ARGOX
Empower the Barcode

PT-9130 Mobile Computer
SDK Programming Manual

DOC NO. UM-PT905-01
Sep. 2012
Version 2.0

http://www.argox.com

Table of Contents

OVERVIEW **2**

SDK FUNCTIONS **3**

SYSAPIAX.DLL **4**

AUDIO RELATED FUNCTIONS 9

Audio_GetVolume *9*

Audio_SetVolume *10*

BATTERY RELATED FUNCTION 11

GetBatteryStatus *11*

DISPLAY RELATED FUNCTIONS 13

BacklightOn *13*

Display_QueryBacklightIntensity *14*

GetBacklightStatus *16*

PowerOnLCD *17*

SetBacklightPWM *18*

EnableTouchPanel *19*

GetTouchPanelStatus *20*

KEYPAD RELATED FUNCTIONS 21

EnablePowerButton *21*

GetKeypadAlphaMode *22*

SendKbdVirtualKey *23*

SetKeypadAlphaMode *24*

LED RELATED FUNCTIONS 25

GetKeypadLEDStatus *25*

GoodReadLEDOn *26*

KeypadLEDOn *27*

QueryKeypadLEDIntensity *28*

SetKeypadPWM *30*

SYSTEM RELATED FUNCTIONS 31

CallSuspend *31*

EnableAutoConnect *32*

RegisterAlphaKeyNotification *33*

ShowChineseIME *34*

ShowDesktop *35*

ShowExploreToolbar *36*

ShowTaskbar *37*

UnRegisterAlphaKeyNotification 38
RegisterTriggerKeyNotification 39
UnregisterTriggerKeyNotification 40
VIBRATOR RELATED FUNCTIONS 41
VibratorOn 41
WLAN RELATED FUNCTION 42
WL_Enable 42
WL_Disable 43
BLUETOOTH RELATED FUNCTION 44
BT_On 44
BT_Off 45
SetDiscoverMode 46
GetDiscoverMode 47
SetSPPService 48
GetSPPService 49
SetFTPService 50
GetFTPService 51
SetFTPWriteable 52
GetFTPWriteable 53
SetFTPShareFolder 54
GetFTPShareFolder 55
InitSearchBTDevice 56
FindNextBTDevice 57
EndSearchBTDevice 59
InitSearchFTPDevice 60
FindFirstFTPDevice 61
FindNextFTPDevice 63
PairDevice 65
UnPairDevice 66
GetComInfo 67
ConnectDevice 69
GetConnectStatus 70
GetSPPClientChannel 71
FindFirstFTPFile 72
FindNextFTPFile 73
GetFTPFile 74
PutFTPFile 75
CreateFTPFolder 76
DeleteFTPFolder 77

DeleteFTPFile 78
CAMERA RELATED FUNCTION 79
Camera_On 79
Camera_Off 80
SetPreviewSize 81
GetPreviewSize 82
EnablePreview 83
SetStillCaptureSize 84
GetStillCaptureSize 85
StartStillCapture 86
SetFlash 87
GetFlash 88
SetDarkMode 89
GetDarkMode 90
SetInvert 91
GetInvert 92
GPRS RELATED FUNCTION 93
GPRS_On 93
GPRS_Off 94
GetGPRSPowerStatus 95
SendATCommand 96
GetATResponse 97
GetIMEINumber 98
GetIMSINumber 99
GetModuleInfo 100
GetSignalQuality 101
ConnectRAS 102
DisconnectRAS 103
GetRASConnStatus 104
CreateRASEntry 105
DeleteRASEntry 106
ChangeRASEntryName 107
ChangeRASEntryProperty 108
GetRASEntryProperty 109
GetRASEntryCount 110
MenuRASEntries 111
GetPINCounter 112
GetPINStatus 113
SetPINLock 114

GetPINLockStatus 115
CheckPINCode 116
CheckPUKCode 117
ChangePINCode 118
ReadPhonebook 119
WritePhonebook 120
DeletePhonebook 121
ReadMultiPhonebook 122
GetPhonebookTotal 124
GetPhoneMaxLength 125
SendSMS 126
ReadSMS 127
DeleteSMS 129
ReadMultiSMS 130
WriteStorageSMS 132
SendStorageSMS 133
GetSMSTotal 134
SetSMSCentre 135
GetSMSCentre 136
SMS_Register 137
SMS_UnRegister 138
GPS 139
BLUETOOTH STRUCTURE 140
CONNECT_INFO Structure 140
FTP_FILE Structure 141
GPRS STRUCTURE 142
RAS_ENTRY Structure 142
PHONEBOOK_INFO Structure 144
SMS_INFO Structure 145

RFID.DLL 146

RFID RELATED FUNCTION 148
OpenComPort 148
CloseComPort 149
GetFWVersion 150
APIVersion 151
SetWorkingType 152
AntennaContro 153
Inventory15693 154

ReadSingleBlock *155*
WriteSingleBlock *156*
WriteAFI *157*
WriteDSFID *158*
LockAFI *159*
LockDSFID *160*
LockBlock *161*
TagSystemInfo *162*
TagStayQuiet *163*
TagSelect *164*
TagResetToReady *165*
OpenCard14443A *166*
ReadMifareBlock *167*
WriteMifareBlock *168*
ReadUltraLightBlock *169*
WriteUltraLightBlock *170*
GetUid14443B *171*
SRIX4KChipID *172*
SRIX4KReadBlock *173*
SRIX4KWriteBlock *174*
RFID REQUEST FLAGS 175
SRIX4K FLOW CHART 176

SCANAPIAX.DLL 177

API_SCAN RELATED FUNCTIONS 180
API_Register *180*
API_Unregister *181*
API_GetBarData *182*
API_GetBarDataLength *184*
API_GetBarType *185*
API_GetError *186*
API_GetSysError *187*
API_GoodRead *188*
API_LoadSettingsFromFile *189*
API_Reset *190*
API_ResetBarData *191*
API_SaveSettingsToFile *192*
API_SaveSettingsToScanner *193*
S2K_IsLoad *194*

S2K_Load *195*
SCAN_QueryStatus *196*
SCAN_SendCommand *197*
SCAN_ResumeSystem *198*
SCAN_BatchSetting *199*
SCAN_BatchRead *200*
SCAN2KEY RELATED FUNCTIONS 201
PT_OpenScan2Key *201*
PT_CloseScan2Key *202*
PT_SetToDefault *203*
SCANNER RELATED FUNCTIONS 204
PT_EnableScanner *204*
PT_DisableScanner *205*
PT_CheckBarcodeData *206*
PT_GetBarcodeData *207*
PT_SetDefault *209*
SCAN KEY RELATED FUNCTIONS 210
EnableTriggerKey *210*
GetLibraryVersion *211*
GetTriggerKeyStatus *212*
PressTriggerKey *213*
TriggerStatus *214*
SCAN STRUCTURE 215
ScannerSetting Structure *215*
GeneralSetting Structure *218*
Code11_Setting Structure *219*
Code39_Setting Structure *220*
Code93_Setting Structure *221*
Code128_Setting Structure *222*
Codabar_Setting Structure *223*
EAN8_Setting Structure *224*
EAN13_Setting Structure *225*
Industrial25_Setting Structure *226*
Interleaved25_Setting Structure *227*
MSI_Setting Structure *228*
UK_Setting Structure *229*
Telepen_Setting Structure *230*
UPCA_Setting Structure *231*
UPCE_Setting Structure *232*

Matrix25_Setting Structure *233*

UEGeneral_Setting Structure *234*

IATA25_Setting Structure *235*

Trioptic_Setting Structure *236*

RSS_Setting Structure *237*

SCAN COMMAND TABLE **238**

FUNCTION RETURN VALUES **246**

Overview

The ***Argox*** PT-90 Mobile Computer Software Developer Kit (SDK) Programming Manual is prepared to assist programmers on developing application programs using ***Argox*** PT-90 Mobile Computers under Microsoft® Windows® CE6.0 Operating System. It gives all the details needed to call functional subroutines controling the devices on the ***Argox*** PT-90 Mobile Computer or access value-added devices on board such as Scanning and Wireless module.

This Programming Manual is organized as two major sections, one for the system related functions and the other for value-added scanning functions with the following information:

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

  Audio

  Display

  Keypad

  Led and Vibrator Indicators

  Battery Status

  System Settings

  Bluetooth

  WLAN

- Argox Scanning module Application Programming Interface (API) Definitions

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

SDK Functions

When using SDK to develop their own application program, the programmer should link DLL file or LIB file, then, include header file SYSAPIAX.H.
The following two examples are given to show how to use LIB file and DLL file while developing an application program. We will use ***Visual Studio 2005*** to illustrate.

Example 1: Using LIB file.

First, programmer should include sysapiax.lib in the application 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 PT-90 SDK, we provide SYSAPIAX.DLL which includes several functions to allow programmer to control device drivers and system functions. Programmer can use WINCE develop tool like ***Visual Studio 2005*** to develop application programs. Descriptions of all these functions are given below.

Audio Related Functions

- Audio_GetVolume – Query current volume setting.
- Audio_SetVolume – Set level of audio volume.

Battery Related Function

- GetBatteryStatus – Gets main battery status.

Display Related Functions

- BacklightOn – Turn ON or OFF screen backlight.
- Display_QueryBacklightIntensity – Query back-light intensity.
- GetBacklightStatus – Gets screen backlight status.
- PowerOnLCD – Turn ON or OFF the power of LCD.
- SetBacklightPWM – Adjusts screen back-light brightness.
- EnableTouchPanel – ENABLE or DISABLE touch panel.
- GetTouchPanelStatus – Get touch panel status.

KeyPad Related Functions

- EnablePowerButton – ENABLE or DISABLE Power button.
- GetAlphaMode – Get the current keypad input MODE.
- SendKbdVisualKey – Sends a virtual key to key buffer.
- SetAlphaMode – Change keypad input MODE.

LED Related Functions

- GetKeypadLEDStatus – Gets keypad backlight LED status.
- GoodReadLEDOn – Turn ON or OFF good read LED.
- KeypadLEDOn – Turn ON or OFF keypad backlight LED.
- QueryKeypadLEDIntensity – Query keypad backlight LED brightness.
- SetKeypadPWM – Adjusts keypad backlight LED brightness.

System Related Functions

- CallSuspend – Enter SUSPEND mode.
- EnableAutoConnect – Turn auto-connect ON or OFF.
- RegisterAlphaKeyNotification – Register a request to send a prompt message when the ALPHA key is pressed.
- ShowChineseIME – DISPLAY or HIDE the Chinese IME.
- ShowDeskTop – DISPLAY or HIDE all icons on desktop.
- ShowExploreToolbar – DISPLAY or HIDE toolbar on windows explorer.
- ShowTaskbar –DISPLAY or HIDE taskbar.
- UnregisterAlphaKeyNotification – UNREGISTER prompt message request.
- RegisterTriggerKeyNotification – Register a request to send a prompt message when the trigger key is pressed.
- UnregisterTriggerKeyNotification – UNREGISTER prompt message request for trigger key.

Vibrator Related Functions

- VibratorOn – ON or OFF vibration indicator.

WLAN Related Functions

- WL_Enable – Enable WLAN.
- WL_Disable – Disable WLAN.

BlueTooth Related Functions

- BT_On – Enable Bluetooth.
- BT_Off – Disable Bluetooth.
- SetDiscoverMode – Enable/Disable the terminal is discoverable.
- GetDiscoverMode –Query terminal discoverable status.
- SetSPPService – Enable/Disable SPP Service.
- GetSPPService –Query SPP Service.
- SetFTPService – Enable/Disable FTP service.
- GetFTPService – Query FTP service status.
- SetFTPWriteable – Enable/Disable FTP service writeable.
- GetFTPWriteable – Query FTP service writeable status.
- SetFTPShareFolder – Setup the FTP share folder in terminal.
- GetFTPShareFolder – Query current FTP share folder in terminal.
- InitSearchBTDevice – Initial search information.
- FindNextBTDevice – retrieves the results of an Bluetooth device.

- EndSearchBTDevice – frees the search handle.
- InitSearchFTPDevice – Initial search the device supported FTP service.
- FindFirstFTPDevice – Get first device supported FTP service position.
- FindNextFTPDevice – Get next device supported FTP service position.
- PairDevice – Pair with device.
- UnPairDevice – Unpair with device.
- GetComInfo – Get com identifier index and amount.
- ConnectDevice – Connect to Bluetooth device for SPP or FTP.
- GetConnectStatus – Query the device connected status.
- GetSPPClientChannel – Get SPP channel.
- FindFirstFTPFile – Get first file information from share folder in connected device.
- FindNextFTPFile – Get next file information from share folder in connected device.
- GetFTPFile – Get file from share folder in the connected device.
- PutFTPFile – Send file to share folder in the connected device.
- CreateFTPFolder – Create a new folder to share folder in the connected device.
- DeleteFTPFolder – Delete folder from share folder in connected device.
- DeleteFTPFile – Delete file from share folder in connected device.

Camera Related Functions

- Camera_On – Enable Camera.
- Camera_Off – Disable Camera.
- SetPreviewSize – Set preview window x-axis coordinate、y-axis coordinate、width、height.
- GetPreviewSize – Get preview window x-axis coordinate、y-axis coordinate、width、height.
- EnablePreview – Enable/Disable preview window.
- SetStillCaptureSize – Set still capture image pixel.
- GetStillCaptureSize – Get still capture image pixel.
- StartStillCapture – Start execute still capture active.
- SetFlash – Enable/Disable flash light.
- GetFlash – Query flash light status.
- SetDarkMode – Enable/Disable dark mode.
- GetDarkMode – Query dark mode status.
- SetInvert – Set media stream invert, include flip、mirror.
- GetInvert – Get media stream invert, include flip、mirror.

GPRS Related Functions

- GPRS_On – Enable GPRS.
- GPRS_Off – Disable GPRS.
- GetGPRSPowerStatus – Query GPRS current power status.
- SendATCommand – Send AT command and verify required response.
- GetATResponse – Get response from module buffer.
- GetIMEINumber – Query IMEI number.
- GetIMSINumber – Query IMSI number.
- GetModuleInfo – Query module information.
- GetSignalQuality – Query signal strength of GSM/GPRS network.
- ConnectRAS – Establishes a RAS connection.
- DisconnectRAS – Disconnect current RAS connection.
- GetRASConnStatus – Query RAS connection status.
- CreateRASEntry – Create a new RAS entry.
- DeleteRASEntry – Delete a RAS entry.
- ChangeRASEntryName – Change RAS entry name.
- ChangeRASEntryProperty – Change RAS entry property.
- GetRASEntryProperty – Query RAS entry property.
- GetRASEntryCount – Query RAS entry count.
- MenuRASEntries – Menu all RAS entries, and get specified entry name.
- GetPINCounter – Query still available count for entering the currently required password.
- GetPINStatus – Query current authentication code status.
- SetPINLock – Set PIN code lock or unlock.
- GetPINLockStatus – Query the PIN code lock status.
- CheckPINCode – Enter PIN code passwords, and check correctness.
- CheckPUKCode – Enter PUK code passwords, and check correctness.
- ChangePINCode – Allows defining new PIN code password.
- ReadPhonebook – Read a phonebook entry.
- WritePhonebook – Write a phonebook entry.
- DeletePhonebook – Delete a phonebook entry.
- ReadMultiPhonebook – Read multitude phonebook entries.
- GetPhonebookTotal – Query phonebook entry amount, include used、total.
- GetPhoneMaxLength – Query maximum length of number field and text field.
- SendSMS – Send a SMS message.
- ReadSMS – Read a SMS message.
- DeleteSMS – Delete a SMS message.

- ReadMultiSMS – Read multitude SMS messages.
- WriteStorageSMS – Write a SMS message to memory storage.
- SendStorageSMS – Send a SMS message from memory storage.
- GetSMSTotal – Query SMS message amount, include used、total.
- SetSMSCentre – Set phone number of SMS centre.
- GetSMSCentre – Get phone number of SMS centre.
- SMS_Register – Register the application to SYSAPIAX.dll.
- SMS_UnRegister – Un-register the application from SYSAPIAX.dll.

Bluetooth Structure

- CONNECT_INFO Structure – CONNECT_INFO Information used by ConnectDevice.
- FTP_FILE Structure – FTP_FILE Information used by FindFirstFTPFile and FindNextFTPFile.

GPRS Structure

- RAS_ENTRY Structure – RAS_ENTRY Information used by CreateRASEntry、ChangeRASEntryProperty and GetRASEntryProperty.
- PHONEBOOK_INFO Structure – PHONEBOOK_INFO Information used by ReadMultiPhonebook.
- SMS_INFO Structure – SMS_INFO Information used by ReadMultiSMS.

Audio Related Functions

Audio_GetVolume

To query the current audio volume level setting.

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

Parameters

lpdwVolume

[out] The current volume level setting.

Returned Values

If this action succeeds, the returned value is E_FUNC_SUCCEED. If this action fails, the returned 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 6.0 or beyond.
Header: sysapiax.h
Link Library: sysapiax.lib
Link DLL: sysapiax.dll
Device: PT90

Audio_SetVolume

To set the audio volume level.

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

Parameters

dwVolume

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

Returned Values

If this action succeeds, the returned value is E_FUNC_SUCCEED. If this action fails, the returned 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 6.0 or beyond

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

Battery Related Function

GetBatteryStatus

To get main battery status.

```
int GetBatteryStatus
{
}
```

Parameters

None.

Returned Values

The returned value can be one of the values in the table below.

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 6.0 or beyond

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll
Device: PT90

Display Related Functions

BacklightOn

To turn ON or OFF the LCD screen back-light.

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

Parameters

bOn

[in] Flag that indicates whether to turn ON screen back-light(TRUE) or turn OFF screen back-light(FALSE).

Returned Values

If this action succeeds, the returned value is E_FUNC_SUCCEED. If this action fails, the returned values are E_FUNC_ERROR, E_FUNC_PAR_ERROR.

Remarks

After this action turning ON or OFF the screen back-light, the back-light will be always ON or OFF. The back-light setting of display properties in control panel does not work until the terminal been reseted.

Example

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

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

Display_QueryBacklightIntensity

To return the back-light 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.

Returned Values

If this action succeeds, the returned value is E_FUNC_SUCCEED. If this action fails, possible returned values 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 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

GetBacklightStatus

To get screen back-light status.

```
DWORD GetBacklightStatus
{
}
```

Parameters

None.

Returned Values

The returned value indicates whether screen back-light is:

1 = screen back-light is ON; or

0 = screen back-light is oOFF.

Example

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

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

PowerOnLCD

To turn ON or OFF the LCD screen power.

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

Parameters

bOn

[in] Flag that indicates whether to turn ON (TRUE) or OFF (FALSE) the LCD power.

Returned Values

If this action succeeds, the returned value is E_FUNC_SUCCEED. If this action fails, possible returned values are E_FUNC_ERROR, E_FUNC_PAR_ERROR.

Remarks

After calling this function with "bOn" FALSE, terminal will only turn OFF the LCD power. It means that terminal is still working. You should either call this function again to turn ON the LCD power or to reset terminal to use the terminal with the LCD screen ON.

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 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

SetBacklightPWM

To adjusts the LCD screen back-light brightness.

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

Parameters

nACPowerPercent,nBatteryPercent

[in] One is the brightness level setting when the terminal is using AC power and the other is the brightness level setting when the terminal is using battery power. These two settings must be one of the values in the table below.

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

Returned Values

If this action succeeds, the returned value is E_FUNC_SUCCEED. If this action fails, possible returned values are E_FUNC_ERROR, E_FUNC_PAR_ERROR.

Remarks

The Back-light Setting function in the Control Panel sets LCD screen back-light brightness level. Calling this function will also change the brightness level in Back-light Setting. You can use this function or Back-light Setting function in the Control Panel to adjust back-light brightness level.

Example

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

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

EnableTouchPanel

To ENABLE or DISABLE the touch panel.

```
DWORD EnableTouchPanel
{
    BOOL bEnable
}
```

Parameters

bEnable

[in] Flag that indicates whether to enable(TRUE) or disable(FALSE) the touch panel.

Returned Values

If this action succeeds, the returned value is E_FUNC_SUCCEED. If this action fails, possible returned values are E_FUNC_ERROR.

Example

```
DWORD dwResult = EnableTouchPanel(TRUE);
if(dwResult != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Enable touch panel fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

GetTouchPanelStatus

To get touch panel status.

```
DWORD GetTouchPanelStatus
{
    LPBOOL lpbEnable
}
```

Parameters

lpbEnable

[out] Receive the touch panel status. The returned value can be one of the values in the table below

Value	Touch panel status
0	Touch panel disable
1	Touch panel enable

Returned Values

If this action succeeds, the returned value is E_FUNC_SUCCEED. If this action fails, possible returned values are E_FUNC_ERROR.

Example

```
BOOL bEnable;
DWORD dwResult = GetTouchPanelStatus(&bEnable);
if(dwResult != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Get touch panel status fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

KeyPad Related Functions

EnablePowerButton

To ENABLE or DISABLE the POWER button.

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

Parameters

bOn

[in] Flag that indicates whether to ENABLE the POWER button(TRUE) or to DISABLE the POWER button(FALSE).

Returned Values

If this action succeeds, the returned value is E_FUNC_SUCCEED. If this action fails, possible returned values are E_FUNC_ERROR, E_FUNC_PAR_ERROR.

Remarks

If the *bOn* parameter is FALSE, the POWER button will be DISABLED. The POWER button will not work when been pressed. If the terminal enters suspend mode, the POWER button will work one time only to wake up the terminal. When the terminal wakes up, the POWER button will be DISABLED again until this function been called with parameter TRUE to ENABLE the POWER button.

Example

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

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

GetKeypadAlphaMode

To get the current keypad INPUT mode.

```
DWORD GetKeypadAlphaMode
{
}
```

Parameters

None.

Returned Values

The returned value can be one of the values in the table below.

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 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

SendKbdVirtualKey

To send a VIRTUAL KEY to key buffer.

```
DWORD SendKbdVirtualKey
{
    BYTE Key
}
```

Parameters

Key

[in] Specifies a virtual-key code.

Returned Values

If this action succeeds, the returned value is E_FUNC_SUCCEED. If this action fails, possible returned value is 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 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

SetKeypadAlphaMode

To change keypad INPUT mode.

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

Parameters

nMode

[in] Flags for setting INPUT mode. This parameterer must be one of the values in the table below.

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

Returned Values

If this action succeeds, the returnedvalue is E_FUNC_SUCCEED. If this action fails, possible returned values 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 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

LED Related Functions

GetKeypadLEDStatus

To get keypad back-light LED status.

```
BOOL GetKeypadLEDStatus
{
}
```

Parameters

None.

Returned Values

The returned value indicates whether keypad back-light LED is ON(TRUE) or 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 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

GoodReadLEDOn

To turn ON or OFF the goodread LED.

```
DWORD GoodReadLEDOn
{
    BOOL bOn
}
```

Parameters

bOn

[in] Flag that indicates whether to turn ON(TRUE) or OFF(FALSE) the goodread LED.

Returned Values

If this action succeeds, the returned value is E_FUNC_SUCCEED. If this action fails, possible returned values are E_FUNC_ERROR, E_FUNC_PAR_ERROR.

Example

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

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

KeypadLEDOn

To always turn ON or OFF the keypad LED.

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

Parameters

bOn

[in] Flag that indicates whether to turn ON (TRUE) or OFF(FALSE) the keypad LED.

Returned Values

If this action succeeds, the returned value is E_FUNC_SUCCEED. If this action fails, possible returned values 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 actual application requirements. Calling this function will set the KeyPad LED to always ON or OFF. Programmer can use this function or KeyPad LED Setting in the Control Panel to always turn ON or OFF the keypad LED.

Example

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

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

QueryKeypadLEDIntensity

To return the Keypad LED Intensity Setting when using external power and using battery power.

```
DWORD QueryKeypadLEDIntensity
{
    LPDWORD lpdwACKeypadLED,
    LPDWORD lpdwBatteryKeypadLED
}
```

Parameters

lpdwACKeypadLED

[out] The Keypad LED Intensity Setting using external power.

lpdwBatteryKeypadLED

[out] The Keypad LED Intensity Setting using battery power.

Returned Values

If this action succeeds, the returned value is E_FUNC_SUCCEED. If this action fails, possible returned values are E_FUNC_ERROR, E_FUNC_NULLPTR.

Remarks

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

Keypad LED Intensity	Keypad LED Brightness
1	on
0	off

Example

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

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll
Device: PT90

SetKeypadPWM

To adjusts Keypad LED Brightness.

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

Parameters

nACPowerPercent,nBatteryPercent

[in] One is to set Keypad LED Brightness setting when using AC power and the other is to set Keypad LED Brightness setting when using battery. These two parameters must be one of the values in the table below.

nPercent	keypad LED brightness
100	on
0	off

Returned Values

If this action succeeds, the returned value is E_FUNC_SUCCEED. If this action fails, possible returned values are E_FUNC_ERROR, E_FUNC_PAR_ERROR.

Remarks

The Timeout&Brightness function in Control Panel can set Keypad LED Brightness. Calling this function will also change the Keypad LED Brightness in Timeout&Brighness function. Programmer can use either this function or Timeout&Brightness function in Control Panel to adjust the Keypad LED Brightness level.

Example

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

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

System Related Functions

CallSuspend

To force the terminal entering SUSPEND mode.

```
void CallSuspend
{
}
```

Parameters

None.

Returned Values

None.

Example

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

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

EnableAutoConnect

To turn the AUTOCONNECT function ON or OFF.

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

Parameters

bEnable

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

Returned Values

Returning TRUE if the operation is successful. otherwise, FALSE.

Remarks

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

Example

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

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

RegisterAlphaKeyNotification

To Register the application to SYSAPIAX.dll, so that SYSAPIAX.dll will send a window message to the application when the Alpha Key is pressed.

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

Parameters

hWnd

[in] The handling window of the application to receive the message.

uMsg

[in] The message value to be sent when Alpha Key is pressed.

Returned Values

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

Remarks

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

Example

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

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

ShowChineseIME

To SHOW or HIDE ChineseIME function display

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

Parameters

bShow

[in] Flag that indicates whether to SHOW(TRUE) or HIDE (FALSE) the Chinese IME function display.

Returned Values

Returning TRUE if the operation is successful, otherwise FALSE.

Remarks

The Chinese IME is only supported in Chinese OS. It will work after calling this function then reset the terminal.

Example

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

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

ShowDesktop

To SHOW or HIDE Desktop function icon display.

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

Parameters

bShow

[in] Flag that indicates whether to SHOW(TRUE) or HIDE(FALSE) the Desktop function icon display.

Returned Values

Returning TRUE if the operation is successful; otherwise FALSE.

Remarks

After calling this function with parameter FALSE, the terminal will HIDE all icons on Desktop. After calling this function with parameter TRUE, the terminal will SHOW all icons on Desktop.

Example

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

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

ShowExploreToolbar

To SHOW or HIDE Internet ExploreToolbar function display in Windows IE.

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

Parameters

bShow

[in] Flag that indicates whether to SHOW(TRUE) or HIDE(FALSE) the Internet Explorer toolbar in Windows IE.

Returned Values

Returning TRUE if the operation is successful; otherwise FALSE.

Remarks

The *ShowExploreToolbar* function only affects Windows Internet Explorers been opened already.

Example

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

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

ShowTaskbar

To SHOW or HIDE Taskbar function display.

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

Parameters

bShow

[in] Flag that indicates whether to SHOW(TRUE) or HIDE(FALSE) Taskbar display.

Returned Values

Returning TRUE if the operation is successful; otherwise FALSE.

Remarks

After calling this function, the terminal will SHOW or HIDE Taskbar. If Taskbar is hidden by this function, it needs 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 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

UnRegisterAlphaKeyNotification

To Unregister AlphaKey Notification function request so that the application will no longer receive Alpha Key been pressed notification messages.

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

Parameters

hWnd

[in] The handling window of the application.

Returned Values

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

Example

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

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

RegisterTriggerKeyNotification

To Register the application to SYSAPIAX.dll, so that SYSAPIAX.dll will send a window message to the application when the trigger Key is pressed or released

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

Parameters

hWnd

[in] The handling window of the application to receive the message.

uMsg

[in] The message value to be sent when trigger Key is pressed or released

Returned Values

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

Remarks

The application should call UnregisterTriggerKeyNotification function to unregister the prompt message from the dll. The wParam parameter of window message return the status of trigger key.

wParam	Trigger Key
0	The trigger key is released.
1	The trigger key is pressed.

Example

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

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

UnregisterTriggerKeyNotification

To Unregister triggerkey Notification function request so that the application will no longer receive trigger key been pressed or released notification messages.

```
DWORD UnregisterTriggerKeyNotification
{
    HANDLE hWnd
 }
```

Parameters

hWnd

[in] The handling window of the application.

Returned Values

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

Example

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

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

Vibrator Related Functions

VibratorOn

To turn ON or OFF the Vibration indicator

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

Parameters

bOn

[in] Flag that indicates whether to turn ON(TRUE) or OFF(FALSE) the vibration indicator.

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned values are E_FUNC_ERROR, E_FUNC_PAR_ERROR.

Remarks

You can use this function to activate the vibration indicator of the terminal to alert the user that something is happening. Calling 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 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

WLAN Related Function

WL_Enable

To ENABLE the WLAN function.

```
BOOL WL_Enable
{
}
```

Parameters

None.

Returned Values

Returning TRUE if the operation is successful; otherwise FALSE.

Example

```
BOOL bResult;
bResult = WL_Enable();
if(bResult == FALSE)
    AfxMessageBox(_T("Wireless enable fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

WL_Disable

To DISABLE the WLAN function.

```
BOOL WL_Disable
{
}
```

Parameters

None.

Returned Values

Returning TRUE if the operation is successful; otherwise FALSE.

Example

```
BOOL bResult;
bResult = WL_Disable();
if(bResult == FALSE)
    AfxMessageBox(_T("Wireless disable fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

Bluetooth Related Function

BT_On

To ENABLE the Bluetooth function and power.

```
BOOL BT_On
{
}
```

Parameters

None.

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned values are BT_ERR_CREATE_FAIL, BT_ERR_INUSING.

Example

```
BOOL bResult;
bResult = BT_On();
if(bResult != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Bluetooth enable fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

BT_Off

To DISABLE the Bluetooth function and power.

```
void BT_Off
{
}
```

Parameters

None.

Returned Values

None

Example

BT_Off();

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

SetDiscoverMode

Enable or disable terminal discovered mode.

```
DWORD SetDiscoverMode
{
    BOOL bEnable
}
```

Parameters

bEnable

[in] Flag that indicates whether to enable (TRUE) or disable (FALSE) the terminal discovered mode

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned value is BT_ERR_SETTING_FAIL.

Example

```
If(SetDiscoverMode(TRUE) != E_FUNC_SUCCEED)
    AfxMessageBox("Setting fail");
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

GetDiscoverMode

Get terminal current discovered status.

```
BOOL GetDiscoverMode
{
}
```

Parameters

None

Returned Values

Return TRUE if terminal can be discovered, otherwise return FALSE.

Example

```
if(GetDiscoverMode())
    AfxMessageBox(_T("Discover mode is enable"));
Else
    AfxMessageBox(_T("Discover mode is disable"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

SetSPPService

Enable or disable Bluetooth serial port profile service.

```
DWORD SetSPPService
{
    BOOL bEnable
}
```

Parameters

bEnable

[in] Flag that indicates whether to enable (TRUE) or disable (FALSE) the serial port profile service mode

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned values are BT_ERR_SETTING_FAIL, BT_ERR_REG_DEV_FAIL, BT_ERR_SPP_COM_FAIL.

Example

```
If(SetSPPService(TRUE) != E_FUNC_SUCCEED)
    AfxMessageBox("Setting fail");
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

GetSPPService

Get terminal current serial port profile service status.

```
BOOL GetSPPService
{
}
```

Parameters

None

Returned Values

Return TRUE if SPP service is enable, otherwise return FALSE.

Example

```
if(GetSPPService())
    AfxMessageBox(_T("SPP service is enable"));
Else
    AfxMessageBox(_T("SPP service is disable"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

SetFTPService

Enable or disable File Transfer Profile service.

```
DWORD SetFTPService
{
    BOOL bEnable
}
```

Parameters

bEnable

[in] Flag that indicates whether to enable (TRUE) or disable (FALSE) the File Transfer Profile service mode

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned values are BT_ERR_SETTING_FAIL.

Example

```
if(SetFTPService(TRUE) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Set FTP service fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

GetFTPService

Get terminal current File Transfer Profiel service status.

```
BOOL GetFTPService
{
}
```

Parameters

None

Returned Values

Return TRUE if FTP service is enable, otherwise return FALSE.

Example

```
if(GetFTPService())
    AfxMessageBox(_T("FTP service is enable"));
Else
    AfxMessageBox(_T("FTP service is disable"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

SetFTPWriteable

Enable or disable File Transfer Profile writable.

```
DWORD SetFTPWriteable
{
    BOOL bWriteable
}
```

Parameters

bWriteable

[in] Flag that indicates whether to enable (TRUE) or disable (FALSE) the File Transfer Profile writeable mode

Returned Values

Return E_FUNC_SUCCEED if the operation is successful.

Example

```
if(SetFTPWriteable(TRUE) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Set FTP writeable fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

GetFTPWriteable

Get terminal current File Transfer Profile writeable status.

```
BOOL GetFTPWriteable
{
}
```

Parameters

None

Returned Values

Return TRUE if FTP writeable is enable, otherwise return FALSE.

Example

```
if(GetFTPWriteable())
        AfxMessageBox(_T("FTP service is writeable"));
else
        AfxMessageBox(_T("FTP service is diswriteable));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

SetFTPShareFolder

Setup the File Transfer Profile share folder.

```
DWORD SetFTPShareFolder
{
    WCHAR *strShareFolder
}
```

Parameters

strShareFolder

[in] The folder for File Transfer Profile.

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned value is E_FUNC_PAR_ERROR.

Example

```
if(SetFTPShareFolder("\\Temp") != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Set FTP Share Folder fail!!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

GetFTPShareFolder

Get terminal current File Transfer Profile share folder.

```
DWORD GetFTPShareFolder
{
    WCHAR *strShareFolder,
    int *nFolderLen
}
```

Parameters

strShareFolder

[out] The buffer to receive the share folder string

nFolderLen

[in/out] The **strShareFolder** buffer max size. If terminal current share folder length > nFolderLen, the nFolderLen receive current share folder length.

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned values are E_FUNC_PAR_ERROR, BT_ERR_INSUFFICIENT.

Remarks

If function return BT_ERR_INSUFFICIENT, nFolderLen will receive the share folder length of terminal.

Example

```
WCHAR *strFolder;
int nFolderLen = 256;
strFolder = new WCHAR[nMax];
DWORD dwErr = GetFTPShareFolder(strFolder, & nFolderLen);;
If(dwErr == BT_ERR_INSUFFICIENT){
    Delete strFolder;
    strFolder = new WCHAR[nFolderLen];
    GetFTPShareFolder(strFolder, & nFolderLen);
}
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

InitSearchBTDevice

This function initiates search information.

```
DWORD InitSearchBTDevice
{
    HANDLE *hLookup
}
```

Parameters

hLookup

[out] Handle to be used when calling the **FindNextBTDevice** & **EndSearchBTDevice** function

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned values are E_FUNC_ERROR., BT_ERR_DEVICE_ERROR.

Remarks

Must call **EndSearchBTDevice** function frees the handle after calls to the **InitSearchBTDevice** and **FindNextBTDevice** function.

Example

```
DWORD dwRe;
HANDLE hLookup;
ULONGLONG btAddress;
WCHAR szDeviceName[128];
dwRe = InitSearchBTDevice(&hLookup)
while(dwRe == E_FUNC_SUCCEED){
    dwRe = FindNextBTDevice(hLookup, szDeviceName, &btAddress, 256);
    if(dwRe == BT_ERR_DEVICE_ERROR)
        break;
        ……
}
EndSearchBTDevice(hLookup);
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

FindNextBTDevice

This function retrieves the results of an nearby Bluetooth device search.

```
DWORD FindNextBTDevice
{
   HANDLE hLookup,
   LPTSTR szDeviceName,
   ULONGLONG *btAddress,
   int nNameLen
}
```

Parameters

hLookup

[in] Handle obtained from **InitSearchBTDevice** function

szDeviceName

[out] The buffer to receive the device name string

btAddress

[out] Receive the device address of 64-bit unsigned integer

nNameLen

[in] The **szDeviceName** buffer max size. If terminal device name length > nNameLen, the **szDeviceName** buffer store data of nNameLen length

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned values are E_FUNC_ERROR, E_FUNC_PAR_ERROR, BT_ERR_DEVICE_ERROR.

Remarks

Must call **EndSearchBTDevice** function frees the handle after calls to the **InitSearchBTDevice** and **FindNextBTDevice** function.

Example

```
#define GET_NAP(_bt_addr)   ((USHORT)(((_bt_addr) & (ULONGLONG)0xFFFF00000000) >> (8*4)))
#define GET_SAP(_bt_addr)   ((ULONG)(((_bt_addr) & (ULONGLONG)0x0000FFFFFFFF) >> (0)))
DWORD dwRe;
HANDLE hLookup;
ULONGLONG btAddress;
WCHAR szAddress[16], szDeviceName[128];
dwRe = InitSearchBTDevice(&hLookup)
while(dwRe == E_FUNC_SUCCEED){
```

```
        dwRe = FindNextBTDevice(hLookup, szDeviceName, &btAddress, 256);
        if(dwRe == BT_ERR_DEVICE_ERROR)
            break;
            ……
        wsprintf(szAddress, L"%04X%08X", GET_NAP(btAddress), GET_SAP(btAddress));
            ……
    }
    EndSearchBTDevice(hLookup);
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

EndSearchBTDevice

This function frees the search handle.

```
DWORD EndSearchBTDevice
{
    HANDLE hLookup
}
```

Parameters

hLookup

[in] Handle obtained from **InitSearchBTDevice** function

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned values are E_FUNC_ERROR, BT_ERR_DEVICE_ERROR.

Example

```
DWORD dwRe;
HANDLE hLookup;
ULONGLONG btAddress;
WCHAR szDeviceName[128];
dwRe = InitSearchBTDevice(&hLookup)
while(dwRe == E_FUNC_SUCCEED){
    dwRe = FindNextBTDevice(hLookup, szDeviceName, &btAddress, 256);
    if(dwRe == BT_ERR_DEVICE_ERROR)
        break;
        ……
}
EndSearchBTDevice(hLookup);
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

InitSearchFTPDevice

Initial search Bluetooth device support File Transfer Profile service.

```
DWORD InitSearchFTPDevice
{
}
```

Parameters

None

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned values are E_FUNC_ERROR, BT_ERR_DEVICE_ERROR.

Example

```
DWORD dwPos, dwRe;
WCHAR szDeviceName[128];
ULONGLONG btAddress;
If(InitSearchFTPDevice() == E_FUNC_SUCCEED){
    dwRe = FindFirstFTPDevice(&dwPos, szDeviceName, & btAddress, 256);
    while(dwRe == E_FUNC_SUCCEED){
        FindNextFPTDevice(&dwPos, szDeviceName, & btAddress, 256);
        if(dwRe != E_FUNC_SUCCEED)
        break;
        ……
    }
}
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

FindFirstFTPDevice

Get the first searched device position information after calling InitSearchFTPDevice().

```
DWORD FindFirstFTPDevice
{
   DWORD *dwPos,
   LPTSTR szDeviceName,
   ULONGLONG *btAddress,
   int nNameLen
}
```

Parameters

dwPos

[in/out] a reference to a position value returned by **FindFirstBTDevice** or **FindNextBTDevice** function

szDeviceName

[out] The buffer to receive the device name string

btAddress

[out] Receive the device address of 64-bit unsigned integer

nNameLen

[in] The **szDeviceName** buffer max size. If terminal device name length > nNameLen, the **szDeviceName** buffer store data of nNameLen length

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned values are E_FUNC_ERROR, E_FUNC_PAR_ERROR, BT_ERR_DEVICE_ERROR.

Example

```
#define GET_NAP(_bt_addr)   ((USHORT)(((_bt_addr) & (ULONGLONG)0xFFFF00000000) >> (8*4)))
#define GET_SAP(_bt_addr)   ((ULONG)(((_bt_addr) & (ULONGLONG)0x0000FFFFFFFF) >> (0)))
DWORD dwPos, dwRe;
WCHAR szDeviceName[128], szAddress[16];
ULONGLONG btAddress;
If(InitSearchFTPDevice() == E_FUNC_SUCCEED){
    dwRe = FindFirstFTPDevice(&dwPos, szDeviceName, & btAddress, 256);
    while(dwRe == E_FUNC_SUCCEED){
        FindNextFPTDevice(&dwPos, szDeviceName, & btAddress, 256);
        if(dwRe != E_FUNC_SUCCEED)
        break;
        ……
```

```
                wsprintf(szAddress, L"%04X%08X", GET_NAP(btAddress), GET_SAP(btAddress));
                ……
            }
        }
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

FindNextFTPDevice

Get the next searched device position information.

```
DWORD FindNextFTPDevice
{
   DWORD *dwPos,
   LPTSTR szDeviceName,
   ULONGLONG *btAddress,
   int nNameLen
}
```

Parameters

dwPos

[in/out] a reference to a position value returned by **FindFirstBTDevice** or **FindNextBTDevice** function

szDeviceName

[out] The buffer to receive the device name string

btAddress

[out] Receive the device address of 64-bit unsigned integer

nNameLen

[in] The **szDeviceName** buffer max size. If terminal device name length > nNameLen, the **szDeviceName** buffer store data of nNameLen length

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned values are E_FUNC_ERROR, E_FUNC_PAR_ERROR, BT_ERR_DEVICE_ERROR.

Example

```
#define GET_NAP(_bt_addr)   ((USHORT)(((_bt_addr) & (ULONGLONG)0xFFFF00000000) >> (8*4)))
#define GET_SAP(_bt_addr)   ((ULONG)(((_bt_addr) & (ULONGLONG)0x0000FFFFFFFF) >> (0)))
DWORD dwPos, dwRe;
WCHAR szDeviceName[128], szAddress[16];
ULONGLONG btAddress;
If(InitSearchFTPDevice() == E_FUNC_SUCCEED){
    dwRe = FindFirstFTPDevice(&dwPos, szDeviceName, & btAddress, 256);
    while(dwRe == E_FUNC_SUCCEED){
        FindNextFPTDevice(&dwPos, szDeviceName, & btAddress, 256);
        if(dwRe != E_FUNC_SUCCEED)
        break;
        ……
```

```
                wsprintf(szAddress, L"%04X%08X", GET_NAP(btAddress), GET_SAP(btAddress));
                ……
            }
        }
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

PairDevice

Pair terminal with other device.

```
DWORD PairDevice
{
    ULONGLONG btAddress
    unsigned char PinCode[16]
}
```

Parameters

btAddress

[in] The device address for pair with

PinCode

[in] The pin code for connection

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned values are BT_ERR_PAIR_FAIL, BT_ERR_DEVICE_ERROR.

Example

```
if(PairDevice(btAddress, PinCode) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Pair fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

UnPairDevice

Unpair terminal with other device.

```
DWORD UnPairDevice
{
    ULONGLONG btAddress
}
```

Parameters

btAddress

[in] The device address for unpair

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned value is BT_ERR_DEVICE_ERROR.

Example

```
PairDevice(btAddress, PinCode);
    ..........
UnPairDevice(btAddress);
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

GetComInfo

Get com identifier index and amount from device hardware.

```
DWORD GetComInfo
{
    int *nComSum,
    LPCWSTR pComValue,
    int *nComValueLen
}
```

Parameters

nComSum

[out] Receive the device com amount

pComValue

[out] The buffer to receive the device com identifier index

nComValueLen

[in/out] The **pComValue** buffer max size. If terminal com value length > nComValueLen, the nComValueLen receive current com value length.

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned value is E_FUNC_ERROR, BT_ERR_INSUFFICIENT.

Remarks

If function return BT_ERR_INSUFFICIENT, nComValueLen will receive the com value length of terminal.

Example

```
WCHAR * pComValue;
int nComSum = 0, nComValueLen = 10;
pComValue = new WCHAR[nComValueLen];
DWORD dwErr = GetComInfo(&nComSum, pComValue, & nComValueLen);
If(dwErr == BT_ERR_INSUFFICIENT){
    Delete pComValue;
    pComValue = new WCHAR[nComValueLen];
    GetComInfo(&nComSum, pComValue, & nComValueLen);
}
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

ConnectDevice

Connect to Bluetooth device for SPP or FTP.

```
DWORD ConnectDevice
{
    ULONGLONG btAddress,
    CONNECT_INFO *Info,
    BOOL nConnect
}
```

Parameters

btAddress

[in] The device address for connect

Info

[in] The device connect information., see CONNECT_INFO data structure.

nConnect

[in] Connect status. 1 → connect, 0 → disconnect

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned values are E_FUNC_ERROR, E_FUNC_PAR_ERROR, BT_ERR_CHANNEL, BT_ERR_REG_DEV_FAIL, BT_ERR_DEVICE_ERROR.

Example

```
if(ConnectDevice(btAddrecc, &Info, 1) == E_FUNC_SUCCEED){
    ……….…
}
ConnectDevice(btAddress, &Info, 0);
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

GetConnectStatus

Query the device connect status.

```
DWORD GetConnectStatus
{
    ULONGLONG btAddress,
    int nConnectType,
    LPCWSTR pCom,
    int *nStatus
}
```

Parameters

btAddress

[in] Bluetooth device address

nConnectType

[in] Connect profile type. 1→ Serial port profile, 2 → File transfer profile

pCom

[in] The connect com for Serial port profile, must be four characters long.. Contains "COM"+ com identifier index, for example "COM7". If **nConnectType** parameter is 2 (FTP), **pCom** ben't to check

nStatus

[out] The device connect status

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned value is E_FUNC_PAR_ERROR.

Example

```
GetConnectStatus(btAddress, 1, _T("COM7"), &nStatus);
if(nStatus)
    AfxMessageBox(_T("SPP Connect!!"));
else
    AfxMessageBox(_T("SPP Disconnect"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

GetSPPClientChannel

Get the device serial port profile channel.

```
DWORD GetSPPClientChannel
{
    ULONGLONG btAddress,
    int *nChannel
}
```

Parameters

btAddress

[in] The device address which to get SPP channel

nChannel

[out] Receive queried channel

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned values are E_FUNC_ERROR, BT_ERR_DEVICE_ERROR.

Example

```
if(GetSPPClientChannel(btAddress, &nChannel) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Get channel fail!!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

FindFirstFTPFile

Get first file information from share folder in the connected device.

```
DWORD FindFirstFTPFile
{
    WCHAR *path,
    FTP_FILE *File
}
```

Parameters

path

[in] The path of connected device for search file.

File

[out] The first searched file information in the path, see FTP_FILE data structure.

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned values are E_FUNC_PAR_ERROR, BT_ERR_DEVICE_NOT_CONNECT, BT_ERR_FTP_DIR_FAIL, BT_ERR_FTP_EMPTY_FILE.

Example

```
FTP_FILE File;
DWORD dwErr = FindFirstFTPFile(_T("\\"), &File);
If(dwErr==E_FUNC_SUCCEED){
    Do{
        ……
        dwErr = FindNextFTPFile(&File);
    }while(dwErr == E_FUNC_SUCCEED);
}
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

FindNextFTPFile

Get next file information from share folder in the connected device.

```
DWORD FindNextFTPFile
{
    FTP_FILE *File
}
```

Parameters

File

[out] The received file information, see FTP_FILE data structure.

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned values are E_FUNC_PAR_ERROR, BT_ERR_DEVICE_NOT_CONNECT, BT_ERR_FTP_EMPTY_FILE.

Example

```
FTP_FILE File;
DWORD dwError = FindFirstFTPFile(_T("\\"), &File);
If(dwErr == E_FUNC_SUCCEED){
    Do{
        ……
        dwErr = FindNextFTPFile(&File);
    }while(dwErr == E_FUNC_SUCCEED);
}
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

GetFTPFile

Get file from share folder in the connected device.

```
DWORD GetFTPFile
{
    LPCWSTR pTargetFile
}
```

Parameters

pTargetFile

[in] The file to get from connected device

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned values are E_FUNC_ERROR, E_FUNC_PAR_ERROR, BT_ERR_DEVICE_NOT_CONNECT.

Example

```
if(GetFTPFile(_T(\\record.txt)) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Get file fail!!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

PutFTPFile

Send file to share folder in the connected device.

```
DWORD PutFTPFile
{
    LPCWSTR pSourceFile,
    LPCWSTR pTargetPath
}
```

Parameters

pSourceFile

[in] The source file in the share folder to transfer to connected device.

pTargetPath

[in] The target path in the connected device to save file.

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned values are E_FUNC_ERROR, E_FUNC_PAR_ERROR, BT_ERR_DEVICE_NOT_CONNECT.

Example

```
if(PutFTPFie(_T(\\Temp\\record.txt, _T(\\Collect)) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Put file fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

CreateFTPFolder

Create a new folder to share folder in the connected device.

```
DWORD CreateFTPFolder
{
    LPCWSTR pTarget
}
```

Parameters

pTarget

[in] The folder which be created to share folder in the connected device

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned values are E_FUNC_ERROR, E_FUNC_PAR_ERROR, BT_ERR_DEVICE_NOT_CONNECT.

Example

```
if(CreateFTPFolder(_T("\\FTP Folder") != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Create folder fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

DeleteFTPFolder

Delete folder from share folder in connected device.

```
DWORD DeleteFTPFolder
{
    LPCWSTR pTarget
}
```

Parameters

pTarget

[in] The folder will be deleted from share folder in the connected device

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned values are E_FUNC_ERROR, E_FUNC_PAR_ERROR, BT_ERR_DEVICE_NOT_CONNECT.

Example

```
if(DeleteFTPFolder(_T("\\FTP Folder")) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Delete folder fail!!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

DeleteFTPFile

Delete file from share folder in connected device.

```
DWORD DeleteFTPFile
{
    LPCWSTR pTarget
}
```

Parameters

pTarget

[in] The file will be deleted from share folder in the connected device.

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned values are E_FUNC_ERROR, E_FUNC_PAR_ERROR, BT_ERR_DEVICE_NOT_CONNECT.

Example

```
if(DeleteFTPFile(_T("\\FTP Folder\\record.txt")) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Delete file fail!!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

Camera Related Function

Camera_On

To ENABLE the Camera function and initial.

```
DWORD Camera_On
{
    HWND hVideoWnd
}
```

Parameters

hVideoWnd

[in] The handle of new owner window. This parameter can set NULL.

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned values are CAM_ERR_CREATE_FAIL, CAM_ERR_VERSION_ERROR, CAM_ERR_GET_DEVICE_FAIL, CAM_ERR_VIDEO_WINDOW_FAIL, CAM_ERR_STILL_SETTING_FAIL.

Example

```
if(Camera_On(NULL) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Camera on fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

Camera_Off

To DISABLE the Camera function and release.

```
void Camera_Off
{
}
```

Parameters

None.

Returned Values

None

Example

BT_Off();

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

SetPreviewSize

To set the x-axis coordinate、y-axis coordinate、width、height of the preview window.

```
DWORD SetPreviewSize
{
    UINT uiPosX
    UINT uiPosY
    UINT uiWidth
    UINT uiHeight
}
```

Parameters

uiPosX

[in] Specifies the x-axis coordinate to be set. Range: 0~240.

uiPosY

[in] Specifies the y-axis coordinate to be set. Range: 0~320.

uiWidth

[in] Specifies the width to be set. Range: 0~240.

uiHeight

[in] Specifies the height to be set. Range: 0~320.

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned values are E_FUNC_ERROR, E_FUNC_PAR_ERROR, CAM_ERR_DEVICE_ERROR.

Example

```
UINT uiPosX, uiPosY, uiWidth, uiHeight;
uiPosX = 0;
uiPosY = 0;
uiWidth = 240;
uiHeight = 320;
if(SetPreviewSize(uiPosX, uiPosY, uiWidth, uiHeight) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Set preview size fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

GetPreviewSize

To get the x-axis coordinate、y-axis coordinate、width、height of the preview window.

```
DWORD GetPreviewSize
{
    UINT *uiPosX
    UINT *uiPosY
    UINT *uiWidth
    UINT *uiHeight
}
```

Parameters

uiPosX

[out] Specifies the x-axis coordinate to be retrieved.

uiPosY

[out] Specifies the y-axis coordinate to be retrieved.

uiWidth

[out] Specifies the width to be retrieved.

uiHeight

[out] Specifies the height to be retrieved.

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned values are CAM_ERR_DEVICE_ERROR.

Example

```
UINT uiPosX, uiPosY, uiWidth, uiHeight;
if(GetPreviewSize(&uiPosX, &uiPosY, &uiWidth, &uiHeight) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Get preview size fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

EnablePreview

Change to ENABLE or DISABLE of the preview window.

```
DWORD EnablePreview
{
    BOOL bEnable
}
```

Parameters

bEnable

[in] Flag that indicates whether to ENABLE(TRUE) or DISABLE(FALSE) preview window.

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned values are E_FUNC_ERROR, CAM_ERR_DEVICE_ERROR.

Example

```
if(EnablePreview(TRUE) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Enable preview fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

SetStillCaptureSize

To set the image pixel for still picture capture.

```
DWORD SetStillCaptureSize
{
    DWORD dwImageSize
}
```

Parameters

dwImageSize

[in] Flags for setting image pixel. This parameterer must be one of the values in the table below.

Value	Image pixel
0	QSXGA(2560*1944)
1	SXGA(1280*960)

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned values are E_FUNC_ERROR, E_FUNC_PAR_ERROR, CAM_ERR_STILL_SETTING_FAIL, CAM_ERR_DEVICE_ERROR.

Example

```
DWORD dwSize;
dwSize = 1;
if(SetStillCaptureSize(dwSize) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Set still capture size fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

GetStillCaptureSize

To get the image pixel setting value for still picture capture.

```
DWORD GetStillCaptureSize
{
    DWORD *dwImageSize
}
```

Parameters

dwImageSize

[out] Receive the image pixel setting value.

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned values are E_FUNC_ERROR, CAM_ERR_DEVICE_ERROR.

Example

```
DWORD dwSize;
if(GetStillCaptureSize(&dwSize) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Get still capture size fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

StartStillCapture

To Start with still picture capture and save to file path.

```
DWORD StartStillCapture
{
    LPTSTR lpPathName
    BOOL bSound
}
```

Parameters

lpPathName

[in] Pointer to the file save path after capture still picture. Max size of 256.

bSound

[in] Flag that indicates whether to ENABLE(TRUE) or DISABLE(FALSE) shutter sound.

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned values are E_FUNC_ERROR, E_FUNC_PAR_ERROR, CAM_ERR_NOT_SAVE_PATH, CAM_ERR_STILL_SOUND_FAIL, CAM_ERR_DEVICE_ERROR.

Example

```
WCHAR szPictureName[256];
wcscpy(szPictureName, L"\\My Documents\\Img_01.jpg");
if(StartStillCapture(szPictureName, TRUE)   != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Still capture fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

SetFlash

To set ENABLE or DISABLE the flash light.

```
DWORD SetFlash
{
    BOOL bEnable
}
```

Parameters

bEnable

[in] Flag that indicates whether to ENABLE(TRUE) or DISABLE(FALSE) flash light.

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned values are E_FUNC_ERROR, E_FUNC_PAR_ERROR.

Example

```
if(SetFlash(TRUE) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Enable flash fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

GetFlash

To get the flash light setting value.

```
BOOL GetFlash
{
}
```

Parameters

None.

Returned Values

Return TRUE if flash light is enable, otherwise return FALSE

Example

```
if(GetFlash () != TRUE)
    AfxMessageBox(_T("Flash is disable));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

SetDarkMode

To set ENABLE or DISABLE the dark mode

```
DWORD SetDarkMode
{
    BOOL bEnable
}
```

Parameters

bEnable

[in] Flag that indicates whether to ENABLE(TRUE) or DISABLE(FALSE) dark mode.

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned values are E_FUNC_ERROR, E_FUNC_PAR_ERROR.

Example

```
if(SetDarkMode (TRUE) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Dark mode is disable"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

GetDarkMode

To get the dark mode setting value.

```
BOOL GetDarkMode
{
}
```

Parameters

None.

Returned Values

Return TRUE if dark mode is enable, otherwise return FALSE

Example

```
if(GetDarkMode() != TRUE)
    AfxMessageBox(_T("Dark mode is disable));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

SetInvert

To set the media stream invert, include flip、mirror.

```
DWORD SetInvert
{
    BOOL bFlip
    BOOL bMirror
}
```

Parameters

bFlip

[in] Flag that indicates whether to ENABLE(TRUE) or DISABLE(FALSE) media stream invert of upside down.

bMirror

[in] Flag that indicates whether to ENABLE(TRUE) or DISABLE(FALSE) media stream invert of left to right.

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned values are E_FUNC_ERROR, E_FUNC_PAR_ERROR.

Example

```
if(SetInvert(1, 1) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Set invert fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

GetInvert

To get the media stream invert setting value, include flip、mirror..

```
DWORD GetInvert
{
    BOOL *bFlip
    BOOL *bMirror
}
```

Parameters

bFlip

[out] Receive the media stream flip setting value.

bMirror

[out] Receive the media stream mirror setting value.

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned values are E_FUNC_ERROR.

Example

```
BOOL bFlip, bMirror;
if(GetInvert(&bFlip, &bMirror) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Get invert fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

GPRS Related Function

GPRS_On

To ENABLE the GPRS function and initial.

```
DWORD GPRS_On
{
}
```

Parameters

None

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned values are GPRS_ERR_CREATE_FAIL, GPRS_ERR_VERSION_ERROR.

Example

```
if(GPRS_On(NULL) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("GPRS on fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

GPRS_Off

To DISABLE the GPRS function and release.

```
void GPRS_Off
{
}
```

Parameters

None.

Returned Values

None

Example

GPRS_Off();

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

GetGPRSPowerStatus

Get current power status of GPRS module.

```
BOOL GetGPRSPowerStatus
{
    BOOL *bStatus
}
```

Parameters

bStatus

[out] Flag that indicates whether to ENABLE(TRUE) or DISABLE(FALSE) GPRS power.

Returned Values

Returning TRUE if the operation is successful; otherwise FALSE.

Example

```
BOOL bStatus;
if(GetGPRSPowerStatus(&bStatus) != TRUE)
    AfxMessageBox(_T("Get GPRS power status fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

SendATCommand

Send specify AT command and verify required response.

```
DWORD SendATCommand
{
    char *lpCmd,
    char *lpResponse,
    DWORD dwResLen,
    DWORD dwWaitTime
}
```

Parameters

lpCmd

[in] Specifies AT command. Max size of 256.

lpResponse

[out] Receive the required response data.

dwResLen

[in] The **lpResponse** buffer max size.

dwWaitTime

[in] The time interval for wait response, in milliseconds.

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned values are E_FUNC_ERROR, E_FUNC_PAR_ERROR, GPRS_ERR_DEVICE_ERROR, GPRS_ERR_BUFFER_INSUFFICIENT.

Remarks

.If receive data incomplete, maybe matter of wait time. Can adjust **dwWaitTime** parameter to try.

Example

```
char szResponse[512];
if(SendATCommand("at+gsn", szResponse, sizeof(szResponse), 0) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Send AT command fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

GetATResponse

Get response data from module buffer.

```
DWORD GetATResponse
{
    char *lpResponse,
    DWORD dwResLen,
    DWORD dwWaitTime
}
```

Parameters

lpResponse

[out] Receive the response data.

dwResLen

[in] The **lpResponse** buffer max size.

dwWaitTime

[in] The time interval for wait response, in milliseconds.

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned values are E_FUNC_ERROR, GPRS_ERR_DEVICE_ERROR, GPRS_ERR_BUFFER_INSUFFICIENT.

Example

```
char szResponse[512];
if(GetATResponse(szResponse, sizeof(szResponse), 0) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Get AT response fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

GetIMEINumber

Get module International Mobile Equipment Identity (IMEI) number.

```
DWORD GetIMEINumber
{
    LPTSTR lpIMEI
}
```

Parameters

lpResponse

[out] Receive the module IMEI number.

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned values are E_FUNC_ERROR, GPRS_ERR_DEVICE_ERROR.

Example

```
WCHAR szIMEI[128];
if(GetIMEINumber(szIMEI) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Get IMEI number fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

GetIMSINumber

Get SIM card International Mobile Subscriber Identity (IMSI) number.

```
DWORD GetIMSINumber
{
    LPTSTR lpIMSI
}
```

Parameters

lpIMSI

[out] Receive the SIM card IMSI number.

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned values are E_FUNC_ERROR, GPRS_ERR_DEVICE_ERROR.

Example

```
WCHAR szIMSI[128];
if(GetIMSINumber(szIMSI) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Get IMSI number fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

GetModuleInfo

Get module identification information, include manufacture、module、revision.

```
DWORD GetATResponse
{
    LPTSTR lpManufacture,
    LPTSTR lpModule,
    LPTSTR lpRevision
}
```

Parameters

lpManufacture

[out] Receive the manufacture identification.

lpModule

[out] Receive the product model identification.

lpRevision

[out] Receive the firmware version identification.

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned values are E_FUNC_ERROR, GPRS_ERR_DEVICE_ERROR.

Example

```
WCHAR szManufacture[128], szModule[128], szRevision[128];
if(GetModuleInfo(szManufacture, szModule, szRevision) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Get module information fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

GetSignalQuality

Get signal strength from the module to the GSM/GPRS network.

```
DWORD GetSignalQuality
{
    DWORD *dwQuality
}
```

Parameters

dwQuality

[out] Receive the signal strength.of GSM/GPRS network.

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned values are E_FUNC_ERROR, GPRS_ERR_DEVICE_ERROR.

Example

```
DWORD dwQuality;
if(GetSignalQuality(&dwQuality) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Get signal quality fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

ConnectRAS

Establishes a RAS connection. Supports callback information.

```
DWORD ConnectRAS
{
    LPTSTR lpEntryName,
    HWND hNotifyWnd
}
```

Parameters

lpEntryName

[in] The entry name to use to establishes the connect.

hNotifyWnd

[in] Pointer to a window handles to receive **RasDial** event notifications (for WinCE is WM_RASDIALEVENT). If **lpEntryName** is not NULL, the **RasDial** call operates asynchronously, send a message for each RAS event. If **lpEntryName** is NULL, the **RasDial** call operates synchronously, **RasDial** does not return until the connection has succeed or failed. See Microsoft documentation for WM_RASDIALEVENT message related.

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned values are E_FUNC_ERROR, E_FUNC_PAR_ERROR, GPRS_ERR_DEVICE_ERROR.

Example

```
if(ConnectRAS(_T("GPRS Connection"), NULL) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Connect RAS fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

DisconnectRAS

Disconnect current RAS connection.

```
DWORD DisconnectRAS
{
}
```

Parameters

None

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned values are E_FUNC_ERROR, GPRS_ERR_DEVICE_ERROR.

Example

```
if(DisconnectRAS() != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Disconnect RAS fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

GetRASConnStatus

Get status from the RAS connection opened with ConnectRAS.

```
DWORD GetRASConnStatus
{
    DWORD *dwStatus
}
```

Parameters

dwStatus

[out] Receive the current status of the specified RAS connection.

Value	Connect Status
0	Disconnection or failed connection
1	Successfully established connection
2	Other connect status

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned values are E_FUNC_ERROR, GPRS_ERR_DEVICE_ERROR.

Example

```
DWORD dwStatus;
if(GetRASConnStatus(&dwStatus) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Get RAS connect status fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

CreateRASEntry

Create a new RAS entry.

```
DWORD CreateRASEvtry
{
    LPTSTR lpEntryName,
    RAS_ENTRY *Entry
}
```

Parameters

lpEntryName

[in] The RAS entry name of an new entry to be create.

Entry

[in] The connect entry information, see RAS_ENTRY data structure.

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned values are E_FUNC_ERROR, E_FUNC_PAR_ERROR, GPRS_ERR_ENTRY_ALREADY_EXIST, GPRS_ERR_ENTRY_NOT_EXIST.

Example

```
RAS_ENTRY lpEntry;
lpEntry.dwCountryCode = 1;
wcscpy(lpEntry.szAreaCode, _T("425"));
    ……..…..…
if(CreateRASEntry(_T("My GPRS"), &lpEntry) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Create RAS entry fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

DeleteRASEntry

Delete a RAS entry.

```
DWORD DeleteRASEntry
{
    LPTSTR lpEntryName
}
```

Parameters

lpEntryName

[in] The RAS entry name of an existing entry to be deleted.

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned values are E_FUNC_ERROR, E_FUNC_PAR_ERROR.

Example

```
if(DeleteRASEntry(_T("My GPRS")) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Delete RAS entry fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

ChangeRASEntryName

Change the name of a RAS entry.

```
DWORD ChangeRASEntryName
{
    LPTSTR lpOldEntry,
    LPTSTR lpNewEntry
}
```

Parameters

lpOldEntry

[in] The old RAS entry name of an existing entry.

lpNewEntry

[in] The new RAS entry name of an not exist entry.

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned values are E_FUNC_ERROR, E_FUNC_PAR_ERROR, GPRS_ERR_ENTRY_ALREADY_EXIST, GPRS_ERR_ENTRY_NOT_EXIST.

Example

```
if(ChangeRASEntryName(_T("My GPRS"), _T("My GPRS 2")) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Change RAS entry name fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

ChangeRASEntryProperty

Change the property of a RAS entry.

```
DWORD ChangeRASEvtryProperty
{
    LPTSTR lpEntryName,
    RAS_ENTRY *Entry
}
```

Parameters

lpEntryName

[in] The RAS entry name of an existing entry to be change.

Entry

[in] The connect entry information, see RAS_ENTRY data structure.

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned values are E_FUNC_ERROR, E_FUNC_PAR_ERROR, GPRS_ERR_ENTRY_ALREADY_EXIST, GPRS_ERR_ENTRY_NOT_EXIST.

Example

```
RAS_ENTRY lpEntry;
lpEntry.dwCountryCode = 2;
wcscpy(lpEntry.szAreaCode, _T("850"));
    ……..…..…
if(ChangeRASEntryProperty(_T("My GPRS"), &lpEntry) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Change RAS entry property fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

GetRASEntryProperty

Get the property of a RAS entry.

```
DWORD GetRASEvtryProperty
{
    LPTSTR lpEntryName,
    RAS_ENTRY *Entry
}
```

Parameters

lpEntryName

[in] The RAS entry name of an existing entry to be get information.

Entry

[out] Receive the connect entry information, see RAS_ENTRY data structure.

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned values are E_FUNC_ERROR, E_FUNC_PAR_ERROR, GPRS_ERR_ENTRY_NOT_EXIST.

Example

```
RAS_ENTRY lpEntry;
if(GetRASEntryProperty(_T("My GPRS"), &lpEntry) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Get RAS entry property fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

GetRASEntryCount

Get entry count for RAS connection.

```
DWORD GetRASEntryCount
{
    int *nCount
}
```

Parameters

nCount

[out] Receive the RAS entry count.

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned values are E_FUNC_ERROR.

Example

```
int iCount;
if(GetRASEntryCount(&iCount) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Get RAS entry count fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

MenuRASEntries

Menu all existing RAS entries, and get specified entry name.

```
DWORD MenuRASEntries
{
    int nIndex,
    LPTSTR lpEntryName
}
```

Parameters

nIndex

[in] The zero-based index of a RAS entry.

lpEntryName

[out] Receive the specified RAS entry name.

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned values are E_FUNC_ERROR, E_FUNC_PAR_ERROR.

Example

```
WCHAR szEntryName[20];
if(MenuRASEntries(1, szEntryName) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Menu RAS entries fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

GetPINCounter

Get count of attempts still available for entering the currently required password, for example the PIN、PUK…

```
DWORD GetPINCounter
{
    int *nCount
}
```

Parameters

nCount

[out] Receive the number of attempts still available.

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned values are E_FUNC_ERROR, GPRS_ERR_DEVICE_ERROR.

Example

```
int iCount;
if(GetPINCounter(&iCount) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Get PIN counter fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

GetPINStatus

Get the current authentication code status of SIM card

```
DWORD GetPINStatust
{
    DWORD *dwStatus
}
```

Parameters

dwStatus

[out] Receive the authentication code status. The return value can be one of the values in the following table.

Value	Authentication Code Status
0	"ERROR", SIM card maybe not exist
1	"READY", PIN has already been entered
2	"SIM PIN", wait SIM PIN enter
3	"SIM PUK", wait SIM PUK enter
4	Other authentication code

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned values are E_FUNC_ERROR, GPRS_ERR_DEVICE_ERROR.

Example

```
DWORD dwStatus;
if(GetPINStatus(&dwStatus) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Get PIN status fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

SetPINLock

Set PIN code lock or unlock.

```
DWORD SetPINLock
{
    LPTSTR lpPIN,
    BOOL bLock
}
```

Parameters

lpPIN

[in] PIN code passwords, size of 4~8 character.

bLock

[in] Flag that indicates whether to lock (TRUE) or unlock(FALSE) the PIN code.

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned values are E_FUNC_ERROR, E_FUNC_PAR_ERROR, GPRS_ERR_DEVICE_ERROR.

Example

```
if(SetPINLock(_T("0000"), TRUE) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Set PIN lock fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

GetPINLockStatus

Get the PIN code lock status of SIM card

```
DWORD GetPINLockStatus
{
    BOOL *bStatus
}
```

Parameters

bStatus

[out] Receive the PIN code lock status. TRUE indicates lock is active, FALSE indicates lock is inactive.

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned values are E_FUNC_ERROR, GPRS_ERR_DEVICE_ERROR.

Example

```
BOOL bStatus;
if(GetPINLockStatus(&bStatus) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Get PIN lock status fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

CheckPINCode

Enter PIN code passwords, and check correctness.

```
DWORD GheckPINCode
{
    LPTSTR lpPIN,
    BOOL *bMatch
}
```

Parameters

lpPIN

[in] PIN code passwords, size of 4~8 character.

bMatch

[out] Receive the check result. TRUE indicates passwords is ok, FALSE indicates passwords is error.

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned values are E_FUNC_ERROR, E_FUNC_PAR_ERROR, GPRS_ERR_DEVICE_ERROR.

Example

```
BOOL bMatch;
if(CheckPINCode (_T("0000"), &bMatch) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Check PIN code fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

CheckPUKCode

Enter PUK code passwords, and check correctness.

```
DWORD CheckPUKCode
{
    LPTSTR lpPUK,
    LPTSTR lpNewPIN,
    BOOL *bMatch
}
```

Parameters

lpPUK

[in] PUK code passwords, size of 8 character.

lpNewPIN

[in] New PIN code passwords, size of 4~8 character.

bMatch

[out] Receive the check result. TRUE indicates passwords is ok, FALSE indicates passwords is error.

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned values are E_FUNC_ERROR, E_FUNC_PAR_ERROR, GPRS_ERR_DEVICE_ERROR.

Example

```
BOOL bMatch;
if(CheckPUKCode(_T("12345678"), _T("0000"), &bMatch) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Check PUK code fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

ChangePINCode

Allows defining new PIN code password.

```
DWORD ChangePINCode
{
    LPTSTR lpOldPIN,
    LPTSTR lpNewPIN
}
```

Parameters

lpOldPIN

[in] Old PIN code passwords, size of 4~8 character.

lpNewPIN

[in] New PIN code passwords, size of 4~8 character.

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned values are E_FUNC_ERROR, E_FUNC_PAR_ERROR, GPRS_ERR_DEVICE_ERROR.

Example

```
if(ChangePINCode(_T("0000"), _T("1111")) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Change PIN code fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

ReadPhonebook

Read a phonebook entry in SIM card.

```
DWORD ReadPhonebook
{
    int nIndex,
    LPTSTR lpNumber,
    LPTSTR lpName
}
```

Parameters

nIndex

[in] The one-based index of a phonebook entry.

lpNumber

[out] Receive the phonebook entry number

lpName

[out] Receive the phonebook entry text.

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned values are E_FUNC_ERROR, E_FUNC_PAR_ERROR, GPRS_ERR_DEVICE_ERROR.

Example

```
WCHAR szNumber[30], szName[30];
if(ReadPhonebook(1, szNumber, szName) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Read phonebook fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

WritePhonebook

Write a phonebook entry in SIM card.

```
DWORD WritePhonebook
{
    int nIndex,
    LPTSTR lpNumber,
    LPTSTR lpName,
    int nNumType
}
```

Parameters

nIndex

[in] The one-based index of a phonebook entry. If this parameter is 0, the phonebook entry wills automatic write empty location.

lpNumber

[in] Phonebook entry number

lpName

[in] Phonebook entry text.

nNumType

[in] Define the used type of number.

Value	Number Type
129	Normal code
145	International access code

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned values are E_FUNC_ERROR, E_FUNC_PAR_ERROR, GPRS_ERR_DEVICE_ERROR.

Example

```
if(WritePhonebook(0, _T("+886912345678"), _T("test"), 145) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Write phonebook fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

DeletePhonebook

Delete a phonebook entry in SIM card.

```
DWORD DeletePhonebook
{
    int nIndex
}
```

Parameters

nIndex

[in] The one-based index of a phonebook entry.

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned values are E_FUNC_ERROR, E_FUNC_PAR_ERROR, GPRS_ERR_DEVICE_ERROR.

Example

```
if(DeletePhonebook(1) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Delete phonebook fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

ReadMultiPhonebook

Read multitude phonebook entries in SIM card.

```
DWORD ReadMultiPhonebook
{
    int nStartIndex,
    int nEndIndex,
    DWORD *dwResLen,
    PHONEBOOK_INFO *Info,
    DWORD *dwInfoLen,
    int *nCount
}
```

Parameters within

nStartIndex

[in] The first location number to start reading, one-based index.

nEndIndex

[in] The last location number to stop reading, one-based index.

dwResLen

[in/out] The response buffer max size. If response buffer length > dwResLen, the dwResLen receive current response buffer length.

Info

[in/out] Pointer to a buffer that receives an array of PHONEBOOK_INFO structure. Before calling the function, an application must set the **dwSize** member of the first **PHONEBOOK_INFO** structure in the buffer to **sizeof(PHONEBOOK_INFO)** in order to identify the version of the structure being passed.

dwInfoLen

[in/out] Pointer to a variable that contains the size, in bytes, of the buffer specified by **Info**. On return, the function sets this variable to the number of bytes required to successfully complete the call.

nCount

[out] Receive the **Info** count.

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned values are E_FUNC_ERROR, E_FUNC_PAR_ERROR, GPRS_ERR_DEVICE_ERROR, GPRS_ERR_BUFFER_INSUFFICIENT.

Example

```
#define LPPHONEBOOK_INFO PHONEBOOK_INFO*
    ……
```

```
int iCount;
DWORD dwRet, dwInfoLen, dwResLen=10000;
PHONEBOOK_INFO PhonebookInfo={0};
LPPHONEBOOK_INFO lpPhonebookInfo=&PhonebookInfo;
dwInfoLen = sizeof(PHONEBOOK_INFO);
PhonebookInfo.dwSize = sizeof(PHONEBOOK_INFO);
if(dwRet = ReadMultiPhonebook(1, 10, &dwResLen, lpPhonebookInfo, &dwInfoLen, &iCount))
{
    if(dwRet == GPRS_ERR_BUFFER_INSUFFICIENT)
    {
        lpPhoneInfo = new PHONEBOOK_INFO[dwInfoLen / sizeof(PHONEBOOK_INFO)];
        lpPhonebookInfo->dwSize=sizeof(PHONEBOOK_INFO);
        dwRet = ReadMultiPhonebook(1, 10, &dwResLen, lpPhonebookInfo, &dwInfoLen, &iCount);
        if(dwRet != E_FUNC_SUCCEED)
            AfxMessageBox(_T("Read multitude phonebook fail!"));
    }
    else
        AfxMessageBox(_T("Read multitude phonebook fail!"));
}
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

GetPhonebookTotal

Get amount of phonebook entry in SIM card, include used、total.

```
DWORD GetPhonebookTotal
{
    int *nUsed,
    int *nTotal
}
```

Parameters

nUsed

[out] Receive the amount of used.

nTotal

[out] Receive the amount of allow maximum.

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned values are E_FUNC_ERROR, GPRS_ERR_DEVICE_ERROR.

Example

```
int iUsed, iTotal;
if(GetPhonebookTotal (&iUsed, &iTotal) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Get phonebook total fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

GetPhoneMaxLength

Get maximum length of number field and text field by SIM card.

```
DWORD GetPhoneMaxLength
{
    int *nNumberLen,
    int *nNameLen
}
```

Parameters

nNumberLen

[out] Receive the maximum length of phone number.

nNameLen

[out] Receive the maximum length of phone text.

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned values are E_FUNC_ERROR, GPRS_ERR_DEVICE_ERROR.

Example

```
int iNumberLen, iNameLen ;
if(GetPhoneMaxLength (&iNumberLen, &iNameLen) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Get phone maximum length fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

SendSMS

Send SMS message to specified phone number.

```
DWORD SendSMS
{
    LPTSTR lpCenterNum,
    LPTSTR lpSendNum,
    LPTSTR lpData
}
```

Parameters

lpCenterNum

[in] The phone number of short message service center (SMSC), max size of 30. If this parameter is NULL, will use storage SMSC in terminal to send.

lpSendNum

[in] The phone number of recipient, max size of 30.

lpData

[in] SMS message data, max size of 70.

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned values are E_FUNC_ERROR, E_FUNC_PAR_ERROR, GPRS_ERR_DEVICE_ERROR.

Example

```
if(SendSMS(_T(""), _T("0912345678"), _T("Send Test")) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Send SMS fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

ReadSMS

Read a SMS message in SIM card.

```
DWORD ReadSMS
{
    int nIndex,
    int *nStatus,
    LPTSTR lpNumber,
    LPTSTR lpData,
    LPTSTR lpTimeStamp
}
```

Parameters

nIndex

[in] The one-based index of a SMS message.

nStatus

[out] Receive the message status. The return value can be one of the values in the following table.

Value	Message Status
0	"REC UNREAD", received unread message
1	"REC READ", received read message
2	"STO UNSENT", stored unsent message
3	"STO SENT", stored sent message

lpNumber

[out] Receive the sender phone number.

lpData

[out] Receive the SMS message data.

lpTimeStamp

[out] Receive the service center time stamp. Format: "yy/MM/dd,hh:mm:ss", where characters indicate year (two last digits), month, day, hour, minute, second.

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned values are E_FUNC_ERROR, E_FUNC_PAR_ERROR, GPRS_ERR_DEVICE_ERROR.

Example

```
int iStatus;
WCHAR szNumber[30], szData[512], szTimpStamp[30];
if(ReadSMS(1, &iStatus, szNumber, szData, szTimpStamp) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Read SMS fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

DeleteSMS

Delete a SMS message in SIM card.

```
DWORD DeleteSMS
{
    int nIndex
}
```

Parameters

nIndex

[in] The one-based index of a SMS message.

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned values are E_FUNC_ERROR, E_FUNC_PAR_ERROR, GPRS_ERR_DEVICE_ERROR.

Example

```
if(DeleteSMS(1) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Delete SMS fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

ReadMultiSMS

Read multitude SMS messages in SIM card.

```
DWORD ReadMultiSMS
{
    int nStatus,
    DWORD *dwResLen,
    SMS_INFO *Info,
    DWORD *dwInfoLen,
    int *nCount
}
```

Parameters

nStatus

[in] The message status can be one of the values in the following table.

Value	Message Status
0	"REC UNREAD", received unread messages
1	"REC READ", received read messages
2	"STO UNSENT", stored unsent messages
3	"STO SENT", stored sent messages
4	"ALL", all messages

dwResLen

[in/out] The response buffer max size. If response buffer length > dwResLen, the dwResLen receive current response buffer length.

Info

[in/out] Pointer to a buffer that receives an array of SMS_INFO structure. Before calling the function, an application must set the **dwSize** member of the first **SMS_INFO** structure in the buffer to **sizeof(SMS_INFO)** in order to identify the version of the structure being passed.

dwInfoLen

[in/out] Pointer to a variable that contains the size, in bytes, of the buffer specified by **Info**. On return, the function sets this variable to the number of bytes required to successfully complete the call.

nCount

[out] Receive the **Info** count.

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned values are E_FUNC_ERROR, E_FUNC_PAR_ERROR, GPRS_ERR_DEVICE_ERROR, GPRS_ERR_BUFFER_INSUFFICIENT.

Example

```
#define LPSMS_INFO SMS_INFO*
        ……
int iCount;
DWORD dwRet, dwInfoLen, dwResLen=10000;
SMS_INFO SMSInfo={0};
LPSMS_INFO lpSMSInfo=&SMSInfo;
dwInfoLen = sizeof(SMS_INFO);
SMSInfo.dwSize = sizeof(SMS_INFO);
if(dwRet = ReadMultiSMS (4, &dwResLen, lpSMSInfo, &dwInfoLen, &iCount))
{
    if(dwRet == GPRS_ERR_BUFFER_INSUFFICIENT)
    {
        lpSMSInfo = new SMS_INFO[dwInfoLen / sizeof(SMS _INFO)];
        lpSMSInfo->dwSize=sizeof(SMS_INFO);
        dwRet = ReadMultiSMS (4, &dwResLen, lpSMSInfo, &dwInfoLen, &iCount);
        if(dwRet != E_FUNC_SUCCEED)
            AfxMessageBox(_T("Read multitude SMS fail!"));
    }
    else
        AfxMessageBox(_T("Read multitude SMS fail!"));
}
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

WriteStorageSMS

Write a SMS message to SIM memory storage.

```
DWORD WriteStorageSMS
{
    int *nIndex,
    LPTSTR lpCenterNum,
    LPTSTR lpSendNum,
    LPTSTR lpData
}
```

Parameters

nIndex

[out] Receive the one-based index of storage SMS message.

lpCenterNum

[in] The phone number of short message service center (SMSC) , max size of 30. If this parameter is NULL, will use storage SMSC in terminal to send.

lpSendNum

[in] The phone number of recipient, max size of 30.

lpData

[in] SMS message data, max size of 70.

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned values are E_FUNC_ERROR, E_FUNC_PAR_ERROR, GPRS_ERR_DEVICE_ERROR.

Example

```
int iIndex;
if(WriteStorageSMS(&iIndex, _T(""), _T("0912345678"), _T("Write Storage Test")) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Write storage SMS fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

SendStorageSMS

Send a SMS message from SIM memory storage.

```
DWORD SendStorageSMS
{
    int nIndex
}
```

Parameters

nIndex

[in] The one-based index of storage SMS message.

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned values are E_FUNC_ERROR, E_FUNC_PAR_ERROR, GPRS_ERR_DEVICE_ERROR.

Example

```
if(SendStorageSMS(1) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Send storage SMS fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

GetSMSTotal

Get amount of SMS message in SIM card, include used、total.

```
DWORD GetSMSTotal
{
    int *nUsed,
    int *nTotal
}
```

Parameters

nUsed

[out] Receive the amount of used.

nTotal

[out] Receive the amount of allow maximum.

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned values are E_FUNC_ERROR, GPRS_ERR_DEVICE_ERROR.

Example

```
int iUsed, iTotal;
if(GetSMSTotal (&iUsed, &iTotal) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Get SMS total fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

SetSMSCentre

Set phone number of short message service center (SMSC).

```
DWORD SetSMSCentre
{
    LPTSTR lpNumber
}
```

Parameters

lpNumber

[in] SMS centre phone number.

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned values are E_FUNC_ERROR, E_FUNC_PAR_ERROR, GPRS_ERR_DEVICE_ERROR.

Example

```
if(SetSMSCentre(_T("+886912345678")) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Set SMS centre fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

GetSMSCentre

Get phone number of short message service center (SMSC).

```
DWORD GetSMSCentre
{
    LPTSTR lpNumber,
    int nMaxLen
}
```

Parameters

lpNumber

[out] Receive the SMS centre phone number.

nMaxLen

[in] The **lpNumber** buffer max size.

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned values are E_FUNC_ERROR, GPRS_ERR_DEVICE_ERROR.

Example

```
WCHAR szNumber[30];
if(GetSMSCentre(szNumber, 30) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("Get SMS centre fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

SMS_Register

To register the SMS application to SYSAPIAX.dll so that SYSAPIAX.dll can communicate with the application.

```
DWORD SMS_Register
{
    HWND hNotifyWnd,
    UINT uNewSMSNotify,
    UINT uNewPhoneNotify
}
```

Parameters

hNotifyWnd

[in] The window handling the function library will send message to report.

uNewSMSNotify

[in] Application-defined message identifier. The system uses this for notification message that it sends to the window identified in **hNotifyWnd**. These notifications are sent when receive a new SMS of GPRS module.

uNewPhoneNotify

[in] Application-defined message identifier. The system uses this for notification message that it sends to the window identified in **hNotifyWnd**. These notifications are sent when receive a new phone call of GPRS module.

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned values are E_FUNC_ERROR, E_FUNC_PAR_ERROR, GPRS_ERR_DEVICE_ERROR.

Example

```
if(SMS_Register(hWnd, WM_APP+1, WM_APP+2) != E_FUNC_SUCCEED)
    AfxMessageBox(_T("SMS_Register fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

SMS_UnRegister

To unregister the SMS application from SYSAPIAX.dll.

```
DWORD SMS_UnRegister
{
}
```

Parameters

None

Returned Values

If the action succeeds, the returned value is E_FUNC_SUCCEED. If the action fails, possible returned values are E_FUNC_ERROR, GPRS_ERR_DEVICE_ERROR.

Example

```
if(SMS_UnRegister() != E_FUNC_SUCCEED)
    AfxMessageBox(_T("SMS_UnRegister fail!"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: sysapiax.h

Link Library: sysapiax.lib

Link DLL: sysapiax.dll

Device: PT90

GPS

The GPS receiver on the PT90 uses COM8 to output NMEA messages, when the COM port is opened. Using Microsoft Standard API to open serial port and receive NMEA data. For details, please refer to NMEA 0183 ver3.0.

Serial Configuration	GPS receiver
Com port	8
Baud rate	9600
Data bits	8
Parity	None
Stop bits	1

Requirements

OS Versions: Windows CE 6.0 or beyond.

Device: PT90

Bluetooth Structure

CONNECT_INFO Structure

This setting file contains information used by ConnectDevice .

```
Struct CONNECT_INFO
{
    int nChannel;
    int nConnectType;
    WCHAR strCom[6];
}
```

Members

nChannel

The connect channel for Serial port profile

nCannectType

Connect profile type. 1→ Serial port profile, 2 → File transfer profile

strCom

The connect com for Serial port profile, must be four characters long.. Contains "COM"+ com identifier index, for example "COM7". If **nConnectType** member is 2 (FTP), **strCom** ben't to check

Structure Information

Header: sysapiax.h
Device: PT90

FTP_FILE Structure

This setting file contains information used by FindFirstFTPFile, FindNextFTPFile .

```
Struct FTP_FILE
{
    int nFileType;
    WCHAR strPath[260];
    WCHAR strFile[260];
    DWORD dwFileSize;
}
```

Members

nFileType

File object profile type. 0→ File, 1→ Folder

strPath

The file path

strFile

The file name; It will be null if the object is a folder

dwFileSize

The file size , in bytes; It will be 0 if the object is a folder.

Structure Information

Header: sysapiax.h

Device: PT90

GPRS Structure

RAS_ENTRY Structure

This setting file contains information used by CreateRASEntry, ChangeRASEntryProperty, GetRASEntryProperty.

```
Struct CONNECT_INFO
{
    DWORD dwCountryCode;
    WCHAR szAreaCode[10];
    WCHAR szPhoneNumber[128];
    WCHAR szExtraCmd[330];
    WCHAR szUserName[256];
    WCHAR szPassword[256];
    WCHAR szDomain[15];
}
```

Members

dwCountryCode

Specifies the country/region code portion of the phone number.

szAreaCode

Null.-terminated string that contains the area code.

szPhoneNumber

The phone number of RAS entry.

szExtraCmd

Spicifies the special modem commands may be inserted into the dial string.

szUserName

Null.-terminated string that contains the user's name. This string is used to authenticate the user's access to the remote access server.

szPassword

Null.-terminated string that contains the user's password. This string is used to authenticate the user's access to the remote access server.

szDomain

Null.-terminated string that contains the domain on which authenticate is to occur. An empty string ("") specifies the domain in which the remote access server is a member. An asterisk specifies the domain stored in the RAS for the entry.

Structure Information

Header: sysapiax.h

Device: PT90

PHONEBOOK_INFO Structure

This setting file contains information used by ReadMultiPhonebook.

```
Struct PHONEBOOK_INFO
{
    DWORD dwSize;
    int nIndex;
    WCHAR szNumber[30];
    WCHAR szName[30];
}
```

Members

dwSize

Specifies the structure size, in bytes.

nIndex

The one-based index of a phonebook entry.

szNumber

Phonebook entry number.

szName

Phonebook entry text.

Structure Information

Header: sysapiax.h

Device: PT90

SMS_INFO Structure

This setting file contains information used by ReadMultiSMS.

```
Struct SMS_INFO
{
    DWORD dwSize;
    int nIndex;
    int nStatus;
    WCHAR szNumber[30];
    WCHAR szData[160];
    WCHAR szTimeStamp[30];
}
```

Members

dwSize

Specifies the structure size, in bytes.

nIndex

The one-based index of a SMS message.

nStatus

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

Value	Message Status
0	"REC UNREAD", received unread message
1	"REC READ", received read message
2	"STO UNSENT", stored unsent message
3	"STO SENT", stored sent message

szNumber

Sender phone number.

szData

SMS message data.

szTimeStamp

Service center time stamp. Format: "yy/MM/dd,hh:mm:ss", where characters indicate year (two last digits), month, day, hour, minute, second.

Structure Information

Header: sysapiax.h

Device: PT90

RFID.DLL

We supply RFID.DLL which includes several functions to allow programmer to control the on-board RFID device. Programmer can use WINCE develop tool like ***Visual Studio 2005*** to develop application programs. Descriptions of all these functions are given below.

RFID Related Functions

- OpenComPort – Open com port to connect RFID module.
- CloseComPort – Close com port and disconnect RFID module.
- GetFWVersion – Query RFID module firmware version.
- APIVersion – Query RFID API version.
- SetWorkingType – Set RFID module working type with different tag type.
- AntennaContro – Enable/Disable antenna to save power.
- Inventory15693 – Query 15693 tag unique identifier (UID).
- ReadSingleBlock – Read data of single block from 15693 tag.
- WriteSingleBlock – Write data of single block to 15693 tag.
- WriteAFI – Write application family identifier (AFI) to 15693 tag.
- WriteDSFID – Write data structure format identifier (DSFID) to 15693 tag.
- LockAFI – Lock AFI on 15693 tag.
- LockDSFID – Lock DSFID on 15693 tag.
- LockBlock – Lock single block on 15693 tag.
- TagSystemInfo – Query tag information from 15693 tag.
- TagStayQuiet – Set 15693 tag to stay quiet state.
- TagSelect – Set 15693 tag to select state.
- TagResetToReady – Set select or quiet tag to ready state.
- OpenCard14443A – Open 14443A tag of mifare type and get tag UID.
- ReadMifareBlock – Read data of single block from 14443A mifare S50/S70 tag.
- WriteMifareBlock – Write data of single block to 14443A mifare S50/S70 tag.
- ReadUltraLightBlock – Read data of single block from 14443A mifare ultralight tag.
- WriteUltraLightBlock – Write data of single block to 14443A mifare ultralight tag.
- GetUid14443B – Query 14443B tag UID.
- SRIX4KChipID – Query 14443B SRIX4K tag chip ID.
- SRIX4KReadBlock – Read data of single block from 14443B SRIX4K tag.
- SRIX4KWriteBlock – Write data of single block to 14443B SRIX4K tag.

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

RFID Related Function

OpenComPort

Open com port to connect the RFID module.

```
int OpenComPort
{
    int iCom
}
```

Parameters

iCom

[in] The RFID module com port number. PT90 only specified com 6.

Returned Values

Returning zero if the operation is successful, otherwise, indicates failure.

Example

```
if(OpenComPort(6) != 0)
    AfxMessageBox(_T("RFID open com fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: RFID.h

Link Library: RFID.lib

Link DLL: RFID.dll

Device: PT90

CloseComPort

Close com port and disconnect the RFID module connection.

```
int CloseComPort
{
}
```

Parameters

None.

Returned Values

Returning zero if the operation is successful, otherwise, indicates failure.

Example

```
if(CloseComPort() != 0)
    AfxMessageBox(_T("RFID close com fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: RFID.h

Link Library: RFID.lib

Link DLL: RFID.dll

Device: PT90

GetFWVersion

Get RFID module firmware version.

```
int GetFWVersion
{
    char lpOutVersion[]
}
```

Parameters

lpOutVersion

[out] Receive the RFID module firmware version.

Returned Values

Returning zero if the operation is successful, otherwise, indicates failure.

Example

```
char cVersion[20];
if(GetFWVersion(cVersion) != 0)
    AfxMessageBox(_T("Get FW version fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: RFID.h

Link Library: RFID.lib

Link DLL: RFID.dll

Device: PT90

APIVersion

Get RFID API version.

```
intAPIVersion
{
    charlpOutVersion[]
}
```

Parameters

lpOutVersion

[out] Receive the RFID API version.

Returned Values

Returning zero if the operation is successful, otherwise, indicates failure.

Example

```
char cVersion[20];
if(APIVersion(cVersion) != 0)
    AfxMessageBox(_T("Get API version fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: RFID.h

Link Library: RFID.lib

Link DLL: RFID.dll

Device: PT90

SetWorkingType

Set RFID module working type with different tag type, and this function should be called before read card.

```
int SetWorkingType
{
    int iType,
    int iHalfPower
}
```

Parameters

iType

[in] The RFID module working type. This parameter must be one of the values in the table below.

Value	Working type
1	15693
2	14443A
3	14443B

iHalfPower

[in] The RFID module power type. This parameter must be one of the values in the table below.

Value	Power type
0	full power
1	Half power

Returned Values

Returning zero if the operation is successful, otherwise, indicates failure.

Example

```
if(SetWorkingType(1,0) != 0)
    AfxMessageBox(_T("Set working type fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: RFID.h

Link Library: RFID.lib

Link DLL: RFID.dll

Device: PT90

AntennaContro

Enable/Disable antenna to save power.

```
int AntennaContro
{
    int iSelect
}
```

Parameters

iSelect

[in] Flag that indicates whether to Enable(1) or Disable(0) the antenna.

Returned Values

Returning zero if the operation is successful, otherwise, indicates failure.

Example

```
if(AntennaContro(1) != 0)
    AfxMessageBox(_T("Antenna control fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: RFID.h

Link Library: RFID.lib

Link DLL: RFID.dll

Device: PT90

Inventory15693

Get ISO 15693 tag unique identifier (UID).

```
int Inventory15693
{
    char lpFlag[],
    char lpAFI[],
    char lpOutUID[]
}
```

Parameters

lpFlag

[in] Set request flag, please refer to table below. This parameter displayed in base hexadecimal notation.

Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1
0	0	Slots_flag 0 - 16 slots 1 - 1 slots	0	0	1	0	0

lpAFI

[in] This parameter not support and should be set to "".

lpOutUID

[out] Receive the tag unique identifier.

Returned Values

Returning zero if the operation is successful, otherwise, indicates failure.

Example

```
char cUID[40];
if(Inventory15693("04", "", cUID) != 0)
    AfxMessageBox(_T("Inventory 15693 fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: RFID.h

Link Library: RFID.lib

Link DLL: RFID.dll

Device: PT90

ReadSingleBlock

Read data of single block from ISO 15693 tag.

```
int ReadSingleBlock
{
    int iBlock,
    char lpFlag[],
    char lpAddressUID[],
    char lpOutData[]
}
```

Parameters

iBlock

[in] The block index of ISO 15693 tag. Please refer to tag respective document.

lpFlag

[in] Set request flag, please refer to table belowand RFID Request Flags. This parameter displayed in base hexadecimal notation.

Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1
0	Option_flag	Address_flag	Select_flag	0	0	0	0

lpAddressUID

[in] The unique identifier of ISO 15693 tag. This parameter apply determined by the **Address_flag**, please refer to ISO 15693 document.

lpOutData

[out] Receive the block data.

Returned Values

Returning zero if the operation is successful, otherwise, indicates failure.

Example

```
char cData[128];
if(ReadSingleBlock(8, "00", "", cData) != 0)
    AfxMessageBox(_T("Read single block fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: RFID.h

Link Library: RFID.lib

Link DLL: RFID.dll

Device: PT90

WriteSingleBlock

Write data of single block to ISO 15693 tag.

```
int WriteSingleBlock
{
    int iBlock,
    char lpFlag[],
    char lpAddressUID[],
    char lpData[]
}
```

Parameters

iBlock

[in] The block index of ISO 15693 tag. Please refer to tag respective document.

lpFlag

[in] Set request flag, please refer to table below and RFID Request Flags. This parameter displayed in base hexadecimal notation.

Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1
0	Option_flag	Address_flag	Select_flag	0	0	0	0

lpAddressUID

[in] The unique identifier of ISO 15693 tag. This parameter apply determined by the **Address_flag**, please refer to ISO 15693 document.

lpData

[in] The block data. Data lengths refer to tag respective document.

Returned Values

Returning zero if the operation is successful, otherwise, indicates failure.

Example

```
if(WriteSingleBlock(8, "40", "", "12345678") != 0)
    AfxMessageBox(_T("Write single block fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: RFID.h

Link Library: RFID.lib

Link DLL: RFID.dll

Device: PT90

WriteAFI

Write application family identifier (AFI) to ISO 15693 tag.

```
int WriteAFI
{
    char lpFlag[],
    char lpAddressUID[],
    char lpData[]
}
```

Parameters

lpFlag

[in] Set request flag, please refer to table below and RFID Request Flags. This parameter displayed in base hexadecimal notation.

Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1
0	Option_flag	Address_flag	Select_flag	0	0	0	0

lpAddressUID

[in] The unique identifier of ISO 15693 tag. This parameter apply determined by the **Address_flag**, please refer to ISO 15693 document.

lpData

[in] The AFI data, 2 byte.

Returned Values

Returning zero if the operation is successful, otherwise, indicates failure.

Example

```
if(WriteAFI("40","","11") != 0)
    AfxMessageBox(_T("Write AFI fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: RFID.h

Link Library: RFID.lib

Link DLL: RFID.dll

Device: PT90

WriteDSFID

Write data structure format identifier (DSFID) to ISO 15693 tag.

```
int WriteDSFID
{
    char lpFlag[],
    char lpAddressUID[],
    char lpData[]
}
```

Parameters

lpFlag

[in] Set request flag, please refer to table below and RFID Request Flags. This parameter displayed in base hexadecimal notation.

Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1
0	Option_flag	Address_flag	Select_flag	0	0	0	0

lpAddressUID

[in] The unique identifier of ISO 15693 tag. This parameter apply determined by the **Address_flag**, please refer to ISO 15693 document.

lpData

[in] The DSFID data, 2 byte.

Returned Values

Returning zero if the operation is successful, otherwise, indicates failure.

Example

```
if(WriteDSFID("40", "", "22") != 0)
    AfxMessageBox(_T("Write DSFID fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: RFID.h

Link Library: RFID.lib

Link DLL: RFID.dll

Device: PT90

LockAFI

Lock application family identifier (AFI) on ISO 15693 tag.

```
int LockAFI
{
    char lpFlag[],
    char lpAddressUID[],
}
```

Parameters

lpFlag

[in] Set request flag, please refer to table below and RFID Request Flags. This parameter displayed in base hexadecimal notation.

Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1
0	Option_flag	Address_flag	Select_flag	0	0	0	0

lpAddressUID

[in] The unique identifier of ISO 15693 tag. This parameter apply determined by the **Address_flag**, please refer to ISO 15693 document.

Returned Values

Returning zero if the operation is successful, otherwise, indicates failure.

Remarks

When applies this function, AFI will lock and be unable to restore on ISO 15693 tag.

Example

```
char cUID[40];
Inventory15693("04", "", cUID);
if(LockAFI("20", cUID) != 0)
    AfxMessageBox(_T("Lock AFI fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: RFID.h

Link Library: RFID.lib

Link DLL: RFID.dll

Device: PT90

LockDSFID

Lock data structure format identifier (DSFID) on ISO 15693 tag.

```
int LockDSFID
{
    char lpFlag[],
    char lpAddressUID[],
}
```

Parameters

lpFlag

[in] Set request flag, please refer to table below and RFID Request Flags. This parameter displayed in base hexadecimal notation.

Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1
0	Option_flag	Address_flag	Select_flag	0	0	0	0

lpAddressUID

[in] The unique identifier of ISO 15693 tag. This parameter apply determined by the **Address_flag**, please refer to ISO 15693 document.

Returned Values

Returning zero if the operation is successful, otherwise, indicates failure.

Remarks

When applies this function, DSFID will lock and be unable to restore on ISO 15693 tag.

Example

```
char cUID[40];
Inventory15693("04", "", cUID);
if(LockDSFID("20", cUID) != 0)
    AfxMessageBox(_T("Lock DSFID fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: RFID.h

Link Library: RFID.lib

Link DLL: RFID.dll

Device: PT90

LockBlock

Lock specific block on ISO 15693 tag.

```
int LockBlock
{
    int iBlock,
    char lpFlag[],
    char lpAddressUID[],
}
```

Parameters

iBlock

[in] The block index of ISO 15693 tag. Please refer to tag respective document.

lpFlag

[in] Set request flag, please refer to table below and RFID Request Flags. This parameter displayed in base hexadecimal notation.

Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1
0	Option_flag	Address_flag	Select_flag	0	0	0	0

lpAddressUID

[in] The unique identifier of ISO 15693 tag. This parameter apply determined by the **Address_flag**, please refer to ISO 15693 document.

Returned Values

Returning zero if the operation is successful, otherwise, indicates failure.

Remarks

When applies this function, the block will lock and be unable to restore on ISO 15693 tag.

Example

```
char cUID[40];
Inventory15693("04", "", cUID);
if(LockBlock(8, "20", cUID) != 0)
    AfxMessageBox(_T("Lock block fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: RFID.h

Link Library: RFID.lib

Link DLL: RFID.dll

Device: PT90

TagSystemInfo

Get tag information from ISO 15693 tag.

```
int TagSystemInfo
{
    char lpFlag[],
    char lpAddressUID[],
    char lpOutData[]
}
```

Parameters

lpFlag

[in] Set request flag, please refer to table belowand RFID Request Flags. This parameter displayed in base hexadecimal notation.

Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1
0	Option_flag	Address_flag	Select_flag	0	0	0	0

lpAddressUID

[in] The unique identifier of ISO 15693 tag. This parameter apply determined by the **Address_flag**, please refer to ISO 15693 document.

lpOutData

[out] Receive the tag information.

Returned Values

Returning zero if the operation is successful, otherwise, indicates failure.

Example

```
char cData[128];
if(TagSystemInfo("00", "", cData) != 0)
    AfxMessageBox(_T("Get tag information fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: RFID.h

Link Library: RFID.lib

Link DLL: RFID.dll

Device: PT90

TagStayQuiet

Set ISO 15693 tag to stay quiet state.

```
int TagStayQuiet
{
    char lpFlag[],
    char lpAddressUID[],
}
```

Parameters

lpFlag

[in] Set request flag, please refer to table belowand RFID Request Flags. This parameter displayed in base hexadecimal notation.

Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1
0	0	Address_flag	Select_flag	0	0	0	0

lpAddressUID

[in] The unique identifier of ISO 15693 tag. This parameter apply determined by the **Address_flag**, please refer to ISO 15693 document.

Returned Values

Returning zero if the operation is successful, otherwise, indicates failure.

Example

```
char cUID[40];
Inventory15693("04", "", cUID);
if(TagStayQuiet("20", cUID) != 0)
    AfxMessageBox(_T("Set stay quiet state fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: RFID.h

Link Library: RFID.lib

Link DLL: RFID.dll

Device: PT90

TagSelect

Set ISO 15693 tag to select state.

```
int TagSelect
{
    char lpFlag[],
    char lpAddressUID[],
}
```

Parameters

lpFlag

[in] Set request flag, please refer to table belowand RFID Request Flags. This parameter displayed in base hexadecimal notation.

Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1
0	0	Address_flag	Select_flag	0	0	0	0

lpAddressUID

[in] The unique identifier of ISO 15693 tag. This parameter apply determined by the **Address_flag**, please refer to ISO 15693 document.

Returned Values

Returning zero if the operation is successful, otherwise, indicates failure.

Example

```
char cUID[40];
Inventory15693("04", "", cUID);
if(TagSelect("20", cUID) != 0)
    AfxMessageBox(_T("Set select state fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: RFID.h

Link Library: RFID.lib

Link DLL: RFID.dll

Device: PT90

TagResetToReady

Set select or ouiet tag to ready state.

```
int TagResetToReady
{
    char lpFlag[],
    char lpAddressUID[],
}
```

Parameters

lpFlag

[in] Set request flag, please refer to table belowand RFID Request Flags. This parameter displayed in base hexadecimal notation.

Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1
0	0	Address_flag	Select_flag	0	0	0	0

lpAddressUID

[in] The unique identifier of ISO 15693 tag. This parameter apply determined by the **Address_flag**, please refer to ISO 15693 document.

Returned Values

Returning zero if the operation is successful, otherwise, indicates failure.

Example

```
char cUID[40];
Inventory15693("04", "", cUID);
if(TagResetToReady("20", cUID) != 0)
    AfxMessageBox(_T("Set ready state fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: RFID.h

Link Library: RFID.lib

Link DLL: RFID.dll

Device: PT90

OpenCard14443A

Open ISO 14443A tag of mifare type and get the tag UID.

```
int OpenCard14443A
{
    char lpOutUID[],
    char lpOutType[]
}
```

Parameters

lpOutUID

[out] Receive the tag unique identifier.

lpOutData

[out] Receive the tag type. Please refer to mifare based document For example common values in the following table.

Value	Tag type
0400	Mifare Classic 1k
0200	Mifare Classic 4k
4400	Mifare Ultralight

Returned Values

Returning zero if the operation is successful, otherwise, indicates failure.

Example

```
char cUID[40], cType[20];
if(OpenCard14443A(cUID, cType) != 0)
    AfxMessageBox(_T("Open card 14443A fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: RFID.h

Link Library: RFID.lib

Link DLL: RFID.dll

Device: PT90

ReadMifareBlock

Read data of single block from ISO 14443A mifare S50/S70 tag,

```
int ReadMifareBlock
{
    int iKeyType,
    char lpKey[],
    int iBlock,
    char lpOutData[]
}
```

Parameters

iKeyType

[in] The key type. This parameter must be one of the values in the table below.

Value	Key type
0	Key A
1	Key B

lpKey

[in] The key string, 12 byte.

iBlock

[in] The block index of mifare S50/S70 tag. Please refer to tag respective document.

lpOutData

[out] Receive the block data.

Returned Values

Returning zero if the operation is successful, otherwise, indicates failure.

Example

```
char cData[128];
if(ReadMifareBlock(0, "FFFFFFFFFFFF", 8, cData) != 0)
    AfxMessageBox(_T("Read mifare block fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: RFID.h

Link Library: RFID.lib

Link DLL: RFID.dll

Device: PT90

WriteMifareBlock

Write data of single block to ISO 14443A mifare S50/S70 tag.

```
int WriteMifareBlock
{
    int iKeyType,
    char lpKey[],
    int iBlock,
    char lpData[]
}
```

Parameters

iKeyType

[in] The key type. This parameter must be one of the values in the table below.

Value	Key type
0	Key A
1	Key B

lpKey

[in] The key string, 12 byte.

iBlock

[in] The block index of mifare S50/S70 tag. Please refer to tag respective document.

lpData

[in] The block data, 32byte.

Returned Values

Returning zero if the operation is successful, otherwise, indicates failure.

Example

```
if(WriteMifareBlock(0, "FFFFFFFFFFFF", 8, "11111111112222222222333333333344") != 0)
    AfxMessageBox(_T("Write mifare block fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: RFID.h

Link Library: RFID.lib

Link DLL: RFID.dll

Device: PT90

ReadUltraLightBlock

Read data of single block from ISO 14443A mifare ultralight tag.

```
int ReadUltraLightBlock
{
    int iBlock,
    char lpOutData[]
}
```

Parameters

iBlock

[in] The block index of mifare ultralight tag. Please refer to tag respective document.

lpOutData

[out] Receive the block data.

Returned Values

Returning zero if the operation is successful, otherwise, indicates failure.

Example

```
char cData[128];
if(ReadUltraLightBlock(8, cData) != 0)
    AfxMessageBox(_T("Read ultralight block fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: RFID.h

Link Library: RFID.lib

Link DLL: RFID.dll

Device: PT90

WriteUltraLightBlock

Write data of single block to ISO 14443A mifare ultralight tag.

```
int WriteUltraLightBlock
{
    int iBlock,
    char lpData[]
}
```

Parameters

iBlock

[in] The block index of mifare ultralight tag. Please refer to tag respective document.

lpData

[in] The block data, 8byte.

Returned Values

Returning zero if the operation is successful, otherwise, indicates failure.

Example

```
if(WriteUltralightBlock(8, "12345678") != 0)
    AfxMessageBox(_T("Write ultralight block fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: RFID.h

Link Library: RFID.lib

Link DLL: RFID.dll

Device: PT90

GetUid14443B

Get ISO 14443B tag UID.

```
int GetUid14443B
{
    int iType,
    char lpOutUID[]
}
```

Parameters

iType

[in] The tag type. This parameter must be one of the values in the table below.

Value	Key type
1	14443B
2	SR176
3	SRIX4K

lpOutUID

[out] Receive the tag unique identifier.

Returned Values

Returning zero if the operation is successful, otherwise, indicates failure.

Remarks

When tag type is SRIX4K, must read chip ID before get UID. Refer to SRIX4K Flow Chart.

Example

```
char cUID[40];
if(GetUid14443B(1, cUID) != 0)
    AfxMessageBox(_T("Get UID 14443B fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: RFID.h

Link Library: RFID.lib

Link DLL: RFID.dll

Device: PT90

SRIX4KChipID

Get ISO 14443B SRIX4K tag chip ID.

```
int SRIX4KChipID
{
    char lpOutID[]
}
```

Parameters

lpOutData

[out] Receive the tag chip ID.

Returned Values

Returning zero if the operation is successful, otherwise, indicates failure.

Example

```
char cID[4];
if(SRIX4KChipID(cID) != 0)
    AfxMessageBox(_T("Get SRIX4K chip ID fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: RFID.h

Link Library: RFID.lib

Link DLL: RFID.dll

Device: PT90

SRIX4KReadBlock

Read data of single block from ISO 14443B SRIX4K tag.

```
int SRIX4KReadBlock
{
    int iBlock,
    char lpOutData[]
}
```

Parameters

iBlock

[in] The block index of SRIX4K tag. Please refer to tag respective document.

lpOutData

[out] Receive the block data.

Returned Values

Returning zero if the operation is successful, otherwise, indicates failure.

Remarks

When applies this function, after must remove tag of RFID field. Otherwise, **SRIX4KChipID** function will not be able to get chip ID. Refer to SRIX4K Flow Chart.

Example

```
char cData[128];
if(SRIX4KReadBlock(8, cData) != 0)
    AfxMessageBox(_T("Read SRIX4K block fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: RFID.h

Link Library: RFID.lib

Link DLL: RFID.dll

Device: PT90

SRIX4KWriteBlock

Write data of single block to ISO 14443B SRIX4K tag.

```
int SRIX4KWriteBlock
{
    int iBlock,
    char lpData[]
}
```

Parameters

iBlock

[in] The block index of SRIX4K tag. Please refer to tag respective document.

lpOutData

[in] The block data, 8byte.

Returned Values

Returning zero if the operation is successful, otherwise, indicates failure.

Remarks

When applies this function, after must remove tag of RFID field. Otherwise, **SRIX4KChipID** function will not be able to get chip ID. Refer to SRIX4K Flow Chart.

Example

```
if(SRIX4KWriteBlock(8, "12345678") != 0)
    AfxMessageBox(_T("Write SRIX4K block fail"));
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: RFID.h

Link Library: RFID.lib

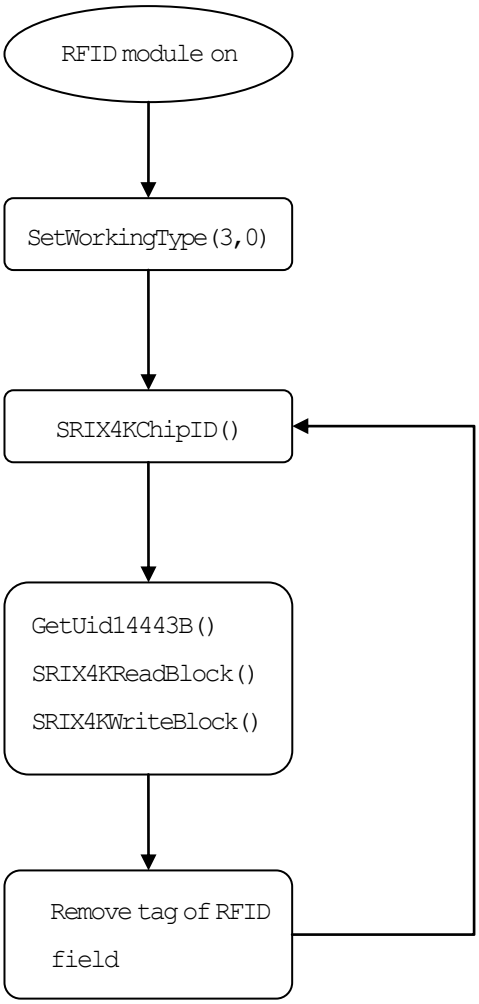
Link DLL: RFID.dll

Device: PT90

RFID Request Flags

Bit	Flag Name	Value	Description
Bit 5	Select_flag	0	Request executed by any tag according to the setting of Address flag.
		1	Request executed only by tag in selected state. The Address flag is set to 0 and the UID field is not included in the request.
Bit 6	Address_flag	0	Request is not addressed. UID field is not included. It can be executed by any tag.
		1	Request is addressed. UID field is included. It is exected only by the tag whose UID matches the UID specified in the request.
Bit 7	Option_flag	0	Meaning is defined by the command description. It is set to 0 if not otherwise defined by the command.
		1	Meaning is defined by the command description.

SRIX4K Flow Chart



SCANAPIAX.DLL

We supply SCANAPIAX.DLL to allow programmer to control the on-board scanning device. There are several functions available for programmer to tailor the application. Programmer can also use Windows CE development tool, such as Visual Studio 2005, to develop application program and control the on-board scanning device.

In the SCANAPIAX.DLL library, there are three functional groups can be used to control the scanning device. They are API_SCAN, Scan2Key, and Scanner related functions. Each functional group can be used to control the scanning device in different ways. These three functional groups can not be used at the same time. Programmer should select the most appropriate way to develop target application. The following shows function list of each functional group.

API_SCAN Related Functions

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

- 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 length of the scanned data.
- 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 the LED.
- API_LoadSettingFromFile – Load scanner settings from file.
- API_Reset – Reset scanner to default settings.
- API_ResetBarData – Clear data buffer so that the next new scanned data can be load into the buffer.
- API_SaveSettingToFile – Save the current scanner settings to file.
- API_SaveSettingsToScanner – Write scanner settings into scanner.
- S2K_IsLoad – Check if the scan.exe is running or not.
- S2K_Load – Load or unload the scan.exe.
- SCAN_QueryStatus – Query scanner settings.
- 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 settings in batch command.

Scan2Key Related Functions

Programmer can use Scan2Key related functions to control scan.exe program. When scan.exe is loaded, scanned data will be sent to key buffer. Target application program can retrieve scanned 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 scanner settings to default status.

Scanner Related Functions

Programmer can use Scanner related functions to control scanner module without messages. When target application is using Scanner related functions, the scanned data will be stored in system buffer.

- PT_EnableScanner – Enable scanner to scan barcode data.
- PT_DiableScanner – Disable scanner.
- PT_CheckBarcodeData – Check whether there is scanned data in system buffer.
- PT_GetBarcodeData – Get barcode data and type from system buffer.
- PT_SetDeault – Reset scanner settings to default status.

Scan Key Related Functions

- EnableTriggerKey – Enable and disable scan key.
- GetLibraryVersion – Get the library version.
- GetTriggerKeyStatus – Get scan key status.
- PressTriggerKey – Trigger scan key.
- TriggerStatus – Get scan key trigger status.

Scan Structure

- ScannerSetting Structure – Scanner Setting Information used by SCAN_BatchSetting and SCAN_BatchRead.
- GeneralSetting Structure – Information of Indication, Transmission, Scan, and String settings.
- Code11_Setting Structure – Information of Code11 settings.
- Code39_Setting Structure – Information of Code39 settings.
- Code93_Setting Structure – Information of Code93 settings.
- Code128_Setting Structure – Information of Code128 settings.
- Codabar_Setting Structure – Information of Codabar settings.

- EAN8_Setting Structure – Information of EAN8 settings.
- EAN13_Setting Structure – Information of EAN13 settings.
- Industrial25_Setting Structure – Information of Industrial 2 of 5 settings.
- Interleaved25_Setting Structure –Information of Interleaved 2 of 5 settings.
- MSI_Setting Structure – Information of MSI Plessey settings.
- UK_Setting Structure – Information of UK Plessey settings.
- Telepen_Setting Structure – Information of Telepen settings.
- UPCA_Setting Structure – Information of UPCA settings.
- UPCE_Setting Structure – Information of UPCE settings.
- Matrix25_Setting Structure – Information of Matrix25 settings.
- UEGeneral_Setting Structure – Information of UPC/EAN General settings.
- IATA25_Setting Structure – Information of IATA 2 of 5 settings.
- Trioptic_Setting Structure – Information of TRI-OPTIC settings.
- RSS_Setting Structure – Information of RSS settings.

Scan Command Table

The Scan Command Table of PT-90 terminal is used for SCAN_QueryStatus and SCAN_SendCommand functions. The Scan Command provides a different way to setup the scanning device.

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

To Register the target application to SCANAPIAX.dll so that SCANAPIAX.dll can communicate with the application. It will also set the scanning device to working mode.

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

Parameters

hwnd

[in] the window handling the function library will send message to report all activities of the scanning device.

Returned Values

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

Remarks

The target application must call API_Unregister to unregister from the dll and close the scanning device after the action been done. The messages can be one of the following:

SM_DATA_READY : Indicating that the barcode data is successfully read and ready for retrieval.

SM_ERROR_SYS : Indicating a system error is caused by calling system function. Calling API_GetSysError can get the system error code.

SM_ERROR_API : Indicating an error. Calling API_GetError can get the error code.

Example

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

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT90

API_Unregister

To Unregister the target application from SCANAPIAX.dll and close the scanning device.

```
void API_Unregister
{
}
```

Parameters

None

Return Values

None.

Example

API_Unregister();

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll
Device: PT90

API_GetBarData

To Get Barcode into the buffer. When receiving the message SM_DATA_READY, call this function can get the barcode data.

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

Parameters

buffer

[out] buffer for scanned data string.

uiLength

[in/out] buffer size

uiBarType

[out] barcode type.

Returned Values

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

Remarks

If the buffer size is smaller than the scanned data, this function will return 0 and the parameter uiLength will return the size of the buffer to adopt the scanned data.

Example

```
if(message == SM_DATA_READY){
    CString strBarData,strBarType;
    UINT  uiSize, 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 6.0 or beyond.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT90

API_GetBarDataLength

To Get Length of the scanned data

```
UINT API_GetBarDataLength
{
}
```

Parameters

None

Returned Values

Length of the scanned data

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 6.0 or beyond.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT90

API_GetBarType

To Get the barcode type.

```
UINT API_GetBarType
{
}
```

Parameters

None

Returned Values

Always return zero

Remarks

value	Barcode	value	Barcode
BC_CODE11(100)	Code 11	BC_UK_PLESSEY(111)	UK Plessey
BC_CODE39(101)	Code 39	BC_TELEPEN(112)	Telepen
BC_CODE93(102)	Code 93	BC_UPCA(113)	UPCA
BC_CODE128(103)	Code 128	BC_UPCE(114)	UPCE
BC_CODABAR(104)	Codabar	BC_MATRIX_25(115)	Matrix 25
BC_EAN8(105)	EAN8	BC_IATA_25(125)	IATA2 of 5
BC_EAN13(106)	EAN13	BC_TRIOPTIC(126)	TRI-OPTIC
BC_INDUSTRIAL_25(107)	Industrial 2 of 5	BC_RSS(127)	RSS
BC_INTERLEAVED_25(108)	Interleaved 2 of 5	BC_UCCEAN128(128)	UCC/EAN 128
BC_MSI_PLESSEY(110)	MSI Plessey		

Example

uiType = API_GetBarType();

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT90

API_GetError

To Get the error code.

```
DWORD API_GetError
{
}
```

Parameters

None

Returned Values

The returned value can be one of those in the table below:

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 6.0 or beyond.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT90

API_GetSysError

To Get the system error code.

```
DWORD API_GetSysError
{
}
```

Parameters

None

Returned Values

Returning the system error code that is returned by GetLastError(). Descriptions of system error code can be found in MSDN.

Example

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

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT90

API_GoodRead

To activate a buzzer sound when the buzzer of the scanning device is enabled and to flash the LED when the good-read LED of the scanning device is enabled.

```
void API_GoodRead
{
}
```

Parameters

None

Returned Values

None.

Remarks

Use API_GoodRead() to notify the user that a barcode data is successfully scanned. The buzzer function of the scanning device can be set by Scan Configuration in the control panel. The good-read LED function of the scanning device can be set by SCAN_SendCommand() function. If the buzzer and good-read LED functions are disabled, the API_GoodRead will do nothing.

Example

API_GoodRead();

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT90

API_LoadSettingsFromFile

To Load scanner settings from file.

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

Parameters

filename

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

Returned Values

Returning 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 6.0 or beyond.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT90

API_Reset

To Reset the scanner settings to default.

```
BOOL API_Reset
{
}
```

Parameters

None

Returned Values

Returning 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 6.0 or beyond.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT90

API_ResetBarData

To Clear the data buffer to allow the next scanned data coming in.

```
void API_ResetBarData
{
}
```

Parameters

None

ReturnedValues

None.

Example

API_ResetBarData();

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll
Device: PT90

API_SaveSettingsToFile

To Save current scanner settings to file. The extension file name is "axs".

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

Parameters

filename

[in] file name of the scanner settings.

Returned Values

Returning 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 6.0 or beyond.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT90

API_SaveSettingsToScanner

To Write current scanner settings into the scanner.

```
BOOL API_SaveSettingsToScanner
{
}
```

Parameters

None

Returned Values

Returning 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 6.0 or beyond.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT90

S2K_IsLoad

To Check if the application program scan.exe(scan barcode and send the scanned data into key buffer) is executing.

```
BOOL S2K_IsLoad
{
}
```

Parameters

None

Returned Values

The returned value TRUE indicates that scan.exe is running, The returned 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 6.0 or beyond.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT90

S2K_Load

To Load or Unload the the application program 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 been closed or Timeout by this parameter.

Returned Values

Returning 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 6.0 or beyond.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT90

SCAN_QueryStatus

To Query current scanner settings.

```
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 settings. This buffer size must be larger than 100.

ReturnedValues

Returning 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 settings 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 6.0 or beyond.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT90

SCAN_SendCommand

To Send scanner command to change the 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.

Returned Values

Returning 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 6.0 or beyond.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT90

SCAN_ResumeSystem

To Enable/Disable scan key to resume system

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

Parameters

bOn

[in] Flag that indicates whether to Enable(TRUE) scan key to resume system or Disable(FALSE) scan key to resume system.

Returned Values

If this action succeeds, the returned value is E_FUNC_SUCCEED. If this action fails, possible returned values are E_FUNC_ERROR, E_FUNC_PAR_ERROR, E_FUNC_SCANNER_NOT_OPEN.

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 6.0 or beyond.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT90

SCAN_BatchSetting

To Setup all scanner settings in batch command

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

Parameters

setting

[in] The ScannerSetting data structure.

Returned Values

If this action succeeds, the returned value is E_FUNC_SUCCEED.. If this action fails, possible returned values are E_FUNC_ERROR, E_FUNC_PAR_ERROR, E_FUNC_SCANNER_NOT_OPEN, E_FUNC_SETTING_FAIL.

Example

```
ScannerSetting          setting;
setting.generalsetting.m_uiLED = 0;
setting.Code11.m_uiRead = 1;
setting.Code39.m_uiRead = 1;
        ……..…..
SCAN_BatchSetting(setting);
```

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT90

SCAN_BatchRead

To Read all scanner settings in batch command

```
DWORD SCAN_BatcRead
{
    ScannerSetting *setting
}
```

Parameters

setting

[out] Pointer to ScannerSetting data structure.

Returned Values

If this action succeeds, the returned value is E_FUNC_SUCCEED. If this action fails, possible returned values are E_FUNC_SCANNER_NOT_OPEN, E_FUNC_PAR_ERROR.

Example

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

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT90

Scan2Key Related Functions

PT_OpenScan2Key

To Execute scan.exe to scan barcode and send the scanned data into terminal key buffer.

```
BOOL PT_OpenScan2Key
{
}
```

Parameters

None

Returned Values

Returning 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 6.0 or beyond.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT90

PT_CloseScan2Key

To Close application program scan.exe.

```
void PT_CloseScan2Key
{
}
```

Parameters

None

Returned Values

None.

Example

PT_CloseScan2Key()

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT90

PT_SetToDefault

To Reset all the scanner settings to default value.

```
int PT_SetToDefault
{
}
```

Parameters

None

Returned Values

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

Example

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

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT90

Scanner Related Functions

PT_EnableScanner

To Enable scanner to scan barcode. This function will also get scanned data from scanning device and store in the system buffer. Application can use the other function, *PT_GetBarcodeData* , to retrieve scanned data from system buffer.

```
int PT_EnableScanner
{
}
```

Parameters

None

Returned Values

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

Example

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

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll
Device: PT90

PT_DisableScanner

To close the scanning device.

```
void PT_DisableScanner
{
}
```

Parameters

None

Returned Values

None.

Example

PT_DisableScanner();

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT90

PT_CheckBarcodeData

To Check whether there is scanned barcode data available in system buffer.

```
BOOL PT_CheckBarcodeData
{
}
```

Parameters

None

Returned Values

This function returns TRUE if there is scanned barcode data in system buffer and FALSE if there is no scanned 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 6.0 or beyond

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT90

PT_GetBarcodeData

To Get Barcode data and barcode 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 maximum buffer size

Returned Values

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

Remarks

If the buffer size is smaller than scanned data, this function will return 0 and the parameter *uiMaxBufferLen* will return the length of the scanned 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 6.0 or beyond.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT90

PT_SetDefault

To Reset the scanner settings to default.

```
BOOL PT_SetDefault
{
}
```

Parameters

None

Returned Values

Returning 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 6.0 or beyond.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT90

Scan Key Related Functions

EnableTriggerKey

To Enable or Disable the scan Trigger Key.

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

Parameters

bEnable

[in] Flag that indicates whether to Enable(TRUE) or Disable(FALSE) the scan Trigger Key.

Returned Values

If this action succeeds, the returned value is E_FUNC_SUCCEED. If this action fails, the returned values are E_FUNC_ERROR, E_FUNC_PAR_ERROR.

Remarks

This function is valid only if the scanning device is enabled. A Warm Reset will enable the scan Trigger Key automatically.

Example

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

Requirements

OS Versions: Windows CE 6.0 or beyond

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT90

GetLibraryVersion

To Get Library Version information.

```
int GetLibraryVersion
{
}
```

Parameters

None

Returned Values

The version information, for example, if the returned value is 301, it means that dll version is 3.01

Example

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

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT90

GetTriggerKeyStatus

To Get scan Trigger Key Status.

```
DWORD GetTriggerKeyStatus
{
}
```

Parameters

None.

Returned Values

Returned value 1 indicates that the scan Trigger Key is enabled. Returned value 0 indicates that the scan Trigger Key is disabled.

Example

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

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT90

PressTriggerKey

To Trigger the Scan Key.

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

Parameters

bPress

[in] Flag that indicates whether to Press(TRUE) or Release(FALSE) the scan Trigger Key.

Returned Values

If this action succeeds, the returned value is E_FUNC_SUCCEED. If this action fails, the returned value is E_FUNC_ERROR.

Remarks

This function is valid only if the scanning device is enabled.

Example

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

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT90

TriggerStatus

To get the scan Trigger Key status.

```
DWORD TriggerStatus
{
}
```

Parameters

None.

Returned Values

The returned value 1 indicates that the scan Trigger Key is pressed and 0 indicates that scan Trigger Key is released.

Example

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

Requirements

OS Versions: Windows CE 6.0 or beyond.

Header: scanapiax.h

Link Library: scanapiax.lib

Link DLL: scanapiax.dll

Device: PT90

Scan Structure

ScannerSetting Structure

This setting file contains information used by SCAN_BatchSetting and SCAN_BatchRead..

```
Struct ScannerSetting
{
    DWORD cbSize;
    struct GeneralSetting    generalsetting;
    struct Code11_Setting    Code11;
    struct Code39_Setting    Code39;
    struct Code93_Setting    Code93;
    struct Code128_Setting   Code128;
    struct Codabar_Setting   Codabar;
    struct EAN8_Setting      EAN8;
    struct EAN13_Setting     EAN13;
    struct Industrial25_Setting   Indust25;
    struct Interleaved25_Setting  Inter25;
    struct MSI_Setting       MSIPlessey;
    struct UK_Setting        UKPlessey;
    struct Telepen_Setting   Telepen;
    struct UPCA_Setting      UPCA;
    struct UPCE_Setting      UPCE;
    struct Matrix25_Setting  Matrix25;
    struct UEGeneral_Setting UEGeneral;
    struct IATA25_Setting    IATA25;
    struct Trioptic_Setting  Trioptic;
    struct Rss_Setting       RSS;
}
```

Members

cbSize

This is required. It is the size of structure *ScannerSetting* , in bytes.

generalsetting

GeneralSetting Structure.

Code11

Code11_Setting Structure.

Code39

Code39_Setting Structure.

Code93

Code93_Setting Structure.

Code128

Code128_Setting Structure.

Codabar

Codabar_Setting Structure.

EAN8

EAN8_Setting Structure.

EAN13

EAN13_Setting Structure.

Indust25

Industrial25_Setting Structure.

Inter25

Interleaved25_Setting Structure.

MSIPlessey

MSI_Setting Structure.

UKPlessey

UK_Setting Structure.

Telepen

Telepen_Setting Structure.

UPCA

UPCA_Setting Structure.

UPCE

UPCE_Setting Structure.

Matrix25

Matrix25_Setting Structure.

UEGeneral

UEGeneral_Setting Structure.

IATA25

IATA25_Setting Structure.

Trioptic

Trioptic_Setting Structure.

RSS

RSS_Setting Structure.

Remarks

The *cbSize* must be the size of Structure *ScannerSetting*, in bytes.

Structure Information

Header: scanapiax.h

Device: PT90

GeneralSetting Structure

This setting file contains information used by *Structure ScannerSetting*.

```
Struct GeneralSetting
{
    UINT    m_uiLED;
    UINT    m_uiBeep;
    UINT    m_uiVibrator;
    UINT    m_uiScanResume;
    UINT    m_uiPosition;
    UINT    m_uiAimID;
    UINT    m_uiGlobalMin;
    UINT    m_uiGlobalLock;
    UINT    m_ui1Type;
    UINT    m_ui1Length;
    UINT    m_ui2Type;
    UINT    m_ui2Length;
    UINT    m_ui3Type;
    UINT    m_ui3Length;
    UINT    m_ui4Type;
    UINT    m_ui4Length;
    UINT    m_ui5Type;
    UINT    m_ui5Length;
    UINT    m_ui6Type;
    UINT    m_ui6Length;
    UINT    m_ui7Type;
    UINT    m_ui7Length;
    UINT    m_uiScanTout;
    UINT    m_uiIdleTout;
    Unsigned char    m_strPreamble[10];
    Unsigned char    m_strPostamble[10] ;
}
```

Members

All members

See Indication, Transmission, Scan and String setting settings in Scan Command Table .

Structure Information

Header: scanapiax.h

Device: PT90

Code11_Setting Structure

This setting contains information used by *Structure ScannerSetting*.

```
Struct Code11_Setting
{
    UINT    m_uiRead;
    UINT    m_uiChkDig;
    UINT    m_uiXmitChkDig;
    UINT    m_uiCodeID;
}
```

Members

All members

See Code11 settings in Scan Command Table .

Structure Information

Header: scanapiax.h

Device: PT90

Code39_Setting Structure

This setting file contains information used by *Structure ScannerSetting*.

```
Struct Code39_Setting
{
    UINT    m_uiRead;
    UINT    m_uiChkDig;
    UINT    m_uiXmitChkDig;
    UINT    m_uiCodeID;
    UINT    m_uiFullASCII;
    UINT    m_uiXmitStarStop;
    UINT    m_uiPharmacode;
    UINT    m_uiPharmacodeOnly;
}
```

Members

All members

See Code39 settings in Scan Command Table .

Structure Information

Header: scanapiax.h

Device: PT90

Code93_Setting Structure

This setting file contains information used by *Structure ScannerSetting*.

```
Struct Code93_Setting
{
    UINT    m_uiRead;
    UINT    m_uiCodeID;
}
```

Members

All members

See Code93 settings in Scan Command Table .

Structure Information

Header: scanapiax.h

Device: PT90

Code128_Setting Structure

This setting file contains information used by *Structure ScannerSetting*.

```
Struct Code128_Setting
{
    UINT    m_uiRead;
    UINT    m_uiCodeID;
    UINT    m_uiUCCEAN128;
    UINT    m_uiISBT128;
    UINT    m_uiISBT128XmitID;
    UINT    m_uiISBT128Concat;
    UINT    m_uiISBT128Form;
}
```

Members

All members

See Code128 settings in Scan Command Table .

Structure Information

Header: scanapiax.h

Device: PT90

Codabar_Setting Structure

This setting file contains information used by *Structure ScannerSetting*.

```
Struct Codabar_Setting
{
    UINT    m_uiRead;
    UINT    m_uiChkDig;
    UINT    m_uiXmitChkDig;
    UINT    m_uiCodeID;
    UINT    m_uiXmitStarStop;
    UINT    m_uiDualField;
}
```

Members

All members

See Codabar settings in Scan Command Table .

Structure Information

Header: scanapiax.h

Device: PT90

EAN8_Setting Structure

This setting file contains information used by *Structure ScannerSetting*.

```
Struct EAN8_Setting
{
    UINT    m_uiRead;
    UINT    m_uiXmitChkDig;
    UINT    m_uiCodeID;
    UINT    m_uiConvertToEAN13;
}
```

Members

All members

See EAN8 settings in Scan Command Table .

Structure Information

Header: scanapiax.h

Device: PT90

EAN13_Setting Structure

This setting file contains information used by *Structure ScannerSetting*.

```
Struct EAN13_Setting
{
    UINT    m_uiRead;
    UINT    m_uiXmitChkDig;
    UINT    m_uiCodeID;
    UINT    m_uiISBNConvert;
    UINT    m_uiISBNSuppReq;
    UINT    m_uiISMNConvert;
    UINT    m_uiISMNSuppReq;
}
```

Members

All members

See EAN13 settings in Scan Command Table .

Structure Information

Header: scanapiax.h

Device: PT90

Industrial25_Setting Structure

This setting file contains information used by *Structure ScannerSetting*.

```
Struct Industrial25_Setting
{
    UINT    m_uiRead;
    UINT    m_uiCodeID;
}
```

Members

All members

See Industrial 2 of 5 settings in Scan Command Table .

Structure Information

Header: scanapiax.h

Device: PT90

Interleaved25_Setting Structure

This setting file contains information used by *Structure ScannerSetting*.

```
Struct Interleaved25_Setting
{
    UINT    m_uiRead;
    UINT    m_uiChkDig;
    UINT    m_uiXmitChkDig;
    UINT    m_uiCodeID;
}
```

Members

All members

See Interleaved 2 of 5 settings in Scan Command Table .

Structure Information

Header: scanapiax.h

Device: PT90

MSI_Setting Structure

This setting file contains information used by *Structure ScannerSetting*.

```
Struct MSI_Setting
{
    UINT    m_uiRead;
    UINT    m_uiChkDig;
    UINT    m_uiXmitChkDig;
    UINT    m_uiCodeID;
}
```

Members

All members

See MSI Plessey settings in Scan Command Table .

Structure Information

Header: scanapiax.h

Device: PT90

UK_Setting Structure

This setting file contains information used by *Structure ScannerSetting*.

```
Struct UK_Setting
{
    UINT    m_uiRead;
    UINT    m_uiXmitChkDig;
    UINT    m_uiCodeID;
}
```

Members

All members

See UK Plessey settings in Scan Command Table .

Structure Information

Header: scanapiax.h

Device: PT90

Telepen_Setting Structure

This setting file contains information used by *Structure ScannerSetting*.

```
Struct Telepen_Setting
{
    UINT    m_uiRead;
    UINT    m_uiCodeID;
    UINT    m_uiFullASCII;
}
```

Members

All members

See Telepen settings in Scan Command Table .

Structure Information

Header: scanapiax.h

Device: PT90

UPCA_Setting Structure

This setting file contains information used by *Structure ScannerSetting*.

```
Struct UPCA_Setting
{
    UINT    m_uiRead;
    UINT    m_uiXmitChkDig;
    UINT    m_uiCodeID;
    UINT    m_uiConvertToEAN13;
    UINT    m_uiCoupon;
    UINT    m_uiCouponXmitID;
    UINT    m_uiXmitNumSys;
}
```

Members

All members

See UPCA settings in Scan Command Table .

Structure Information

Header: scanapiax.h

Device: PT90

UPCE_Setting Structure

This setting file contains information used by *Structure ScannerSetting*.

```
Struct UPCE_Setting
{
    UINT    m_uiRead;
    UINT    m_uiXmitChkDig;
    UINT    m_uiCodeID;
    UINT    m_uiConvertToUPCA;
    UINT    m_uiXmitNumSys;
}
```

Members

All members

See UPCE settings in Scan Command Table .

Structure Information

Header: scanapiax.h

Device: PT90

Matrix25_Setting Structure

This setting file contains information used by *Structure ScannerSetting*.

```
Struct Matrix25_Setting
{
    UINT    m_uiRead;
    UINT    m_uiChkDig;
    UINT    m_uiXmitChkDig;
    UINT    m_uiCodeID;
}
```

Members

All members

See Matrix25 settings in Scan Command Table .

Structure Information

Header: scanapiax.h

Device: PT90

UEGeneral_Setting Structure

This setting file contains information used by *Structure ScannerSetting*.

```
Struct UEGeneral_Setting
{
    UINT    m_uiSuppReq;
    UINT    m_ui2DigSupp;
    UINT    m_ui5DigSupp;
    UINT    m_ui2DigRedu;
    UINT    m_ui5DigRedu;
    UINT    m_uiGTIN;
}
```

Members

All members

See UPC/EAN General settings in Scan Command Table .

Structure Information

Header: scanapiax.h

Device: PT90

IATA25_Setting Structure

This setting file contains information used by *Structure ScannerSetting*.

```
Struct IATA25_Setting
{
    UINT    m_uiRead;
    UINT    m_uiCodeID;
}
```

Members

All members

See IATA2 of 5 settings in Scan Command Table .

Structure Information

Header: scanapiax.h

Device: PT90

Trioptic_Setting Structure

This setting file contains information used by *Structure ScannerSetting*.

```
Struct Trioptic_Setting
{
    UINT    m_uiRead;
    UINT    m_uiCodeID;
    UINT    m_uiConversion;
}
```

Members

All members

See TRI-OPTIC settings in Scan Command Table .

Structure Information

Header: scanapiax.h

Device: PT90

RSS_Setting Structure

This setting file contains information used by *Structure ScannerSetting*.

```
Struct RSS_Setting
{
    UINT    m_uiRead;
    UINT    m_uiXmitAppID;
    UINT    m_uiXmitChkDig;
    UINT    m_uiXmitSymID;
}
```

Members

All members

See RSS settings in Scan Command Table .

Structure Information

Header: scanapiax.h

Device: PT90

Scan Command Table

Command1	Command2	Value
5 Indication	2 LED indication	0: Disable 1: Enable *
	3 Buzzer indication	0: Disable 1: Enable *
	4: Vibrator	0: Disable * 1: Enable
	5 Resume System Using ScanKey	0: Disable * 1: Enable
6 Transmission	7 Code ID position	0: Before code data * 1: After code data
	12 AIM ID	0: Disable * 1: Enable
	13 Code ID	0: Clear 1: Proprietary See Note 1
7 Scan	6: Global min. code length	0~80 (0: Disable) 3 *
	9: Global lock code length	0~80 (0: Disable) 0 *
	10: Configurable code length #1	See Note 2
	11: Configurable code length #2	See Note 2
	12: Configurable code length #3	See Note 2
	13: Configurable code length #4	See Note 2
	14: Configurable code length #5	See Note 2
	15: Configurable code length #6	See Note 2
	16: Configurable code length #7	See Note 2

	17: Scan Timeout (Sec)	1~30 See Note 3 2 *
	18: Idle Timeout (Sec)	0~255 (0: Disable) See Note 3 5 *
8 String setting	3 Preamble characters settings	0x01 ~ 0x7F ASCII code (1-10 characters) 00 *
	4 Postamble characters settings	0x01 ~ 0x7F ASCII code (1-10 characters) 00 *
10 Code 11	1 Read	0: Disable * 1: Enable
	2 Check Digit	1: One digit * 2: Two digits
	3 Transmit Check Digit	0: Disable * 1: Enable
	8 Code ID setting	0: Disable * 0x01 ~ 0x7F ASCII code(1 bytes)
11 Code 39	1 Read	0: Disable 1: Enable *
	2 Check Digit	0: Disable * 1: Enable
	3 Transmit Check Digit	0: Disable * 1: Enable
	8 Code ID setting	0: Disable * 0x01 ~ 0x7F ASCII code(1 bytes)
	10 Full ASCII	0: Disable * 1: Enable
	13 Transmit Start/Stop Characters	0: Disable * 1: Enable
	14 Italian Pharmacode	0: Disable * 1: Enable
	15 Italian Pharmacode Only	0: Disable 1: Enable *
12 Code 93	1 Read	0: Disable 1: Enable *
	8 Code ID setting	0: Disable * 0x01 ~ 0x7F ASCII code(1 bytes)
13 Code 128	1 Read	0: Disable 1: Enable *

	8 Code ID setting	0: Disable * 0x01 ~ 0x7F ASCII code(1 bytes)
	10 UCC/EAN 128	0: Disable * 1: Enable
	14 ISBT 128	0: Disable * 1: Enable
	15 ISBT 128 Transmit Identifier Data	0: Disable 1: Enable *
	16 ISBT 128 Concatenation	0: Disable 1: Enable *
	17 ISBT 128 Form:	0: =A+=% * 1: =A+&; 2: =A+&! 3: =<+=> 4: =<+&> 5: &<+=> 6: &<+&>
14 Codabar	1 Read	0: Disable * 1: Enable
	2 Check Digit	0: Disable * 1: Enable
	3 Transmit Check Digit	0: Disable * 1: Enable
	8 Code ID setting	0: Disable * 0x01 ~ 0x7F ASCII code(1 bytes)
	11 Transmit Start/Stop Characters	0: Disable * 1: Enable
	12 Dual Field	0: Disable * 1: Enable
15 EAN 8	1 Read	0: Disable 1: Enable *
	3 Transmit Check Digit	0: Disable 1: Enable *
	8 Code ID setting	0: Disable * 0x01 ~ 0x7F ASCII code(1 bytes)
	13 Convert to EAN-13	0: Disable * 1: Enable
16	1	0: Disable

EAN 13	Read	1: Enable *
	3 Transmit Check Digit	0: Disable 1: Enable *
	8 Code ID setting	0: Disable * 0x01 ~ 0x7F ASCII code(1 bytes)
	12 ISBN conversion	0: Disable * 1: Enable
	13 ISBN Supplement Required	0: Disable * 1: Enable
	14 ISMN conversion	0: Disable * 1: Enable
	15 ISMN Supplement Required	0: Disable * 1: Enable
17 Industrial 2 of 5	1 Read	0:Disable * 1:Enable
	8 Code ID setting	0: Disable * 0x01 ~ 0x7F ASCII code(1 bytes)
18 Interleaved 2 of 5	1 Read	0: Disable 1: Enable *
	2 Check Digit	0: Disable * 1: Enable
	3 Transmit Check Digit	0: Disable * 1: Enable
	8 Code ID setting	0: Disable * 0x01 ~ 0x7F ASCII code(1 bytes)
20 MSI Plessey	1 Read	0: Disable * 1: Enable
	2 Check Digit	0: Disable 1: One digit (MOD 10) * 2: Two digit (MOD10/10)
	3 Transmit Check Digit	0: Disable * 1: Enable
	8 Code ID setting	0: Disable * 0x01 ~ 0x7F ASCII code(1 bytes)
21 UK Plessey	1 Read	0: Disable * 1: Enable
	3 Transmit Check Digit	0: Disable * 1: Enable

	8 Code ID setting	0: Disable * 0x01 ~ 0x7F ASCII code(1 bytes)
22 Telepen	1 Read	0: Disable * 1: Enable
	8 Code ID setting	0: Disable * 0x01 ~ 0x7F ASCII code(1 bytes)
	10 Full ASCII	0: Disable * 1: Enable
23 UPCA	1 Read	0: Disable 1: Enable *
	3 Transmit Check Digit	0: Disable 1: Enable *
	8 Code ID setting	0: Disable * 0x01 ~ 0x7F ASCII code(1 bytes)
	12 Convert to EAN-13	0: Disable * 1: Enable
	13 Coupon	0: Disable * 1: Enable See Note 4
	14 Coupon Transmit "]C1"	0: Disable 1: Enable *
	15 Transmit Number System	0: Disable 1: Enable *
24 UPCE	1 Read	0: Disable 1: Enable *
	3 Transmit Check Digit	0: Disable 1: Enable *
	8 Code ID setting	0: Disable * 0x01 ~ 0x7F ASCII code(1 bytes)
	14 Convert to UPC-A	0: Disable * 1: Enable
	15 Transmit Number System	0: Disable 1: Enable *
25 Matrix 25	1 Read	0: Disable * 1: Enable
	2 Check Digit	0: Disable * 1: Enable
	3 Transmit Check Digit	0: Disable * 1: Enable

	8 Code ID setting	0: Disable * 0x01 ~ 0x7F ASCII code(1 bytes)
35 UPC/EAN General	1 Supplements Required	0: Disable * 1: Enable
	2 Two Digit Supplements	0: Disable * 1: Enable
	3 Five Digit Supplements	0: Disable * 1: Enable
	4 Two Digit Redundancy	0: Disable * 1: Enable
	5 Five Digit Redundancy	0: Disable * 1: Enable
	6 GTIN Formatting	0: Disable * 1: Enable
36 IATA 2 of 5	1 Read	0: Disable * 1: Enable See Note 5
	8 Code ID setting	0: Disable * 0x01 ~ 0x7F ASCII code(1 bytes)
37 TRI-OPTIC	1 Read	0: Disable * 1: Enable
	8 Code ID setting	0: Disable * 0x01 ~ 0x7F ASCII code(1 bytes)
	10 Conversion	0: Disable * 1: Enable
38 RSS	1 Read	0: Disable * 1: Enable
	2 Transmit Application ID	0: Disable 1: Enable *
	3 Transmit Check Digit	0: Disable 1: Enable *
	4 Transmit Symbology ID "]e0"	0: Disable * 1: Enable

Note1:

This command will set all barcode ID settings to proprietary value. It can't be queried.

Barcode	value	Barcode	value
Code 11	100	MSI Plessey	110
Code 39	101	UK Plessey	111
Code 93	102	Telepen	112
Code 128	103	UPCA	113
Codabar	104	UPCE	114
EAN8	105	Matrix 25	115
EAN13	106	IATA 25	125
Industrial 2 of 5	107	Trioptic	126
Interleaved 2 of 5	108		

Note 2:

There are seven barcode length-lock available. Specific barcode type can be assigned with a length-lock.

Code type:

	Hex Value
CODE11	64
CODE39	65
CODE93	66
CODE128	67
CODABAR	68
INDUSTRIAL_25	6B
INTERLEAVED_25	6C
MSI_PLESSEY	6E
UK_PLESSEY	6F
TELEPEN	70
MATRIX_25	73
TRIOPTIC	7E

Length:

0~80 (0: Disable)

Default setting:

0000

Example:

Cmd1	Cmd2	Value
7	10	670C
7	11	6514
7	12	6710

Code 128: Can only read barcode of 12- or 16-digit.

Code 39: Can only read barcode of 20-digit.

Note 3:

ScanTimeout: the maximum time, in seconds, allowed during which the scanning beam remains ON without decoding any barcode.

IdleTimeout: the maximum time, in seconds, allowed during which the scanning device remains idle without any action.

Note 4:

When this setting is Enabled, the UCC / EAN 128 barcode can not be read.

Note 5:

IATA 2 of 5 only support 13- or 15-digit.

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	An invalid parameter was passed to the function.
E_FUNC_SCANNER_NOT_OPEN	0x00000004	The scanning device is not enabled.
E_FUNC_SETTING_FAIL	0x00000005	The function setting failed.
BT_ERR_CREATE_FAIL	0x00001001	BlueTooth module startup fail
BT_ERR_INUSING	0x00001002	BlueTooth module is using by other application
BT_ERR_DEVICE_ERROR	0x00001003	BlueTooth Initial setting fail
BT_ERR_SETTING_FAIL	0x00001004	BlueTooth setup fail
BT_ERR_REG_DEV_FAIL	0x00001005	Register communication port fail
BT_ERR_SPP_COM_FAIL	0x00001006	SPP service com open fail
BT_ERR_INSUFFICIENT	0x00001007	The buffer for receivc data is insufficient
BT_ERR_PAIR_FAIL	0x00001008	Pait to device fail
BT_ERR_CHANNEL	0x00001009	SPP channel error
BT_ERR_FTP_SERVER_REJECT	0x00001010	FTP server reject connect request
BT_ERR_DIVICE_NOT_CONNECT	0x00001011	FTP service device not connect
BT_ERR_FTP_DIR_FAIL	0x00001012	Search the direction fail
BT_ERR_FTP_EMPTY_FILE	0x00001013	No more file data
BT_ERR_CONNECTED	0x00001014	The device had connected