USER MANUAL

SmartPIN K100/ C100/ B100
API User Manual

80111502-001-51
09-01-2011

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 ID TECH, Incorporated, 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. The software and documentation provided were current at the time of publication; however, they are subject to change at any time without prior notice.

Proprietary & Trademark Statements

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.

SmartPIN K100/ C100/ B100 API Reference Manual

Revision History

Rev.	Date	Description of Changes	By
50	07/01/2011	Initial Draft	JW
51	09/01/2011	Revised and added function examples	JW

SmartPIN K100/ C100/ B100 API Reference Manual

Table of Contents

1.0 Introduction

The SmartPIN K100/ C100/ B100 API is to be used by the host applications to access the PIN Pad functions. This document provides information for using the API, which supports both USB and RS232 interfaces SmartPIN K100/ C100/ B100 device. The operation and parameters for each API functions are listed in this document.

1.1 Availability

Sample dll calls are provided to show how to use the API commands in single thread method and in a multi-threaded method.

1.2 Usage

This API can be use in any Windows based development environment, such as Visual Studio, Delphi etc.

Put the header files, library files and DLL required for API into your setup program folder.

To successfully use all the functions that DLL includes, it is required to open the COM port using function COM_OpenPort, or open the USB port using function com_OpenUSBHID.

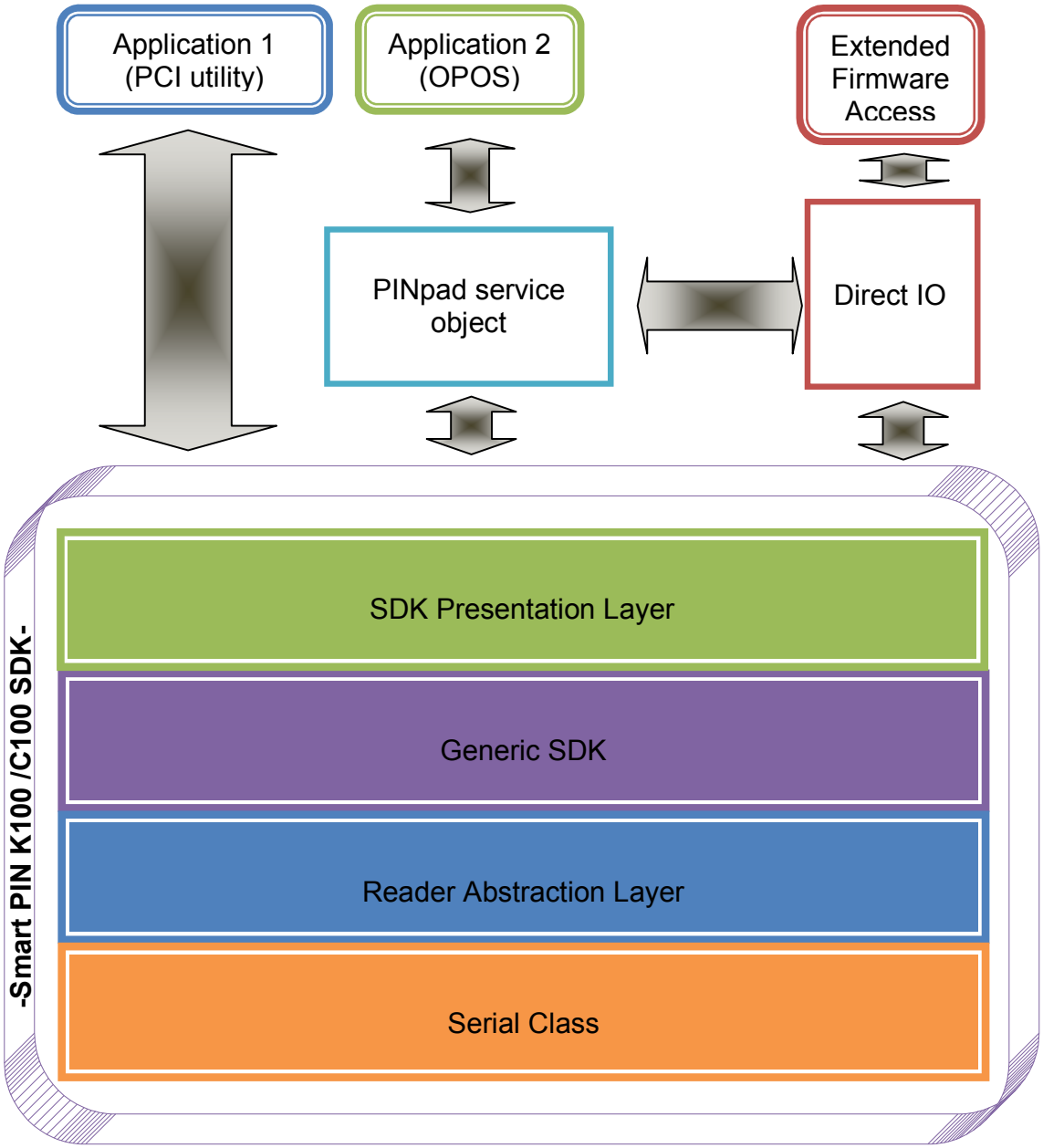
Function Com_OpenPort and function Com_ClosePort must be called again after changing any communication parameters (Baud rate, Stop bit, Parity, Data bit) and the updated parameters must be used to open the COM port.

2.0 Important Notes

- For functions with required preconditions, please ensure the preconditions are satisfied before making the function calls. Failure to meet the precondition might result in unexpected errors.
- Do not rely on observed or incidental behavior. Incidental behavior is behavior observed by experiment or in practice, and is not guaranteed by specification.
- Do not treat a null as an object or parameter. Assume nulls are being used unless specifically required in the API specifications.
- The application must assume that the function post-conditions are met upon the return of function calls.
- In function calls, do not modify a mutable object when it's being passed as a function parameter.
- Thread safety must be taken into consideration when programming a multi-threaded application.

3.0 SmartPIN K100/ C100/ B100 SDK Levels

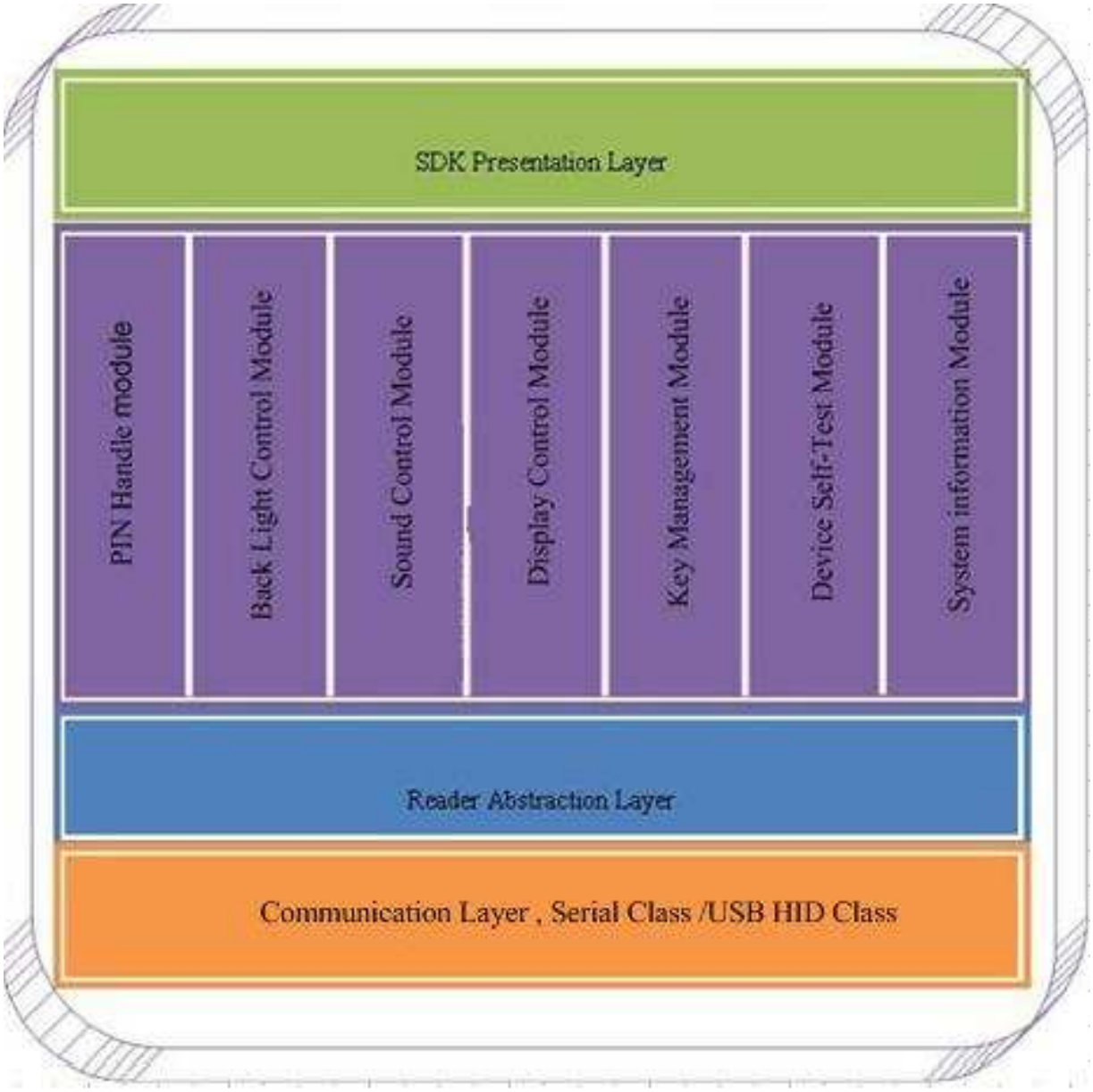
SmartPIN K100/ C100/ B100 General Systems Architecture



General System Architecture – SmartPIN K100/ C100/ B100 SDK

SDK Layer	Responsibility	Description
Serial Class/USB HID Class	- Facilitate the low-level communication interface between RS232 or USB and the SmartPIN device.	This Class handles the Serial/ USB HID communication between the host side and device. This is capable of handling RS232 or USB type devices.
Reader Abstraction Layer	- Processing SmartPIN device specific byte level data for commands. - Raw data processing to/from SmartPIN device and first level data verification and validation.	- Data encoding / decoding with the valid communication packet structure - Make Direct IO calls independent on data processing mechanism and provide raw data to/from SmartPIN device.
Generic SDK	- Consists with 8 modules for key controls on SmartPIN device	- Abstracts the 7 modules; 1. Serial Communication 2. Sound control 3. Display control 4. Backlight control 5. PIN control 6. Device self-test control 7. Key management control 8. General information
SDK Presentation Layer	- Export SmartPIN K100/ C100/ B100 Native SDK API functions	- Exports the SmartPIN K100/ C100/ B100 Native SDK API functions

4.0 SmartPIN K100/C100 SDK Systems Architecture



System Architecture –SmartPIN K100/C100 SDK

SmartPIN K100/ C100/ B100 API Reference Manual

The SmartPIN K100/ C100/ B100 SDK is consisted of seven modules: PIN Handle Module, Backlight Control Module, Sound Control Module, Display Control Module, Key Management Module, Device Self-test Module, and System Information Module. The features of SmartPIN K100/ C100/ B100 devices are grouped in those seven modules. The modules need to be initialized before the function can be used. Here is a list of the modules and the functions included.

Module Name	**Function Name**
Communication Control	Com_OpenPort
	Com_ClosePort
	Com_SetBaud
	Com_GetBaud
	Com_SetParity
	Com_GetParity
	Com_SetStopBit
	Com_GetStopBit
	Com_OpenUSBHID
Sound_Control	Sound_Control
	Sound_Tone
Display Control	Display_LoadPrompt
	Display_SavePrompt
	Display_Clear
BackLight Control	BackLight_SetControl
	BackLight_ReviewControl
	BackLight_SetTimer
	BackLight_ReviewTimer
PIN Control	PIN_GetPINBlock
	PIN_GetFuncKey
	PIN_CancelEnter
	PIN_SetPINLength
	PIN_GetPINLength

KEY Management Control	KEY_GetRemovalRecord
	KEY_LoadMACKEY
	KEY_EncryptData
System Information	Sys_GetVersion
	Sys_GetModel
	Sys_Reset
	Sys_LoadTime
	Sys_GetRealTime
	GetDllVersion

The details for each function can be found in the function description section.

5.0 SmartPIN K100/C100/ B100 SDK Error Code Description

SmartPIN K100/ C100/ B100 SDK return error codes are listed below:

Error Code	Error Name	Error Description
0	FAIL	Fail
1	SUCCESS	Success
50	ERR_NOT_SUPPORTED	Device not support this function
99	ERR_PARAMETER	Input parameter is not supported by device
100	ERR_UNKNOWN	Unknown error
101	ERR_INVALID_COMMAND	Protocol is right, but task ID is invalid
102	ERR_COMMAND	Protocol is wrong
103	ERR_TIME_OUT	Timeout for "Get Fun key" & "Get Encrypted PIN" & "Get Plaintext PIN",
104	ERR_UNIT_SUSPENDED	Unit is suspended
105	ERR_CANCELED	Exit from Get PIN mode or Get plaintext mode by cancel command
106	ERR_ABORTED	Exit from Get PIN mode or Get plaintext mode by press ”cancel” key
107	ERR_KEY_TYPE	Key type was not supported by device
108	ERR_NO_KEY	No DUKPT key or MKSK key
109	ERR_NO_PROMPT	No prompt during get numeric key to device
110	ERR_CONFIG_FAIL	Save or Config Failed / Or Read Config Error
111	ERR_BOOTLOAD_FAIL	Bootloader Block Number update error
112	ERR_CHECKVALUE_LOADED	Device was working under loading check value mode
113	ERR_LOADING_KEY_PROCESS	Device was working under loading key mode

114	ERR_SELF_TEST_PROCESS	Device was working under self test mode
115	ERR_MAXQ_DISCONNECTTED	Security Chip is not connect,
116	ERR_UNIT_DEACTIVE	Security Chip and MCU were both deactivation
117	ERR_MAXQ_DEACTIVE	Security Chip was d deactivation
118	ERR_MCU_DEACTIVE	MCU was d deactivation
119	ERR_NO_FIRMWARE_KEY	Firmware Key was not existed in the device
121	ERR_MAXQ_SUSPEND	Security Chip is suspend
122	ERR_DUKPT_KEY_NO_USED	DUKPT is stop (21 bit 1).
125	ERR_FULL_LOG	Fix/Removal records are more than 20.
126	ERR_TIME_LOADED	Device time has been loaded to the device
127	ERR_ST_DEACTIVE	Security Chip are all deactivation
128	ERR_ST_AND_MAXQ_DEACTIVE	MCU and Security Chip are all deactivation
200	PORT_OPENED	Communication port was opened
201	PORT_CLOSED	Communication port was closed
202	ERR_DEVICE_BUSY	Device was working under get pin mode or get plaintext mode
203	CLEAR_TEXT_MODEL	Plain Text mode
204	ERR_NO_FILE	Software was not found the special file

6.0 Getting Started with SmartPIN K100/C100/ B100 SDK

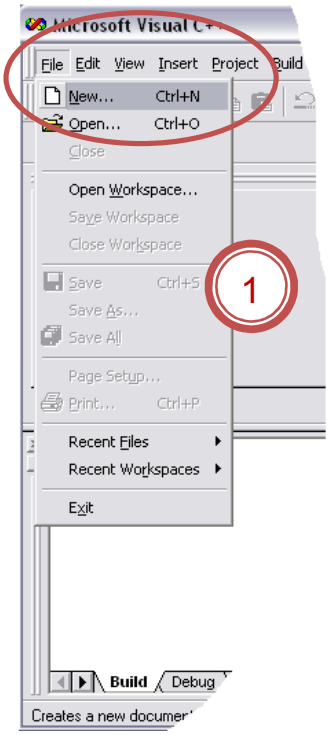
Few examples with source code samples will be covered here for application developer's reference. The documentation will illustrate start on application development with SmartPIN SDK using two programming languages; Visual C++ 6 and Visual Basic 6. SDK will provide an 'Include' package, which consists with the required dynamic link library files (dll), library files (.lib), and header files (.h).

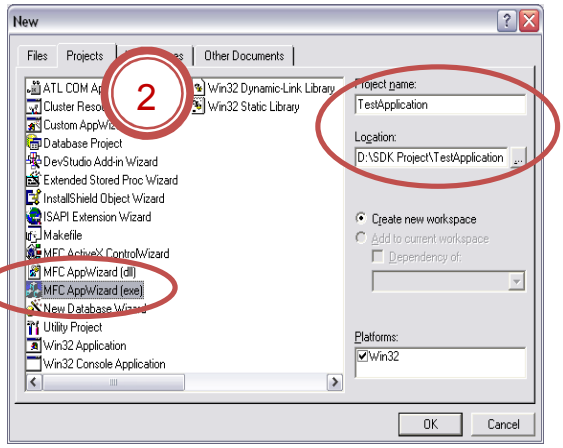
6.1 Sample Application development using VC++ 6

Create a Test Application for SmartPIN K100/ C100/ B100 SDK to retrieve Device Firmware Version.

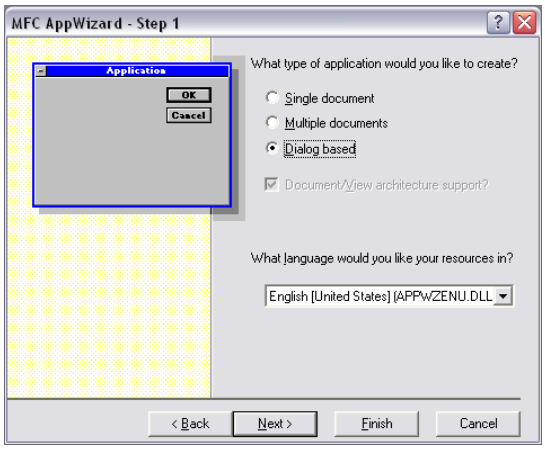
Please follow the below steps:

1. Put "pinDll.dll","Pin.h" and "PinDll.lib" into develop folder.
2. Run Microsoft Visual C++ 6 and create a dialog based MFC AppWizard (exe) using File → New → Projects (Tab). Give a suitable name (TestApplication) for the project and the location. Moreover, click next button while customizing the dialog application (follow the steps given).

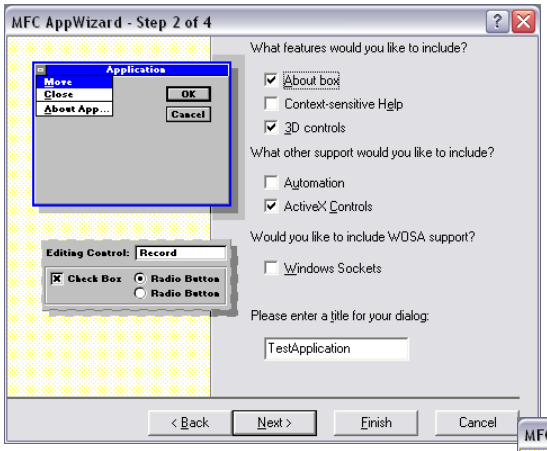




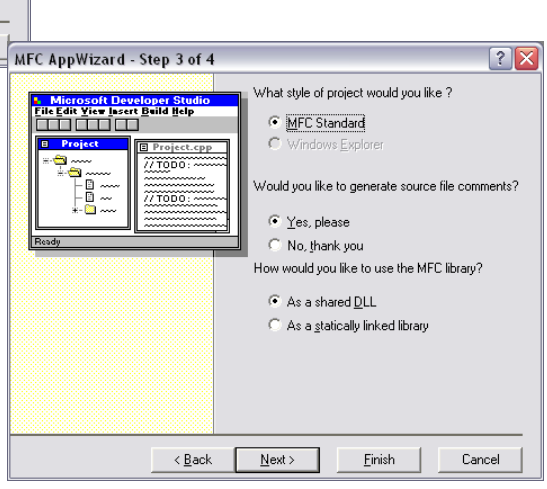
3

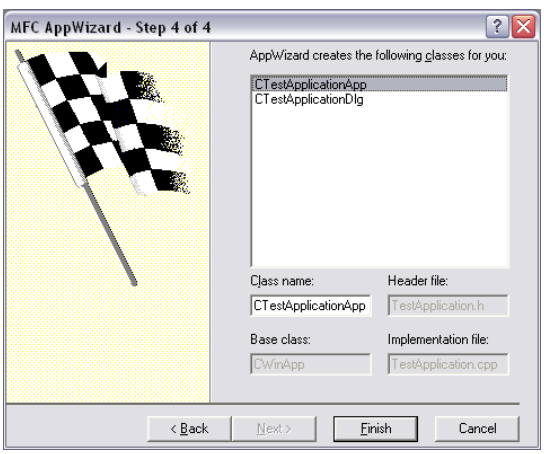


4

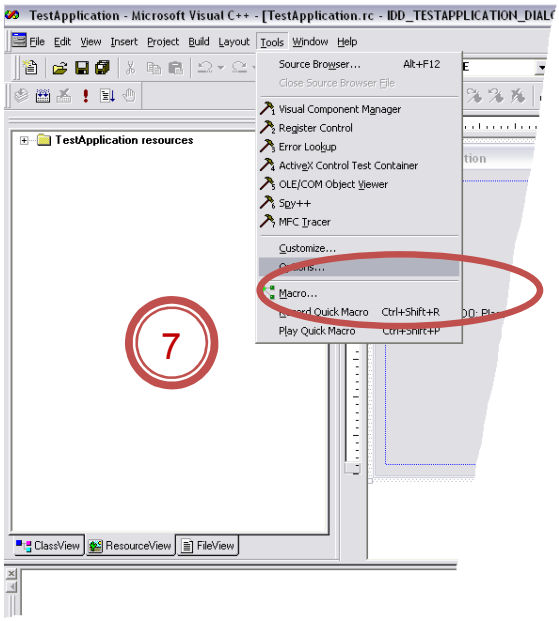


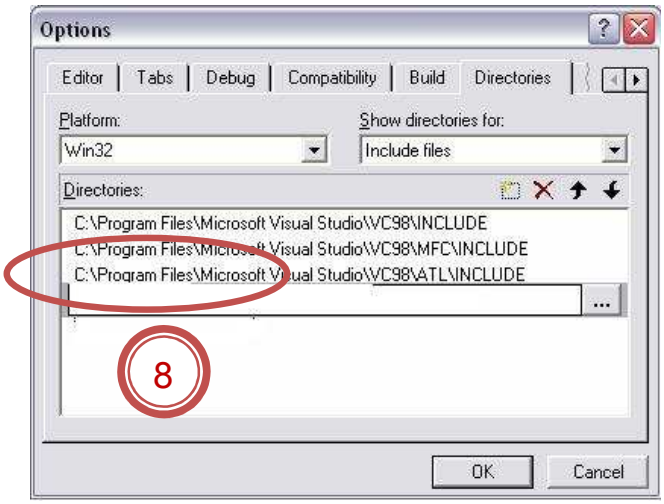
5



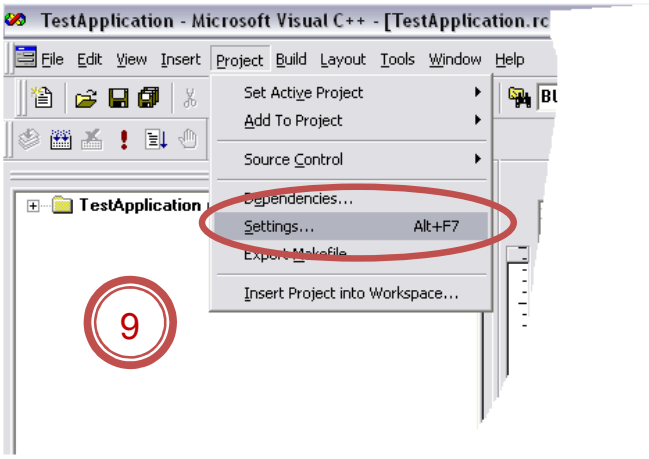


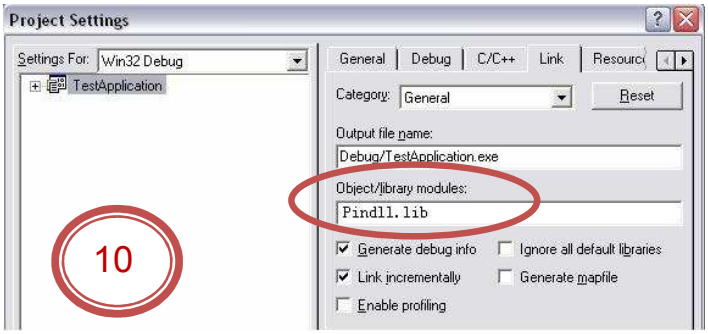
3. Now the PinDll.dll is willing to link statically with the ‘TestApplication’ just created.
4. Provide Header file path for pin.h and pinDll.lib as path of the ‘Include’ folder to Tools -> Options ->Directories.





5. Provide Library path for pinDll.lib as path of the 'Include' folder to Project->Settings->Link->Object/library modules





7.0 Function Description

7.1 Detailed Description on Functions of Reader Control Module

7.1.1 COM_OpenPort

Function Name:	COM_OpenPort		
Description:	Function used to establish a communication for a given RS232 port with its communication parameters.		
Format:	BYTE WINAPI Com_OpenPort(int Comport,long Baud, char Parity, int Stop, int DataBit)		
Parameters:	Comport	Pointer to a null-terminated string that specifies the name of the COM port in format "x". Here x should be an integer.	
	Baud	Specifies the baud rate at which the communication device operates	
	DataBit	Specifies the number of bits in the bytes transmitted and received	
	Parity	Specifies the parity scheme to be used	
	StopBit	Specifies the number of stop bits to be used	
Parameter Values:	Baud Rates:	Baud Rate Index:	Description:
		BAUDRATE_2400	Baud Rate 2400
		BAUDRATE_4800	Baud Rate 4800
		BAUDRATE_9600	Baud Rate 9600
		BAUDRATE_19200	Baud Rate 19200
		BAUDRATE_38400	Baud Rate 38400
		BAUDRATE_115200	Baud Rate 115200
	Parity:	Parity Index:	Description:
		PARITY_NONE	None
		PARITY_ODD	ODD
		PARITY_EVEN	EVEN
	Stop Bits:	Stop Bit Index:	Description:
		STOP_BIT_ONE	1 stop bit
		STOP_BIT_TWO	2 stop bits
Return Values:	TRUE:	Open port success.	
	FALSE:	Fail to open port, or port was opened.	

	Others	Please refer to error code table.
Remarks:	Function could fail if the requested COM port is not available. Function could fail if device is not available Function could fail if the requested COM port was opened.	
Example:	BYTE res = Com_OpenPort(1,38400, 'N', 1, 8);	

7.1.2 COM_ClosePort

Function Name:	COM_ClosePort	
Description:	This function is used to terminate the communication, which is already established byCOM_OpenPort or Com_OpenUSBHID()	
Format:	BOOL WINAPI Com_ClosePort();	
Parameters:	N/A	
Return Values:	Return value	Description
	TRUE	Success to close established port
	FALSE	Fail to close established port
	Others	Please refer to error code table.
Remarks:	- Function could fail if the COM_ClosePort established port cannot be closed	
Example:	bool res = Com_ClosePort();	

7.1.3 COM_OpenUSBHID

Function Name:	COM_OpenUSBHID	
Description:	This function is used to control the device beeper with user defined duration	
Format:	BYTE WINAPI Com_OpenUSBHID()	
Parameters:	none	
Return Values:	Return Value	Description
	TRUE	Success to open the required USB device

	FALSE	Fail to open the required USB device.
	Others	Please refer to error code table.
Remarks:	-Function could fail if the required USB Device cannot be found by PC	
Example:	BYTE res = Com_OpenUSBHID();	

7.1.4 COM_SetBaud

Function Name:	COM_SetBaud		
Description:	This function is used to set device baud		
Format:	BYTE WINAPI Com_SetBaud(long Baud)		
Parameters:	Baud	**Parameters**	**Description:**
		BAUDRATE_2400	Baud rate 2400
		BAUDRATE_4800	Baud rate 4800
		BAUDRATE_9600	Baud rate 9600
		BAUDRATE_19200	Baud rate 19200
		BAUDRATE_38400	Baud rate 38400
		BAUDRATE_115200	Baud rate115200
Return Values:	Return Value	Description	
	TRUE	Succeed to set new baud rate to device	
	FALSE	Fails to set new baud to device	
	Others	Please refer to error code table.	
Remarks:	-If this function was performed successfully, serial communication should be set up with new baud rate		
Example:	long baud = BAUDRATE_9600; BYTE res= Com_SetBaud(baud);		

7.1.5 COM_SetParity

Function Name:	COM_SetParity
Description:	This function is used to set new parity to device

Format:	BYTE WINAPI Com_SetParity(char Parity);		
Parameters:	Parity	**Parameters**	**Description**
		PARITY_NONE	None
		PARITY_ODD	Odd
		PARITY_EVEN	Even
Return Values:	**Return Value**	**Description**	
	TRUE	Succeed to set new parity to device	
	FALSE	Fails to set new parity to device	
	Others	Please refer to error code table.	
Remarks:	-If this function was performed successfully, serial communication should be set up with new parity		
Example:	char parity = PARITY_NONE; BYTE res= Com_SetParity(parity):		

7.1.6 Com_SetStopBit

Function Name:	Com_SetStopBit		
Description:	This function is used to set new stop bit to device		
Format:	BYTE WINAPI Com_SetStopBit(int Stopbit)		
Parameters:	Stopbit	**Parameters**	**Description**
		STOP_BIT_ONE	1 bit
		STOP_BIT_TWO	2 bits
Return Values:	**Return Value**	**Description**	
	TRUE	Succeed to set new stop bit to device	
	FALSE	Fails to set new stop bit to device	
Remarks:	-If this function was performed successfully, serial communication should be set up with new stop bit		
Example:	int Stopbit = STOP_BIT_ONE; BYTE res = Com_SetStopBit(Stopbit);		

7.1.7 Com_GetBaud

Function Name:	Com_GetBaud		
Description:	This function is used to get the current baud rate from device		
Format:	BYTE WINAPI Com_GetBaud(char *Baud);		
Parameters:	Baud	**Parameters**	**Description:**
		BAUDRATE_2400	Baud rate 2400
		BAUDRATE_4800	Baud rate 4800
		BAUDRATE_9600	Baud rate 9600
		BAUDRATE_19200	Baud rate 19200
		BAUDRATE_38400	Baud rate 38400
		BAUDRATE_115200	Baud rate115200
Return Values:	**Return Value**	**Description**	
	TRUE	Succeed to get baud rate.	
	FALSE	Fail to get baud rate.	
	Others	Please refer to error code table.	
Remarks:	Shall be used after COM_OpenPort();		
Example	char *baud; BYTE res = Com_GetBaud(baud);		

7.1.8 Com_GetParity

Function Name:	Com_GetParity		
Description:	This function is used to get the current parity rate from device		
Format:	BYTE WINAPI Com_GetParity(char *Parity);		
Parameters:	Parity Parity	**Parameters**	**Description**
		PARITY_NONE	None
		PARITY_ODD	Odd
		PARITY_EVEN	Even

Return Values:	**Return Value**	**Description**
	TRUE	Succeed to review the current parity rate of device.
	FALSE	Fails to review the current parity rate of device
	Others	Please refer to error code table.
Remarks:	Shall be used after COM_OpenPort();	
Example:	char *parity; BYTE res = Com_GetParity(parity);	

7.1.9 Com_GetStopBit

Function Name:	Com_GetStopBit		
Description:	This function is used to get the current stop bit in the device		
Format:	BYTE WINAPI Com_GetStopBit(char *Stopbit);		
Parameters:	Parity	**Parameters**	**Description**
		PARITY_NONE	None
		PARITY_ODD	Odd
		PARITY_EVEN	Even
Return Values:	**Return Value**	**Description**	
	TRUE	Succeed to review the current stop bit in the device	
	FALSE	Fails to review the current stop bit in the device	
	Others	Please refer to error code table.	
Remarks:	Shall be used after COM_OpenPort();		
Example:	char *stopBit; BYTE res = Com_GetStopBit (stopBit);		

7.1.10 Sys_GetVersion

Function Name:	Sys_GetVersion
Description:	This function is used to get device firmware version

Format:	Sys_GetVersion(char *Version, int *Length);	
Parameters:	**Parameters**	**Description**
	Version	Version string
	Length	The length of version string
Return Values:	**Return Value**	**Description**
	TRUE	Succeed to get device version number
	FALSE	Fails to get device version number
	Others	Please refer to error code table.
Remarks:	Shall be used after COM_OpenPort() or COM_OpenUSBHID()	

Example:	char *version;
	int length=0;
	version=(char *)malloc(128);
	BYTE res = Sys_GetVersion(version, &length);	

7.1.11 Sys_GetModel

Function Name:	Sys_GeModel	
Description:	This function is used to get device model number	
Format:	Sys_GetModel(char *ModelNumber, int *Length);	
Parameters:	**Parameters**	**Description**
	Model number	Model number string
	Length	The length of model number string
Return Values:	**Return Value**	**Description**
	TRUE	Succeed to get device model number
	FALSE	Fails to get device model number
	Others	Please refer to error code table.
Remarks:	Shall be used after COM_OpenPort() or COM_OpenUSBHID()	
Example:	char *version;	

	int length=0;
	version=(char *)malloc(128);
	BYTE res = Sys_GetVersion(version, &length);

7.1.12 Sys_Reset

Function Name:	Sys_Reset	
Description:	This function is used to reset device	
Format:	Sys_Reset()	
Parameters:	**Parameters**	**Description**
	None	None
Return Values:	**Return Value**	**Description**
	TRUE	Succeed to reset device
	FALSE	Fails to reset device
	Others	Please refer to error code table.
Remarks:	Shall be used after COM_OpenPort() or COM_OpenUSBHID()	
Example:	BYTE res = Sys_Reset();	

7.1.13 Sys_ LoadTime

Function Name:	Sys_LoadTime	
Description:	This function is used to load time to device	
Format:	BYTE WINAPI Sys_LoadTime(char *TimeStr, int Length);	
Parameters:	**Parameters**	**Description**
	TimeStr	Time string. If current time is 2011-05-23 08:21:45,, Time string should be 20110523082145
	Length	The length of time string.
Return	**Return Value**	**Description**

Values:	TRUE	Succeed to finish loading time process
	FAIL	Fail to finish loading time process
	Others	Please refer to error code table.
Remarks:	Shall be used after COM_OpenPort() or COM_OpenUSBHID() This is supported by SmartPIN K100 and B100.	

Example:	CString CurTime;
	char *timeStr;
	CTime t = CTime::GetCurrentTime();
	int d = t.GetDay();
	int y = t.GetYear();
	int m = t.GetMonth();
	int h = t.GetHour();
	int mm = t.GetMinute();
	int s = t.GetSecond();
	CurTime.Format("%4d%02d%02d%02d%02d%02d",y,m,d,h,mm,s);
	timeStr = (char *)malloc(14);
	for(int I = 0;i<14;i++)
	  timeStr[i]=CurTime[i];
	BYTE res = Sys_LoadTime(timeStr,14);	

7.1.14 Sys_GetRealTime

Function Name:	Sys_GetRealTime	
Description:	This function is used to get real time form device.	
Format:	BYTE WINAPI Sys_GetRealTime(char *RealTime, int *Length);	
Parameters:	**Parameters**	**Description**
	TimeStr	Return string's format should beYear/Month/Date Hour:Minute:Second
	Length	The length of time string.
Return Values:	**Return Value**	**Description**
	TRUE	Succeed to get real time from device
	FAIL	Fail to get device time
	Others	Please refer to error code table.
Remarks:	Shall be used after COM_OpenPort() or COM_OpenUSBHID() This is supported by SmartPIN K100 and B100	

Example:	char *realTime;
	int res,Length = 0;
	CString str = "";
	realTime = (char *)malloc(128);
	BYTE res = Sys_GetModel(realTime, &Length);	

7.1.15 Sys_GetRemovalRecord

Function Name:	Sys_GetRemovalRecord	
Description:	This function is used to get removal record.	
Format:	BYTE WINAPI Sys_GetRemovalRecord(char *LogString, int *Length);	
Parameters:	**Parameters**	**Description**
	LogString	<Records Number> (<Record Block>…) ● <Records Number> is Number of Record Block. If it is 0, there is not <Record Block> ● <Record Block> has the following format of <UserID> <State> <-> <4 bytes Year> <2 bytes Month> <2 bytes Date> <-> <2 bytes Hour> <2 bytes Minute> </> Where: ◆ <UserID> is 0x31 (User1) or 0x32 (User2) ◆ <State> is 0x30 (Fix) or 0x31 (Removal) ◆ Year, Month, Date, Hour, and Minute need be ASCII code.
	Length	The length of log string. The Max length is 341
Return Values:	**Return Value**	**Description**
	TRUE	Succeed to load numeric key into device
	FAIL	Fail to load numeric key into device
	Others	Please refer to error code table.
Remarks:	Shall be used after COM_OpenPort() or COM_OpenUSBHID() This is supported by SmartPIN K100 and B100	

Example:	char *logStr;
	int Length = 0;
	CString str;
	logStr = (char *)malloc(500);
	BYTE res = Sys_GetRemovalRecord(logStr, &Length);	

7.1.16 Sound_Control

Function Name:	Sound_Control	
Description:	This function is used to open or close beeper	
Format:	BYTE WINAPI Sound_Control(bool Flag);	
Parameters:	**Parameters**	**Description**
	TRUE	Beep on
	FALSE	Beep off
Return Values:	**Return Value**	**Description**
	TRUE	Succeed to open or close beeper
	FALSE	Fails to open or close beeper
	Others	Please refer to error code table.
Remarks:	Shall be used after COM_OpenPort() or COM_OpenUSBHID()	
Example:	bool Flag = true; BYTE res = Sound_Control(Flag);	

7.1.17 Sound_Tone

Function Name:	Sound Tone	
Description:	This function is used to set the frequency or duration of the beeper sound.	
Format:	BYTE WINAPI Sound_Tone(long Frequency, long Duration);	
Parameters:	**Parameters**	**Description**
	Frequency	Smart PIN K100/ B100 :Frequency need be more than 500 and less than 10000 Smart PIN C100 :Frequency need be more than 1000 and less than 20000
	Duration	Duration need be more than 16 and less than 65535.
Return Values:	**Return Value**	**Description**
	TRUE	Succeed to set sound
	FALSE	Fails to set sound

	Others	Please refer to error code table.
Remarks:	Shall be used after COM_OpenPort() or COM_OpenUSBHID()	
Example:	BYTE res = Sound_Tone(2000, 300);	

7.1.18 Display_LoadPrompt

Function Name:	Display_LoadPrompt	
Description:	This function is used to display the required Prompt at the required line number of LCD	
Format:	BYTE WINAPI Display_LoadPrompt(int LineNum, int PromptNum);	
Parameters:	**Parameters**	**Description**
	LineNum	To indicate the line number which show the prompt. The value should be 0 or 1
	Prompt	To indicate the prompt number which is going to showed The number range is 0~9.
Return Values:	**Return Value**	**Description**
	TRUE	Succeed to display the prompt
	FALSE	Fails to display the prompt
	Others	Please refer to error code table.
Remarks:	Shall be used after COM_OpenPort() or COM_OpenUSBHID() This function is only support by SmartPIN C100.	
Example:	BYTE res = Display_LoadPrompt(0, 3);	

7.1.19 Display_SavePrompt

Function Name:	Display_SavePrompt	
Description:	This function is used to save prompts into device	
Format:	BYTE WINAPI Display_SavePrompt(int Num, char *Message);	
Parameters:	**Parameters**	**Description**
	Num	Prompt number is an index number of the prompt. The

		number range is 0~9.
	Prompt	Prompt message. The maximum of the prompt length is 12.
Return Values:	**Return Value**	**Description**
	TRUE	Succeed to save prompt into the device
	FALSE	Fails to save the prompt into the device
	Others	Please refer to error code table.
Remarks:	Shall be used after COM_OpenPort() or COM_OpenUSBHID() This function is only support by SmartPIN C100.	
Example:	char *message=”IDTECH”; Int lineNumber=3; BYTE res = Display_SavePrompt(lineNumber,message);	

7.1.20 Display_Clear

Function Name:	Display_Clear	
Description:	This function is used to clear the prompt in the specified line.	
Format:	BYTE WINAPI Display_Clear(int Line);	
Parameters:	**Parameters**	**Description**
	0x00	Clear the prompt in the first line
	0x01	Clear the prompt in the second line
	0xFF	Clear the prompt in both lines.
Return Values:	**Return Value**	**Description**
	TRUE	Succeed to clear prompt
	FALSE	Fails to clear prompt
	Others	Please refer to error code table.
Remarks:	Shall be used after COM_OpenPort() or COM_OpenUSBHID() This function is only support by SmartPIN C100.	

Example:	BYTE res=0;
	//Clear top line message
	res=Display_Clear(LCD_TOP_LINE);
	//Clear bottom line message	

	res=Display_Clear(LCD_BOTTOM_LINE);
	//Clear all message.
	res=Display_Clear(LCD_BOTH_LINES);

7.1.21 BackLight_ReviewControl

Function Name:	Backlight_ReviewControl	
Description:	This function is used to review the status of LCD back light.	
Format:	BYTE WINAPI BackLight_ReviewControl(bool *Flag);	
Parameters:	**Parameters**	**Description**
	TRUE	LCD back light is on
	FALSE	LCD back light is off
Return Values:	**Return Value**	**Description**
	TRUE	Succeed to get LCD back light status
	FALSE	Fails to get LCD back light status.
	Others	Please refer to error code table.
Remarks:	Shall be used after COM_OpenPort() or COM_OpenUSBHID() This function is only support by SmartPIN C100.	
Example:	bool backlight; BYTE res=BackLight_ReviewControl(&backlight);	

7.1.22 BackLight_ReviewTimer

Function Name:	Backlight_ReviewTimer	
Description:	This function is used to review the timer of back light on	
Format:	BYTE WINAPI BackLight_ReviewTimer(int *Timer);	
Parameters:	**Parameters**	**Description**
	0	LCD back light never off
	1~255	LCD back light is on. After timer time, turn off
Return	**Return Value**	**Description**

Values:	TRUE	Succeed to get LCD back light timer setting.
	FALSE	Fail to get LCD back light timer setting.
	Others	Please refer to error code table.
Remarks:	Shall be used after COM_OpenPort() or COM_OpenUSBHID() This function is only support by SmartPIN C100.	
Example:	int backlightTimer; BYTE res=BackLight_ReviewTimer(backlightTimer);	

7.1.23 BackLight_SetControl

Function Name:	Backlight_SetControl	
Description:	This function is used to change back light on or off	
Format:	BYTE WINAPI BackLight_SetControl(bool Flag);	
Parameters:	**Parameters**	**Description**
	TRUE	LCD back light is on
	OFF	LCD back light is off.
Return Values:	**Return Value**	**Description**
	TRUE	Succeed to set LCD back light status.
	FALSE	Fail to set LCD back light status
	Others	Please refer to error code table.
Remarks:	Shall be used after COM_OpenPort() or COM_OpenUSBHID() This function is only support by SmartPIN C100.	
Example:	bool Flag=true; BYTE res=BackLight_ SetControl (Flag);	

7.1.24 BackLight_ SetTimer

Function Name:	Backlight_ SetTimer
Description:	This function is used to set the during time of back light on.
Format:	BYTE WINAPI BackLight_SetTimer(int Timer);

Parameters:	**Parameters**	**Description**
	0	LCD back light is never off
	1~255	LCD back light is on during timer time.
Return Values:	**Return Value**	**Description**
	TRUE	Succeed to set LCD back light timer
	FALSE	Fail to set LCD back light timer
	Others	Please refer to error code table.
Remarks:	Shall be used after COM_OpenPort() or COM_OpenUSBHID() This function is only support by SmartPIN C100.	
Example:	int backlightTimer=10; BYTE res=BackLight_SetTimer(m_backlightTimer);	

7.1.25 PIN_GetPINBlock

Function Name:	PIN_GetPINBlock	
Description:	This function is used to get PIN block	
Format:	PIN_GetPINBlock(char Key, char EncryptMethod,char *AccountNumber,int Plength, char *EncryptedPIN, int *Length)	
Parameters:	**Parameters**	**Description**
	Key	Key type. Value shall be KEY_DUKPT or Key_MKSK
	Encrypt Method	Encrypt PIN block method. Value shall be ENCRYPTION_TDES or ENCRYPTION_DES
	AccountNumber	Account Number string
	pLength	Account number length. Length should be less than 20 and more than 12.
	EncryptPIN	Device returned the encrypt PIN block data.
	Length	Returned PIN block data length
Return Values:	**Return Value**	**Description**
	TRUE	Succeed to get PIN Block

	FAIL	Fail to get PIN Block
	Others	Please refer to error code table.
Remarks:	Shall be used after COM_OpenPort() or COM_OpenUSBHID() The encrypt PIN block shall be returned after enter PIN and "Enter key"from device. Returned PIN block is encrypted block.	

Example:	//Call thread TheadProc_GetPIN
	AfxBeginThread(ThreadProc_GetPIN, this);
	
	//Define TreadProc_GetPIN
	static UINT ThreadProcGetPIN(LPVOID pParam)
	{
	char key = KEY_DUKPT;;
	char encrypt = DK_ENCRYPTION_TDES;;
	int length = 0;
	char EncryptedPIN[256];
	char *AccountNumber = "40123456789012345678";
	BYTE res = PIN_GetPINBlock(key, encrypt,AccountNumber,20, EncryptedPIN, &length);
	}	

7.1.26 PIN_GetFunKey

Function Name:	PIN_GetFunKey		
Description:	This function is used to get function key		
Format:	BYTE WINAPI PIN_GetFuncKey(char *Key, int *Length);		
Parameters:	**Parameters**	**Value**	**Description**
	Key	0x43	Cancel key
		0x42	Backspace Key
		0x45	Enter Key
		0x46 0x31	F1 Key
		0x46 0x32	F2 Key
		'*'	* key
		'#'	'#'Key
		'?'	'?' key

Return Values:	**Return Value**	**Description**
	TRUE	Succeed to get function key
	FAIL	Fail to get function key
	Others	Please refer to error code table.
Remarks:	Shall be used after COM_OpenPort() or COM_OpenUSBHID() SmartPIN C100 no '*','#' and '?' key. SmartPIN K100/ B100 no 'F1' and 'F2' key	

Example:	char key[128];
	char encrypt;
	int length = 0;
	int res = 0;
	res = PIN_GetFuncKey(key, &length);	

7.1.27 PIN_CancelEnter

Function Name:	PIN_CancelEnter		
Description:	This function is used to cancel get PIN proess or get plaintext process		
Format:	BYTE WINAPI PIN_CancelEnter();		
Parameters:	**Parameters**	**Value**	**Description**
	No parameter`		
Return Values:	**Return Value**	**Description**	
	TRUE	Succeed to cancel PIN process	
	FAIL	Fail to cancel PIN process	
	Others	Please refer to error code table.	
Remarks:	Shall be used after COM_OpenPort() or COM_OpenUSBHID() This function shall return with an error if device isn't working on get PIN or get plaintext process.		
Example:	BYTE res=PIN_CancelEnter();		

7.1.28 PIN_SetPINLength

Function Name:	PIN_SetPINLength		
Description:	This function is used to set the maximum and minimum length during PIN		
Format:	BYTE WINAPI PIN_SetPINLength(int Minlen, int Maxlen);		
Parameters:	**Parameters**	**Value**	**Description**
	minLen	4~maxLen	The Minimum of the PIN length
	maxLen	minLen~12	The Maximum of the PIN length
Return Values:	**Return Value**	**Description**	
	TRUE	Succeed to set the length of PIN Block	
	FAIL	Fail to set the length of PIN Block	
	Others	Please refer to error code table.	
Remarks:	Shall be used after COM_OpenPort() or COM_OpenUSBHID()		

Example:	int minLen=4;
	int maxLen=12;
	BYTE res=PIN_SetPINLength(minLen, maxLen);		

7.1.29 PIN_GetPINLength

Function Name:	PIN_GetPINLength	
Description:	This function is used to get the maximum and minimum length during PIN	
Format:	BYTE WINAPI PIN_GetPINLength(int *Minlen, int *Maxlen);	
Parameters:	**Parameters**	**Description**
	*minLen	The Minimum of the PIN length
	*maxLen	The Maximum of the PIN length
Return Values:	**Return Value**	**Description**
	TRUE	Succeed to get the length of PIN Block
	FAIL	Fail to get the length of PIN Block

	Others	Please refer to error code table.
Remarks:	Shall be used after COM_OpenPort() or COM_OpenUSBHID()	
Example:	int *minLen; int *maxLen; BYTE res = PIN_GetPINLength(&minLen, &maxLen);	

7.1.30 PIN_ GetNumeric

Function Name:	PIN_ GetNumeric	
Description:	This function is used to get the clear text with special prompt	
Format:	BYTE WINAPI PIN_GetClearText(char *InputMessage,int MessageLength,char *ClearTextPIN, int *Length);	
Parameters:	**Parameters**	**Description**
	InputMessage	Show message.
	MessageLength	Show message string length should be less than 12
	ClearTextPIN	Return clear text string
	length	Return the length of clear text.
Return Values:	**Return Value**	**Description**
	TRUE	Succeed to get Plain text
	FAIL	Fail to get Plain text
	Others	Please refer to error code table.
Remarks:	Shall be used after COM_OpenPort() or COM_OpenUSBHID()	

Example:	//Call thread TheadProc_GetNumeric
	AfxBeginThread(ThreadProc_GetNumeric, this);
	
	//Define TreadProc_GetNumeric
	static UINT ThreadProc_GetNumeric(LPVOID pParam)
	{
	char key[128];
	char *inputMessage;
	int length = 0;
	//Encrypt LCD message by numeric key
	unsigned char ebuf[] =
	{0x1e,0x78,0x19,0xa6,0xd9,0xa3,0xb8,0x35,0xce,0x47,0xcb,0x41,0xc4,0x	

	a0,0x5e,0xa4,0x3e,0x87,0x34,0x5a,0x86,0x80,0xd7,0x99,0xa2,0x02,0x6f,0x87,0xb9,0xf2,0x5a,0x2d,0x8d,0x33,0x2a,0x02,0x1a,0x89,0x18,0x08,0xed,0x3a,0x65,0x63,0x65,0xb7,0x70,0x46,0xea,0x31,0xe1,0x9f,0x89,0x5b,0x29,0x45,0x69,0x9d,0xac,0x22,0x48,0x7c,0xef,0xbd,0xf6,0x65,0x9a,0xb4,0x0a,0x76,0x55,0xd5,0x48,0x10,0x67,0x2e,0xd2,0x61,0x4b,0xdd,0x98,0x2c,0x46,0xaa,0x5d,0x21,0x10,0xe0,0x37,0xbd,0xe7,0x63,0xce,0xcc,0xb0,0xeb,0x92,0x65,0x44,0x95,0xec,0x38,0x95,0xc7,0xc6,0xe9,0x7e,0x38,0x5f,0x57,0x3e,0x23,0xc0,0x15,0x33,0x58,0xef,0x19,0xca,0x38,0xf0,0xf1,0x61,0x61,0x2d,0x8a,0x28,0xa1};
	for(int i=0;i<128;i++)
	inputMessage[i]=ebuf[i];
	BYTE res = PIN_GetNumeric(inputMessage,128,key, &length);
	}

7.1.31 PIN_GetClearText

Function Name:	PIN_GetClearText	
Description:	This function is used to get the clear text	
Format:	BYTE WINAPI PIN_GetClearText(char *ClearTextPIN, int *Length);	
Parameters:	**Parameters**	**Description**
	ClearTextPIN	Return clear text string
	length	Return the length of clear text.
Return Values:	**Return Value**	**Description**
	TRUE	Succeed to get Plain text
	FAIL	Fail to get Plain text
	Others	Please refer to error code table.
Remarks:	Shall be used after COM_OpenPort() or COM_OpenUSBHID() This one is supported by SmartPIN B100 and K100.	

Example:	//Call thread TheadProc_GetClearText
	AfxBeginThread(ThreadProc_GetClearText, this);
	
	//Define TreadProc_GetClearText
	static UINT ThreadProc_GetClearText(LPVOID pParam)
	{
	CString temp;
	char key[128];	

	char encrypt;
	int length = 0;
	BYTE res = PIN_GetClearText(key, &length);
	}

7.1.32 KEY_LoadMACKEY

Function Name:	KEY_LoadMACKEY	
Description:	This function is used to load Mac key	
Format:	BYTE WINAPI KEY_LoadMACKEY(char *Key,int length);	
Parameters:	**Parameters**	**Description**
	*Key	Mac key
	length	Mac key length
Return Values:	**Return Value**	**Description**
	TRUE	Succeed to load mac key to device
	FAIL	Fail to load mac key to device
	Others	Please refer to error code table.
Remarks:	Shall be used after COM_OpenPort() or COM_OpenUSBHID()	

Example:	unsigned char buf[] = {0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,
	0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF};
	unsigned int len=0;
	char *key;
	key=(char *)malloc(16);
	for(int i=0;i<16;i++)
	key[i]=buf[i];
	int res= KEY_LoadMACKEY(key,16);	

7.1.33 KEY_EncryptData

Function Name:	KEY_EncryptData
Description:	This function is used to encrypt message by Mac key
Format:	BYTE WINAPI KEY_EncryptData(char *RawData,int *length);

Parameters:	**Parameters**	**Description**
	*RawData	Input : plaintext Output:encrypt data
	length	Encrypt data block length
Return Values:	**Return Value**	**Description**
	TRUE	Succeed to encrypt raw data
	FAIL	Fail to encrypt raw data
	Others	Please refer to error code table.
Remarks:	Shall be used after COM_OpenPort() or COM_OpenUSBHID()	

Example:	int len = 0;
	char *RawData;
	CString str = "Welcome to Dll Demo!";
	m_Result = "";
	len = str.GetLength();
	RawData = (char *)malloc(len);
	for(int i=0;i<len;i++)
	RawData[i]=str[i];
	BYTE res=KEY_EncryptData(RawData,&len);	

7.1.34 GetDllVersion

Function Name:	GetDllVersion	
Description:	This function is used to get the current version of PinDll	
Format:	BYTE WINAPI GetDllVersion(char *DllVersion, int *Length);	
Parameters:	**Parameters**	**Description**
	*DllVersion	Return version string of Dll version
	Length	Return the length of Dll version
Return Values:	**Return Value**	**Description**
	TRUE	Succeed to get Dll version number
	FAIL	Fail to get Dll version number

	Others	Please refer to error code table.
Remarks:	None	
Example	char *DllVersion; int Length = 0; DllVersion = (char *)malloc(128); BYTE res = GetDllVersion(DllVersion,&Length);	