	int MSR_SetSentinel(BYTE m_bSentinel)	
Parameter	m_bSentinel	0x00: Don't send start/end sentinel and send all data on Track 2 0x01: **Send start/end sentinel and send all data on Track 2** 0x02: Don't send start/end sentinel and send account # on Track 2

		0x03: Send start/end sentinel and send account number on Track 2
Return:	Appendix A	
Example:	MSR_SetSentinel(0x01)	

Example for DLL call:

```
//include head file
#include "PINB100_SecureHead_SDK.h"
//add Lib(PINB100_SecureHead_SDK.lib):
Add PINB100_SecureHead_SDK.lib to Project->Settings->Link->Object/library
//Call DLL functions using single-thread method:
//Com_OpenDevice
    //Open SecureHead (RS232) and SmarPIN B100(HID)
        char *SecureHead_Port = "COM1/baud=38400/parity=N/stop=1/data=8";
        char *PIN_Port = "";
        int res = Com_OpenDevice(SecureHead_Port,PIN_Port);
    // Open SecureHead (RS232) and SmarPIN B100(RS232)
        char *SecureHead_Port = "COM1/baud=38400/parity=N/stop=1/data=8";
        char *PIN_Port = " COM2/baud=38400/parity=N/stop=1/data=8";
        int res = Com_OpenDevice(SecureHead_Port,PIN_Port);
//Com_Close
    int res = Com_Close();
//Com_PairingKey
    int res = Com_PairingKey();
//PIN_Cancel
    res   = PIN_Cancel();
//PIN_ControlAudio
    res   = PIN_ControlAudio(true)
//PIN_GenerateTone
    res   =   PIN_GenerateTone(2000,300);
//PIN_GetModelNumber
    m_strModelNum = "";
    char ModelNum[128];
    int len = 0;
    int res = PIN_GetModelNumber(ModelNum, &len);
    if(res == 1)
    {
        for(int i = 0; i < len; i++)
            m_strModelNum += ModelNum[i];
    }
    UpdateData(false);
//PIN_SetPINLen
```

```
        res =PIN_SetPINLen(4, 12);
//PIN_GetPINLen
        res = PIN_GetPINLen (&m_bMinLen, &m_bMaxLen)
//PIN_Reset
        res    =    PIN_Reset();
//PIN_GetFirmware
        m_strModelNum = "";
        char Firmware[128];
        int len = 0;
        int res = PIN_GetFirmware(Firmware, &len);
        if(res == 1)
        {
              for(int i = 0; i < len; i++)
                    m_strModelNum += Firmware[i];
        }
        UpdateData(false);
//PIN_GetSerialNumber
        m_strModelNum = "";
        char SerialNum[128];
        int len = 0;
        int res = PIN_GetSerialNumber(SerialNum, &len);
        if(res == 1)
        {
              for(int i = 0; i < len; i++)
                    m_strModelNum += SerialNum[i];
        }
        UpdateData(false);
//PIN_SetSerialNumber
        res = PIN_SetSerialNumber("IDTECH2012");
//PIN_BaudRate
        long baud = 38400;
        i res = PIN_BaudRate(false,&baud);
//PIN_Parity
        res = PIN_Parity(true,'E');
//PIN_StopBits
        res = PIN_StopBits(true,1);
//MSR_DisableSecureHead
        res = MSR_DisableSecureHead();
```

// Call DLL functions using multi-threads methods:

```
//PIN_GetEncryptedPIN
        static UINT ThreadProc( LPVOID pParam )
    {
        CString temp;
```

```
    CPINB100_SecureHead_DemoDlg* pthis = (CPINB100_SecureHead_DemoDlg*)pParam;

    char EncryptedPIN[256];
    int length = 0;
    int res = 0;

    res = PIN_GetEncryptedPIN(0x03,"1234567890123456",EncryptedPIN, &length);
    if(res == 1)
    {
        pthis->m_strPIN = "";
        for(int j = 0; j < length; j++)
        {
            pthis->m_strPIN += EncryptedPIN[j];
        }

    }
    else
    {
        pthis->m_strPIN = "";
        CString str;
        str.Format("res = %d",res);
        pthis->m_strPIN = str;

    }

    pthis->SendMessage(WM_SWITCH_UPDATE, 0, 0);

    if(res == 108)
        pthis->MessageBox("KEYS_NOT_LOADED");
    return 0;
}
void CPINB100_SecureHead_DemoDlg::OnButtonPin()
{
    // TODO: Add your control notification handler code here
    AfxBeginThread(ThreadProc, this);
}
//PIN_GetFuncKey
    static UINT ThreadProc_FuncKey( LPVOID pParam )
{
    CString temp;
    CPINB100_SecureHead_DemoDlg* pthis = (CPINB100_SecureHead_DemoDlg*)pParam;

    char FuncKey[64];
    int length = 0;
```

```
    int res = 0;

    res = PIN_GetFuncKey(FuncKey, &length);
    if(res == 1)
    {
        pthis->m_strPIN = "";
        for(int j = 0; j < length; j++)
        {
            pthis->m_strPIN += FuncKey[j];
        }

    }
    else
    {
        pthis->m_strPIN = "";
        CString str;
        str.Format("res = %d",res);
        pthis->m_strPIN = str;

    }

    pthis->SendMessage(WM_SWITCH_UPDATE, 0, 0);

    return 0;
}

void CPINB100_SecureHead_DemoDlg::OnButtonFunckey()
{
    AfxBeginThread(ThreadProc_FuncKey, this);

}
// MSR_EnableSecureHead
void __stdcall SecureHead_handle (unsigned char *buf, int rev, LPVOID pParam)
{
    CPINB100_SecureHead_DemoDlg* pthis = (CPINB100_SecureHead_DemoDlg*)pParam;

    pthis->m_msrData = "";
    pthis->SendMessage(WM_SWITCH_UPDATE, 0, 0);
    CString str,temp;
    for(int i = 0; i < rev; i++)
    {
        temp.Format("%02X ",buf[i]);
        str += temp;
```

```
        }
        pthis->m_msrData = str;
        pthis->SendMessage(WM_SWITCH_UPDATE, 0, 0);

}

static UINT ThreadProc_MSR( LPVOID pParam )
{

        CPINB100_SecureHead_DemoDlg*                pthis                =
(CPINB100_SecureHead_DemoDlg*)pParam;
        int res = 0;

        MSR_EnableSecureHead(SecureHead_handle,pParam);


        return 0;
}

void CPINB100_SecureHead_DemoDlg::OnButtonEnable()
{
        // TODO: Add your control notification handler code here
        AfxBeginThread(ThreadProc_MSR, this);
}
```

Appendix A:

Return Value

Return Value	Description
0	FAIL
1	SUCCESS
50	NOT_SUPPORTED
81	NO_CARD_ACOUNT
82	DECRYPTDATA_FAIL
83	OPERATION_ERROR
84	RANDOMDATA_NOT_MATCH
85	SECURELEVEL_NOT_MATCH
86	EEPROM_ERROR
99	PARAMETER_ERR
100	COMMAND_UNSUPPORTED
101	INVALID_COMMAND
102	COMMAND_PROCESS
103	TIME_OUT
104	NO_DATA_AVAILABLE
105	ACTION_CANCELED
106	ACTION_ABORTED
107	WRONG_KEY_TYPE
108	KEYS_NOT_LOADED
109	KEY_EXIST
110	NO_BDK_PAIRING
111	NO_PAIRING_KEY
112	PAN_ERROR
113	PAIRING_FAIL
114	OTHER_PAIRING_ERROR
115	KEY_HAS_LOAD
116	CHIP_NOT_CONNECT
117	CHIP_ERROR
118	CONFIG_ERROR
119	NO_SERIAL_NUMBER
120	DEVICE_SUSPEND
121	PIN_DUKPT_STOP
200	PORT_OPENED
201	PORT_CLOSED
202	PIN_MODEL
203	SECUREHEAD_MODEL
206	FunKey_MODEL