



PT-60 Mobile Computer Programming Manual

DOC NO. UM-PT605-02

Jan. 2010

Version 1.2

Table of Contents

OVERVIEW	2
SDK FUNCTIONS	3
SYSAPIX.DLL	4
AUDIO RELATED FUNCTIONS.....	6
<i>Audio_GetVolume</i>	6
<i>Audio_SetVolume</i>	7
BATTERY RELATED FUNCTION	8
<i>GetBatteryStatus</i>	8
DISPLAY RELATED FUNCTIONS	10
<i>BacklightOn</i>	10
<i>Display_QueryBacklightIntensity</i>	11
<i>GetBacklightStatus</i>	13
<i>PowerOnLCD</i>	14
<i>SetBacklightPWM</i>	15
KEYPAD RELATED FUNCTIONS	16
<i>EnablePowerButton</i>	16
<i>GetKeypadAlphaMode</i>	17
<i>SendKbdVisualKey</i>	18
<i>SetKeypadAlphaMode</i>	19
LED RELATED FUNCTIONS.....	20
<i>GetKeypadLEDStatus</i>	20
<i>GoodReadLEDOn</i>	21
<i>KeypadLEDOn</i>	22
SYSTEM RELATED FUNCTIONS	23
<i>CallSuspend</i>	23
<i>EnableAutoConnect</i>	24
<i>RegisterAlphaKeyNotification</i>	25
<i>ShowChineseIME</i>	26
<i>ShowDesktop</i>	27
<i>ShowExploreToolbar</i>	28
<i>ShowTaskbar</i>	29
<i>UnRegisterAlphaKeyNotification</i>	30
VIBRATOR RELATED FUNCTIONS.....	31
<i>VibratorOn</i>	31

BLUETOOTH RELATED FUNCTION.....	32
<i>BT_Enable</i>	32
<i>BT_Disable</i>	33
WIRELESS RELATED FUNCTION	34
<i>WL_Enable</i>	34
<i>WL_Disable</i>	35
SCANAPIAX.DLL	36
API_SCAN RELATED FUNCTIONS	38
<i>API_Register</i>	38
<i>API_Unregister</i>	39
<i>API_GetBarData</i>	40
<i>API_GetBarDataLength</i>	42
<i>API_GetBarType</i>	43
<i>API_GetError</i>	45
<i>API_GetSysError</i>	46
<i>API_GoodRead</i>	47
<i>API_LoadSettingsFromFile</i>	48
<i>API_Reset</i>	49
<i>API_ResetBarData</i>	50
<i>API_SaveSettingsToFile</i>	51
<i>API_SaveSettingsToScanner</i>	52
<i>S2K_IsLoad</i>	53
<i>S2K_Load</i>	54
<i>SCAN_QueryStatus</i>	55
<i>SCAN_SendCommand</i>	56
<i>SCAN_ResumeSystem</i>	57
<i>SCAN_BatchSetting</i>	58
<i>SCAN_BatchRead</i>	59
<i>GetScannerDevice</i>	60
SCAN2KEY RELATED FUNCTIONS	61
<i>PT_OpenScan2Key</i>	61
<i>PT_CloseScan2Key</i>	62
<i>PT_SetToDefault</i>	63
SCANNER RELATED FUNCTIONS	64
<i>PT_EnableScanner</i>	64
<i>PT_DisableScanner</i>	65
<i>PT_CheckBarcodeData</i>	66
<i>PT_GetBarcodeData</i>	67

<i>PT_SetDefault</i>	69
SCAN KEY RELATED FUNCTIONS.....	70
<i>EnableTriggerKey</i>	70
<i>GetLibraryVersion</i>	71
<i>GetTriggerKeyStatus</i>	72
<i>PressTriggerKey</i>	73
<i>TriggerKeyStatus</i>	74
SCAN COMMAND TABLE	75
FUNCTION RETURN VALUES	87

Overview

The **Argox** Mobile Computer Software Developer Kit (SDK) Help is intended to assist programmers with the creation of applications for **Argox** Mobile Computers running a Microsoft® Windows® .NET CE5.0 Operating System. It gives all of the details necessary for calling functions which control the devices on the **Argox** Mobile Computer or access the Value-added device module, such as Scanning and Wireless.

The help file is organized into two sections, one is the system related, and the other one is the value-added scanning functions providing the following information:

- Argox Mobile Computer standard Application Programming Interface (API) Definitions for system related.

Audio

Display

Keypad

Led and Vibrator Indicators

Battery Status

System Settings

- Argox Scanning device Application Programming Interface (API) Definitions

API definitions illustrate how to call a given function. The API definitions are structured with some information including prototypes, parameters, return values, examples and requirements for each API. The “Requirements” section gives the information on whether or not a device supports a specific API function and the files to be included.

SDK Functions

When user wants to use SDK to develop their own program, they should link DLL file or LIB file, and include header file SYSAPIAX.H.

There are two examples to show how to use LIB file and DLL file on their project. We will use Visual Studio 2005 to explain.

Example 1: Using LIB file.

At first you should include sysapiax.lib in your project.

```
#include "Sysapiax.h"
main()
{
    .....
    SetBacklightPWM(100, 100);
    .....
}
```

Example 2: Using DLL file.

```
HINSTANCE dllHandle = NULL;
typedef DWORD (_stdcall *pfnSetBacklightPWM)(int nACPowerPercent, int
nBatteryPercent);
pfnSetBacklightPWM    m_SetBacklightPWM;

main()
{
    dllHandle = LoadLibrary(L"SYSAPIAX.dll");
    m_SetBacklightPWM = (pfnSetBacklightPWM) ::GetProcAddress(dllHandle,
_T("SetBacklightPWM"));
    m_SetBacklightPWM(0, 0);
    FreeLibrary(dllHandle);
}
```

SYSAPIAX.DLL

In this SDK, we supply SYSAPIAX.DLL which includes several functions to allow programmer to control device drivers and system functions. User can use WINCE develop tool like Visual Studio 2005 to develop application program. The function description is given below.

Audio Related Functions

- [Audio_GetVolume](#) – Query the current volume setting.
- [Audio_SetVolume](#) – Set the volume setting.

Battery Related Function

- [GetBatteryStatus](#) – Gets main battery status.

Display Related Functions

- [BacklightOn](#) – Turn on or off screen backlight.
- [Display_QueryBacklightIntensity](#) – Query backlight intensity.
- [GetBacklightStatus](#) – Gets screen backlight status.
- [PowerOnLCD](#) – Turn on or off LCD power.
- [SetBacklightPWM](#) – Adjusts screen backlight brightness.

KeyPad Related Functions

- [EnablePowerButton](#) – Enable and disable power button.
- [GetAlphaMode](#) – Get the current input mode.
- [SendKbdVisualKey](#) – Sends a visual key to key buffer.
- [SetAlphaMode](#) – Change input mode.

LED Related Functions

- [GetKeypadLEDStatus](#) – Gets keypad LED status.
- [GoodReadLEDOn](#) – Turn on and off good read LED.
- [KeypadLEDOn](#) – Turn on or off keypad LED.

System Related Functions

- [CallSuspend](#) – Enter suspend mode.
- [EnableAutoConnect](#) – Turn auto-connect on and off.
- [RegisterAlphaKeyNotification](#) – Register a request with send a message when the alpha key pressed.
- [ShowChineseIME](#) – Display and hide the Chinese IME.

-
- [ShowDeskTop](#) – Display and hide all icons on desktop.
 - [ShowExploreToolbar](#) – Display and hide toolbar on windows explorer.
 - [ShowTaskbar](#) – Display and hide taskbar.
 - [UnregisterAlphaKeyNotification](#) – Unregister message request.

Vibrator Related Functions

- [VibratorOn](#) – On and off vibration indicator.

Audio Related Functions

Audio_GetVolume

This function queries the current volume setting.

```
DWORD Audio_GetVolume
{
    LPDWORD lpdwVolume
}
```

Parameters

lpdwVolume

[out] The current volume setting.

Return Values

If function succeeds, the return value is [E_FUNC_SUCCEED](#). If function fails, the return value is [E_FUNC_ERROR](#).

Example

```
DWORD dwResult, dwVolume;
dwResult = Audio_GetVolume(&dwVolume);
if(dwResult != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Audio_GetVolume fail"));
else
{
    CString strTemp;
    strTemp.Format(_T("Volume:      %d"),dwVolume);
    AfxMessageBox(strTemp);
}
```

Requirements

OS Versions: Windows CE 5.0 and later.
Header: sysapi.h
Link Library: sysapi.lib
Link DLL: sysapi.dll
Device: PT60

Audio_SetVolume

This function sets the current volume setting.

```
DWORD Audio_SetVolume
{
    DWORD dwVolume
}
```

Parameters

dwVolume

[in] Specifies a new volume setting. The default setting is 0x99999999.

Return Values

If function succeeds, the return value is [E_FUNC_SUCCEED](#). If function fails, the return value is [E_FUNC_ERROR](#).

Example

```
DWORD dwResult,dwVolume;
dwVolume = 0x11111111;
dwResult = Audio_SetVolume(dwVolume);
if(dwResult != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Audio_SetVolume fail"));
```

Requirements

OS Versions: Windows CE 5.0 and later.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT60

Battery Related Function

GetBatteryStatus

This function gets main battery status.

```
int GetBatteryStatus
{
}
```

Parameters

None.

Return Values

The return value can be one of the values in the following table.

Return value	Description
0	battery high
1	battery low
2	battery critical
3	battery charging
4	no battery
5	battery unknown

Example

```
switch (GetBatteryStatus())
{
case 0:
    AfxMessageBox(_T("Battery High"));
    break;
case 1:
    AfxMessageBox(_T("Battery Low"));
    break;
case 2:
    AfxMessageBox(_T("Battery Critical"));
    break;
case 3:
```

```
        AfxMessageBox(_T("Battery Charging"));
        break;
    case 4:
        AfxMessageBox(_T("No Battery"));
        break;
    case 5:
        AfxMessageBox(_T("Battery Unknown"));
        break;
}
```

Requirements

OS Versions: Windows CE 5.0 and later.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT60

Display Related Functions

BacklightOn

This function will always turn on or off screen backlight.

```
DWORD BacklightOn
{
    BOOL bOn
}
```

Parameters

bOn

[in] Flag that indicates whether turn on screen backlight(TRUE) or turn off screen backlight(FALSE).

Return Values

If function succeeds, the return value is [E_FUNC_SUCCEEDED](#). If function fails, possible return value are [E_FUNC_ERROR](#), [E_FUNC_PAR_ERROR](#).

Remarks

After this function turn on or off backlight, the backlight will always on or off. The backlight setting of display properties in control panel does not work until terminal resets.

Example

```
DWORD dwResult;
dwResult = BacklightOn(TRUE);
if(dwResult != E_FUNC_SUCCEEDED)
    AfxMessageBox(_T("BacklightOn fail"));
```

Requirements

OS Versions: Windows CE 5.0 and later.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT60

Display_QueryBacklightIntensity

This function will return the backlight intensity of external power and battery power.

```
DWORD Display_QueryBacklightIntensity
{
    LPDWORD lpdwACBacklight,
    LPDWORD lpdwBatteryBacklight
}
```

Parameters

lpdwACBacklight

[out] The backlight intensity of external power.

lpdwBatteryBacklight

[out] The backlight intensity of battery power.

Return Values

If function succeeds, the return value is [E_FUNC_SUCCEED](#). If function fails, possible return value are [E_FUNC_ERROR](#), [E_FUNC_NULLPTR](#).

Remarks

The parameters will be one of the values in the following table.

Backlight intensity	Backlight brightness
4	super
3	normal
2	fine
1	micro
0	off

Example

```
DWORD dwResult, dwValue1, dwValue2;
dwResult = Display_QueryBacklightIntensity(&dwValue1, &dwValue2);
if(dwResult != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Display_QueryBacklightIntensity fail"));
else
{
    CString strTemp;
    strTemp.Format(_T("AC backlight intensity: %d, Battery backlight intensity: %d"), dwValue1, dwValue2);
    AfxMessageBox(strTemp);
}
```

Requirements

OS Versions: Windows CE 5.0 and later.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT60

GetBacklightStatus

This function gets screen backlight status.

```
DWORD GetBacklightStatus  
{  
}  
}
```

Parameters

None.

Return Values

The return value indicates whether screen backlight is 1 = screen backlight on or screen backlight is 0 = screen backlight off.

Example

```
DWORD dwResult;  
dwResult = GetBacklightStatus();  
if(dwResult == 1)  
    AfxMessageBox(_T("Backlight on"));  
else  
    AfxMessageBox(_T("Backlight off"));
```

Requirements

OS Versions: Windows CE 5.0 and later.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT60

PowerOnLCD

This function turns on or off LCD power.

```
DWORD PowerOnLCD
{
    BOOL bOn
}
```

Parameters

bOn

[in] Flag that indicates whether turn on LCD power(TRUE) or turn off LCD power(FALSE).

Return Values

If function succeeds, the return value is [E_FUNC_SUCCEED](#). If function fails, possible return value are [E_FUNC_ERROR](#), [E_FUNC_PAR_ERROR](#).

Remarks

After calling this function with *bOn* is FALSE, terminal will only turn off LCD power. It means that terminal still works. You should call this function to turn on LCD power or reset terminal.

Example

```
DWORD dwResult;
dwResult = PowerOnLCD(FALSE); //power off LCD
if(dwResult != E_FUNC_SUCCEED)
    AfxMessageBox(_T("PowerOnLCD fail"));
Sleep(3000);
dwResult = PowerOnLCD(TRUE); //power on LCD
if(dwResult != E_FUNC_SUCCEED)
    AfxMessageBox(_T("PowerOnLCD fail"));
```

Requirements

OS Versions: Windows CE 5.0 and later.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT60

SetBacklightPWM

This function adjusts screen backlight brightness.

```
DWORD SetBacklightPWM
{
    int nACPowerPercent,
    int nBatteryPercent
}
```

Parameters

nACPowerPercent, nBatteryPercent

[in] One is brightness setting using AC power and the other is brightness setting using battery. These two members must be one of the values in the following table.

nPercent	Backlight brightness
100	super
75	normal
50	fine
25	micro
0	off

Return Values

If function succeeds, the return value is [E_FUNC_SUCCEED](#). If function fails, possible return value are [E_FUNC_ERROR](#), [E_FUNC_PAR_ERROR](#).

Remarks

The Backlight Setting program in Control Panel sets screen backlight brightness. Called this function will also change the brightness in Backlight Setting. You can use this function or Backlight Setting program in Control Panel to adjust backlight brightness.

Example

```
DWORD dwResult = SetBacklightPWM(100,100);
if(dwResult != E_FUNC_SUCCEED)
    AfxMessageBox(_T("SetBacklightPWM fail"));
```

Requirements

OS Versions: Windows CE 5.0 and later.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT60

KeyPad Related Functions

EnablePowerButton

This function will enable or disable power button.

```
DWORD EnablePowerButton
{
    BOOL bOn
}
```

Parameters

bOn

[in] Flag that indicates whether enable power button(TRUE) or disable power button(FALSE).

Return Values

If function succeeds, the return value is [E_FUNC_SUCCEEDED](#). If function fails, possible return value are [E_FUNC_ERROR](#), [E_FUNC_PAR_ERROR](#).

Remarks

If the *bOn* parameter is FALSE, power button is disabled. The power button will not work when power button pressed. If terminal enters suspend mode, the power button will work once to wake up. When terminal wakes up, the power button is still disabled. Until this function calls with parameter TRUE to enable power button.

Example

```
DWORD dwResult;
dwResult = EnablePowerButton(FALSE);
if(dwResult != E_FUNC_SUCCEEDED)
    AfxMessageBox(_T("EnablePowerButton fail"));
```

Requirements

OS Versions: Windows CE 5.0 and later.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT60

GetKeypadAlphaMode

This function will get the current input mode.

```
DWORD GetKeypadAlphaMode  
{  
}  

```

Parameters

None.

Return Values

The return value can be one of the values in the following table.

Return value	Alpha mode
0	numeric mode
1	lowercase letter mode
2	uppercase lette mode

Example

```
DWORD dwResult;  
dwResult = GetKeypadAlphaMode();  
switch (dwResult){  
case 0:  
    AfxMessageBox(_T("Numeric mode"));  
    break;  
case 1:  
    AfxMessageBox(_T("Lowercase letter mode"));  
    break;  
case 2:  
    AfxMessageBox(_T("Uppercase letter mode"));  
    break;  
}
```

Requirements

OS Versions: Windows CE 5.0 and later.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT60

SendKbdVisualKey

This function sends a visual key to key buffer.

```
DWORD SendKbdVisualKey
{
    BYTE Key
}
```

Parameters

Key

[in] Specifies a virtual-key code.

Return Values

If function succeeds, the return value is [E_FUNC_SUCCEED](#). If function fails, possible return value are [E_FUNC_ERROR](#), [E_FUNC_PAR_ERROR](#).

Example

```
CString strTemp;
strTemp = "VisualKey";
for(int i=0;i<strTemp.GetLength();i++)
    SendKbdVisualKey((unsigned char)strTemp.GetAt(i));
```

Requirements

OS Versions: Windows CE 5.0 and later.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT60

SetKeypadAlphaMode

This function will change input mode.

```
DWORD SetKeypadAlphaMode
{
    int nMode
}
```

Parameters

nMode

[in] Flags for set input mode. This member must be one of the values in the following table.

Value	Alpha mode
0	numeric mode
1	lowercase letter mode
2	uppercase letter mode

Return Values

If function succeeds, the return value is [E_FUNC_SUCCEED](#). If function fails, possible return value are [E_FUNC_ERROR](#), [E_FUNC_PAR_ERROR](#).

Example

```
DWORD dwResult;
dwResult = SetKeypadAlphaMode(1);
if(dwResult != E_FUNC_SUCCEED)
    AfxMessageBox(_T("SetKeypadAlphaMode fail"));
```

Requirements

OS Versions: Windows CE 5.0 and later.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT60

LED Related Functions

GetKeypadLEDStatus

This function gets keypad LED status.

```
BOOL GetKeypadLEDStatus  
{  
}  
}
```

Parameters

None.

Return Values

The return value indicates whether keypad LED is on(TRUE) or keypad LED is off(FALSE).

Example

```
BOOL bResult;  
bResult = GetKeypadLEDStatus();  
if(bResult == TRUE)  
    AfxMessageBox(_T("Keypad LED on"));  
else if(bResult == FALSE)  
    AfxMessageBox(_T("Keypad LED off"));
```

Requirements

OS Versions: Windows CE 5.0 and later.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT60

GoodReadLEDOOn

This function turns on and off goodread LED.

```
DWORD GoodReadLEDOOn
{
    BOOL bOn
}
```

Parameters

bOn

[in] Flag that indicates whether turn on goodread LED(TRUE) or turn off goodread LED(FALSE).

Return Values

If function succeeds, the return value is [E_FUNC_SUCCEEDED](#). If function fails, possible return value are [E_FUNC_ERROR](#), [E_FUNC_PAR_ERROR](#).

Example

```
DWORD dwResult;
dwResult = GoodReadLEDOOn(TRUE);
if(dwResult != E_FUNC_SUCCEEDED)
    AfxMessageBox(_T("GoodReadLEDOOn fail"));
```

Requirements

OS Versions: Windows CE 5.0 and later.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT60

KeypadLEDOn

This function always turns on or off keypad LED.

```
DWORD KeypadLEDOn
{
    BOOL bOn
}
```

Parameters

bOn

[in] Flag that indicates whether turn on keypad LED(TRUE) or turn off keypad LED(FALSE).

Return Values

If function succeeds, the return value is [E_FUNC_SUCCEEDED](#). If function fails, possible return value are [E_FUNC_ERROR](#), [E_FUNC_PAR_ERROR](#).

Remarks

The KeyPad LED setting in Control Panel is used to set the Keypad LED operation to meet requirements. Called this function will change the KeyPad LED setting to always on or off. You can use this function or KeyPad LED setting in Control Panel to always turn on or off keypad LED.

Example

```
DWORD dwResult;
dwResult = KeypadLEDOn(TRUE);
if(dwResult != E_FUNC_SUCCEEDED)
    AfxMessageBox(_T("KeypadLEDOn fail"));
```

Requirements

OS Versions: Windows CE 5.0 and later.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT60

System Related Functions

CallSuspend

After called this function, terminal will enter suspend mode.

```
void CallSuspend  
{  
}
```

Parameters

None.

Return Values

None.

Example

```
//suspend device  
CallSuspend();
```

Requirements

OS Versions: Windows CE 5.0 and later.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT60

EnableAutoConnect

The EnableAutoConnect function turns Autoconnect on and off.

```
BOOL EnableAutoConnect
{
    BOOL bEnable
}
```

Parameters

bEnable

[in] Flag that indicates whether ActiveSync is being automatically executed (TRUE) or ActiveSync is being not automatically executed (FALSE) when user plug cable into terminal.

Return Values

Return TRUE if the operation is successful; otherwise FALSE.

Remarks

After called EnableAutoConnect with bEnable set to TRUE, terminal will automatically execute ActiveSync program when user plug cable into terminal. After called EnableAutoConnect with bEnable set to FALSE, terminal will not automatically execute ActiveSync program when user plug cable into terminal.

Example

```
BOOL bResult;
bResult = EnableAutoConnect(TRUE);
if(bResult == FALSE)
    AfxMessageBox(_T("EnableAutoConnect fail"));
```

Requirements

OS Versions: Windows CE 5.0 and later.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT60

RegisterAlphaKeyNotification

Register the application to SYSAPIAX.dll, so that SYSAPIAX.dll will send a windows message to the application when the alpha key pressed.

```
DWORD RegisterAlphaKeyNotification
{
    HANDLE hWnd,
    UINT uMsg
}
```

Parameters

hWnd

[in] The window handle of the application that is to receive the message.

uMsg

[in] The message value that is to be sent when alpha key pressed.

Return Values

Return 0 if the operation is successful, otherwise return 1.

Remarks

The application should call UnregisterAlphaKeyNotification function to unregister message from the dll.

Example

```
if(RegisterAlphaKeyNotification(this->m_hWnd,WM_USER+0x0001))
    AfxMessageBox(_T("RegisterAlphaKeyNotification FAIL!!"));
```

Requirements

OS Versions: Windows CE 5.0 and later.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT60

ShowChineseIME

The ShowChineseIME function display and hide the Chinese IME.

```
BOOL ShowChineseIME
{
    BOOL bShow
}
```

Parameters

bShow

[in] Flag that indicates whether display the Chinese IME(TRUE) or hide the Chinese IME(FALSE).

Return Values

Return TRUE if the operation is successful; otherwise FALSE.

Remarks

The Chinese IME only support in Chinese OS. It will work after call this function and reset terminal.

Example

```
BOOL bResult;
bResult = ShowChineseIME(TRUE);
if(bResult == FALSE)
    AfxMessageBox(_T("ShowChineseIME fail"));
```

Requirements

OS Versions: Windows CE 5.0 and later.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT60

ShowDesktop

The ShowDesktop function display and hide all icons on desktop.

```
BOOL ShowDesktop
{
    BOOL bShow
}
```

Parameters

bShow

[in] Flag that indicates whether display the desktop(TRUE) or hide the desktop(FALSE).

Return Values

Return TRUE if the operation is successful; otherwise FALSE.

Remarks

After called this function with parameter FALSE, terminal will hide all icons on desktop. After called this function with parameter TRUE, terminal will display all icons which had already showed on desktop.

Example

```
BOOL bResult;
bResult = ShowDesktop(TRUE);
if(bResult == FALSE)
    AfxMessageBox(_T("ShowDesktop fail"));
```

Requirements

OS Versions: Windows CE 5.0 and later.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT60

ShowExploreToolbar

The ShowExploreToolbar function display and hide toolbar on windows explore.

```
BOOL ShowExploreToolbar
{
    BOOL bShow
}
```

Parameters

bShow

[in] Flag that indicates whether display the toolbar on windows explore (TRUE) or hide the toolbar on windows explore (FALSE).

Return Values

Return TRUE if the operation is successful; otherwise FALSE.

Remarks

The ShowExploreToolbar function only effect the windows explorers that opened already.

Example

```
BOOL bResult;
bResult = ShowExploreToolbar(TRUE);
if(bResult == FALSE)
    AfxMessageBox(_T("ShowExploreToolbar fail"));
```

Requirements

OS Versions: Windows CE 5.0 and later.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT60

ShowTaskbar

The ShowTaskbar function display and hide the taskbar.

```
BOOL ShowTaskbar
{
    BOOL bShow
}
```

Parameters

bShow

[in] Flag that indicates whether display the taskbar(TRUE) or hide the taskbar(FALSE).

Return Values

Return TRUE if the operation is successful; otherwise FALSE.

Remarks

After called this function, terminal will display or hide taskbar. If taskbar is hide by this function, it need to call this function to display taskbar again.

Example

```
BOOL bResult;
bResult = ShowTaskbar(TRUE);
if(bResult == FALSE)
    AfxMessageBox(_T("ShowTaskbar fail"));
```

Requirements

OS Versions: Windows CE 5.0 and later.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT60

UnRegisterAlphaKeyNotification

The UnregisterAlphaKeyNotification function requests that the application no longer receive alpha key pressed notification messages.

```
DWORD UnregisterAlphaKeyNotification
{
    HANDLE hWnd,
}
```

Parameters

hWnd

[in] The window handle of the application.

Return Values

Return 0 if the operation is successful, otherwise return 1.

Example

```
if(UnregisterAlphaKeyNotification(this->m_hWnd))
    AfxMessageBox(_T("UnregisterAlphaKeyNotification FAIL!!"));
```

Requirements

OS Versions: Windows CE 5.0 and later.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT60

Vibrator Related Functions

VibratorOn

This function turns on or off Vibration indicator

```
DWORD VibratorOn
{
    BOOL bOn
}
```

Parameters

bOn

[in] Flag that indicates whether turn on vibrator(TRUE) or turn off vibrator LED(FALSE).

Return Values

If function succeeds, the return value is [E_FUNC_SUCCEED](#). If function fails, possible return value is [E_FUNC_ERROR](#), [E_FUNC_PAR_ERROR](#).

Remarks

You can use this function to vibrate the terminal to indicate operator that some thing happened. Called this function will not change the “Scanner Vibrator” setting.

Example

```
DWORD dwResult;
dwResult = VibratorOn(TRUE);
if(dwResult != E_FUNC_SUCCEED)
    AfxMessageBox(_T("VibratorOn fail"));
```

Requirements

OS Versions: Windows CE 5.0 and later.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT60

■ Bluetooth Related Function

BT_Enable

This function enable Bluetooth device

```
BOOL BT_Enable  
{  
}
```

Parameters

None.

Return Values

If function succeeds, the return value is TRUE. If function fails, the return value is FALSE.

Example

```
DWORD dwResult;  
dwResult = BT_Enable();  
if(dwResult == 0)  
    AfxMessageBox(_T("Open Bluetooth fail"));
```

Requirements

OS Versions: Windows CE 5.0 and later.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT60B

BT_Disable

This function disable Bluetooth device

```
BOOL BT_Disable  
{  
}
```

Parameters

None.

Return Values

If function succeeds, the return value is TRUE. If function fails, the return value is FALSE.

Example

```
DWORD dwResult;  
dwResult = BT_Disable();  
if(dwResult == 0)  
    AfxMessageBox(_T("Close Bluetooth fail"));
```

Requirements

OS Versions: Windows CE 5.0 and later.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT60B

Wireless Related Function

WL_Enable

This function enable wireless

```
BOOL WL_Enable  
{  
}
```

Parameters

None.

Return Values

If function succeeds, the return value is TRUE. If function fails, the return value is FALSE.

Example

```
DWORD dwResult;  
dwResult = WL_Enable();  
if(dwResult == 0)  
    AfxMessageBox(_T("Open wireless device fail"));
```

Requirements

OS Versions: Windows CE 5.0 and later.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT60W

WL_Disable

This function disable Bluetooth device

```
BOOL WL_Disable  
{  
}
```

Parameters

None.

Return Values

If function succeeds, the return value is TRUE. If function fails, the return value is FALSE.

Example

```
DWORD dwResult;  
dwResult = WL_Disable();  
if(dwResult == 0)  
    AfxMessageBox(_T("Close wireless device fail"));
```

Requirements

OS Versions: Windows CE 5.0 and later.

Header: sysapi.h

Link Library: sysapi.lib

Link DLL: sysapi.dll

Device: PT60W

SCANAPIAX.DLL

We supply SCANAPIAX.DLL to allow programmer to control scan device status. There are several functions for user to use. User can use WINCE develop tool which like Visual Studio 2005 to develop application program to control scanner.

In this library, there are three different ways to control scanner module. These are API_SCAN, Scan2Key and Scanner related functions. Each related function can be used to control scanner module in different way. These three related functions can not be used at the same time. User should decide a suitable way to develop application. The following shows functions description.

API_SCAN Related Functions

User use API_SCAN related functions to register application to SCANAPIAX.dll. API_SCAN functions will send messages to report all activities, including error messages and scan data ready.

- [API_Register](#) – Register the application to SCANAPIAX.dll
- [API_Unregister](#) – Un-register the application from SCANAPIAX.dll
- [API_GetBarData](#) – Get barcode data into the buffer.
- [API_GetBarDataLength](#) – Return the scan data length.
- [API_GetBarType](#) – Return the barcode type.
- [API_GetError](#) – Get the error code.
- [API_GetSysError](#) – Return the system error code.
- [API_GoodRead](#) – Play sound and flash LED.
- [API_LoadSettingFromFile](#) – Loader scanner setting form file.
- [API_Reset](#) – Reset the scanner setting to default status.
- [API_ResetBarData](#) – Clear the data buffer that the next new scan data can come in.
- [API_SaveSettingToFile](#) – Save current scanner setting to file.
- [API_SaveSettingsToScanner](#) – Write the current scanner setting into scanner.
- [S2K_IsLoad](#) – Check the scan.exe is running or not.
- [S2K_Load](#) – Load or unload the scan.exe.
- [SCAN_QueryStatus](#) – Query scanner setting.
- [SCAN_SendCommand](#) – Send scanner command to change scanner status.
- [SCAN_ResumeSystem](#) – Enable/Disable scan key to resume system.
- [SCAN_BatchSetting](#) – Setup scanner in batch command.
- [SCAN_BatchRead](#) – Read scanner setting in batch command.
- [GetScannerDevice](#) – Get scanner engine type.

Scan2Key Related Functions

User use Scan2Key related functions to control scan.exe program. When scan.exe is loaded, scan data will send to key buffer. User application can be get scan data just like standard keyboard input.

- [PT_OpenScan2Key](#) – Execute scan.exe to scan barcode data into Terminal key buffer.
- [PT_CloseScan2Key](#) – Close scan.exe.
- [PT_SetToDefault](#) – Reset the scanner setting.

Scanner Related Functions

User use Scanner related functions to control scanner module without messages. When user use Scanner related functions, scan data will store in system buffer.

- [PT_EnableScanner](#) – Enable scanner to scan barcode.
- [PT_DisableScanner](#) – Disable scanner.
- [PT_CheckBarcodeData](#) – Check whether there is barcode data on system buffer.
- [PT_GetBarcodeData](#) – Get barcode data and type from system buffer.
- [PT_SetDefault](#) – Reset the scanner setting to default value.

Scan Key Related Functions

- [EnableTriggerKey](#) – Enable and disable scan key.
- [GetLibraryVersion](#) – Get the library version.
- [GetTriggerKeyStatus](#) – Get scan key status.
- [PressTriggerKey](#) – Trigger scan key.
- [TriggerKeyStatus](#) – Get scan key press status.

Scan Command Table

The scan command table of terminal is used for SCAN_QueryStatus and SCAN_SendCommand functions. The scan command provides the different way to setup scan settings.

When user wants to use this library, user should link SCANAPIAX.DLL, SCANAPIAX.LIB and the relate functions header file (SCANAPIAX.H).

■ API_SCAN Related Functions

API_Register

Register the application to SCANAPIAX.dll, so that SCANAPIAX.dll can communication with the application. It will also open scanner module to working mode.

```
BOOL API_Register
{
    HWND hwnd
}
```

Parameters

hwnd

[in] the window handle which library will send message to report all activities of scanner.

Return Values

Return TRUE if the operation is successful, otherwise return FALSE.

Remarks

The application must call API_Unregister to unregister from the dll and close scanner module after done with scanner. The messages can be one of the followings:

SM_DATA_READY : Indicates that scan data is successfully reading and ready to retrieve.

SM_ERROR_SYS : Indicates a system error, which is caused by calling system function. Call API_GetSysError to get the system error code.

SM_ERROR_API : Indicates an error. Call API_GetError to get error code.

Example

```
if(!API_Register(theApp.GetMainWnd()->m_hWnd))
    AfxMessageBox(_T("API_Register FAIL!!"));
```

Requirements

OS Versions: Windows CE 5.0 and later.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT60

API_Unregister

Unregister the application from SCANAPIAX.dll and close scanner module.

```
void API_Unregister  
{  
}
```

Parameters

None

Return Values

None.

Example

```
API_Unregister();
```

Requirements

OS Versions: Windows CE 5.0 and later.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT60

API_GetBarData

Get Barcode into the buffer. When you get the message SM_DATA_READY, call this function to get the barcode data.

```
UINT API_GetBarData
{
    LPBYTE buffer,
    UINT * uiLength,
    UINT * uiBarType
}
```

Parameters

buffer

[out] buffer for string scanned data.

uiLength

[in/out] buffer size

uiBarType

[out] barcode type

Return Values

Return 1 if the operation is successful, otherwise return 0.

Remarks

If the buffer size is less than scan data, function return 0 and the parameter *uiLength* return the size of the buffer to get barcode data.

Example

```
if(message == SM_DATA_READY){
    CString    strBarData,strBarType;
    UINtSize, uiType, i;
    char  *pBuf;

    uiSize = uiType = 0;
    API_GetBarData(NULL,  &uiSize, &uiType);
    if(uiSize == 0)
        strBarData = _T("No Data");
    else{
```

```
pBuf = (char *)new char[uiSize+1];
memset(pBuf, 0, uiSize+1);
API_GetBarData((LPBYTE)pBuf, &uiSize, &uiType);
strBarType.Format(_T("%d"), uiType);
for(i = 0 ; i < strlen(pBuf); i++)
    strBarData += *(pBuf+i);
}
AfxMessageBox( _T("Type:") + strBarType + _T("\r\nBarcode:") +
strBarData);
return 0;
}
```

Requirements

OS Versions: Windows CE 5.0 and later.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT60

API_GetBarDataLength

Get the scan data length.

```
UINT API_GetBarDataLength
{
}
```

Parameters

None

Return Values

Scan data length

Example

```
if(message == SM_DATA_READY){
    CString strData;
    UINT uiSize, uiType, i,uiLength;
    char *pBuf;
    uiLength = API_GetBarDataLength();
    if(uiLength == 0)
        strData = _T("No Data");
    else{
        uiSize = uiLength+1;
        pBuf = (char *)new char[uiSize];
        memset(pBuf, 0, uiSize);
        API_GetBarData((LPBYTE)pBuf, &uiSize, &uiType);
        for(i = 0 ; i < strlen(pBuf); i++)
            strData += *(pBuf+i);
    }
    AfxMessageBox(strData);
    return 0;
}
```

Requirements

OS Versions: Windows CE 5.0 and later.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT60

API_GetBarType

Get the barcode type.

```
UINT API_GetBarType
{
}
```

Parameters

None

Return Values

Return the barcode type

Remarks

value	Barcode	value	Barcode
BC_CODE11(100)	Code 11	BC_UPCA(113)	UPCA
BC_CODE39(101)	Code 39	BC_UPCE(114)	UPCE
BC_CODE93(102)	Code 93	BC_MATRIX_25(115)	Matrix 25
BC_CODE128(103)	Code 128	BC_PDF417(116)	PDF 417
BC_CODABAR(104)	Codabar	BC_CODE16K(117)	Code 16k
BC_EAN8(105)	EAN8	BC_CHINAPOST(118)	China Post
BC_EAN13(106)	EAN13	BC_RSS14(119)	RSS 14
BC_INDUSTRIAL_25(107)	Industrial 2 of 5	BC_RSS_LIMITED(120)	RSS Limited
BC_INTERLEAVED_25(108)	Interleaved 2 of 5	BC_RSS_EXPANDED(121)	RSS Expanded
BC_STANDARD_25(109)	Standard 2 of 5	BC_PHARMACODE39(122)	Pharama code39
BC_MSI_PLESSEY(110)	MSI Plessey	BC_MICRO_PDF(123)	Micro PDF
BC_UK_PLESSEY(111)	UK Plessey	BC_EANUCC(124)	UCC.EAN composite
BC_TELEPEN(112)	Telepen		

Example

```
uiType = API_GetBarType();
```

Requirements

OS Versions: Windows CE 5.0 and later.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT60

API_GetError

Get the error code.

```
DWORD API_GetError  
{  
}  
}
```

Parameters

None

Return Values

The return value can be one of the following table:

Constant	Value	Description
ERR_WRITE_FAIL	WM_USER+1	Send commands to scanner module failed.
ERR_SETTING_FAIL	WM_USER+2	Set scanner setting failed.
ERR_SCANNER_NOT_OPEN	WM_USER+3	Open scanner module failed.
ERR_INVALID_FILE	WM_USER+4	Invalid setting file.

Example

```
dwError = API_GetError();  
strMess.Format(_T("API Error Code: %d"), dwError);  
AfxMessageBox(strMess);
```

Requirements

OS Versions: Windows CE 5.0 and later.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT60

API_GetSysError

Get the system error code.

```
DWORD API_GetSysError  
{  
}  
}
```

Parameters

None

Return Values

Return the system error code that is returned by GetLastError(). The description of system error code can be find in MSDN.

Example

```
dwError = API_GetSysError();  
strMess.Format(_T("System Error Code: %d"), dwError);  
AfxMessageBox(strMess);
```

Requirements

OS Versions: Windows CE 5.0 and later.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT60

API_GoodRead

This function plays a sound when buzzer indication of scan module is enable and flashes the goodread LED when the LED indication of scan module is enable.

```
void API_GoodRead
{
}
```

Parameters

None

Return Values

None.

Remarks

Use *API_GoodRead()* to indicate user barcode data is scanned. The buzzer indication of scan module can be set by scan configuration program in control panel. The LED indication of scan module can be set by *SCAN_SendCommand()* function. If buzzer and LED indication are disable, the *API_GoodRead* will do nothing.

Example

```
API_GoodRead();
```

Requirements

OS Versions: Windows CE 5.0 and later.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT60

API_LoadSettingsFromFile

Load scanner setting from file.

```
BOOL API_LoadSettingsFromFile
{
    LPCTSTR filename
}
```

Parameters

filename

[in] the scanner setting file(*.axs)

Return Values

Return TRUE if the operation is successful, otherwise return FALSE.

Example

```
CString strFile;
CFileDialog    dlg(TRUE, NULL, NULL, OFN_FILEMUSTEXIST |
OFN_PATHMUSTEXIST);

if(dlg.DoModal() != IDOK)
    return;

strFile = dlg.GetPathName();
if(theApp.m_API_LoadSettingsFromFile(strFile))
    AfxMessageBox(_T("Load form file Succeed"));
else
    AfxMessageBox(_T("Load from file Fail"));
```

Requirements

OS Versions: Windows CE 5.0 and later.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT60

API_Reset

Reset the scanner setting to the default.

```
BOOL API_Reset  
{  
}  
}
```

Parameters

None

Return Values

Return TRUE if the operation is successful, otherwise return FALSE.

Example

```
if(API_Reset())  
    AfxMessageBox(_T("Reset Succeed"));  
else  
    AfxMessageBox(_T("Reset Fail"));
```

Requirements

OS Versions: Windows CE 5.0 and later.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT60

API_ResetBarData

Clear the data buffer that the next new scan data can come in.

```
void API_ResetBarData
{
}
```

Parameters

None

Return Values

None.

Example

```
API_ResetBarData();
```

Requirements

OS Versions: Windows CE 5.0 and later.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT60

API_SaveSettingsToFile

Save current scanner settings to file. The extension file name is “axs”.

```
BOOL API_SaveSettingsToFile
{
    LPCTSTR filename
}
```

Parameters

filename

[in] the file name for the setting file.

Return Values

Return TRUE if the operation is successful, otherwise return FALSE.

Example

```
CString strFile;
CFileDialog dlg(FALSE, _T("axs"), NULL, OFN_CREATEPROMPT, _T("Scanner
Settings Files (*.axs) |*.axs || "));

if(dlg.DoModal() != IDOK)
    return;

strFile = dlg.GetPathName();
if(API_SaveSettingsToFile(strFile))
    AfxMessageBox(_T("Save to file Succeed"));
else
    AfxMessageBox(_T("Save to file Fail"));
```

Requirements

OS Versions: Windows CE 5.0 and later.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT60

API_SaveSettingsToScanner

Write the current scanner setting into scanner.

```
BOOL API_SaveSettingsToScanner  
{  
}  
}
```

Parameters

None

Return Values

Return TRUE if the operation is successful, otherwise return FALSE.

Example

```
if(API_SaveSettingsToScanner())  
    AfxMessageBox(_T("Save to Scanner Succeed"));  
else  
    AfxMessageBox(_T("Save to Scannere Fail"));
```

Requirements

OS Versions: Windows CE 5.0 and later.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT60

S2K_IsLoad

Check the application scan.exe(scan barcode data into key buffer) is running.

```
BOOL S2K_IsLoad
{
}
```

Parameters

None

Return Values

The return value TRUE indicates that scan.exe is running. The return value FALSE indicates that scan.exe is not running.

Example

```
if(S2K_IsLoad()){
    AfxMessageBox(_T("scan.exe load"));
}
else
    AfxMessageBox(_T("scan.exe does not load"));
```

Requirements

OS Versions: Windows CE 5.0 and later.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT60

S2K_Load

Load or unload the scan.exe.

```
BOOL S2K_Load
{
    BOOL bLoad,
    DWORD dwTimeOut
}
```

Parameters

bLoad

[in] To set true to load scan.exe and false to unload scan.exe

dwTimeOut

[in] When unload scan.exe it will wait until the scan.exe closed or timeout by this parameter.

Return Values

Return TRUE if the operation is successful, otherwise return FALSE.

Example

```
if(S2K_Load(FALSE,1000)){
    AfxMessageBox(_T("unload scan.exe success"));
}
else
    AfxMessageBox(_T("unload scan.exe failed"));
```

Requirements

OS Versions: Windows CE 5.0 and later.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT60

SCAN_QueryStatus

Query current scanner setting.

```
BOOL SCAN_QueryStatus
{
    int nCommand1,
    int nCommand2,
    char *pReturn
}
```

Parameters

nCommand1

[in] See [scan command table](#).

nCommand2

[in] See [scan command table](#).

pReturn

[out] The current scanner setting. This buffer size must be large than 100.

Return Values

Return TRUE if the operation is successful, otherwise return FALSE.

Remarks

The *pReturn* value is depending on *nCommand1* and *nCommand2*. The *nCommand1* and *nCommand2* decide which scanner setting to be queried.

Example

```
char      *pValue;
pValue = (char *)new char[100];
memset(pValue, 0, 100);
//query Buzzer indication setting
SCAN_QueryStatus(5, 3, pValue);
```

Requirements

OS Versions: Windows CE 5.0 and later.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT60

SCAN_SendCommand

Send scanner command to change scanner status.

```
BOOL SCAN_SendCommand
{
    int nCommand1,
    int nCommand2,
    char *pValue
}
```

Parameters

nCommand1

[in] See [scan command table](#).

nCommand2

[in] See [scan command table](#).

pValue

[in] See [scan command table](#).

Return Values

Return TRUE if the operation is successful, otherwise return FALSE.

Example

```
//Enable Buzzer indication setting
if(SCAN_SendCommand(5, 3, "1"))
    AfxMessageBox(_T("Setup complete"));
else
    AfxMessageBox(_T("Setup false"));
```

Requirements

OS Versions: Windows CE 5.0 and later.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT60

SCAN_ResumeSystem

Enable/Disable scan key to resume system

```
DWORD SCAN_ResumeSystem
{
    BOOL bOn
}
```

Parameters

bOn
[in]

Return Values

Return 0 if function succeed; return the system error code that is returned by GetLastError(). The description of system error code can be find in MSDN..

Example

```
//Enable scan key to resume system
if(SCAN_ResumeSystem(1) == 0)
    AfxMessageBox(_T("Enable scan key to resume system succeed"));
else
    AfxMessageBox(_T("Enable scan key to resume system fail"));
```

Requirements

OS Versions: Windows CE 5.0 and later.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT60

SCAN_BatchSetting

Setup scanner in batch command

```
DWORD SCAN_BatchSetting
{
    ScannerSetting setting
}
```

Parameters

setting

[in] The scanner setting configure. The barcode setting refer to [scan command table](#).

Return Values

Return: FALSE if the operation is success, otherwise return SM_ERROR_SYSTEM, SM_ERROR_API,

Example

```
ScannerSetting    setting;
setting.Code11.m_uiRead = 1;
setting.Code39.m_uiRead = 1;
.....
SCAN_BatchSetting(setting);
```

Requirements

OS Versions: Windows CE 5.0 and later.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT60

SCAN_BatchRead

Read scanner setting in batch command

```
DWORD SCAN_BatchRead  
{  
    ScannerSetting *setting  
}
```

Parameters

setting

[in] The buffer is to receive scanner setting configure. The barcode setting refer to [scan command table](#).

Return Values

Return FALSE if the operation is success, ERR_SCANNER_NOT_OPEN if the scanner is not opened.

Example

```
ScannerSetting    setting;  
SCAN_BatchRead(&setting);
```

Requirements

OS Versions: Windows CE 5.0 and later.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT60

GetScannerDevice

Get scanner device type

```
DWORD GetScannerDevice  
{  
}  
}
```

Parameters

None

Return Values

Return FALSE if the scanner device is not ready

1 → 1D scanner engine

2 → 2D scan engine

Example

```
int nDevice;  
nDevice = GetScannerDevice();  
If(nDevice == 1){ // 1D scanner  
}  
Else if(nDevice == 2){ // 2D scanner  
}
```

Requirements

OS Versions: Windows CE 5.0 and later.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT60

Scan2Key Related Functions

PT_OpenScan2Key

Execute scan.exe to scan barcode data into Terminal key buffer.

```
BOOL PT_OpenScan2Key  
{  
}
```

Parameters

None

Return Values

Return TRUE if the operation is successful, otherwise return FALSE.

Example

```
BOOL bResult;  
bResult = PT_OpenScan2Key();  
if(!bResult)  
    AfxMessageBox(_T("PT_OpenScan2Key fail"));
```

Requirements

OS Versions: Windows CE 5.0 and later.

Header: scanapi.h

Link Library: scanapi.lib

Link DLL: scanapi.dll

Device: PT60

PT_CloseScan2Key

Close scan.exe.

```
void PT_CloseScan2Key
{
}
```

Parameters

None

Return Values

None.

Example

PT_CloseScan2Key()

Requirements

OS Versions: Windows CE 5.0 and later.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT60

PT_SetToDefault

Reset the scanner setting. All scanner setting will reset to default value.

```
int PT_SetToDefault
{
}
```

Parameters

None

Return Values

Return 1 if the operation is successful, otherwise return 0.

Example

```
if(!PT_SetToDefault())
    AfxMessageBox(_T("PT_SetToDefault fail"));
```

Requirements

OS Versions: Windows CE 5.0 and later.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT60

Scanner Related Functions

PT_EnableScanner

Enable scanner to scan barcode. This function creates a thread to get scan data from scanner module and store scan data in the system buffer. Application can use function call `PT_GetBarcodeData` to get scan data from system buffer.

```
int PT_EnableScanner
{
}
```

Parameters

None

Return Values

Return 0 if the operation is successful, otherwise return 1.

Example

```
if(PT_EnableScanner())
    AfxMessageBox(_T("PT_EnableScanner fail"));
```

Requirements

OS Versions: Windows CE 5.0 and later.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT60

PT_DisableScanner

This function will close scanner module.

```
void PT_DisableScanner
{
}
```

Parameters

None

Return Values

None.

Example

```
PT_DisableScanner();
```

Requirements

OS Versions: Windows CE 5.0 and later.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT60

PT_CheckBarcodeData

Check whether there is available barcode data in system buffer.

```
BOOL PT_CheckBarcodeData  
{  
}  
}
```

Parameters

None

Return Values

This function returns TRUE if there are barcode data in system buffer. This function returns FALSE if there are no barcode data in system buffer.

Example

```
if(PT_CheckBarcodeData())  
    m_strScanData = _T("There are barcode data in system buffer");  
else  
    m_strScanData = _T("There are no barcode data in system buffer ");
```

Requirements

OS Versions: Windows CE 5.0 and later.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT60

PT_GetBarcodeData

Get Barcode data and type from system buffer.

```
BOOL PT_GetBarcodeData
{
    UINT * uiBarType,
    Char * pBuffer,
    UINT * uiMaxBufferLen
}
```

Parameters

uiBarType

[out] barcode type.

pBuffer

[out] buffer for storing scanned data..

uiMaxBufferLen

[in/out] The max buffer size

Return Values

Return TRUE if the operation is successful, otherwise return FALSE.

Remarks

If the buffer size is less than scan data, function return 0 and the parameter *uiMaxBufferLen* return the size of barcode data.

Example

```
if(PT_CheckBarcodeData()){
    if(PT_GetBarcodeData(&uiBarType, pBarData, &uiMaxLen)){
        for(i = 0 ; i < strlen(pBarData) ; i++)
            m_strScanData += *(pBarData + i);
    }
    else
        m_strScanData = _T("Can't get scan data");
}
else
    m_strScanData = _T("No Scan Data");
```

Requirements

OS Versions: Windows CE 5.0 and later.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT60

PT_SetDefault

Reset the scanner setting to default value.

```
BOOL PT_SetDefault  
{  
}
```

Parameters

None

Return Values

Return TRUE if the operation is successful, otherwise return FALSE.

Example

```
if(PT_SetDefault())  
    AfxMessageBox(_T("PT_SetDefault succeed"));  
else  
    AfxMessageBox(_T("PT_SetDefault fail"));
```

Requirements

OS Versions: Windows CE 5.0 and later.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT60

Scan Key Related Functions

EnableTriggerKey

This function will enable or disable scan key.

```
DWORD EnableTriggerKey
{
    BOOL bEnable
}
```

Parameters

bEnable

[in] Flag that indicates whether enable scan key(TRUE) or disable scan key(FALSE).

Return Values

If function succeeds, the return value is [E_FUNC_SUCCEEDED](#). If function fails, the return value is [E_FUNC_ERROR](#), [E_FUNC_PAR_ERROR](#).

Remarks

This function is meaningful only if scanner is opened. The warm reset will enable scan key automatically.

Example

```
BOOL bResult;
bResult = EnableTriggerKey(TRUE);
if(bResult)
    AfxMessageBox(_T("EnableTriggerKey Succeed"));
Else
    AfxMessageBox(_T("EnableTriggerKey Fail"));
```

Requirements

OS Versions: Windows CE 5.0 and later.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT60

GetLibraryVersion

Get library version number.

```
int GetLibraryVersion
{
}
```

Parameters

None

Return Values

The version number. If the return value is 101, it means that dll version is 1.01

Example

```
int nVersion;
CString strTemp;
nVersion = GetLibraryVersion();
strTemp.Format(_T("Version = %d"), nVersion);
AfxMessageBox(strTemp);
```

Requirements

OS Versions: Windows CE 5.0 and later.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT60

GetTriggerKeyStatus

This function will get scan key status.

```
DWORD GetTriggerKeyStatus  
{  
}  
}
```

Parameters

None.

Return Values

The return value 1 indicates that scan key is enable. The return value 0 indicates that scan key is disable.

Example

```
if(GetTriggerKeyStatus())  
    AfxMessageBox(_T("scan key ensable!"));  
else  
    AfxMessageBox(_T("scan key disable!"));
```

Requirements

OS Versions: Windows CE 5.0 and later.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT60

PressTriggerKey

This function will trigger scan key.

```
DWORD PressTriggerKey
{
    BOOL bPress
}
```

Parameters

bPress

[in] Flag that indicates whether press scan key(TRUE) or release scan key(FALSE).

Return Values

If function succeeds, the return value is [E_FUNC_SUCCEED](#). If function fails, the return value is [E_FUNC_ERROR](#).

Remarks

This function is meaningful only if scanner is opened.

Example

```
PressTriggerKey(TRUE);
Sleep(1000);
PressTriggerKey(FALSE);
```

Requirements

OS Versions: Windows CE 5.0 and later.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT60

TriggerKeyStatus

This function will get scan key press status.

```
DWORD TriggerKeyStatus  
{  
}  
}
```

Parameters

None.

Return Values

The return value 1 indicates that scan key is pressed. The return value 0 indicates that scan key is released.

Example

```
if(TriggerKeyStatus())  
    AfxMessageBox(_T("scan key pressed!"));  
else  
    AfxMessageBox(_T("scan key release!"));
```

Requirements

OS Versions: Windows CE 5.0 and later.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT60

Scan Command Table

Command1	Command2	Value
5 Indication	2 LED indication	0: Disable 1: Enable(*)
	3 Buzzer indication	0: Disable 1: Enable(*)
6 Transmission	7 Code ID position	0: Before code data(*) 1: After code data
	8 Code ID transmission	0: Disable(*) 1: Proprietary ID 2: AIM ID
	9 Code length transmission	0: Disable(*) 1: Enable
	10 Code name transmission	0: Disable(*) 1: Enable
	11 Case conversion	0: Disable(*) 1: Upper case 2: Lower case
7 Scan	4 Double confirm	0 ~ 9(default:0)
	6 Global min. code length	0 ~ 64(default:4)
	7 Global max. code length	0 ~ 64(default:63)
	8 Inverted image scan	0: Disable(*) 1: Enable
8 String setting	2 Suffix characters setting	0x00 ~ 0xff ASCII code(default: 0x0D) 22 characters.
	3 Preamble characters settings	0x00 ~ 0xff ASCII code(default: NULL) 22 characters.
	4 Postamble characters settings	0x00 ~ 0xff ASCII code(default: NULL) 22 characters.

10 Code 11	1 Read	0: Disable(*) 1: Enable
	2 Check-sum verification	0: Disable 1: One digit(*) 2: Two digits
	3 Check-sum transmission	0: Disable(*) 1: Enable
	4 Max. code length	0 ~ 64(default: 0)
	5 Min. code length	0 ~ 64(default: 0)
	6 Truncate leading	0 ~ 15(default: 0)
	7 Truncate ending	0 ~ 15(default: 0)
	8 Code ID setting	0x00 ~ 0xff ASCII code(1 or 2 bytes) (default: 0)
11 Code 39	1 Read	0: Disable 1: Enable(*)
	2 Check-sum verification	0: Disable(*) 1: Enable
	3 Check-sum transmission	0: Disable(*) 1: Enable
	4 Max. code length	0 ~ 64(default: 0)
	5 Min. code length	0 ~ 64(default: 0)
	6 Truncate leading	0 ~ 20(default: 0) 20: Truncate characters before space
	7 Truncate ending	0 ~ 15(default: 0)
	8 Code ID setting	0x00 ~ 0xff ASCII code(1 or 2 bytes) (default: *)
	10 Format	0: Standard(*) 1: Full ASCII
	13 Start/stop transmission	0: Disable(*) 1: Enable

12 Code 93	1 Read	0: Disable(*) 1: Enable
	2 Check-sum verification	0: Disable 1: Enable(*)
	3 Check-sum transmission	0: Disable(*) 1: Enable
	4 Max. code length	0 ~ 64(default: 0)
	5 Min. code length	0 ~ 64(default: 0)
	6 Truncate leading	0 ~ 15(default: 0)
	7 Truncate ending	0 ~ 15(default: 0)
	8 Code ID setting	0x00 ~ 0xff ASCII code(1 or 2 bytes) (default: &)
13 Code 128	1 Read	0: Disable 1: Enable(*)
	2 Check-sum verification	0: Disable 1: Enable(*)
	3 Check-sum transmission	0: Disable(*) 1: Enable
	4 Max. code length	0 ~ 64(default: 0)
	5 Min. code length	0 ~ 64(default: 0)
	6 Truncate leading	0 ~ 15(default: 0)
	7 Truncate ending	0 ~ 15(default: 0)
	8 Code ID setting	0x00 ~ 0xff ASCII code(1 or 2 bytes) (default: #)
	10 Format	0: Standard(*) 1: UCC.EAN 128
	12 UCC/EAN 128 ID setting	0x00 ~ 0xff ASCII code(1 bytes) (default: #)
	13 Concatenation code	0x00 ~ 0xff ASCII code(1 bytes) (default: 0x1D)

14 Codabar	1 Read	0: Disable(*) 1: Enable
	2 Check-sum verification	0: Disable(*) 1: Enable
	3 Check-sum transmission	0: Disable(*) 1: Enable
	4 Max. code length	0 ~ 64(default: 0)
	5 Min. code length	0 ~ 64(default: 0)
	6 Truncate leading	0 ~ 15(default: 0)
	7 Truncate ending	0 ~ 15(default: 0)
	8 Code ID setting	0x00 ~ 0xff ASCII code(1 or 2 bytes) (default: %)
	10 Start/stop type	0: ABCD/ABCD(*) 1: abcd/abcd 2: ABCD/TN*E 3: abcd/tn*e
	11 Start/stop transmission	0: Disable(*) 1: Enable
15 EAN 8	1 Read	0: Disable 1: Enable(*)
	3 Check-sum transmission	0: Disable 1: Enable(*)
	6 Truncate leading	0 ~ 15(default: 0)
	7 Truncate ending	0 ~ 15(default: 0)
	8 Code ID setting	0x00 ~ 0xff ASCII code(1 or 2 bytes) (default: FF)
	10 Supplement digits	0: None(*) 1: 2 digits 2: 5 digits 3: 2, 5 digits 4: UCC/EAN 128 5: 2, UCC/EAN 128

		6: 5, UCC/EAN 128 7: All
	11 Truncation/expansion	0: None(*) 1: Truncate leading zero 2: Expand to EAN 13
	12 Expansion	0: Disable(*) 1: Enable
16 EAN 13	1 Read	0: Disable 1: Enable(*)
	3 Check-sum transmission	0: Disable 1: Enable(*)
	6 Truncate leading	0 ~ 15(default: 0)
	7 Truncate ending	0 ~ 15(default: 0)
	8 Code ID setting	0x00 ~ 0xff ASCII code(1 or 2 bytes) (default: F)
	10 Supplement digits	0: None(*) 1: 2 digits 2: 5 digits 3: 2, 5 digits 4: UCC/EAN 128 5: 2, UCC/EAN 128 6: 5, UCC/EAN 128 7: All
	12 ISBN/ISSN conversion	0: Disable(*) 1: Enable
17 Industrial 2 of 5	1 Read	0:Disable(*) 1:Enable
	4 Max. code length	0 ~ 64(default: 0)
	5 Min. code length	0 ~ 64(default: 0)
	6 Truncate leading	0 ~ 15(default: 0)
	7 Truncate ending	0 ~ 15(default: 0)
	8	0x00 ~ 0xff ASCII code(1 or 2 bytes)

	Code ID setting	(default: i)
18 Interleaved 2 of 5	1 Read	0: Disable 1: Enable(*)
	2 Check-sum verification	0: Disable(*) 1: Enable
	3 Check-sum transmission	0: Disable(*) 1: Enable
	4 Max. code length	0 ~ 64(default: 0)
	5 Min. code length	0 ~ 64(default: 0)
	6 Truncate leading	0 ~ 15(default: 0)
	7 Truncate ending	0 ~ 15(default: 0)
	8 Code ID setting	0x00 ~ 0xff ASCII code(1 or 2 bytes) (default: i)
19 Standard 2 of 5	1 Read	0: Disable(*) 1: Enable
	2 Check-sum verification	0: Disable(*) 1: Enable
	3 Check-sum transmission	0: Disable(*) 1: Enable
	4 Max. code length	0 ~ 64(default: 0)
	5 Min. code length	0 ~ 64(default: 0)
	6 Truncate leading	0 ~ 15(default: 0)
	7 Truncate ending	0 ~ 15(default: 0)
	8 Code ID setting	0x00 ~ 0xff ASCII code(1 or 2 bytes) (default: i)
20 MSI Plessey	1 Read	0: Disable(*) 1: Enable
	2 Check-sum verification	0: Disable 1: Mod 10(*) 2: Mod 10/10

		3: Mod 11/10
	3 Check-sum transmission	0: Disable(*) 1: Enable
	4 Max. code length	0 ~ 64(default: 0)
	5 Min. code length	0 ~ 64(default: 0)
	6 Truncate leading	0 ~ 15(default: 0)
	7 Truncate ending	0 ~ 15(default: 0)
	8 Code ID setting	0x00 ~ 0xff ASCII code(1 or 2 bytes) (default: @)
21 UK Plessey	1 Read	0: Disable(*) 1: Enable
	2 Check-sum verification	0: Disable 1: Enable(*)
	3 Check-sum transmission	0: Disable(*) 1: Enable
	4 Max. code length	0 ~ 64(default: 0)
	5 Min. code length	0 ~ 64(default: 0)
	6 Truncate leading	0 ~ 15(default: 0)
	7 Truncate ending	0 ~ 15(default: 0)
	8 Code ID setting	0x00 ~ 0xff ASCII code(1 or 2 bytes) (default: @)
22 Telepen	1 Read	0: Disable(*) 1: Enable
	2 Check-sum verification	0: Disable(*) 1: Enable
	3 Check-sum transmission	0: Disable(*) 1: Enable
	4 Max. code length	0 ~ 64(default: 0)
	5 Min. code length	0 ~ 64(default: 0)

	Min. code length	
	6 Truncate leading	0 ~ 15(default: 0)
	7 Truncate ending	0 ~ 15(default: 0)
	8 Code ID setting	0x00 ~ 0xff ASCII code(1 or 2 bytes) (default: S)
	10 Format	0: Numeric(*) 1: Full ASCII
23 UPCA	1 Read	0: Disable 1: Enable(*)
	3 Check-sum transmission	0: Disable 1: Enable(*)
	6 Truncate leading	0 ~ 15(default: 0)
	7 Truncate ending	0 ~ 15(default: 0)
	8 Code ID setting	0x00 ~ 0xff ASCII code(1 or 2 bytes) (default: A)
	10 Supplement digits	0: None(*) 1: 2 digits 2: 5 digits 3: 2, 5 digits 4: UCC/EAN 128 5: 2, UCC/EAN 128 6: 5, UCC/EAN 128 7: All
	11 Truncate/expansion	0: None 1: Truncate leading zero(*) 2: Expand to EAN 13
24 UPCE	1 Read	0: Disable 1: Enable(*)
	3 Check-sum transmission	0: Disable 1: Enable(*)
	6 Truncate leading	0 ~ 15(default: 0)
	7 Truncate ending	0 ~ 15(default: 0)

	8 Code ID setting	0x00 ~ 0xff ASCII code(1 or 2 bytes) (default: E)
	10 Supplement digits	0: None(*) 1: 2 digits 2: 5 digits 3: 2, 5 digits 4: UCC/EAN 128 5: 2, UCC/EAN 128 6: 5, UCC/EAN 128 7: All
	11 Truncate/expansion	0: None(*) 1: Truncate leading zero 2: Expand to EAN 13 3: Expand to UPCA
	12 Expansion	0: Disable(*) 1: Enable
	13 UPCE-1	0: Disable(*) 1: Enable
25 Matrix 25	1 Read	0: Disable(*) 1: Enable
	2 Check-sum verification	0: Disable(*) 1: Enable
	3 Check-sum transmission	0: Disable(*) 1: Enable
	4 Max. code length	0 ~ 64(default: 0)
	5 Min. code length	0 ~ 64(default: 0)
	6 Truncate leading	0 ~ 15(default: 0)
	7 Truncate ending	0 ~ 15(default: 0)
	8 Code ID setting	0x00 ~ 0xff ASCII code(1 or 2 bytes) (default: B)
26 PDF-417 (2D mode only)	1 Read	0: Disable 1: Enable(*)
	6 Truncate leading	0 ~ 15(default: 0)

	7 Truncate ending	0 ~ 15(default: 0)
	8 Code ID setting	0x00 ~ 0xff ASCII code(1 or 2 bytes) (default: NULL)
	10 Escape sequence Transmit	0: Disable(*) 1: Enable
27 Code-16K (2D model only)	1 Read	0: Disable(*) 1: Enable
	6 Truncate leading	0 ~ 15(default: 0)
	7 Truncate ending	0 ~ 15(default: 0)
	8 Code ID setting	0x00 ~ 0xff ASCII code(1 or 2 bytes) (default: NULL)
28 China post	1 Read	0: Disable(*) 1: Enable
	4 Max. code length	0 ~ 64(default: 11)
	5 Min. code length	0 ~ 64(default: 11)
	6 Truncate leading	0 ~ 15(default: 0)
	7 Truncate ending	0 ~ 15(default: 0)
	8 Code ID setting	0x00 ~ 0xff ASCII code(1 or 2 bytes) (default: t)
29 RSS 14	1 Read	0: Disable(*) 1: Enable
	6 Truncate leading	0 ~ 15(default: 0)
	7 Truncate ending	0 ~ 15(default: 0)
	8 Code ID setting	0x00 ~ 0xff ASCII code(1 or 2 bytes) (default: R4)
	11 UCC/EAN 128 emulation	0: Disable(*) 1: Enable
30	1	0: Disable(*)

RSS Limited	Read	1: Enable
	6 Truncate leading	0 ~ 15(default: 0)
	7 Truncate ending	0 ~ 15(default: 0)
	8 Code ID setting	0x00 ~ 0xff ASCII code(1 or 2 bytes) (default: RL)
	11 UCC/EAN 128 emulation	0: Disable(*) 1: Enable
31 RSS Expanded	1 Read	0: Disable(*) 1: Enable
	4 Max. code length	0 ~ 99(default: 99)
	5 Min. code length	0 ~ 99(default: 1)
	6 Truncate leading	0 ~ 15(default: 0)
	7 Truncate ending	0 ~ 15(default: 0)
	8 Code ID setting	0x00 ~ 0xff ASCII code(1 or 2 bytes) (default: RX)
	11 UCC/EAN 128 emulation	0: Disable(*) 1: Enable
32 Italian Pharmacode 39	1 Read	0: Disable(*) 1: Enable
	4 Max. code length	0 ~ 64(default: 12)
	5 Min. code length	0 ~ 64(default: 9)
	6 Truncate leading	0 ~ 15(default: 0)
	7 Truncate ending	0 ~ 15(default: 0)
	8 Code ID setting	0x00 ~ 0xff ASCII code(1 or 2 bytes) (default: p)
	10 Leading "A"	0: Disable(*) 1: Enable
33	1	0: Disable(*)

MicroPDF (2D model only)	Read	1: Enable
	6 Truncate leading	0 ~ 15(default: 0)
	7 Truncate ending	0 ~ 15(default: 0)
	8 Code ID setting	0x00 ~ 0xff ASCII code(1 or 2 bytes) (default: U)
	10 Escape sequence transmit	0: None 1: GLI protocol 2: ECI protocol(*)
34 EAN.UCC composite (2D model only)	1 Read	0: Disable(*) 1: Enable
	6 Truncate leading	0 ~ 15(default: 0)
	7 Truncate ending	0 ~ 15(default: 0)
	8 Code ID setting	0x00 ~ 0xff ASCII code(1 or 2 bytes) (default: RC)
	11 UCC/EAN 128 emulation	0: Disable(*) 1: Enable

Function Return Values

Constant	Value	Description
E_FUNC_SUCCEED	0x00000000	The function returned without error.
E_FUNC_ERROR	0x00000001	The function returned error.
E_FUNC_NULLPTR	0x00000002	A null pointer was passed to the function.
E_FUNC_PAR_ERROR	0x00000003	A invalid parameter was passed to the function.
E_SCAN_FUNC_SUCCEED	0x00000000	Scanner work succeed
E_SCAN_INVALID_HANDLE	0x00000001	Scanner not ready maybe other application using
E_SCAN_SETTING_FAIL	0x00000003	Setting scanner fail
E_SCAN_INSUFFICIENT_BUFFER	0x00000004	The buffer to receive scan data is to small
E_SCAN_NODATA	0x00000005	There is no scan data
E_SCAN_TIMEOUT	0x00000006	Get scan data timeout